Finding State-Action Similarities in Tabular Reinforcement Learning Using Low-Dimensional Embeddings

Gabriel Rubin
Pontifical Catholic University of Rio Grande do Sul
gdearrud@gmail.com

Abstract

~~This paper proposes a method to automatically detect state-action similarities in temporal difference learning methods.~~ Previous work showed that domain-specific state-action similarity functions can be used to speed up the training process of these learning methods on many domains, such as the popular video-game *Super Mario Bros*. We intend to use our automatic similarity function to speed up an agent on that same domain, without domain-specific information. In this paper, we will briefly detail the technical aspects of our proposed method, the project deadlines and milestones, and why we believe that this automatic approach is capable of performing at least as well as the domain-specific one.

Introduction

Reinforcement learning (Sutton et al. 1998) methods are the state-of-the-art solutions in a variety of domains, from autonomous robots (Riedmiller et al. 2009) to recent developments that enabled agents to master classic video-games (Mnih et al. 2015). Such methods require an agent to learn by training: the agent explores and interacts with an unknown environment for several time-steps in order to learn how to perform an episodic or continuous task. However, this training process can often be expensive resource-wise and time consuming.

Many methods were developed in order to speed up an agent's training process from a designer's perspective, such as using domain-specific knowledge to shape its rewards (Ng, Harada, and Russell 1999) and custom strategies (Ribeiro 1995). Recent work (Rosenfeld, Taylor, and Kraus 2017) introduced a new method called SASS that can speed up this training process significantly for temporal difference methods, such as Q-Learning (Watkins and Dayan 1992), by *spreading* the Q-function estimates of an state to other similar states. SASS considers a custom similarity function that relies on a designer's input in order to compare state-action pairs. However, it is often desirable to ~~autom~~[illegible]e such designer-dependent processes in order to a[illegible]e a generalized solution as opposed to a custom one for each domain.

This paper proposes a novel method for automatically finding state-action similarities by comparing the distance between their low-dimensional embeddings (Nikol'skii 2012). These embeddings will be generated by an autoencoder (Hinton and Salakhutdinov 2006) that can be trained with state-action data from the domain.

In the following sections we will describe the technical aspects of our solution and how we plan to validate our implementation. Then, we will detail the schedule that we will follow in order to complete this project.

Automatic State-Action Similarity Detection Using Embeddings

Reinforcement learning problems and domains are usually represented through a Markov Decision Process (MDP). MDPs are a classical formalization of sequential decision making, where an agent interact with an environment through a set of actions. Taking actions yield immediate and delayed rewards to the agent, and changes the environment configuration (state). This change is expressed as a transition from one state to another. The environment have an action-space A and a state-space S which contain all the possible actions and states respectively. Temporal difference methods, such as *Q-Learning*, learn to estimate the value of taking action a from state s by means of a function called Q-function that receives an action-state pair as input, or (a, s) in this case .

A typical method for finding similarities in data is to compare the distance of elements in the data using some distance metric, such as the Euclidian distance. In reinforcement learning, a state is often represented as a vector of features that describe the environment's current state of affairs. For complex environments, states can have many features and their vector representation will consequently be in a high dimensional space. Working with elements in high dimensions can lead to the *curse-of-dimensionality* (Aggarwal 2005): it is often computationally expensive, hard to visualize, and prune to over-fitting.

Low-dimensional embeddings are lower dimensional representations of high dimensional data that can hold the *semantics* of the original data. Embeddings can be generated by a type of neural network called autoencoder (Hinton and Salakhutdinov 2006). Figure 1 illustrates the structure of an autoencoder: It receives input data through an encoder that encode the incoming data into a low-dimensional embedding (or encoding), then, it decodes this embedding back to its original dimension. An autoencoder is trained by gradually adjusting the network's weights based on the difference (or error) of the input x to the output x'.

This paper's main contribution is to create a similarity function $\sigma : S \times A \times S \times A \mapsto [0, 1]$, where A is the action-space and S is the state-space of a certain domain, that calculates the similarity of state-action pairs by comparing the distance between their embeddings. This function will then be implemented in an SASS agent (Rosenfeld, Taylor, and Kraus 2017) that will utilize it to *spread* Q-function estimates from one state to other similar states, speeding up the training process.

In order to evaluate our similarity function, we will compare its performance to that of the custom made similarity function from previous work on an SASS agent that will be trained on the Super Mario Bros domain. We chose to test our implementation on the Super Mario domain because it is a popular benchmark for reinforcement learning agents (Karakovskiy and Togelius 2012) and it was one of the domains used to test the base SASS agent, which will serve as the baseline for this paper.

Even though comparing state-actions using their embeddings distances may lead to sub-optimal similarity measures, the SASS agent is still capable of yielding superior results compared to that of other temporal difference techniques (Rosenfeld, Taylor, and Kraus 2017). Because of this, we believe that the proposed approach can yield at least comparable speed-up and winning rate when tested next to the custom SASS solution, which would be a great upgrade to SASS, since the proposed approach is automatic and domain-independent.

Project Management

In order to implement the method proposed in this paper, we divided the project in several milestones that will eventually lead to the whole solution, paper and presentation. Timeline 1 shows the dates and milestones considered, where: the first milestone is the Super Mario domain implementation, which should be complete in 10 days. After that, we have 11 days for the autoencoder implementation and training, finishing all this *preparatory* work before November. In the first week of November, we will focus on the similarity method implementation, followed by another week of optimization and fine-tuning. Finally, we reserve the last week of work for writing the paper, getting the results and preparing our presentation.

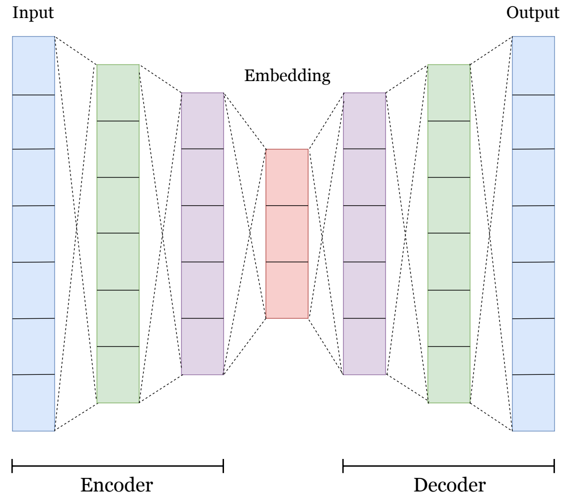

Figure 1: Autoencoder schematic structure.

TIMELINE 1: *Project Milestones*

Oct. 11	Project start
Oct. 21	Super Mario domain implementation
Nov. 1	Autoencoder implementation
Nov. 8	Similarity method implementation
Nov. 19	Optimization and fine-tuning
Nov. 26	Paper and Results
Nov. 29	Presentation

Conclusion

By the end of this project, we hope to confirm our believes that the proposed method is capable of similar or better performance compared to that of the original SASS agent. With a generalized approach to finding state-action similarities, the SASS agent becomes a more viable and interesting option for reinforcement agents, since it would be easier and quicker to implement.

References

Aggarwal, C. C. 2005. On k-anonymity and the curse of dimensionality. In *Proceedings of the 31st international conference on Very large data bases*, 901–909. VLDB Endowment.

Hinton, G. E., and Salakhutdinov, R. R. 2006. Reducing the dimensionality of data with neural networks. *science* 313(5786):504–507.

Karakovskiy, S., and Togelius, J. 2012. The mario ai benchmark and competitions. *IEEE Transactions on Computational Intelligence and AI in Games* 4(1):55–67.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, 278–287.

Nikol'skii, S. M. 2012. *Approximation of functions of several variables and imbedding theorems*, volume 205. Springer Science & Business Media.

Ribeiro, C. H. 1995. Attentional mechanisms as a strategy for generalisation in the q-learning algorithm. In *Proceedings of ICANN*, volume 95, 455–460.

Riedmiller, M.; Gabel, T.; Hafner, R.; and Lange, S. 2009. Reinforcement learning for robot soccer. *Autonomous Robots* 27(1):55–73.

Rosenfeld, A.; Taylor, M. E.; and Kraus, S. 2017. Speeding up tabular reinforcement learning using state-action similarities. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, 1722–1724. International Foundation for Autonomous Agents and Multiagent Systems.

Sutton, R. S.; Barto, A. G.; Bach, F.; et al. 1998. *Reinforcement learning: An introduction*. MIT press.

Watkins, C. J., and Dayan, P. 1992. Q-learning. *Machine learning* 8(3-4):279–292.