IN2349 ADLR: Project Ideas

April 15, 2021

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 Comparing Methods for Uncertainty Estimation

Interesting methods include (but are not limited to) MC-Dropoout [Gal and Ghahramani, 2016], Bootstrapping [Osband et al., 2018], 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.

2 Offline Datasets for Reinforcement Learning

Offline (or Batch) RL has recently gained more and more attention (cf. `https://bair.berkeley.edu/blog/2019/12/05/bear`, [Agarwal et al., 2019], [Nair et al., 2020]).

- Compare differnt Batch RL algorithms.
- Test new environments.
- Benchmark against online algorithms.

3 Geometric Representations in Reinforcement Learning

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

- Similar to [Wang et al., 2018]. 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., 2018].

4 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.

5 Meta Reinforcement Learning for Sim-to-real Domain Adaptation

[Arndt et al., 2019] show that the sim2real problem can be tackled by applying Meta RL to learned low dimensional projections of the action space.

- Implement the algorithm
- 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 how this framework can be applied or extended to classic closed-loop robot control

6 Solving Complex Sparse Reinforcement Learning Tasks

When defining Reinforcement Learning Tasks for robots, it is often desirable to stick to sparse rewards in order to avoid reward shaping. Not only does is it ease the setup since a task can be solely defined by its final desired outcome, but the optimal policy can be found "outside of the box" without a human prior given thru the reward function. Unfortunately, in big state spaces random exploration is not able to find these sparse success signals. Therefore, [Riedmiller et al., 2018] introduce SAC-X.

- Implement the algorithm
- Investigate how the presented method can be used for finer and more dexterous object manipulation e.g. with a hand.

7 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 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?

8 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?

9 Learning to Optimize Motion Planning

Explore the ideas proposed by "Learning to Optimize" [Li and Malik, 2016] in the context of optimization based motion planning [Zucker et al., 2013]. Can RL guide an optimzer to speed up robotic path planning?

- Set up an optimizer for a simple robot (with help from the tutor)
- Test ideas to guide the optimizer with RL
- What are advantages of this hybrid approach over using RL directly on motion planning

10 Exploring Munchausen Reinforcement Learning

Recently, [Vieillard et al., 2020] proposed an appealingly simple, yet surprisingly effective extension to DQN; using the policy for bootstrapping. Exploring the implications of this idea, projects could for example address a (sub) set of the following problems:

- Extend the idea to continuous action spaces (e.g. by augmenting SAC).

- Apply Munchausen RL to robotic tasks and see what it brings to the table. Does it improve the baselines? If yes, under what circumstances? If no, investigate the causes.
- Come up with adaptive strategies to deal with the additional hyperparameters introduced by [Vieillard et al., 2020].

11 Constraint Curricula for Robotic Manipulation Tasks

Consider the setting in which a robotic manipulator is tasked to achieve a goal (e.g. move end effector to a certain position in the workspace), subject to external or internal constraints (like avoiding certain configurations). In RL, this can be formulated as a constrained optimization problem [Achiam and Amodei, 2019]. While converging to a constraint-satisfying policy is often hard enough, ensuring constraints *throughout training* is even more challenging. For a given set of tasks, one approach could be to build an adaptive task curriculum by successively increasing the complexity of tasks and/or constraints during training.

- Come up with a robotic setting with one or multiple constraints. Implement the environment from scratch or modify existing environments to suit your needs.
- Implement or modify a safety-aware RL algorithm for your setup.
- Experiment with ways of adaptively building curricula of increasingly difficult tasks/constraints to minimize constraint-violation during and/or after training.

References

[Achiam and Amodei, 2019] Achiam, J. and Amodei, D. (2019). Benchmarking safe exploration in deep reinforcement learning.

[Agarwal et al., 2019] Agarwal, R., Schuurmans, D., and Norouzi, M. (2019). Striving for simplicity in off-policy deep reinforcement learning. *CoRR*, abs/1907.04543.

[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*.

[Arndt et al., 2019] Arndt, K., Hazara, M., Ghadirzadeh, A., and Kyrki, V. (2019). Meta reinforcement learning for sim-to-real domain adaptation. *CoRR*, abs/1909.12906.

[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., 2018] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforce- ment learning with a stochastic actor. In *International Conference on Machine 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 and Malik, 2016] Li, K. and Malik, J. (2016). Learning to optimize. *CoRR*, abs/1606.01885.

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

[Nair et al., 2020] Nair, A., Dalal, M., Gupta, A., and Levine, S. (2020). Accelerating online reinforcement learning with offline datasets.

[Osband et al., 2018] Osband, I., Aslanides, J., and Cassirer, A. (2018). Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*.

[Riedmiller et al., 2018] Riedmiller, M. A., Hafner, R., Lampe, T., Neunert, M., Degrave, J., de Wiele, T. V., Mnih, V., Heess, N., and Springenberg, J. T. (2018). Learning by playing - solving sparse reward tasks from scratch. *CoRR*, abs/1802.10567.

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

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

[Vieillard et al., 2020] Vieillard, N., Pietquin, O., and Geist, M. (2020). Munchausen reinforcement learning. *arXiv preprint arXiv:2007.14430*.

[Wang et al., 2018] Wang, T., Liao, R., Ba, J., and Fidler, S. (2018). Nervenet: Learning structured policy with graph neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

[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*.

[Zucker et al., 2013] Zucker, M., Ratliff, N., Dragan, A. D., Pivtoraiko, M., Klingensmith, M., Dellin, C. M., Bagnell, J. A., and Srinivasa, S. S. (2013). Chomp: Covariant hamiltonian optimization for motion planning. *International Journal of Robotics Research*, 32(9-10):1164–1193.