Lifelong Learning Applications in Mobile Robotics

David Isele
University of Pennsylvania
3330 Walnut St
Philadelphia, PA 19104
isele@seas.upenn.edu

José Marcio Luna
University of Pennsylvania
3330 Walnut St
Philadelphia, PA 19104
joseluna@seas.upenn.edu

Eric Eaton
University of Pennsylvania
3330 Walnut St
Philadelphia, PA 19104
eeaton@seas.upenn.edu

Gabriel de la Cruz
Washington State University
P.O. Box 642752
Pullman, WA 99164
gabriel.delacruz@wsu.edu

James Irwin
Washington State University
P.O. Box 642752
Pullman, WA 99164
james.irwin@wsu.edu

Brandon Kallaher
Washington State University
P.O. Box 642752
Pullman, WA 99164
brandon.kallaher@wsu.edu

Matthew Taylor
Washington State University
P.O. Box 642752
Pullman, WA 99164
taylor@eecs.wsu.edu

ABSTRACT

Learning controllers for heterogeneous multi-agent systems is often an expensive process when controllers for each system are learned individually. Advances in lifelong learning suggest that information between systems can be shared, improving the quality of the controllers that are learned. However these results have been largely theoretical, with applications limited to benchmark problems with known dynamics. We show that these methods can be extended to robotic platforms. Particularly we validate our assumptions for transfer learning between tasks in order to carry out a disurbance rejection problem.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Delphi theory

Keywords

AAMAS proceedings, LaTeX, text tagging

1. INTRODUCTION

Given differences in manufacturing and wear, even identically designed robots may require very different control policies. Learning a unique control policy for each robot in the system can be very costly. One approach to reduce the amount of learning is to share information between robots. Lifelong learning [8] is a promising approach for accomplishing transfer because it allows the different systems to be encountered consecutively rather than requiring all the systems and models be available prior to training. Also it preserves and possibly improves the models encountered early on, as opposed to transfer methods which only optimize for the new system.

Most work in lifelong learning has focused largely on theory, using benchmark simulations to demonstrate their results [8, 3, 4]. The few examples of lifelong learning on robots focus in skill refinement on a single robot [9] [6] rather than sharing information across multiple robots.

The contribution of this work is to present the first results of lifelong learning a on real robots. We apply the PG-ELLA framework [3] to the problem of disturbance rejection as a step towards solving fault tolerant control problems.

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), John Thangarajah, Karl Tuyls, Stacy Marsella, Catholijn Jonker (eds.), May 9–13, 2016, Singapore.*

2. BACKGROUND

As mentioned before, the current work focuses on learning transfer for reinforcement learning tasks. However, it can be easily extended to th esupervised learing case, *e.g.,* regression and classification problems. In this section we cover the mathematical framework that supports our experiments on lifelong learning.

2.1 Reinforcement Learning

In reinforcement Learning (RL) and agent must select sequential actions to maximize its expected return. RL approaches do not require previous knowledge of the system dynamics, instead, the control policies are learned through the interactions with the system. RL problems are typically formalized as Markov Decision Processes (MDPs) with the form $\langle \mathcal{X}, \mathcal{A}, P, R, \gamma \rangle$ where $\mathcal{X} \subset \mathbb{R}^{d_x}$ is the set of states, $\mathcal{A}$ is the set of actions, $P : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to [0, 1]$ is the state transition probability describing the systems dynamics. $R : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is the reward function and $\gamma \in [0, 1)$ is the reward discount factor. At each time step h, the agent is in the state $\mathbf{x}_h \in \mathcal{X}$ and must choose an action $\mathbf{a}_h \in \mathcal{A}$ so that it transitions to a new state $\mathbf{x}_{h+1}$ with state transition probability $P(\mathbf{x}_{h+1}|\mathbf{x}_h, \mathbf{a}_h)$, thus yielding a reward r_h according to R. The action is selected according to a policy

$\pi : \mathcal{X} \times \mathcal{A} \rightarrow [0,1]$ which specifies a probability distribution over actions given the current state. The goal of RL is to find the optimal policy π^* that maximizes the expectation of the per-step reward.

In Policy Gradient (PG) methods [], the policy is represented as a function defined over a parameter vector $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$ that provides the gains of the control inputs of the system. The goal of PG is to optimize the expected average return,

$$\mathcal{J}(\boldsymbol{\theta}) = \int_{\mathbb{T}} p_{\boldsymbol{\theta}}(\tau)\mathcal{R}(\tau)\mathrm{d}\tau, \qquad (1)$$

where $\mathbb{T}$ is the set of all trajectories and $\mathcal{R}(\tau)$ is the average per-step reward, specifically,

$$\begin{aligned} p_{\boldsymbol{\theta}} &= \prod_{h=0}^{H} p(\mathbf{x}_{h+1}|\mathbf{x}_h, \mathbf{a}_h)\pi(\mathbf{a}_h, \mathbf{x}_h) \\ \mathcal{R}(\tau) &= \frac{1}{H}\sum_{h=0}^{H} r(\mathbf{s}_h, \mathbf{a}_h, \mathbf{s}_{h+1}) \end{aligned}$$

The optimization of the policy is carried out by comparing trajectories generated by a policy $\pi_{\boldsymbol{\theta}}$ with those generated by a new policy $\pi_{\tilde{\boldsymbol{\theta}}}$. For more details see [].

2.2 Finite Differences and Policy Search

The local optimization around existing policies π parameterized by a parameter vector $\boldsymbol{\theta}$ is carried out by computing changes in the policy parameters $\boldsymbol{\Delta\theta}$ that will increase the expected reward, thus producing the iterative update,

$$\boldsymbol{\theta}_{m+1} = \boldsymbol{\theta}_m + \boldsymbol{\Delta\theta}_m.$$

More methods to be added here taken from the survey paper.

Gradient-based methods for policy updates follow the gradient of the expected return $\mathcal{J}$ check the notation for the return and the difference between return and reward given a step-size α in case we talk about the minimizer of the KL divergence which is alpha as well,

$$\boldsymbol{\theta}_{m+1} = \boldsymbol{\theta}_m + \alpha\nabla_{\boldsymbol{\theta}}\mathcal{J}.$$

In finite difference gradients, we have a set of k are we using k for anything else? perturbed policy parameters are assessed to estimate the gradient as follows,

$$\Delta\hat{\mathcal{J}}_{\mathbf{p}} \approx \mathcal{J}(\boldsymbol{\theta}_m + \boldsymbol{\Delta\theta}_{\boldsymbol{p}}) - \mathcal{J}_{ref},$$

where $\mathbf{p} = [1, \ldots, k]$ are the individual perturbations, $\Delta\hat{\mathcal{J}}_{\mathbf{p}}$ is the estimate of their effect on the return, and the $\mathcal{J}_{ref}$ is a reference return which is usually taken as the return of the unperturbed parameters. By using linear regression we get an approximation of the gradient,

$$\nabla_{\boldsymbol{\theta}}\mathcal{J} \approx \left(\boldsymbol{\Delta\Theta}^{\boldsymbol{T}}\boldsymbol{\Delta\Theta}\right)^{-1}\boldsymbol{\Delta\Theta}^{T}\Delta\hat{\boldsymbol{J}}_{\boldsymbol{p}}.$$

where $\Delta\hat{\boldsymbol{J}}_{\boldsymbol{p}}$ contains all the stacked samples of $\Delta\hat{\mathcal{J}}_{\mathbf{p}}$ and $\boldsymbol{\Delta\Theta}$ contains the ones of $\boldsymbol{\Delta\theta}_{\boldsymbol{p}}$. This approach is sensitive to the type and magnitude of the perturbations, as well as to the step size α. Since the number of perturbations needs to be as large as the number of parameters, this method is considered to be noisy and inefficient for problems with large sets of parameters.

2.3 Lifelong Learning

Lifelong learning focuses on learning a set of tasks consecutively while performing well across all tasks. Given a round $t = 1, \ldots, t_{\max}$ a task $T^{(t)}$ is observed. This task is defined acccording to the learning problem in place, *e.g.,* regression and classification. We assume that the model associated to $T^{(t)}$ is parameterized by a parameter vector $\boldsymbol{\theta}^{(t)} \in \mathbb{R}^{d_\theta}$. The ideal goal is that prior knowledge about tasks $T^{(1)}, \ldots, T^{(t-1)}$ should provide enough information so that the lifelong learning algorithm performs better and faster on $T^{(t)}$ while being able to scale as the number of tasks increases.

Based on [Kumar,Ruvolo], we assume there is a shared basis $\boldsymbol{L} \in \mathbb{R}^{d_x \times l}$ I changed notation for hidden layers and a sparse weight vector $\boldsymbol{s}^{(t)} \in \mathbb{R}$, so that the parameter $\boldsymbol{\theta}^{(t)}$ is given by,

$$\boldsymbol{\theta}^{(t)} = \boldsymbol{L}\boldsymbol{s}^{(t)}.$$

Using the return function for PG in (1) we propose the following multitask objective function,

$$\underset{\boldsymbol{L},\boldsymbol{S}}{\operatorname{argmin}} \frac{1}{|\boldsymbol{T}|}\sum_{t}\left[-\mathcal{J}(\boldsymbol{\theta}^{(t)}) + \lambda\|\boldsymbol{s}^{(t)}\|_1\right] + \mu\|\boldsymbol{L}\|_F^2,$$

where $\boldsymbol{S}$ is the set of the sparse vectors $\boldsymbol{s}^{(t)}$, the L1 norm of $\|\boldsymbol{s}^{(t)}\|_1$ provides sparse code for $\boldsymbol{s}^{(t)}$ and the Frobenius norm in $|\boldsymbol{L}\|_F^2$ provides regularization. The coefficients μ and $\lambda \in \mathbb{R}$ are weights for the regularization and sparsity respectively.

The learning objective function is approximated by a second order Taylor expansion around the single task policy, and the optimization problem is solved by using the online ELLA algorithm introduced in [8] and extended for the reinforcement learning setting using Policy Gradients (PG) in [3]. However, PG relies on the indirect maximization of the return function based on the maximization of a lower bound. Although this bound has reportedly been tight enough to guarantee maximization of the return function, we chose to use finite differences to search the optimal policy to make sure we maximize the return function directly. We could summarize the vectorized update rules of PG-ELLA as in the zero transfer paper.

3. EXPERIMENTS

This definitely needs more work. Some printscreen of gazeo an a picture of the turtlebot would be good. Some technical description of the robot would complement the ideas here as well.

We evaluated our approach by modelling the control policies for different turtlebot systems. Turtlebots are an afforable robotics platform that uses the robot operating system ROS and includes a simulator in Gazebo. We artifically create a collection of different turtlebots by applying a constant disturbance to the control signal. Each turtlebot has a different constant disturbance drawn uniformly from $[-.1, .1]$. The turtlebot has a $\mathbb{R}^4$ state and an $\mathbb{R}^2$ action following the model of Aicardi et al. [1] which describes the state as: I can verify this model and possibly try to get a better illustrative image.

Figure 1: Turtlebot in Gazebo Simulator.

$$\boldsymbol{x} = \begin{bmatrix} rcos(\alpha) \\ \alpha \\ \frac{cos(\alpha)sin(\alpha)}{\alpha}(\alpha+\kappa+\phi) \\ 1 \end{bmatrix} \quad (2)$$

it's important to make sure that we aren't reusing notation. where r is the distance to the goal, α is the difference between heading angle and the robot and the goal, ψ is the angle of the robot with respect to the global coordinate frame and κ is the desired angle.

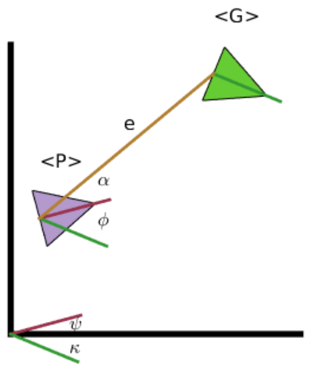

Figure 2: Problem for go to goal task.

The state is a non-linear transformation of the position and heading angle (?). We were not able to learn a linear policy on this non-linear state transformation using either the natural actor critic [7] or episodic REINFORCE [10] policy gradient methods.

3.1 Methodology

We start by generating 20 robots, each with a different constant disturbance and a unique goal, both selected uniformly. The experiments were run for 50 time steps. 15 rollouts were generated for each learning iteration and θ^* was taken from the single task learner policy after 20 iterations. This is fewer iterations than is required for any system to converge to a good controller.

We learned a dictionary with $k = 8$, and compare the learning time of a single task finite difference policy versus the learning time of a system using ELLA.

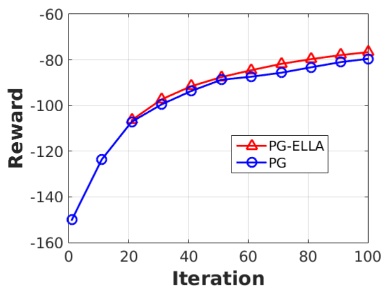

Figure 3: Learning curves for finite differences and finite differences using ELLA to transfer information between tasks.

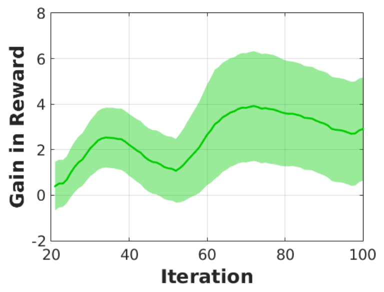

Figure 4: The positive transfer achieved by using ELLA.

4. CONCLUSIONS

there aren't conclusions yet. This paragraph will end the body of this sample document. Remember that you might still have Acknowledgments or Appendices; brief samples of these follow. There is still the Bibliography to deal with; and we will make a disclaimer about that here: with the exception of the reference to the LaTeX book, the citations in this paper are to articles which have nothing to do with the present subject and are used as examples only.

Acknowledgments

This section is optional; it is a location for you to acknowledge grants, funding, editing assistance and what have you. In the present case, for example, the authors would like to thank Gerald Murray of ACM for his help in codifying this *Author's Guide* and the **.cls** and **.tex** files that it describes.

APPENDIX

A. HEADINGS IN APPENDICES

The rules about hierarchical headings discussed above for the body of the article are different in the appendices. In the **appendix** environment, the command **section** is used to indicate the start of each Appendix, with alphabetic order designation (i.e. the first is A, the second B, etc.) and a title (if you include one). So, if you need hierarchical structure *within* an Appendix, start with **subsection** as the highest level. Here is an outline of the body of this document in Appendix-appropriate form:

A.1 Introduction

A.2 The Body of the Paper

A.2.1 Type Changes and Special Characters

A.2.2 Math Equations

Inline (In-text) Equations.

Display Equations.

A.2.3 Citations

A.2.4 Tables

A.2.5 Figures

A.2.6 Theorem-like Constructs

A Caveat for the TEX Expert

A.3 Conclusions

A.4 Acknowledgments

A.5 Additional Authors

Reference section requires a lot more work.

B. ADDITIONAL AUTHORS

REFERENCES

[1] M. Aicardi, G. Casalino, A. Balestrino, and A. Bicchi. Closed loop smooth steering of unicycle-like vehicles. In *Decision and Control, 1994., Proceedings of the 33rd IEEE Conference on*, volume 3, pages 2455–2458. IEEE, 1994.

[2] S. Barrett, M. E. Taylor, and P. Stone. Transfer learning for reinforcement learning on a physical robot. In *Ninth International Conference on Autonomous Agents and Multiagent Systems-Adaptive Learning Agents Workshop (AAMAS-ALA)*. Citeseer, 2010.

[3] H. Bou Ammar, E. Eaton, and P. Ruvolo. Online multi-task learning for policy gradient methods. *International Conference on Machine Learning*, 2014.

[4] H. Bou Ammar, E. Eaton, P. Ruvolo, and M. E. Taylor. Unsupervised cross-domain transfer in policy gradient reinforcement learning via manifold alignment. *International Joint Conference on Artificial Intelligence*, 2015.

[5] S. K. Chalup, C. L. Murch, and M. J. Quinlan. Machine learning with aibo robots in the four-legged league of robocup. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(3):297–310, 2007.

[6] A. Kleiner, M. Dietl, and B. Nebel. Towards a life-long learning soccer agent. In *RoboCup 2002: Robot Soccer World Cup VI*, pages 126–134. Springer, 2002.

[7] J. Peters and S. Schaal. Natural actor-critic. *Neurocomputing*, 71(7):1180–1190, 2008.

[8] P. Ruvolo and E. Eaton. ELLA: An efficient lifelong learning algorithm. *International Conference on Machine Learning*, 28:507–515, 2013.

[9] S. Thrun and T. M. Mitchell. *Lifelong robot learning*. Springer, 1995.

[10] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.