Learning Task-Specific State Representations by Maximizing Slowness and Predictability

Rico Jonschkowski[1] and **Oliver Brock**[1]

Abstract. The success of reinforcement learning in robotic tasks is highly dependent on the state representation – a mapping from high dimensional sensory observations of the robot to states that can be used for reinforcement learning. Even though many methods have been proposed to learn state representations, it remains an important open problem. Identifying the characteristics existing methods are optimizing to find good state representations, combining them, and adding new characteristics will lead to a more robust method for state representation learning. We define a new characteristic – predictability – and combine it with slowness. We implement these characteristics in a neural network and show that this approach can find good state representations from visual input in simulated robotic tasks.

1 Introduction

Reinforcement learning has been very successful in a number of robotic tasks [4]. As for all machine learning methods, the results depend greatly on the representation of the task – in this case the definition of the state s in terms of the usually high dimensional sensory observations of the robot. The state should include all the information necessary for the task at the right level of abstraction while ignoring information that is irrelevant.

Finding the right state representation for a robotic learning task is a difficult problem and a core part of solving the task. Currently, this part is carried out by humans, not by their robots. However, this approach does not scale to autonomously learning robots as we cannot equip the robots with the right representation for every task they might potentially have to learn. To overcome this problem, robots must be able to learn suitable state representations from experience.

The goal of state representation learning is the following:

A good state representation for a given task and reinforcement learning method is a function from past observations and actions to states that allows the robot to learn the task (receive high rewards after learning) by applying the reinforcement learning method on this representation of the task.

This goal description is intuitive and provides a way to compare representations (by training the robot using these representations and comparing the sum of rewards). Unfortunately, we cannot directly use this measure to discover state representations because it takes a long time to estimate and does not provide a gradient towards better representations. Thus, we need a different (heuristic) objective that characterizes good state representations to guide our search for them.

A number of methods have been proposed to find state representations. These methods focus on different characteristics that are thought to describe good state representations and are applicable to different kinds of tasks. However, no single characteristic seems to capture enough information about good state representations in general. We argue that multiple of these characteristics need to be combined and new ones added to build a more robust description of the objective for state representation learning.

Our contributions in this paper are to define a new characteristic – state predictability – and to combine it with the previously proposed slowness characteristic.

The rest of the paper is structured as follows: In the next section, we list and discuss the characteristics that are optimized in other approaches to state representation learning. In section 3, we describe the predictability characteristic, show how it can be combined with slowness, and present our implementation of this approach in a neural network. In section 4, we demonstrate our method on two simulated robotic tasks. The method discovers state representations from visual observations for these tasks. This allows the robot to solve the tasks at a similar level of performance as if it had direct access to a human designed state.

2 Related Work

Many different characteristics of good state representations have been proposed explicitly or implicitly. We first list these characteristics and then explain the intuition behind them.

A good state representation ...

1. ... *provides good features for learning the value function:* Singh et al. [12] form clusters of discrete states to reduce the temporal difference error. Piater et al. [9] partition the state space by learning a decision tree following the same objective. Menache et al. [8] learn a value function by simultaneously changing the basis function features and the linear regression weights.
2. ... *is a compact representation of the observations from which the original observations can be reproduced:* Lange et al. [5] discover state representations by compressing observations using a deep autoencoder and a self-organizing map. They demonstrate this on a real slot car task with raw visual observations.
3. ... *has states that change slowly over time such that consecutive states are proximate in state space*: Legenstein et al. [6] demonstrate that good state representations for reinforcement learning can be found by slow feature analysis. Luciw and Schmidhuber [7] extend this idea such that this representation can be built incrementally.
4. ... *can predict future observations given future actions*: Boots et al. [1] show how predictive state representations can be learned from visual data and used in a reinforcement learning setting.

[1] Robotics and Biology Laboratory, Technical University of Berlin, Germany, email: {rico.jonschkowski, oliver.brock}@tu-berlin.de

5. *. . . can predict future rewards given future actions*: Duell et al. [3] train the weights of a recurrent neural network that simultaneously learns the state representation, the transition function and the reward function to predict future rewards.
6. *. . . is shared by multiple similar tasks:* Caruana [2] finds features for supervised learning that are shared across related tasks. To our best knowledge this idea has not been applied to finding state representations for reinforcement learning.

Characteristic (1) is very general and directly related to the goal description in the introduction. The proposed methods perform reinforcement learning and representation learning simultaneously. However, these methods alone do not seem to solve the problem.

Dimensionality reduction approaches (2) try to find the underlying structure of the data. Applying such methods on observations can potentially result in meaningful state representations. The downside of this method is that it treats every observation independently, ignoring that they are produced in a dynamic process of interactions between robot and environment.

Slowness learning (3) focuses on the temporal property of the observation by maximizing slowness. Slowness is a reasonable assumption for many aspects in our world. Physical objects, for example, do not "teleport" but change their position gradually over time. However the slowest changing features are not necessarily the most important ones for a given task. We would not want a soccer robot to be fascinated by the slowness of the grass growing on the field while the game is running.

Predictive state representations (4) describe the current state as a set of predictions about future observations conditioned on future actions. Such a state representation allows the robot to choose the actions leading to the desired outcome. The problem is that it is unclear which predictions should be included in the state and which should not. (5) focuses on predicting the reward instead of the observations which allows it to focus on important aspects.

(6) shows that state representations do not have to be learned from a single task. While the solution of a task is very specific, the state representation can be less specific and also (partially) work for related tasks. For navigation tasks in a room, for example, the robot can use the same state representation to go to different goal locations if the state encodes the robot's position.

All of these methods try to find the structure that simplifies the given task by focusing on different characteristics. We think that there is not one general characteristic but a combination of them that allows to find this structure for a wide variety of tasks.

3 Our Approach

We want to add a new point to the list of characteristics: 7. *A good state representation can predict future states given future actions.* The difference to (4) and (5) is that this characteristic does not describe the predictiveness, but the predictability of states. This directly encodes the Markov property.

All of these characteristics (and more) are applicable to different kinds of tasks and many will have to be combined to solve state representation learning. We take one step in this direction by combining the predictability characteristic (7) with slowness (3) to find state representations. To avoid the trivial solution that maps all observations to a constant state, we also enforce diversity between states. For now, we restrict ourselves to fully observable tasks which means that the current state can be computed from the current observation alone.

More technically, the robot learns the state representation from a sequence $\{o_i, a_i, r_i\}_{i=1}^{N}$ of observations, actions, and rewards generated by some exploration policy. It does so by simultaneously learning the state representation $s_i = f(o_i)$ and the transition function $s_i = T(s_{i-1}, a_{i-1})$ to minimize the following cost function:

$$\begin{aligned} C(s_{0:N}, a_{0:N}, T) = &\omega_1 \sum_{i=1}^{N} \|s_i - s_{i-1}\|^2 + \\ &\omega_2 \sum_{i=k+1}^{N} \left(\|s_i - s_{i-k}\| - d\right)^2 + \\ &\omega_3 \sum_{i=1}^{N} \|T(s_{i-1}, a_{i-1}) - s_i\|^2 \end{aligned}$$

The first term of this cost function leads to slowness: consecutive states should have a small distance. The second term enforces variability between temporally distant states. It includes two parameters: the number of steps k that lie between distant states and the Euclidean distance d these states are supposed to have. Using this term works better than maximizing the variance or having a soft minimal distance constraint. The third term fosters predictability of future states given future actions of the robot. The terms can be weighted arbitrarily by ω_1, ω_2, and ω_3. We implement this cost function in a neural network (see Figure 1(a)).

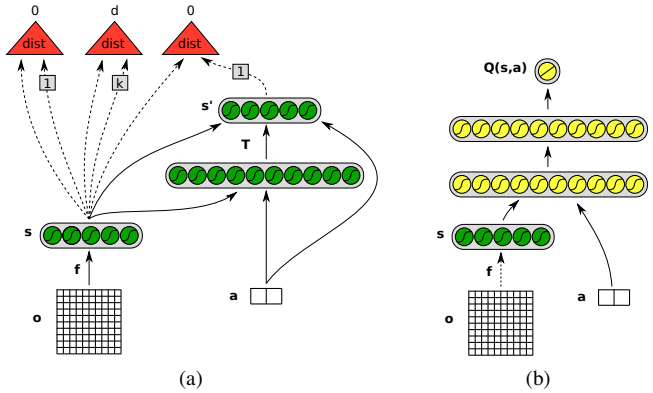


Figure 1: Neural network implementation of our method for state representation learning (a) and Q-learning (b). The green and yellow circles are neurons organized in layers, most of which are sigmoid (more accurately $\tanh$) neurons. Only the top neuron in (b) has a linear activation function. The neurons are colored to illustrate that the state representation learned in (a) is used in (b). Solid arrows symbolize full connections with adjustable weights between layers. The dashed arrows are non-adjustable weights, some of which have delays of 1 or k time steps. The red triangles compute the Euclidean distances that are the outputs of (a) and correspond to the three terms in the cost function. The target values are written on top of them.

After learning the state representation $s = f(o)$, this part of the network is used for neural fitted Q-iteration (NFQ) [10] - a batch version of Q-learning with a neural network as function approximator (see Figure 1(b)). The idea of NFQ is to alternate two steps:

1. The new $\hat{Q}$ targets are computed from transition data $\{s, a, r, s'\}$ and the current estimate of Q (computed by the neural network):

$$\hat{Q}(s,a) = \alpha(r + \gamma \max_{a'} Q(s', a')) + (1 - \alpha)Q(s,a)$$

2. The neural network is trained using $\hat{Q}$ as target values.

After training, the combined network (see Figure 1(b)) can score different action choices of the robot according to its current observation. To execute the learned behavior, the robot computes the score for all its actions and picks the action with the highest value.

4 Evaluation

4.1 Tasks

We performed experiments in simulation with two continuous tasks with high dimensional raw visual observations and stochastic actions.

The first task is to drive a slot car as fast as possible while not going too fast in the right curve as this curve has no side rails (see Figure 2 (a) and (b)). The track consists of two straight segments and two half circles summing to a total length of $2 + \pi$ meters. The slot car can choose its velocity from $\{0, 0.1, 0.2, \ldots, 1\}$ meters per time step to which uniform noise of $[-0.05, 0.05]$ meters per time step is added. Its observation is a ten by ten pixel gray scale image of the scene. When its current action would take the slot car into the right curve and its velocity exceeds 0.5, it gets a reward of -1 and stays in its current position instead (to simulate being thrown out of the track), otherwise the slot car gets a reward equal to its velocity. The optimal strategy for this task is to drive with maximum velocity except just before and in the first part of the curve, where the robot should drive 0.4 meters per time step to avoid being thrown out.

The second task is to maneuver a mobile robot to the top right corner of a room without bumping into the walls (see Figure 2 (c) and (d)). The room has a size of two by two meters. The corner has a size of 0.5 by 0.5 meters. The robot can control its velocity in x- and y-direction, choosing from $\{-0.4, -0.1, 0, 0.1, 0.4\}$ meters per time step. Uniform noise of $[-0.1, 0.1]$ times the total velocity is added to each of the directions independently. If the robot hits a wall it gets a reward of -1, else it gets a reward of +1 if it is in the top right corner, and 0 otherwise.

(a) (b) (c) (d)

Figure 2: The slot car task (a,b) and the mobile robot task (c,d).

4.2 Experiments

In each experiment, we initialize the position of the robot randomly and let it perform random actions for 1000 time steps to collect data $\{o_i, a_i, r_i\}_{i=1}^{1000}$. From this data the robot learns a five-dimensional state representation and a Q-function as described earlier.

We test the quality of the learned behavior in 20 randomly initialized episodes of length ten by computing the average reward over these. We compare the results with using raw image data directly as state and with using the first five slowest features or principal components of the image. Principal component analysis is computed on the same data as our method. The slow features are obtained by performing a linear slow feature analysis on the first ten principal components.

To get an upper and a lower bound on the performance of the robot, we also compare with using the (x,y) coordinates as the state and to a robot performing random actions. The experiment is repeated ten times to compute mean and standard deviation of the average reward.

4.3 Parameters of Our Method

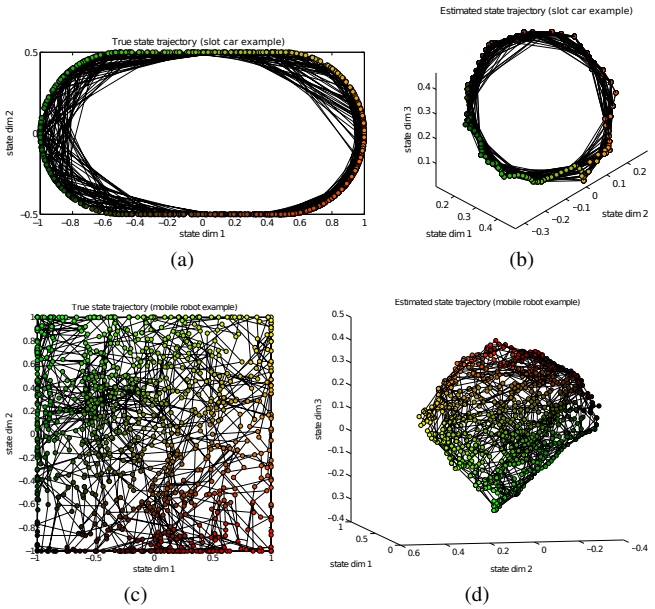
The weights in our cost function are $\omega_1 = 1$, $\omega_2 = 1$, and $\omega_3 = 4$, weighting the predictability term higher than the other terms. We pick $k = 100$ as this is sufficiently large so that states that are 100 time steps apart should be roughly uncorrelated and $d = 0.5$ as this is a distance that does not saturate the $\tanh$ neurons and allows them to use the linear or the nonlinear parts of the sigmoid.

The network for learning the state representation is trained for 20,000 epochs using RPROP [11] with strong regularization $\lambda = 0.01$. To avoid local minima, three restarts are performed and the best network according to the cost function is used as state representation.

For neural fitted Q-iteration, the parameters $\gamma = 0.9$ and $\alpha = 0.5$ are used. The neural network is trained for 100 epochs with RPROP without regularization before the $\hat{Q}$ values are recomputed. This process is iterated 50 times.

4.4 Results

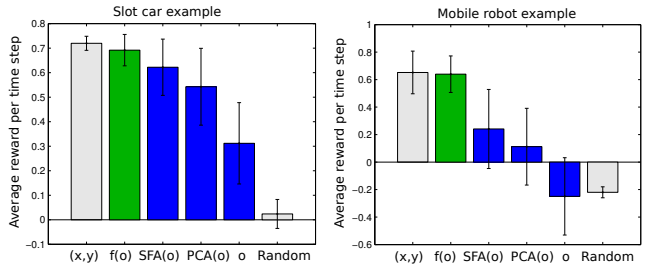
Our method discovers state representations that resemble important structures of the tasks. The state trajectories look similar in the learned state space compared to the coordinates of the robot, which are the intuitive state dimensions in these tasks (see Figure 3). It is promising that our method can discover these structures from raw visual observations and actions.



(a) (b) (c) (d)

Figure 3: Sample state trajectories of the random exploration phase for the slot car task (a,b) and the mobile robot task (c,d). Circles mark visited states. Lines between them indicate which states were visited after another. The plots on the left show the state trajectory in Cartesian coordinates. The plot on the right show the same trajectory in a learned three-dimensional state space. States are colored according to their real coordinates so that it is apparent which states in the different state spaces correspond to each other.

With the state representations that our method discovers, the slot car task can be learned to almost the same performance level as if the robot was given direct access to its coordinates (see Figure 4(a)). When the robot does not perform state representation learning but uses its observations as state, it does not learn the best strategy. It either learns to ignore its observations and always drives slow or it often drives too fast in the right curve. Using slow features or principal components of the observation as state representation leads to reasonable results in this example.



(a) (b)

Figure 4: Reinforcement learning success after neural fitted Q-iteration with different state representations. Using the Cartesian coordinates of the robot as state provides an upper bound (leftmost gray bar). Choosing actions at random provides a lower bound (rightmost gray bar). Our method (green bar) is compared to using slow features, principal components or the observation directly as state (blue bars).

In the mobile robot task, the state representation found by our method enables the robot to learn the task as well as if it could directly observe its coordinates (see Figure 4(b)). Learning this task from raw observations does not lead to a good strategy. Sometimes the robot trained directly on observations learned to stay still to avoid punishment for bumping into walls but often it completely failed and constantly ran into one direction. State representations found by slow feature analysis or principal component analysis worked only in a few trials.

In summary, our experiments show that with our method the robot discovered state representations from raw observations from which it could learn the task almost as well as if it had direct access to its Cartesian coordinates. This was generally not possible by using the observation, its principal components or slow features as state.

5 Conclusion

By optimizing state predictability combined with slowness, the robot was able to discover useful state representations that resemble important structures of the tasks from visual observations. These representations enabled the robot to successfully learn tasks that it could not learn from observations, their slowest features, or principal components directly. The performance of the robot using the learned state representation was comparable to the much easier version of this task in which the robot was given direct access to a human defined state.

The advantage of low-dimensional state representations that include all task-relevant dimensions seems to be that it constrains the solution of the task and thus allows better generalization. However, this information must be represented in such a way that the solution becomes compact and, thus, easy to learn. Combining multiple characteristics of good state representations is a promising way to find such representations.

Our method finds the task-relevant dimensions in the presented tasks because these are the only dimensions that influence the observation. For tasks in which there are also irrelevant information in the observations (e.g., a second slot car if it is not part of the task), another characteristic needs to be added to focus on relevant information. In future work, the reward should be included into the cost function to identify these dimensions.

Furthermore, the presented method needs to be extended to handle partially observable problems for which multiple past observations (and actions) are required to estimate the current state. Finally, the method should be applied on real robots.

References

[1] Byron Boots, Sajid M. Siddiqi, and Geoffrey J. Gordon, 'Closing the learning-planning loop with predictive state representations', *The International Journal of Robotics Research*, **30**(7), 954–966, (2011).

[2] Rich Caruana, 'Multitask learning', *Mach. Learn.*, **28**(1), 41–75, (1997).

[3] Siegmund Duell, Steffen Udluft, and Volkmar Sterzing, 'Solving partially observable reinforcement learning problems with recurrent neural networks', in *Neural Networks: Tricks of the Trade*, volume 7700 of *Lecture Notes in Computer Science*, 709–733, Springer Berlin Heidelberg, (2012).

[4] Jens Kober and Jan Peters, 'Reinforcement learning in robotics: A survey', in *Reinforcement Learning*, volume 12 of *Adaptation, Learning, and Optimization*, 579–610, Springer Berlin Heidelberg, (2012).

[5] S. Lange, M. Riedmiller, and A. Voigtlander, 'Autonomous reinforcement learning on raw visual input data in a real world application', in *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pp. 1–8. IEEE, (2012).

[6] Robert Legenstein, Niko Wilbert, and Laurenz Wiskott, 'Reinforcement learning on slow features of high-dimensional input streams', *PLoS Computational Biology*, **6**(8), (2010).

[7] Matthew Luciw and Juergen Schmidhuber, 'Low complexity proto-value function learning from sensory observations with incremental slow feature analysis', in *Artificial Neural Networks and Machine Learning – ICANN 2012*, volume 7553 of *Lecture Notes in Computer Science*, 279–287, Springer Berlin Heidelberg, (2012).

[8] Ishai Menache, Shie Mannor, and Nahum Shimkin, 'Basis function adaptation in temporal difference reinforcement learning', *Annals of Operations Research*, **134**, 215–238, (2005).

[9] Justus Piater, Sébastien Jodogne, Renaud Detry, Dirk Kraft, Norbert Krüger, Oliver Kroemer, and Jan Peters, 'Learning visual representations for perception-action systems', *Int. J. Rob. Res.*, **30**(3), 294–307, (2011).

[10] Martin Riedmiller, 'Neural fitted q iteration – first experiences with a data efficient neural reinforcement learning method', in *In 16th European Conference on Machine Learning*, pp. 317–328. Springer, (2005).

[11] Martin Riedmiller and Heinrich Braun, 'A direct adaptive method for faster backpropagation learning: The rprop algorithm', in *IEEE International Conference on Neural Networks*, pp. 586–591, (1993).

[12] Satinder P. Singh, Tommi Jaakkola, and Michael I. Jordan, 'Reinforcement learning with soft state aggregation', in *Advances in Neural Information Processing Systems 7*, pp. 361–368. MIT Press, (1995).