IN2349 ADLR: Project Ideas

April 30, 2020

Here you can find a number of ideas for projects we collected. Take this as an inspiration for your own project. Some of the ideas are rather "big", meaning they could result in more than one project. After you have registered your team (including a draft proposal) you will discuss the extent of your final proposal with your assigned tutor.

1 Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience

In [Chebotar et al., 2018] an efficient method for solving the sim2real problem by iteratively adapting the simulation parameters to the real system is proposed.

- Change the experimental setup to a sim2sim setting because no real robot is available (i.e., try to adapt one simulation to a given one with unknown parameters).
- Investigate the convergence of estimated parameters.
- Investigate the influence of a broader starting distribution on the final performance.
- Investigate how re-using learned policies from previous iterations for initialization of the new network shortens training time and restrains the final performance.

2 Comparing Robustness of Learned (Quadruped Walking) Policies from Different Learning Algorithms

[Haarnoja et al., 2018b] claim that RL algorithms with stochastic policies such as SAC are able to express a much more diverse behavior compared to deterministic algorithms like DDPG. As a consequence, stochastic policies should be far more robust when deployed in a perturbed environment.

- Investigate how stochastic policies (SAC, PPO) perform in novel environments when compared to deterministic algorithms like DDPG.
- Investigate the influence of different distributions used for SAC on the performance in novel environments. Early version of SAC employ GMMs which are discarded in favor of a single Gaussian in later papers.
- Investigate the methods for a quadruped with more complex powertrain dynamics than the one used by [Haarnoja et al., 2018b].

3 Investigate Reinforcement Learning with Augmented Networks

In recent years, different methods to tackle problems in robotics have emerged. Especially for POMDPs or complex tasks that employ domain randomization for the sim2real transfer, the incorporation of LSTM cells into the network architecture leads to impressive results as shown by [OpenAI et al., 2019]. One downside to using recurrent architecture is the increased training complexity. Other methods like observation stacking or explicit physical parameter estimation using supervised learning are far easier to implement and train.

- [OpenAI et al., 2019] claim that memory augmented models exhibit 'meta-learning' behavior at test time, meaning that the policy trials the environment to adapt parameters before starting to work on the desired task itself. Investigate these claims in an environment inspired by a physical system.
- Investigate how LSTM cells, observation stacking and explicit parameter estimation perform in the same partially observable physics simulation when simple domain randomization is employed. Some task that require precise knowledge of the domain parameterization are particularly interesting

4 Generalized Hindsight for Reinforcement Learning

Generalized Hindsight for Reinforcement Learning presented by [Li et al., 2020] combines Multi-Task Deep Reinforcement Learning with Sample-Efficient Learning.

- Design a physically inspired and meaningful task that can be used for Multi-Task learning, e.g. a more complex version of the ant environment used by [Li et al., 2020]. This simulation can be used to learn different gaits, pronking as well as yawrate control at different speeds.

5 Comparing different methods for uncertainty estimation

Interesting methods include (but are not limited to) MC-Dropoout [Gal and Ghahramani, 2016] and Normalizing Flows [Louizos and Welling, 2017]. Thsese methods could be compared in vastly different settings.

- Investigate how the uncertainty estimation changes during the training process (relevant to RL since we generally don't update the networks until convergence before collecting more data).
- Investigate which methods are best suited for active learning in the framework proposed by [Gal et al., 2017].
- Investigate which methods perform best for DQNs in simple environments similar to the work by [Touati et al., 2018].
- Come up with your own ideas.

6 Curiosity-Driven Learning

A good starting point for projects in this area is the paper by *Large-Scale Study of Curiosity-Driven Learning* [Burda et al., 2018]. Individual projects could first reporduce some of the experiments and then:

- Compare with other curiosity "measures" (count based, predict next state, predict past action, random network distillation, reachability, goal based, and/or your own ideas).
- Extend the authors comparison to other sets of environments. One key question is whether or not there are certain characteristics that a task needs to fulfill in order to profit from curiosity-driven learning.

7 Geometric Representations in Reinforcement Learning

Note: Requires previous experience with GNNs [Kipf and Welling, 2016].

- Modify PyBullet environments (Hopper, Walker, HalfCheetah, Ant) such that the observations contain a graph representing the robot. Use message passing network(s) in addition or instead of the MLP for value/Q function and policy in standard algorithms like PPO [Schulman et al., 2017] or SAC [Haarnoja et al., 2018a].

8 AlphaZero Implementation & Tweaks

Deep dive into the AlphaGo/AlphaZero algorithm [Silver et al., 2016, Silver et al., 2017, Silver et al., 2017]. Pick a very small game for feasible computational & time cost and quick experimentation. Either implement from scratch or maybe get very familiar with an existing implementation. Then try variations or possible algorithm improvements.

Modification ideas:

- Refined handling of explore/exploit tradeoff in MCTS by adding Uncertainty Estimation (see section 5) in value & prior networks for better upper confidence bounds in PUCT. Can this speed up discovery of good strategies?
- Comparison to model-free RL: Does AlphaZero learn better/worse? Does this depend on environment (game) characteristics? What metrics should be compared here?

9 Make Reinforcement Learning Safe Again

Look at different policy gradient algorithms for a 2D robot badminton example and compare risk-averse vs. risk-seeking gradient directions by examining the reward statistics as in *Entropic Risk Measure in Policy Search* [Nass et al., 2019].

- Modify the environment to include (self-)collision.
- What other options are there to measure/control the safety?

10 Learning the Inverse Kinematic

Look at the possibilities for representing inverse problems with neural networks. *Analyzing Inverse Problems with Invertible Neural Networks* [Ardizzone et al., 2018] compare different flavors of GANs, VAEs and INN(theirs) for Inverse Problems in general. Extend their simple robotic example of a planner arm to 3D / more DoFs / multiple TCPs.

- What is the best approach to represent the high dimensional nullspaces existing for complex robot geometries?
- Predict not only the position of the TCP but also the rotation. Connection to continuous rotation representation [Zhou et al., 2018].
- How to measure the performance if the real nullspace is not known?

11 Harnessing Reinforcement Learning for Neural Motion Planning

Motion planning for a planar robotic arm from a start configuration to a cartesian goal position. Comparison between DDPG, DDPG+HER, and DDPG-MP(theirs) [Jurgenson and Tamar, 2019]. They use RRT* to generate expert knowledge in difficult cases, where random exploration does not find a feasible solution.

- Modify the code and try it for different robots and environments.
- They state that supervised learning is inferior to RL for this problem because of the insufficient data on the boundary of the obstacles. Is it possible to achieve similar results by tweaking the distribution of the supervised examples?

References

[Ardizzone et al., 2018] Ardizzone, L., Kruse, J., J.Wirkert, S., Rahner, D., W.Pellegrini, E., S.Klessen, R., Maier-Hein, L., Rother, C., and Köthe, U. (2018). Analyzing inverse problems with invertible neural networks. *CoRR*.

[Burda et al., 2018] Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. (2018). Large-scale study of curiosity-driven learning.

[Chebotar et al., 2018] Chebotar, Y., Handa, A., Makoviychuk, V., Macklin, M., Issac, J., Ratliff, N. D., and Fox, D. (2018). Closing the sim-to-real loop: Adapting simulation randomization with real world experience. *CoRR*, abs/1810.05687.

[Gal and Ghahramani, 2016] Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proc. Int. Conference on Machine Learning*.

[Gal et al., 2017] Gal, Y., Islam, R., and Ghahramani, Z. (2017). Deep bayesian active learning with image data.

[Haarnoja et al., 2018a] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018a). Soft actor-critic: Off-policy maximum entropy deep reinforce- ment learning with a stochastic actor. In *International Conference on Machine Learning.*

[Haarnoja et al., 2018b] Haarnoja, T., Zhou, A., Ha, S., Tan, J., Tucker, G., and Levine, S. (2018b). Learning to walk via deep reinforcement learning.

[Jurgenson and Tamar, 2019] Jurgenson, T. and Tamar, A. (2019). Harnessing reinforcement learning for neural motion planning. In *Proc. Int. Conf. on Robotics: Science and Systems.*

[Kipf and Welling, 2016] Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks.

[Li et al., 2020] Li, A., Pinto, L., and Abbeel, P. (2020). Generalized hindsight for reinforcement learning.

[Louizos and Welling, 2017] Louizos, C. and Welling, M. (2017). Multiplicative normalizing flows for variational bayesian neural networks.

[Nass et al., 2019] Nass, D., Belousov, B., and Peters, J. (2019). Entropic risk measure in policy search. *CoRR.*

[OpenAI et al., 2019] OpenAI, Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., Schneider, J., Tezak, N., Tworek, J., Welinder, P., Weng, L., Yuan, Q., Zaremba, W., and Zhang, L. (2019). Solving rubik's cube with a robot hand.

[Schulman et al., 2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

[Silver et al., 2016] Silver, D., Huang, A., Maddison, C. J., Guez, A., and et.al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529.

[Silver et al., 2017] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. (2017). Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *ArXiv e-prints.*

[Silver et al., 2017] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359.

[Touati et al., 2018] Touati, A., Satija, H., Romoff, J., Pineau, J., and Vincent, P. (2018). Randomized value functions via multiplicative normalizing flows.

[Zhou et al., 2018] Zhou, Y., Barnes, C., Lu, J., Yang, J., and Li, H. (2018). On the continuity of rotation representations in neural networks. *CoRR.*