Example Projects and Criteria for Alternatives

Project 1: Locomotion control with λ-Soft Actor Critic

Devise an actor-critic algorithm that uses λ-returns for locomotion control in continuous state-action spaces. Take the following paper as basis `https://arxiv.org/abs/1506.02438` and apply it to the Soft Actor-Critic (SAC) pipeline. Evaluate your solution on at least three Mujoco environments, for instance Hopper, Half Cheetah, and Ant. Replicate your experiments multiple times and report the evolution of the evaluation-time total reward as a function of the training steps. Your report should contain:

- All the properties of that limit the applicability of vanilla TD(λ) to the continuous control problem.
- All the design choices you made to overcome these limitations with their technical justification.
- The resulting algorithm as a clean and descriptive pseudo-code.
- How well the resulting model is solving the tasks compared to vanilla SAC.
- What the limitations of your own algorithm are and how they can be improved further in future work.

Project 2: Learning to play Atari games on a discrete latent space

Devise an algorithm that learns to play Atari games from raw pixel observations in two steps: i) A vector-quantized auto-encoder that maps observations to a discrete codebook space. ii) A planner that applies dynamic programming on a Markov Decision Process defined on the codebook space learned in the previous step. Evaluate your algorithm on at least three Atari games supported by the Gymnasium API. For vector quantization, you can take [van den Oort et al., 2017] as basis or design your alternative approach. Replicate your experiments multiple times and report the evolution of the evaluation-time total reward as a function of the gradient-descent steps. Your report should contain:

- A detailed description of all your design choices including the auto-encoder architecture and its training regime, as well as a technical justification of all these choices.
- The resulting algorithm as a clean and descriptive pseudo-code.
- How well your resulting model is solving the task compared to Deep Q Learning.
- What the limitations of your own algorithm are and how they can be improved further in future work.

Project 3: Profiling overestimation bias in ensemble Q-learning

Devise three tabular environments tailored specifically for the overestimation bias problem that have a tunable number of states. Evaluate the overestimation profiles of the following algorithms on these environments:

- `DQN`: Double Q learning as in the original work [Hasselt et al., 2010]
- `DQN-min`: Double Q learning with min-clipping as introduced in TD3 [Fujimoto et al., 2018]

- `Q-ensemble-min`: Apply min-clipping on a group (more than two) of Q tables, i.e. calculat the Bellman target by taking the minimum across all group members.]
- `REDQ`: Ensemble Q learning as practised in the Randomized Ensembled Q-Learning (REDQ) algorithm [Chen et al., 2021], i.e. apply min-clipping on a randomly chosen pair of an ensemble of Q tables.

Evaluate the effect of the environment size (state count starting from 10) and ensemble size (element count starting from 5) on the overestimation bias generated in the course of training. You can compute the overestimation bias by comparing the observed discounted sum of rewards after visiting a particular state to its prediction by the Q table. See Figure 1 of [Fujimoto et al., 2018] for details. Your report should contain:

- A justification of all design choices made in the experiment setup.
- Overestimation plots and learning curves for four methods on different configurations of all three environments.
- A comparison of the overestimation profiles of the four methods in technical terms, i.e. associate their distinguishing properties to the observations in reinforcement learning terms.
- A discussion about the relationship between the overestimation profiles and the corresponding learning curves.
- An explanation of how well the results generalize across the three environments and why.

General remarks

In all projects, train the models as long as your computation resources permit. If a run result is predicted to take unacceptably long, feel free to reduce the maximum number of episodes or the capacities of the used function approximators. Your own project idea is suitable as a course project if your answers to the questions below are "yes":

- Is your project topic relevant to the course material comparable to the projects proposed above?
- Do you modify the existing RL algorithm you build on as much as the projects proposed above?
- Do you evaluate the performance of your algorithm on comparably many different settings as the projects above?
- Do you explain the outcome of your evaluation using the course material at a level of depth comparable to the examples above?