Zero Shot Transfer Learning for Robot Soccer

Extended Abstract

Devin Schwab
Carnegie Mellon University
Pittsburgh, PA
dschwab@andrew.cmu.edu

Yifeng Zhu
Carnegie Mellon University
Pittsburgh, PA
yifengz2@andrew.cmu.edu

Manuela Veloso
Carnegie Mellon University
Pittsburgh, PA
mmv@cs.cmu.edu

ABSTRACT

We present a method for doing zero-shot transfer of multi-agent policies as the number of teammates, opponents, and environment size varies. We apply our approach to RoboCup inspired test domains, where it is necessary for policies to adapt to changing numbers of robots due to in-game breakages. We introduce the concept of encoding not only the states as an image, but also the action space as a multi-channel image, which allows the state and action size to remain fixed across team size changes. We also introduce Fully Convolutional Q-Networks, which represent Q-functions in this space using Fully Convolutional Networks. We present results for zero-shot transfer of these policies across team sizes and field sizes, showing that performance remains consistent as both change.

KEYWORDS

Reinforcement Learning; Multiagent learning; Transfer Learning

ACM Reference Format:
Devin Schwab, Yifeng Zhu, and Manuela Veloso. 2018. Zero Shot Transfer Learning for Robot Soccer. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 3 pages.

1 INTRODUCTION AND RELATED WORK

RoboCup soccer is an international competition where teams of researchers compete to create the best team of autonomous soccer playing robots [13]. The game presents a challenging environment to any type of learning agent due to the multiple agents, adversarial nature, noisy states, noisy actions, and sparse reward signals. *Keepaway* is a simplified sub-game of robot soccer that has been used as a test domain for learning approaches [9, 10]. In keepaway, a team of agents attempts to maintain control of the ball for as long as possible, while a team of adversaries attempts to take the ball.

In this work, we introduce a method to learn multi-agent policies with a fixed team size and field size, and then transfer these policies to new team sizes and new field sizes with zero additional training. We apply our approach to a keepaway inspired domain. During a real robot soccer game, the number of robots will vary due to breakages, battery changes, etc. Therefore, any learned policy must be capable of adapting to new team sizes without a lengthy training process. While there are many existing transfer learning approaches, there are not many multi-agent specific transfer approaches. Most approaches are just slight adaptations on single agent transfer techniques including: multi-agent object oriented MDPs, task mappings, experience sharing, and supervision from more experienced agents in similar contexts [1–3, 12]. Unlike these works, we focus not on transferring between different tasks, but across team sizes and field sizes. To our knowledge, this is the first work in this area.

In this rest of this paper, we introduce the following contributions: 1) a method of encoding the state *and* action space as multi-channel images, 2) the Fully Convolutional Q-Network architecture, which represents a Q-function in this encoding using Fully Convolutional Networks (FCN) [5] and 3) experimental results demonstrating consistent performance after zero shot transfer to different team and field sizes.

2 METHODOLOGY

We model the world as a standard Markov Decision Process (MDP) [8]. For RoboCup, we assume that the raw state information contains positions of all detected robots and the ball and any other information necessary to control the robots (e.g. velocities). We also assume that the robots on each team are homogeneous. We apply our method to a grid-world game inspired by keepaway.

2.1 Naïve State-Action Representation

The simplest state representation for a keepaway like domain, is to concatenate the position of the ball, position of the robots, and any other relevant features into a single state vector. We will assume that the action space for each robot is made up of discrete actions, such as “move left”, and groundings of parameterized actions like “pass to robot 1”. Using the state vector and the set of discrete actions, we could apply any off the shelf deep RL algorithm such as DQN [6, 7].

This representation cannot easily transfer across team sizes, as adding/removing robots will change the state-vector size and the number of actions. Policies are represented as a parameterization of the state-action space, so changing the dimensionality of either states or actions will require new parameters to be learned.

2.2 Image Action Space MDP

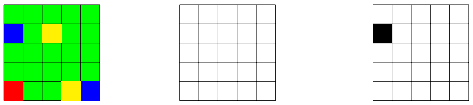
We represent the state as a generated multi-channel image that spans the entire field, with high enough resolution to remain fully-observable. Positions of the robot are marked in this image using different colors to represent teams and which robot has the ball as seen in Figure 1. Adding or removing robots to the field now only changes the colors in the image, not the size.

This research is partially sponsored by DARPA under agreements FA87501620042 and FA87501720152. The views and conclusions contained in this document are those of the authors only.

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden.

Actions are represented as a one-hot, multi-channel image. Each channel corresponds to a discrete action or an action parameterized by a robot. To select a discrete action, in the channel corresponding to the discrete action, the agent will mark its own position. To select a parameterized action, in the channel corresponding to that action, the agent will mark the robot that the action is being applied to. Figure 1 shows an example of what this action might look like for a single discrete action "move" and a single parameterized action "pass to".

The action space has many pixels that can be marked, but only a few are relevant in a given state. Similar to prior work [14], we add an additional channel to the state image that marks all pixels that are valid actions for the given state. The agent must learn to 1) use this extra state information and 2) learn which of the marked pixels is the best choice for the state.



Example State "Move" Channel "Pass to" Channel

Figure 1: (Left) Example of generated state image. Ball holder is red, teammates are blue, opponents are yellow. (Middle, Right) Example action encoding for choosing "Pass to" action with upper left blue teammate as parameter.

2.3 Fully Convolutional Q-Networks

Using a standard DQN style architecture (convolutions followed by fully connected (FC) layers) has two issues: 1) FC layers can only accept a fixed size input 2) flatten for the FC layers throws away spatial information. We instead create Fully Convolutional Q-Networks, by adapting Fully Convolutional Networks (FCN) [5] to represent the Q-function in our new image action space. FCQNs consist only of convolutional layers, so for any size input, a proportionally sized output image will be produced. Therefore an FCQN can be evaluated for any sized field. Each pixel in the output image represents the Q-value of marking that pixel in the one-hot action image. As used in Sukhbaatar et al. [11], we add an averaging layer to our network to help multiple agents coordinate.

2.4 Training and Transferring

States and actions are represented via images as described above and our policy is represented by an FCQN. All agents use the same FCQN layers and train a shared set of weights. Agents can discern their own positions via their masking channels.

Training is performed using the Double DQN [4] training algorithm with the FCQN architecture. An ϵ-greedy policy is used for exploration. After training, to add a new agent, simply execute another copy of the shared FCQN weights. Similarly, to remove an agent, just remove one of the copies of the shared sets of weights.

3 EMPIRICAL RESULTS

We create a grid-world domain inspired by RoboCup Keepaway. A team of agents, one starting with the ball, tries to keep the ball as long as possible. An opposing team, with a fixed policy controlled by the environment, chases after the ball holder, and attempts to intercept passes. The ball is stolen when an opponent is in a cell adjacent to the ball holder. The ball holder can only pass along rows and columns when no opponent is in between the ball holder and the teammate. The rewards during training are: -1 for marking invalid pixel, -1 when the ball is lost, -1 when the ball is captured, 0 otherwise. Episodes are capped at 100 steps. ϵ is decayed from 0.99 to 0.1 over two million steps. We trained with a team size of 3 vs 3 opponents on a field size of 10×10 for 900,000 samples.

Figure 2 shows the results of transferring this policy, with no additional training, across different agent team and opponent team sizes. Each cell represents the average over 100 separate trials with the stated team size combinations on on a 20×20 field. We see that the agents often reach the maximum step count of 100 when there are more teammates than opponents. However, when the teammates are outnumbered, the performance drops because the opponents can corner the ball holder while blocking all passes.

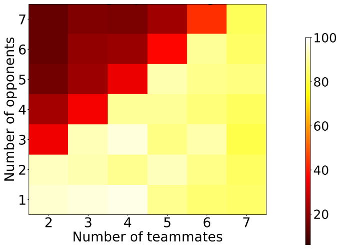


Figure 2: (Left) Transfer of trained policy to different agent team and opponent team sizes. Each cell is the average episode length before ball capture over 100 trials.

Figure 3 shows the results of transferring a policy, with no additional training, across different field sizes. We see that despite large changes in the field size, the policy performance remains consistent, with an average of 84.67 ± 6.52 steps.

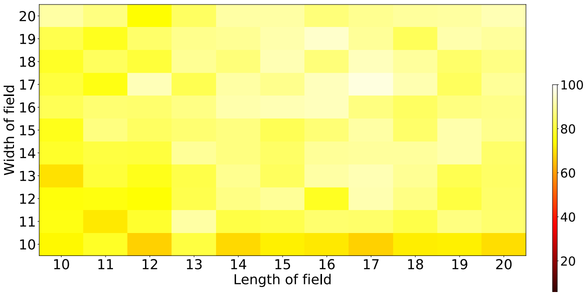


Figure 3: Transfer of a trained policy to different field dimensions. Each cell is average episode length before ball capture over 100 trials.

4 CONCLUSION

We have presented a novel state-action space representation that remains invariant to the number of agents in an environment. We have also presented a novel application of Fully Convolutional Networks to represent Q-functions in this encoding. We have demonstrated on a RoboCup inspired grid keepaway domain that using these techniques, it is possible to do zero shot transfer across team sizes and field sizes, with minimal change in policy performance.

REFERENCES

[1] Georgios Boutsioukis, Ioannis Partalas, and Ioannis Vlahavas. 2011. Transfer learning in multi-agent reinforcement learning domains. In *European Workshop on Reinforcement Learning*. Springer, 249–260.

[2] Felipe Leno Da Silva and Anna Helena Reali Costa. 2016. Transfer Learning for Multiagent Reinforcement Learning Systems.. In *IJCAI*. 3982–3983.

[3] Daniel Garant, Bruno Castro da Silva, Victor Lesser, and Chongjie Zhang. 2015. *Accelerating multi-agent reinforcement learning with dynamic co-learning*. Technical Report.

[4] Hado van Hasselt, Arthur Guez, and David Silver. 2015. Deep Reinforcement Learning With Double Q-Learning. *CoRR* (2015). arXiv:cs.LG/1509.06461 http://arxiv.org/abs/1509.06461v3

[5] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully Convolutional Networks for Semantic Segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[6] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. (2013). arXiv:cs.LG/1312.5602 http://arxiv.org/abs/1312.5602v1

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[8] Martin L Puterman. 2005. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., Hoboken, NJ USA.

[9] Peter Stone, Gregory Kuhlmann, Matthew E. Taylor, and Yaxin Liu. 2006. *Keepaway Soccer: From Machine Learning Testbed to Benchmark*. Springer Science + Business Media, 93–105. https://doi.org/10.1007/11780519_9

[10] Peter Stone, Richard S. Sutton, and Gregory Kuhlmann. 2005. Reinforcement Learning for Robocup Soccer Keepaway. *Adaptive Behavior* 13, 3 (2005), 165–188. https://doi.org/10.1177/105971230501300301

[11] Sainbayar Sukhbaatar, arthur szlam, and Rob Fergus. 2016. Learning Multiagent Communication with Backpropagation. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 2244–2252. http://papers.nips.cc/paper/6398-learning-multiagent-communication-with-backpropagation.pdf

[12] Adam Taylor, Ivana Duparic, Edgar Galván-López, Siobhán Clarke, and Vinny Cahill. 2013. Transfer learning in multi-agent systems through parallel transfer. (2013).

[13] The RoboCup Federation. 2017. RoboCup. (2017). http://www.robocup.org/

[14] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy Lillicrap, Kevin Calderone, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekermo, Jacob Repp, and Rodney Tsing. 2017. Starcraft II: A New Challenge for Reinforcement Learning. *CoRR* (2017). arXiv:cs.LG/1708.04782 http://arxiv.org/abs/1708.04782v1