Report on HW №1 for «Research in LLM»

Вадим Мельников

19 февраля 2026 г.

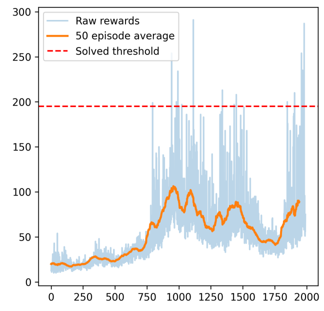
1 Reinforcement Learning

If not stated otherwise, training lasts for 2000 total epochs. After some consideration, AdamW with constant lr=3e-3 was used for optimizing the policy network.

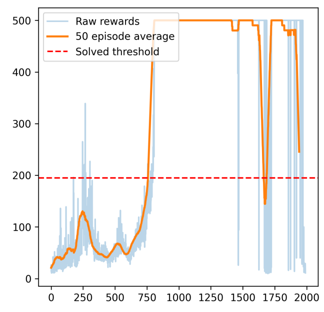
1.1 Vanilla Approach

Vanilla approach yields **very** unstable training results. Below are some statistics gathered after five runs with identical initial parameters, as well as rewards plot during training. Stats correspond to mean and std of Cartpole run lengths, calculated on 100 runs total; maximum possible mean is 500 due to limit on run length. Cartpole is considered (by some) solved, if it consistently scores 195+. The policy network for evaluating mean and std was chosen as the best performing based on running mean on a sliding window of size 50.

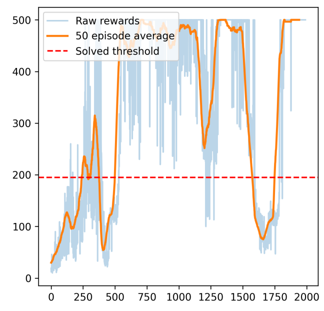
- mean=226.1300, std=148.3487



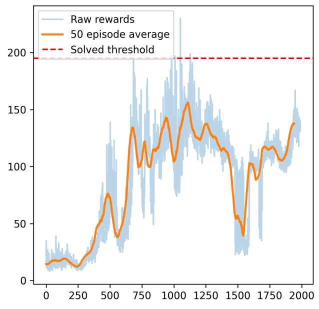
- mean=500.0000, std=0.0000



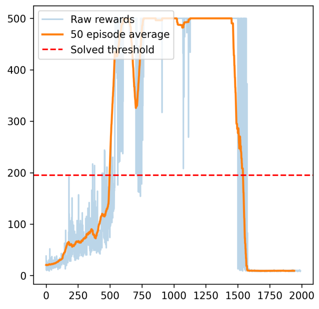
- mean=500.0000, std=0.0000



- mean=148.6000, std=41.7744



- mean=500.0000, std=0.0000



It is clear that the achieved training rewards curves are very unstable, often diverging when reaching perfect scores.

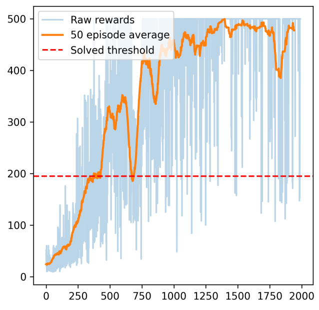
The policy training script is stored in `hw1_vanilla.py`.

Performing policy optimization with all of the following approaches leads to perfect scores during testing, therefore it isn't stated explicitly in each experiment.

1.2 Vanilla w/ baseline

1.2.1 Mean returns as baseline

Using a baseline as simple as a running mean of returns yields much more stable results. It is clear that variance of the curve indeed decreases.

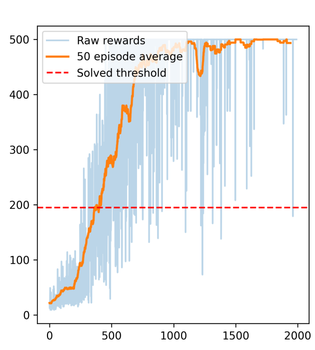


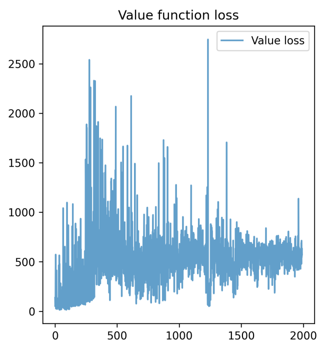
While the rewards curve is still quite erratic, its running mean after convergence remains stable.

The policy training script is stored in `hw1_baseline.py`.

1.2.2 Value function as baseline

Using a trainable value function as a baseline yields more stable rewards during training. It is still true that the model consistently converges to perfect Cartpole scores, but does it both faster and more stable than the previous approach. Curve variance decreases further in relation to the mean returns approach.

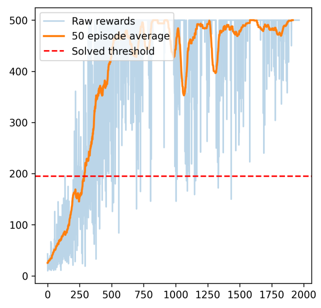




The policy training script is stored in `hw1_baseline_value.py`.

1.2.3 RLOO-style baseline

Despite leave-one-out being one of the simplest ways to implement a baseline, it yielded the fastest convergence of all other approaches, even though it wasn't as stable as learnable value function approach.



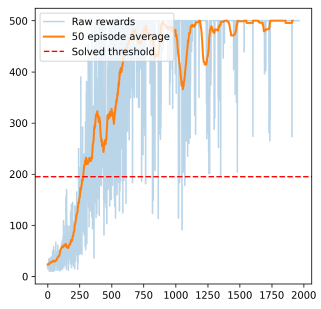
The policy training script is stored in `hw1_baseline_rloo.py`.

1.3 Entropy as regularization

We will be using a RLOO-style baseline as a base for the further studies as it is, in our opinion, a good compromise between convergence speed and stability.

1.3.1 Stable entropy coefficient

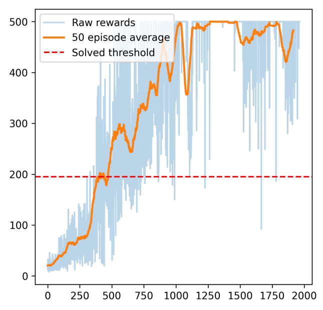
Using logits distribution entropy as a regularization coefficient led to longer convergence but better stability closer to the end of training. The coefficients tried were $\alpha \in \{ 0.001, 0.01, 0.1, 1 \}$. Below is one of the training plots for a middle ground $\alpha = 0.01$.



The policy training script is stored in `hw1_baseline_rloo_entropy.py`.

1.3.2 Linearly scheduled entropy coefficient

Using linear scheduling on the entropy coefficient serves two reasons: better exploration at the start and better convergence at the end of training. However, we didn't manage to achieve results that would perform better than the constant entropy coefficient approach. The figure below depicts train rewards curve for $\alpha\colon 0.01 \to 0.0001$.



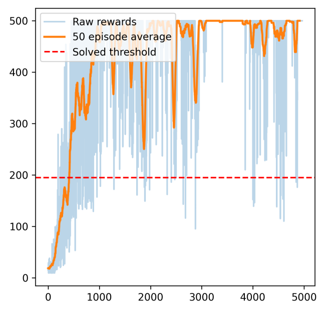
Perhaps the fact that the Cartpole environment is «easy» (nothing to explore in the policies space) and the lack of longer training were the reasons of such underperformance. However for the sake of fair approach comparison it was still decided to cap amount of training episodes to 2000.

The policy training script is stored in `hw1_baseline_rloo_entropy_scheduled.py`.

1.4 Expert model choice

The policy trained with RLOO-style baseline and constant entropy regularization coefficient was selected to serve as expert in further training of a policy in supervised style. In our opinion this training strategy is a good tradeoff between training speed and resulting model efficiency.

It was also decided to increase training episode amount to 5000 to ensure optimality of the expert policy. Below is a plot of train rewards curve for the expert policy.

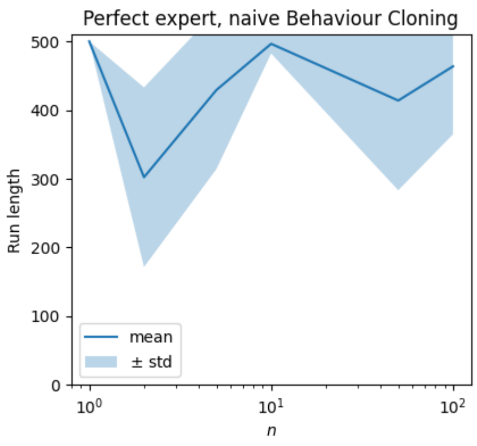


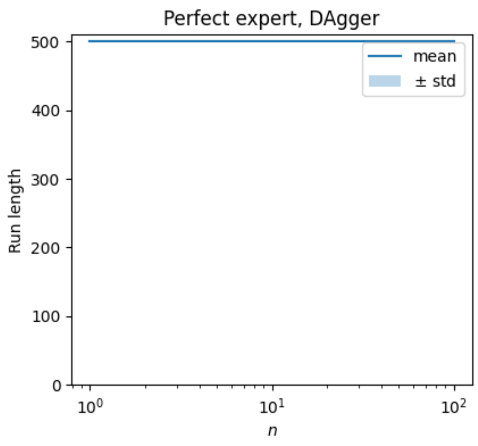
2 Behaviour Cloning

In this section, we are comparing the naive approach to behavior cloning with DAgger. In theory, DAgger should work better than naive behavior cloning, thanks to its ability to reduce compounding errors. We perform two tests: first being supervised-style training of a policy based on n perfect trajectories (i.e. all trajectories of length 500), and the second being same but it may include imperfect trajectories (i.e. trajectories can be both of length 500 and shorter). For both experiments, we consider $n \in \{1, 2, 5, 10, 50, 100\}$. The resulting policies are evaluated on 100 test runs.

2.1 Perfect expert trajectories

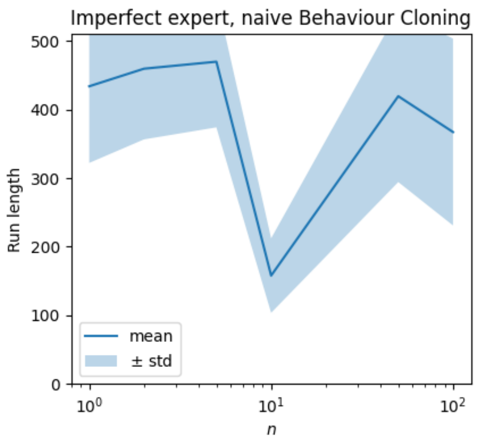
A perfect expert achieves decent but not great results for naive behavior cloning but achieves perfect results using DAgger.

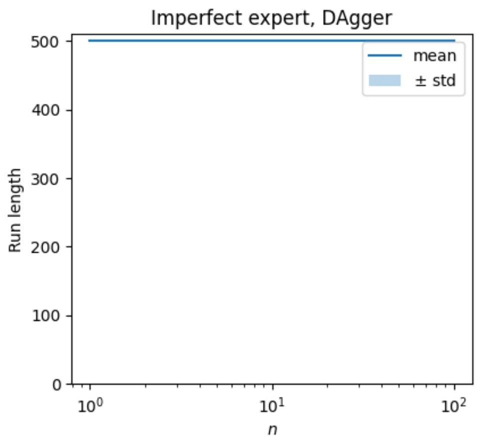




2.2 Imperfect expert trajectories

It is more interesting to consider a scenario that is closer to real life when it's especially hard to collect perfect expert trajectories: an imperfect expert. Now we consider all of the trajectories generated by the expert policy, not just perfect ones. In this case, naive behavior cloning shows slightly worse results, whereas DAgger approach remains perfect.





Training script for both experiments is available in `bc-experiment.py`.

3 Conclusion

In the first part of the report we show that vanilla REINFORCE approach for training a policy is not viable. However, adding even a simplest baseline leads to consistent training, perfecting Cartpole environment every time. Approaches other than using returns mean value as a baseline such as learnable value function or RLOO-style baseline have their own pros and cons, each one exploring a tradeoff between training time and policy space exploring potential.

In the second part of the report we show that DAgger approach is significantly better than naive behavior cloning, leading to perfect scores even for just one trajectory for an imperfect expert model.