STATE2VEC: OFF-POLICY SUCCESSOR FEATURE APPROXIMATORS

Anonymous authors
Paper under double-blind review

ABSTRACT

A major challenge in reinforcement learning (RL) is how to design agents that are able to generalize across tasks that share common dynamics. A viable solution is meta-reinforcement learning, which identifies common structures among past tasks to be then generalized to new tasks (meta-test). In meta-training, the RL agent learns state representations that encode prior information from a set of tasks, used to generalize the value function approximation. This has been proposed in the literature as successor representation approximators. While promising, these methods do not generalize well across optimal policies, leading to sampling-inefficiency during meta-test phases. In this paper, we propose *state2vec*, an efficient and low-complexity framework for learning successor features which *(i)* generalize across policies, *(ii)* ensure sample-efficiency during meta-test. Representing each RL tasks as a graph, we extend the well known nod2vec framework to learn graph embeddings able to capture the discounted future state transitions in RL. The proposed off-policy state2vec captures the geometry of the underlying state space, making good basis functions for linear value function approximation.

1 INTRODUCTION

Reinforcement Learning (RL) can be described as the computational approach to learning from interaction. In RL, an agent acts in an environment and receives observations including a numerical reward. The reward is usually a function of the current state (of the environment) and the action taken by the agent. The goal of the agent is to learn how to act, i.e which control strategy (or *policy*) to adopt in specific situations in order to achieve a long term goal (maximizing the long term expected reward). RL problems are typically modelled as Markov Decision Processes (MDPs).

This general formulation allows us to describe a large variety of tasks of practical interest across diverse fields such as game-playing, robotics and traffic control (Mnih et al., 2015; Levine et al., 2016; Ma et al., 2018; Arel et al., 2010). With the rise of deep learning, RL has seen great success in the recent years with artificial agents now outperforming humans in increasingly challenging tasks (Graepel, 2016; Silver et al., 2017). However, these agents exhibit weak-AI behaviors, being highly task specific and with limited ability to generalise across similar tasks. While some tasks may be very different from one another (e.g., learning how to drive vs learning to master the game of chess), others do not differ much (e.g., taking the train to work vs taking the train home). In the latter case, it is clearly desirable for the agent to be able to leverage the knowledge acquired while solving one task to speed up the solving of other similar tasks. This ability for autonomous agent to re-use previous knowledge is known as *meta-learning* (Mehta et al., 2008; Schmidhuber et al., 1996). The key challenge is for the agent to adapt or generalize to new tasks and new environments that have never been encountered during training time.

In this work, we consider a meta-reinforcement learning (meta-RL) problem, in which tasks are characterized by the same environment (shared structure) but with the reward function changing arbitrarily across tasks. Here, the agent learns at two different time scales: slow meta-learning exploiting the large experienced accumulated across tasks (learning of the shared structure), fast learning on individual tasks. This enables learning how to quickly adapt to a previously unseen task with little data. The successor representation (Dayan, 1993) decouples the environment from the reward in the value function computation, in such a way that one remain fixed should the other change. It is thus a natural tool to consider for achieving meta-RL. In fact, several recent works have

adopted the successor features approach for meta-RL across tasks that share common dynamics (Barreto et al., 2017; 2018; Borsa et al., 2019). These studies are highly promising as they show the ability for autonomous systems to transfer knowledge across tasks. However, they exhibit two major limitations: *(i)* the learning of the successor features (meta-training) is expensive and *(ii)* the meta-testing is not sample-efficient, especially for tasks that do not share a common or similar optimal policy. The reason for this is that the learned successor features are heavily dependent on the on-policy experience, requiring a re-training phase for each individual task Lehnert et al. (2017).

In this paper, we address these challenges by developing an off-policy meta-RL algorithm that disentangles task inference and control. The overall goal is to find the optimal balance between data-training and data-testing, by differentiating the data used in the meta-training with respect to the data used to train the policy. We propose *state2vec*, an efficient yet reliable framework for learning the successor features. We are interested in learning features that capture the underlying geometry of the state space. In particular, we seek the following properties for the features: *(i)* to be learned from data rather than handcrafted − to avoid structural bias, see Madjiheurem & Toni (2019); *(ii)* to be low-dimensional − to ensure a fast adaptation during meta-testing; *(iii)* to be geometry-aware rather than task-aware − to generalize across optimal policies. To learn such features, we extend the well known node2vec algorithm (Grover & Leskovec, 2016) to infer graph embeddings capturing temporal dependencies. In other words, state2vec encodes states in low-dimensional embeddings, defining the similarity of states based on the discounted future transitions. Moreover, to ensure off-policy meta-training, we impose that the data used for training is fully exploratory and dependent on any specific task (it is reward agnostic). This allows us to use the same representation without any retraining of the features to solve tasks with varying reward functions. In the meta-testing phase, the agent will need to simply learn a task-aware coefficient vector to derive a value function approximation. The dimensionality of the coefficient vector is imposed by the embedding dimension, which we constraint to be low to favor sample-efficiency in the meta-testing. We show experimentally that state2vec captures with high accuracy the structural geometry of the environment while remaining reward agnostic. The experiments also support the intuition that off-policy state2vec representations are robust low dimensional basis functions that allow to approximate well the value function.

The reminder of this paper is organized as follows: In Section 2 we formalise the meta-RL scenario in which we are interested and present the background material upon which our work is built. Section 3 introduces *state2vec*, our novel successor features approximation algorithm. The experimental set-up and results are presented and discussed in Section 4. We review the relevant literature in Section 5, relating and comparing prior works to our approach to meta-RL. Finally, in Section 6, we summarise our main contributions and highlight some of the relevant directions for future work.

2 BACKGROUND

2.1 META-REINFORCEMENT LEARNING

In RL, a decision maker, or agent, interacts with an environment by selecting actions with the goal to maximise some long term reward. This is typically modelled as a Markov Decision Process (MDP). A discrete MDP is defined as the tuple $M = (S, A, P, R, \gamma)$, where S is a finite set of discrete states, A a finite set of actions, P describes the transition model − with $P(s, a, s')$ giving the probability of visiting s' from state s once action a is taken, R describes the reward function and $\gamma \in (0, 1]$ defines the discount factor. We consider *finite* MDPs, in which the sets of states, actions, and rewards have a finite number of elements. The random variable R and s have well defined discrete probability distributions that only depend on the previous state and action, hence having the Markovian property. A policy π is a mapping from states to probabilities of selecting each action in A. Formally, for a stochastic policy, $\pi(a|s)$ is the probability that the agent takes action a when the agent is in state s. Given a policy π, an action-value function Q^π is a mapping $S \times A \mapsto \mathbb{R}$ that describes the expected long-term discounted sum of rewards observed when the agent is in a given state s, takes action a, and follows policy π thereafter. Solving an MDP requires to find a policy that defines the optimal action-value function Q^*, which satisfies the following constraints:

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s' \in S} P(s, a, s') \max_{a'} Q^*(s', a') \tag{1}$$

This recursive equation is known as *Bellman's optimality equation*. The optimal policy is a unique solution to the Bellman's equation and can be found by dynamic programming, iteratively evaluating the value functions for all states.

In this work, we are interested in the set of MDPs spanned by the tuple (S, A, P, γ):

$$\mathcal{M} = \{M_1, M_2, \dots, M_n\} \tag{2}$$

with each tasks being a MDP problem $M_i = (S, A, P, R_i, \gamma)$, with $R_i : S \times A \mapsto \mathbb{R}$. In other words, we investigate the meta-learning problem in which tasks M_i share the same MDP components, except the reward function, which is thrown from a common random distribution $M_i \sim p(M_i)$. This is of obvious interest as this formalism potentially model real life applications. This is the same setting adopted in prior related works (Barreto et al., 2017; 2018; Borsa et al., 2019; Lehnert et al., 2017).In this setting, we are interested in finding out if there is an efficient way of learning the underlying dynamics that is shared across all tasks, such that once this information is known, solving a specific MDP becomes a much easier problem.

2.2 Successor Representation

In order to address the meta learning problem, we need to decouple the dynamics of the MDP (common across tasks) from the reward function (task discriminant) in the value function approximation. This decoupling motivates the adoption of the successor representation, or SR, (Dayan, 1993). With the SR, we can factor the action-value function into two independent terms:

$$Q^{\pi}(s, a) = \sum_{s'} \Psi^{\pi}(s, a, s') R(s, a), \tag{3}$$

where the SR $\Psi^{\pi}(s, a, s')$ is defined, for $\gamma < 1$, as:

$$\Psi^{\pi}(s, a, s') = \mathbb{E}_{\pi}\Big[\sum_{t=0}^{\infty} \gamma^t \mathbb{I}(s_t = s') | s_0 = s, a_0 = a\Big], \tag{4}$$

where $\mathbb{I}(s_t = s') = 1$ if $s_t = s'$ and 0 otherwise.

The provided interpretation, is that the SR is a predictive type of representation, which represents a state action pair as a feature vector $\mathbf{\Psi}^{\pi}_{s,a}$ such that, under policy π, the representation $\mathbf{\Psi}^{\pi}_{s,a}$ is similar to the feature vector of successors states. Computing the action-value function given the SR is computationally easier as it becomes a simple linear computation. Furthermore, given the SR, the recomputation of the action value function is robust in changes in the reward function: the new action value function can be quickly recomputed using the current SR. The SR is therefore a natural tool to consider for transfer in reinforcement learning.

2.3 Successor Feature

Barreto et al. (2017) proposed a generalisation of the SR called *successor feature* (SF). They make the assumption that the reward function can be parametrised with

$$R(s, a) = \phi(s, a)^{\top} \boldsymbol{w}\,, \tag{5}$$

where $\phi(s, a)$ is a feature vector for (s, a) and $\boldsymbol{w} \in \mathbb{R}^d$ is a vector of weights. Because no assumption is made about $\phi(s, a)$, the reward function could be recovered exactly, hence (5) is not too restrictive. Under this assumption, the action-value function for the task defined by $\boldsymbol{w}$ can be rewritten as:

$$Q^{\pi}(s, a) = \boldsymbol{\psi}^{\boldsymbol{\pi}}(s, a)^{\top} \boldsymbol{w}\,. \tag{6}$$

where the successor feature $\boldsymbol{\psi}^{\pi}(s, a)$ is defined as

$$\boldsymbol{\psi}^{\pi}(s, a) \doteq \mathbb{E}_{\pi}\Big[\sum_{t=0}^{\infty} \gamma^{t-1} \phi_{i+1} | s_t = s, a_t = a\Big]\,. \tag{7}$$

In Barreto et al. (2017), authors also define the *generalized policy improvement* (GPI) theorem, which shows that given previously computed SF approximation $\hat{\boldsymbol{\psi}}_{\pi_{\boldsymbol{w}}}(s, a)$ for some tasks $M_w \in \mathcal{M}$,

the agent can derive a policy π for a new task $M_i \in \mathcal{M}$ which is guaranteed to preform at least as well as any previously learned policy. More specifically, denoting by $\pi_{\boldsymbol{w}}$ the policy followed by the agent in the task $\boldsymbol{w}$, when a new task $\boldsymbol{w}'$ is experienced, the agent will apply the following strategy

$$\pi_{\boldsymbol{w}'} \in \arg\max_a \max_{\boldsymbol{w}} \hat{\boldsymbol{\psi}}(s,a)^\top \boldsymbol{w}\,. \tag{8}$$

In practice, this means that across all the observed tasks, we will consider the best value function when deciding our policy. The main limitation, however, is the double dependency of the value function on $\boldsymbol{w}$, which defines the task as well as the policy. This means that if all previous tasks are significantly different than the new tasked $\boldsymbol{w}'$, the derived policy $\pi_{\boldsymbol{w}'}$ will be far from the optimal policy for task $\boldsymbol{w}'$, and the knowledge of the previous task is not transferable to the new task.

To disentangle these two contributions, Borsa et al. (2019) define a *general value function* and *universal successor features* respectively by

$$Q(s,a;\boldsymbol{w},\pi) \doteq \boldsymbol{\psi}^\pi(s,a)^\top \boldsymbol{w}\,,$$
$$\boldsymbol{\psi}(s,a;\pi)^\top \doteq \boldsymbol{\psi}^\pi(s,a)^\top\,.$$

They further define a *policy embedding*: $e(\pi) : (S \times A \to \pi) \to \mathbb{R}^k$. The choice of embedding $e(\pi_{\boldsymbol{z}}) = \boldsymbol{z}$, leads to the following approximation:

$$\hat{Q}(s,a;\boldsymbol{w},\boldsymbol{z}) = \boldsymbol{\psi}(s,a;\boldsymbol{z})^\top \boldsymbol{w} \tag{9}$$

The key novelty is that the features $\boldsymbol{\psi}$ are now generalized across policies. This leads to the following policy

$$\pi(s) \in \arg\max_a \max_{\boldsymbol{z}\in\mathcal{C}} \boldsymbol{\psi}(s,a;\boldsymbol{z})^\top \boldsymbol{w}'. \tag{10}$$

where if $\mathcal{C}$ is the set of tasks used to approximate the SF. It is clear that $\mathcal{C} = \{\boldsymbol{w}'\}$ recovers a universal value function approximator, minimising the value function approximation error. Conversely, if $\mathcal{C} = \mathcal{M}$, it means that all experienced tasks are considered for the SF evaluation. This reduces to the GPI of (Barreto et al., 2018), which provides a good generalization across tasks, at a price of a less accurate value function approximation.

3 Off-policy Successor Features Approximators

The main limitation of the proposed methodologies is that they proposed a representation that is transferable only across *similar* policies (Lehnert et al., 2017). In this work, we are rather interested in proposing a fixed low-dimensional representation, which (i) generalizes across tasks; (ii) and is policy independent. Specifically, we target to encapsulate the geometry of the MDP by learning a low-dimensional representation of the states, i.e., *state2vec*. This representation captures the geometry of the problem, which is common across any tasks. At each task, the agent needs to learn only the value function approximation as a function of the low-dimensional state2vec. Our work is a special case of (8) and (10) in which the features are derived from the *state* embedding, aimed at capturing the geometry of the problem, which is the common aspect of the MDPs across different tasks. Denoting by $\Psi(s,a)$ our embeddings we have

$$\pi(s) \in \arg\max_a \max_{\boldsymbol{\theta}_{\boldsymbol{w}'}} \underbrace{\left\{\mathbb{E}_{\pi\in\mathcal{M}}\left[\boldsymbol{\psi}(s,a;\boldsymbol{\pi})^\top\right]\right\}}_{\Psi(s,a)} \boldsymbol{\theta}_{\boldsymbol{w}'}. \tag{11}$$

The key difference of our work is that we observe the structure of the problem (captured by the off-policy SF) and learn the best policy under this structure by optimising a weight vector $\boldsymbol{\theta}_{\boldsymbol{w}'}$. In other words, we seek to maximise the following value function approximation:

$$\hat{Q}^{\pi_{\boldsymbol{w}'}}(s,a) = \mathbb{E}_{\pi\in\mathcal{M}}\left[\boldsymbol{\psi}(s,a;\pi)^\top\right]\boldsymbol{\theta}_{\boldsymbol{w}'} = \Psi(s,a)^\top \boldsymbol{\theta}_{\boldsymbol{w}'}\,, \tag{12}$$

where with an abuse of notation, we use $\mathcal{M}$ to define the set of all policies as well as set of all tasks in a fixed environment.

In the following, we define state2vec and describe how we train Ψ off-policy to capture the geometry of the MDP.

3.1 META-TRAINGING : STATE2VEC

It is now clear that the state of the art provides very interesting low-dimensional representation of the MDP problem. However, as discussed, proposed representations still suffer from limited knowledge transferability. Specifically, since the SF are learned while taking decision, it is intrinsically connected to the task.

Therefore, we propose to approximate the SF off-policy and to use the same representation across all tasks in $\mathcal{M}$. We proposed an efficient framework for learning continuous feature representations of states, such that, similarly to the SF, states that are neighbours in time should have similar representation. Our method is directly inspired by Grover & Leskovec (2016)'s *node2vec*, and hence we refer to it as *state2vec*.

State2vec learns state representations based on sample episodes' statistics. It optimises the representations such that states that are successors have similar representation. It does so by first collecting a data set $\mathcal{D}_\pi$ of n walks $L = \{(s_0, a_0), (s_1, a_1) \dots, (s_n, a_n)\}$ by following a sampling strategy π for maximum T steps (terminating earlier if it results in an absorbing goal state). Then, optimise the following objective function:

$$\max_{\Psi} \sum_{L \in \mathcal{D}_\pi} \sum_{(s,a) \in L} \log Pr(N(s,a) | \Psi(s,a)), \tag{13}$$

where $N(s_i, a_i) = \{(s_{i+1}, a_{i+1}), (s_{i+2}, a_{i+2}), \dots, (s_j, a_j), \dots, (s_{i+T}, a_{i+T})\}$ defines the succession of state action pair (s_i, a_i) of size T. Similarly to Grover & Leskovec (2016), we model the conditional likelihood as

$$Pr(N(s,a) | \Psi(s,a)) = \prod_{(s_j, a_j) \in N(s,a)} Pr(s_j, a_j | \Psi(s,a)), \tag{14}$$

Unlike the *node2vec* algorithm, we account for the fact that neighbours that are further in time should be further discounted. We do so by modelling the the likelihood of every source-neighbour pair as a sigmoid weighted by a discount factor:

$$Pr(s_j, a_j | \Psi(s,a)) = \gamma^{|-j|} \sigma(\Psi(s_j, a_j) \cdot \Psi(s,a)) \tag{15}$$

where σ denotes the sigmoid function.

3.2 META-TESTING WITH STATE2VEC

Once the state2vec representation are learned, we can use them for solving any task in $\mathcal{M}$ without needing to do any retraining. The solving of task $M_{\boldsymbol{w}} \in \mathcal{M}$ given the structural representation $\sigma(\Psi$ reduces to optimising the following value function approximation for the weight vector θ_w:

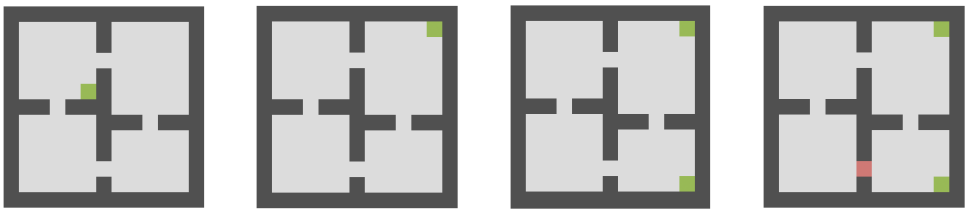
$$\hat{Q}^{\pi_{\boldsymbol{w}}}(s,a) = \Psi(s,a)^\top \boldsymbol{\theta_w} \,. \tag{16}$$

This can be achieve using any parametric RL algorithm, such as fitted Q-learning or LSPI (Riedmiller, 2005; Lagoudakis & Parr, 2004).

4 EXPERIMENTS

4.1 CASE STUDY

We consider the four-room domain (Sutton et al., 1999) shown in Figure 1. It is a two-dimensional space quantized into 169 states, 4 of which are doorways. The agent starts at a random location, and must collect a goal object at a location defined by the task. Depending on the task, the environment also contains "dangerous" zones. The goal object's location is shown in green in Figure 1, while the dangerous states are depicted in red. Collecting an object gives an instantaneous reward of $+100$, and entering a dangerous state gives an instantaneous penalty of -10. The the episode terminates when a goal object is collected or when the agent reaches the maximum of 200 steps.



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

Figure 1: Four-room environment with different configurations.

4.2 RESULTS

4.2.1 META-TRAINING

The meta-training phase is the learning of the state space's geometry by running state2vec. In this feature learning phase, we collect 300 sample walks of length 100 and run state2vec with a window size of 50 and discount factor $\gamma = 0.8$ for varying dimensions d.. Figure 2 visualises the low dimensional (projection onto the first two principal components) representation of the states in the successor representation and in the state2vec feature spaces. As seen in Figure 2, is a close approximation to the exact successor representation. In both cases, we clearly see that the representations have clustered the states within the same room together, while isolating the doorway states. The learned embedding are shown to preserve the geometry of the state space and identify states that have a special structural role (e.g. doorways).

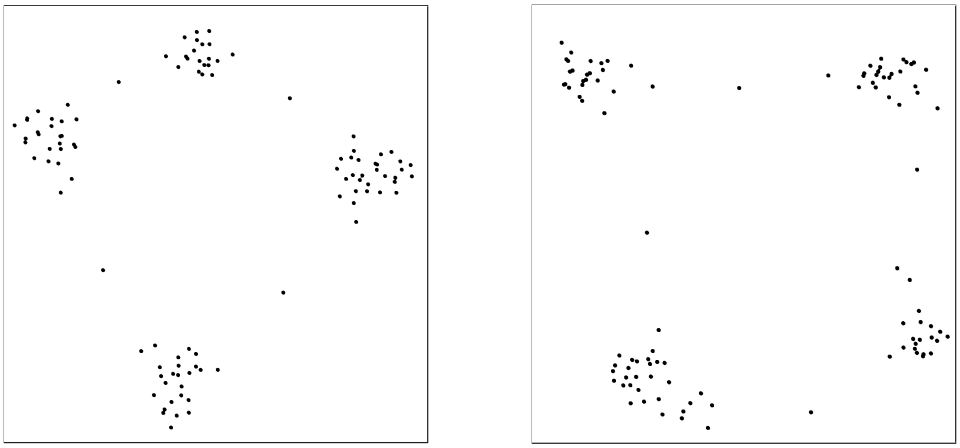


Figure 2: Visualisation of the states representation in feature space (2D PCA projection). **Left**: the exact successor representation, each vector in the original feature space has dimension 169. **Right**: the state2vec approximation of dimension 50 in the embedding space.

4.2.2 META-TESTING

In meta-testing phase, we use the learned state2vec features to learn the optimal policy of each individual. We collect sampled realisations of the form (s, a, s') by simulating 50 episodes of maximum length 200 (terminating earlier if the goal is reach) and run LSPI (Lagoudakis & Parr, 2004) with state2vec representations as basis vectors to learn the weights $\boldsymbol{\theta_w}$ in (16). Figure 4a shows the performance in terms of average cumulative reward for varying value of d. As it can be seen, we are able to achieve strong performance (maximum reward) for all tasks when using the pre-computed state2vec representations of dimensionality 100 with minimal additional exploration per task. Figure 3 depicts the performance when we make the size of the data (number of simulated episodes) used at meta-testing vary. We observe a fast reinforcement learning, with optimal policy learned

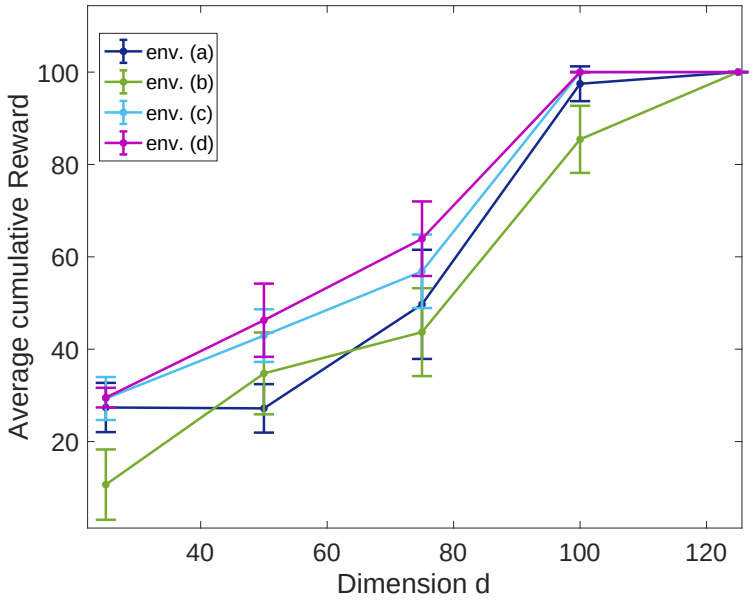
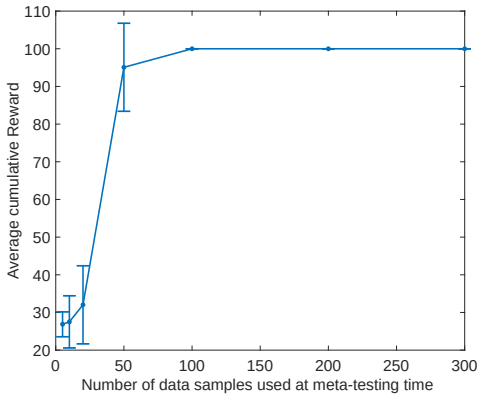
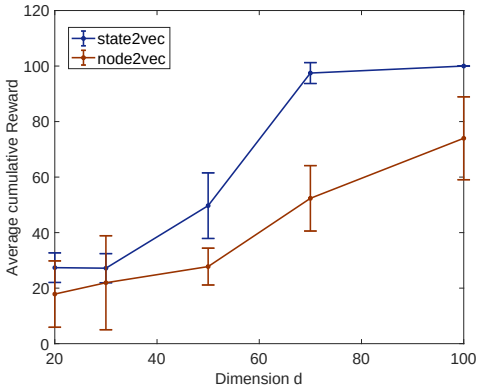


Figure 3: Average cumulative reward after meta-testing using pre-trained state2vec for each of the environment in Figure 1.

with only 50 exploratory episodes collected. These results suggest that information learned during meta-training greatly benefit meta-testing, reducing the need for extensive exploration.



(a) Average cumulative reward after meta-testing using pre-trained state2vec (with $d = 100$) for environment (1a).



(b) Comparison between node2vec and state2vec on environment (1a) (one goal at the corner).

Figure 4

We compare the quality of state2vec embeddings with state of the art low dimensional basis function for linear value function approximation Figure 4b shows an improved performance of state2vec over node2vec in terms of average cumulative reward. We suspect that the gain in performance comes for the fact that state2vec is design for RL, whereas node2vec is a generic graph embedding algorithm. Specifically, in the objective function, the notion of neighborhood in state2vec is such that further states in time are discounted more than the immediate successors.

5 RELATED WORK

The successor feature (SF) as a generalisation of the successor representation (Dayan, 1993) has been introduced in Barreto et al. (2017). Combining the SF with their *generalized policy improvement* algorithm, they showed how information can be transferred, to some extend, across tasks that share the same dynamics but have different reward functions. The initial assumption that rewards can be computed as a linear combination of a set of feature is later relaxed in a follow-up work (Barreto et al., 2018). Since SFs satisfy a Bellman equation, authors adopt TD-learning to learn SFs online. The proposed model generalizes across tasks exploiting the structure of the RL environment. However, the main limitation is that it does not exploit any structural similarity of each single task. Conversely, Ma et al. (2018) introduced the *universal successor representations* (USR) and proposed to model it using a USR approximator. Borsa et al. (2019)'s *universal successor features approximators* (USFAs) exhibits two types of generalisations: one that exploits the structure in the underlying space of value functions and another that exploits the structure of the RL problem.

The above works learn the SFs that better generalize the value function approximator across tasks and/or policy. While sharing the same overarching goal, our work is conceptually different. We target to learn SFs that encode the structure of the environment and we ensure off-policy meta-training. In other words, we aim at learning and leveraging the common structure that is underneath different tasks. The training is performed under off-policy learning. This allows us to decouple the embedding from each single task (either depending on the reward or the optimal policy).

Some previous works proposed other ways of learning sparse representations that capture the geometry of the state space. Mahadevan & Maggioni (2007) introduced *representation policy iteration*, a framework that jointly learns representations and optimal polices. It builds upon spectral graph theory, and relies on a smoothness assumption of the value function over the state graph. As shown by Madjiheurem & Toni (2019), this assumption does not hold when the estimation of the graph is imperfect, leading to poor value function approximation. In their work, Madjiheurem & Toni (2019) adopt *node2vec* as a way of learning the state representations. We have shown that our method state2vec achieves better performance than node2vec. As discussed, the superiority of state2vec for RL probably emerges from the fact that state2vec representations were designed specifically for RL, discounting future occurrences.

6 CONCLUSION

In this work, we focused our effort on the challenging problem of designing RL agents that are able to generalize across tasks that share common dynamics. We considered the meta-reinforcement learning approach, in which the agent, during meta-training, learns a state representation that encodes prior information from the MDP domain, and then leverages this information to solve unseen tasks during meta-testing. With this goal in mind, we proposed state2vec, an efficient and low-complexity framework for learning state representation. We experimentally showed that state2vec is a good approximation of the successor feature. Additionally, we showed that training the state2vec off-policy results in embeddings that capture the geometry of the state space and ensure sample-efficiency during meta-testing. The latter simply needs to estimate a low-dimensional reward-aware vector parameter to learn the optimal value function. While promising, our propose method has the limitation that state representations can only be learned for states encountered at meta-training time. Therefore, if not enough exploration can be allocated to meta-training, it is possible that we explore states during meta-testing which have not been seen during meta-training. Consequently, future work should focus on extending state2vec to a model that can generalise across states (meaning we can compute good approximation of the state2vec vector for an unseen state) for the cases where extensive exploration at meta-training time is not feasible.

We believe the proposed ideas will benefit systems relying on successor representations and will also pave the way for developing meta-reinforcement learning systems that learn representations that are capable of generalising across tasks and are policy-independent, insuring knowledge transferability.

REFERENCES

Itamar Arel, Chuanchang Liu, Tom Urbanik, and Airton G. Kohls. Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems*, 4:128–135(7), June 2010. ISSN 1751-956X. URL https://digital-library.theiet.org/content/journals/10.1049/iet-its.2009.0070.

André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, Hado Van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems*, 2017.

Andre Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Zidek, and Remi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *35th International Conference on Machine Learning, ICML 2018*, 2018. ISBN 9781510867963.

Diana Borsa, Andre Barreto, John Quan, Daniel J. Mankowitz, Hado van Hasselt, Remi Munos, David Silver, and Tom Schaul. Universal successor features approximators. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1VWjiRcKX.

Peter Dayan. Improving Generalization for Temporal Difference Learning: The Successor Representation. *Neural Computation*, 1993. ISSN 0899-7667. doi: 10.1162/neco.1993.5.4.613.

Thore Graepel. AlphaGo - Mastering the game of go with deep neural networks and tree search. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016. ISBN 9783319462264.

Aditya Grover and Jure Leskovec. node2vec. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, pp. 855–864, 2016. ISBN 9781450342322. doi: 10.1145/2939672.2939754. URL http://dl.acm.org/citation.cfm?doid=2939672.2939754.

Michail G. Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 2004. ISSN 15324435. doi: 10.1162/1532443041827907.

Lucas Lehnert, Stefanie Tellex, and Michael L. Littman. Advantages and limitations of using successor features for transfer in reinforcement learning. *CoRR*, abs/1708.00102, 2017. URL http://arxiv.org/abs/1708.00102.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies, 2016. ISSN 15337928.

Chen Ma, Junfeng Wen, and Yoshua Bengio. Universal successor representations for transfer reinforcement learning, 2018. URL https://openreview.net/forum?id=HJ_CpYyDz.

Marlos C. Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption discovery through the deep successor representation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Bk8ZcAxR-.

Sephora Madjiheurem and Laura Toni. Representation Learning on Graphs: A Reinforcement Learning Application. jan 2019. URL http://arxiv.org/abs/1901.05351.

Sridhar Mahadevan and Mauro Maggioni. Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. *Journal of Machine Learning Research*, 2007. ISSN 15324435.

Neville Mehta, Sriraam Natarajan, Prasad Tadepalli, and Alan Fern. Transfer in variable-reward hierarchical reinforcement learning. *Machine Learning*, 73(3):289, Jun 2008. ISSN 1573-0565. doi: 10.1007/s10994-008-5061-y. URL https://doi.org/10.1007/s10994-008-5061-y.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529, feb 2015. URL https://doi.org/10.1038/nature14236http://10.0.4.14/nature14236https://www.nature.com/articles/nature14236{#}supplementary-information.

Martin Riedmiller. Neural fitted q iteration – first experiences with a data efficient neural reinforcement learning method. In *Proceedings of the 16th European Conference on Machine Learning*, ECML'05, pp. 317–328, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3-540-29243-8, 978-3-540-29243-2. doi: 10.1007/11564096_32. URL http://dx.doi.org/10.1007/11564096_32.

Jürgen Schmidhuber, Jieyu Zhao, and Marco Wiering. Simple principles of metalearning. 1996.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George Van Den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 2017. ISSN 14764687. doi: 10.1038/nature24270.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112(1-2):181–211, August 1999. ISSN 0004-3702. doi: 10.1016/S0004-3702(99)00052-1. URL http://dx.doi.org/10.1016/S0004-3702(99)00052-1.