Scalable Soft Optimization

Summary

This project is mainly aimed at a deep reinforcement learning (DRL) implementation. The purpose is to assess selected *soft optimization* methods. Such methods limit the amount of "optimization" in DRL algorithms in order to alleviate the consequences of goal misspecification. The primarily proposed soft optimization method is based on the widely referenced idea of *quantilization*. Broadly speaking, quantilization means sampling options from a reference distribution's top quantile instead of selecting the top option.

The non-summary

Motivation: soft optimization to address over-optimization

Examples of reward "hacking"/"over-optimization", where slight inaccuracies in goal specification lead to undesired behavior, have appeared both in classical DRL applications, and in the field of large language models (LLMs).

Concrete examples of over-optimized behavior can be found on page 45 of this classical paper on LLM training for summarization. There, a reward model to evaluate natural language summaries is trained on human feedback, but its evaluations don't perfectly match the targeted human preferences, especially outside the training distribution. (That is, the rewards are *misspecified*.) Since the space of possible natural language summaries is large, optimization/search finds places where the reward model overestimates desirability to such a degree, that rather undesirable behavior appears to be "optimal".

This type of discrepancy is also known as Goodhart's curse. It is at the core of concerns about "hard" optimization, attempting to find the highest-performing behavior, especially with increasing autonomy and ability (leading to larger and more dangerous behavior spaces and better optimization/search).

Broadly speaking, *soft optimization* describes any method that attempts to implement a limited "degree" of optimization that balances benefits and risks of optimization. Naturally, soft optimization methods may be judged by their ability to strike a good balance.

Indeed, in the case of RL training of generative LLMs by the use of reward LLMs (part of the RLHF (RL from human feedback) pipeline), a regularized objective is usually optimized, yielding a form of soft-optimization. The objective can be recognized as implementing what I call the *classical RL-as-inference* approach and which enjoys popularity in other RL applications as well. This approach is usually not considered promising for soft optimization and I consider it as a baseline to improve on. However, my primarily proposed approach is

itself a (non-classical) version of RL-as-inference, with a smooth transition to the classical version.

Proposed approach: Quantilization by RL as inference

This section is meant to give a high-level description of the proposed approach.

In RL terminology, a trajectory τ describes a sequential interaction between an agent and an environment. Trajectories contain actions sampled from the agent's policy and environment variables (states and rewards) sampled from the environment dynamics, typically for multiple timesteps.
For simplicity, I won't introduce notation for these individual random variables making up τ. Given a policy π, denote the trajectory distribution as $\pi(\tau)$. In the simplest case, the environment dynamics are deterministic and the agent can entirely determine the trajectory τ with its action. In the following discussion, it is often sufficient to imagine τ as a single action. Further, denote the return of a trajectory (i.e., sum of rewards) as r_τ. The return is meant as a proxy for the utility/desirability of the trajectory. Classical RL algorithms usually seek a policy π that maximizes the expected return $E_{\pi(\tau)}\, r_\tau$, thus implementing hard optimization. Such a policy typically selects the most promising action deterministically at each timestep.

Quantilization viewed as inference

The proposed approach requires a prior/reference policy p. The reference policy is typically assumed to be such that most trajectories sampled from the distribution $p(\tau)$ are harmless and their desirability well approximated by their return. It is a starting point for optimization. For LLMs, base models can serve as reference policies, whereas in other applications, simple policies such as uniform distributions may be used.

Consider for the moment the case where τ can be entirely determined by the agent. For a specified return threshold β, the proposed approach recommends selecting trajectories by sampling from the conditional distribution $p(\tau|r_\tau \geq \beta)$, that is, the distribution over trajectories under $p(\tau)$ given that their return exceeds the threshold β.
The threshold β determines the degree of optimization: If β is the maximally attainable return under $p(\tau)$, then sampling from $p(\tau|r_\tau \geq \beta)$ is equivalent to hard optimization. However, as the value of β is reduced, samples from $p(\tau|r_\tau \geq \beta)$ become less and less optimal and more and more typical for samples from $p(\tau)$. By selecting β, we can strike a balance between high return and high likelihood under $p(\tau)$.

This approach is a slightly restrictive reformulation of a frequently mentioned theoretical soft-optimization approach called quantilization: For any β, $p(\tau|r_\tau \geq \beta)$ defines a "quantilizer" with the quantile value $F(\beta)$, where F is the cumulative distribution function of r_τ under $p(\tau)$. Sampling from $p(\tau|r_\tau \geq \beta)$ can be described as sampling from the upper $F(\beta)$-quantile.

As most RL tasks are analytically intractable, the challenge is to find an approximation of quantilization that can be practically implemented. Since finding $p(\tau|r_\tau \geq \beta)$ is an inference problem, it can be cast into the RL-as-inference framework that, in its classical variant, offers several practical RL algorithms. I will refer to this approach as *quantilization by RL-as-inference*.

A smooth RL-as-inference likelihood to approximate quantilization

Note that, by Bayes' rule, $p\left(\tau|r_\tau \geq \beta\right) = \frac{p(\tau)p\left(r_\tau \geq \beta|\tau\right)}{p\left(r_\tau \geq \beta\right)}$. Since the trajectory return r_τ is a deterministic function of the trajectory τ, we simply have $p\left(r_\tau \geq \beta|\tau\right) = 1_{r_\tau \geq \beta}$ as what we call the *likelihood* (of surpassing the return threshold β). We can write this quantilization likelihood in a more general notation (which is also used in RL-as-inference literature) as

$$p(O = o|\tau) \coloneqq p\left(r_\tau \geq \beta|\tau\right) = 1_{r_\tau \geq \beta}$$

and correspondingly write $p(\tau|O=o) = p(\tau|r_\tau \geq \beta)$.
The variable O is just a dummy variable, serving notational convenience, that is assumed to be observed to take the dummy value O=o. In this case O is a Bernoulli variable indicating whether $r_\tau \geq \beta$ is fulfilled, and $O=o \coloneqq 1$ means that this is indeed the case. However, RL-as-inference allows us to choose arbitrary likelihoods that don't require O to have such a clean interpretation:

The above-defined likelihood is not smooth and can take the value zero, which can pose problems for practical algorithms. So we might instead consider the smooth likelihood

$$p(O = o|\tau) \coloneqq \sigma\left(\frac{r_\tau - \beta}{\alpha}\right)$$

for $\alpha > 0$, where σ is a sigmoidal function such as the logistic sigmoid. This likelihood converges to the above non-smooth likelihood as α goes to 0. As β goes to infinity, this likelihood becomes equivalent to the exponential likelihood used in classical RL-as-inference. And, just as α to zero in the classical likelihood retrieves maximization, α to zero in this smooth likelihood retrieves the above non-smooth likelihood and therefore quantilization. In practical algorithms that require the smooth likelihood, α can usually be reduced over the course of training to make the approximation ever finer.

Adapting soft Q-learning to quantilization

Soft Q learning is a classical RL-as-inference algorithm based on learning approximations of "soft Q values" $Q(s,a) = \log p(o|s,a)$ using Bellman equations derived from propagation rules in probabilistic graphical models. These values can be directly used to define action distributions at each state due to $p(a|s,o) \propto p(a|s)p(o|s,a)$. Sampling all actions in a trajectory from the policy $p(a|s,o)$ amounts to sampling from $p(\tau|o)$ as desired (under the assumption of deterministic environment dynamics).

For quantilization, we also want to sample from $p(\tau|o)$, but with the changed definition of the likelihood $p(o|\tau)$. The idea is to emulate the soft Q learning approach for this purpose. However, using a non-classical likelihood creates additional challenges. In particular (since the quantilizer log likelihood is a non-linear function of the sum of rewards) past rewards can not be "forgotten". To preserve notational simplicity we can assume that the sum of past rewards is stored in the state and only paid out at the last timestep.

Indeed, with this view, it is relatively straightforward to adapt soft Q learning for quantilization. Concretely, using the smooth quantilization likelihood $p(o|\tau) = \sigma(r_\tau - \beta)$ instead of the classical exponential likelihood $p(o|\tau) = \exp r_\tau$ simply amounts to (smoothly) capping the

return at β (through replacing r_τ by log $\sigma(r_\tau-\beta)$; all assuming $\alpha=1$), and proceeding in the classical manner.

Dealing with environment stochasticity

So far, τ might have been considered as entirely under the agent's control. This is the context in which the approach truly matches the original quantilization formulation. That original formulation circumvents concerns about environment stochasticity by considering expected utility given a choice of policy. However, this approach appears infeasible to implement in a fashion where the global distribution is broken down into individual action distributions.

Formulating quantilization over trajectories instead of policies, and choosing actions from the policy p(a|s,o), is problematic under stochastic environment dynamics, since the dynamics are not under the agent's control, so trajectory samples will usually not follow p(τ|o). This policy can be described as risk-seeking or risk-avoiding depending on the situation.

The underlying problem is also present in the classical RL-as-inference setup, there exclusively leading to risk-seeking, but the soft Q learning algorithm is actually adjusted to avoid the problem. An adaptation of soft Q learning to quantilization would inherit this adjustment and avoid risk-seeking. However, due to the changed likelihood, further adjustments would be ideal, albeit more involved.

Environments for training and evaluation

Finding suitable environments to test algorithms will be important, and I am open to suggestions. Next to synthetic toy environments to test ideas, the following are the two broad directions that I consider. In accordance with participants' preferences, we can focus on one, attempt both, or consider something else entirely.

First possible direction: Classical RL environments with reward hacks

This direction involves modifying classical reinforcement learning (RL) environments to include deliberate reward hacks, creating a controlled setting to evaluate soft-optimization methods. It would be natural to begin by reimplementing the Soft Q-Learning paper as a baseline. Then, we could adapt the algorithm to the quantilization variant and compare.

Second possible direction: Small-scale RL training of language models from reward models

RL training on reward models naturally provides room for overoptimization to evaluate soft optimization methods on, although it is less straightforward how to measure "true utility", perhaps a larger reward model could sufficiently serve the purpose. Here, the baseline would be the regularized RL training present in RLHF for LLMs, although emulated on a smaller scale. And finally we could adapt the algorithm to the quantilization variant.
Getting language models to a sufficiently small scale for quick and affordable iteration, while keeping the task interesting, may prove challenging.

Project outline

A minimum viable project would involve:

- selecting suitable RL environments.
- designing modifications to these RL environments to feature both a misspecified return/utility function and a "true" return/utility function, and to offer opportunities for severe over-optimization of the misspecified return, in terms of the "true" return.
- reimplementing, as a baseline, established DRL algorithms that follow the classical RL-as-inference approach.
- implementing the simplest quantilization approach, namely by introducing return capping to the baseline algorithms.
- empirically analyzing the relative performance of, and behavioral difference between, approaches.

Additional aspirations are:

- to test a diverse set of environments.
- to find a diverse set of over-optimization failure modes in the environments.
- to implement and evaluate more involved variants of quantilization.
- to publish a clean codebase.
- to write a paper and publish it.

A possible further direction is to use the set up environments and baselines to investigate more involved soft optimizations approaches. (I'd be particularly excited about approaches involving adversarial objectives and/or Bayesian neural networks.)

Output

I think there is a decent chance that we'd get results that can be turned into an academic paper. Additionally or alternatively, a blogpost on LessWrong seems to make sense. A github repo could be an additional output.

Risks and downsides (externalities)

The most salient downside to me is in the opportunity costs that this project creates. The project involves a time investment on the side of its participants that might be better spent elsewhere. Additionally, each participant's research interests might be guided in a way that creates further time investments at the cost of counterfactual research interests. And a successful project might additionally influence the research interests of other alignment researchers.

Acknowledgements

This project proposal was inspired by an earlier AISC project that I participated in.

Team

Team size

Me and 2-3 other team members. If more, then divided into environment-type-specific subteams.

Research Lead

My name Benjamin Kolb and I can be reached under BK-at-AISCVirtual2025@gmx-topmail.de. I'm a master's student in artificial intelligence writing my master's thesis on RL-as-inference. I've participated in the 2023 edition of AISC in a team on uncertainty and soft optimization.

Skill requirements

I'm primarily looking for people who are enthusiastic about:

- reimplementing classic DRL papers.
- developing RL environment setups with reward misspecification that allow to showcase the relative performance of soft optimization methods.
- implementing and evaluating the proposed soft optimization methods.
- pursuing publishable results.

Collaborators should have:

- a sense of commitment to the project.
- the ability to work effectively with the team.
- a good intuitive understanding of (Bayesian) probabilities.
- the ability to code in Python/Pytorch or simply the readiness to quickly learn coding in Python/Pytorch. (Alternative libraries can be proposed.)
- the willingness to use Git and to write readable and well-structured code.

Additionally valuable are:

- experience with deep learning implementations.
- experience with software engineering.
- familiarity with RL algorithms.
- familiarity with language model training.
- strong mathematical/conceptual reasoning skills.
- strong academic writing skills.