DECOUPLING LEARNING FROM NEGATIVE AND POSITIVE FEEDBACK IN PREFERENCE OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing preference optimization methods are mainly designed for directly learning from human feedback with the assumption that paired examples (preferred vs. dis-preferred) are available. In contrast, we propose a method that can leverage unpaired preferred or dis-preferred examples, and works even when only one type of feedback (positive or negative) is available. This flexibility allows us to apply it in scenarios with varying forms of feedback and models, including training generative language models based on human feedback as well as training policies for sequential decision-making problems, where learned (value) functions are available. Our approach builds upon the probabilistic framework introduced in (Dayan & Hinton, 1997), which proposes to use expectation-maximization (EM) to directly optimize the probability of preferred outcomes (as opposed to classic expected reward maximization). To obtain a practical algorithm, we identify and address a key limitation in current EM-based methods: when applied to preference optimization, they solely maximize the likelihood of preferred examples, while neglecting dis-preferred samples. We show how one can extend EM algorithms to explicitly incorporate dis-preferred outcomes, leading to a novel, theoretically grounded, preference optimization algorithm that offers an intuitive and versatile way to learn from both positive and negative feedback.

1 INTRODUCTION

The use of preference annotated data for training machine learning models has a long history going back to early algorithms for recommender systems and market research (Guo & Sanner, 2010; Boutilier, 2002; Bonilla et al., 2010). These days preference optimization algorithms are receiving renewed attention since they are a natural candidate for shaping the outputs of deep learning systems, such as large language models (Ouyang et al., 2022; Team, 2024a) or control policies, via human feedback (Christiano et al., 2017; Rafailov et al., 2023; Azar et al., 2023). Arguably, preference optimization algorithms can also be a natural choice even when direct human feedback is not available but one instead aims to optimize a machine learning model based on feedback from a hand-coded or learned critic function (judging desirability of solutions). Here preference optimization methods are useful since they let us optimize the model to achieve desired outcomes based on relative rankings between outcomes alone (rather than requiring absolute labels or carefully crafted reward functions).

Among preference optimization approaches, those based on directly using preference data – as opposed to casting preference optimization as reinforcement learning from (human) feedback – such as DPO (Rafailov et al., 2023), have emerged as particularly successful since they only require access to an offline dataset of paired preference data, and are fairly robust to application domain and hyperparameter settings. However, algorithms within this class make specific assumptions tailored to their application domain. They were designed to optimize LLMs from human feedback in the form of comparisons of generated sentences and thus, by design, require paired preference data (since they directly model a specific choice of preference distribution). We are interested in finding algorithms that are more flexible, and applicable in settings where the assumptions underlying DPO do not apply.

In this work we take a fresh look at preference optimization from a probabilistic inference perspective that has been used with great success in the literature on KL regularized reinforcement learning (Dayan & Hinton, 1997; Peters et al., 2010; Abdolmaleki et al., 2018). We find that from this perspective a simplified approach to preference optimization can be derived that is intuitive to understand and is capable of leveraging an arbitrary number of unpaired preferred or dis-preferred outcomes, or even solely one type (positive or negative) of preference feedback. In particular, our method is able to learn even if exclusively positive or negative examples are available. Formally, our method involves an objective consisting of three log likelihood terms that are derived from first principles: maximizing the likelihood of preferred outcomes, minimizing the likelihood of dis-preferred outcomes, while staying close to a reference distribution (see equation 10). We show the effectiveness of our method across a wide range of benchmarks including synthetic benchmarks, training policies for continuous control, and training large language models (LLMs) from human feedback.

2 RELATED WORK

2.1 RL AS INFERENCE

Viewing reinforcement learning through the lens of probabilistic inference offers an alternative framing of RL (Dayan & Hinton, 1997). This "RL as inference" perspective has gained considerable attention recently (Levine, 2018) inspiring various expectation-maximization (EM) based RL algorithms (Peters et al., 2010; Abdolmaleki et al., 2018). Essentially, these policy improvement algorithms can be viewed as performing EM to optimize the likelihood of a successful outcome. However, a limitation of these algorithms is their reliance on successes (preferred outcome) data. In this paper, we extend this framework to incorporate dis-preference information; effectively allowing the policy to make unwanted outcomes *less* likely. We show that this alone can have an positive effect on data efficiency and performance on certain tasks, notwithstanding the added flexibility.

2.2 PREFERENCE OPTIMIZATION

Preference optimization methods like Direct Preference Optimization (DPO; Rafailov et al., 2023) and Identity Preference Optimization (IPO; Azar et al., 2023) have enjoyed much attention lately, especially in the LLM training literature. This success is mostly due to a so-called *direct* optimization of human preferences, in contrast to reward model training required in RL from human feedback (RLHF) training pipelines. Nevertheless, these preference optimization methods were designed specifically to learn from a particular type of data: pairs of preferred and dis-preferred data, usually coming from humans indicating their preference over a pair of LLM responses to their query. This can be restrictive in scenarios where multiple outcomes need to be considered, and DPO has since been extended to multiple generations and compared to a novel method Efficient Exact Optimization (EXO; Ji et al., 2024), both shown to outperform the RLHF baseline in cases where a reward model is available. In this paper, we leverage the RL as inference framework to generalize preference optimization even further, allowing for more general algorithms derived from first principles. Our approach can not only handle scenarios with multiple generations but it also naturally handles cases where only one type of feedback is accessible (i.e. all generations are failures), which can be particularly useful for challenging task with binary success/failure outcomes (e.g. code, math, safety assurance).

3 USING PREFERRED AND DIS-PREFERRED OUTCOMES FOR POLICY OPTIMIZATION

In this section we present an approach to optimising policies based on preference data and show how it can be used to tackle a variety of problem settings in Machine Learning; e.g. policy optimisation in a standard RL setting or learning from human preference feedback. We build upon a large body of existing work in probabilistic inference for policy optimization. In particular we make use of a formulation underlying many existing KL regularized RL approaches that are motivated from the RL as inference perspective such as REPS (Peters et al., 2010), AWR (Peng et al., 2019) and MPO (Abdolmaleki et al., 2018). We will show that, when applied to preference optimization, the Expectation-Maximization (EM) approach employed by this class of algorithms results in a natural formulation of maximizing (weighted) likelihood of preferred outcomes. Since such a formulation is appealing due to its simplicity but cannot effectively use information about dis-preferred outcomes (unless we make the often unrealistic assumption of having access to the full probability distribution over preferences) we finally derive a simple extension that enables the use of dis-preferred/disliked data-points.

The resulting algorithm has multiple intriguing properties: it can make use of preference data containing positive and negative outcomes but it does not require paired outcomes (i.e. it can make use of data for which we only know whether it is either good or bad, without knowing about relative preference with respect to other data-points) and can thus also naturally utilize unbalanced datasets (where e.g. we have multiple preferred options for each dis-preferred example, or vice-versa). Due to the close relationship of our algorithm to the existing MPO algorithm (Abdolmaleki et al., 2018) *we refer to it as preference based MPO (PMPO).*

3.1 BACKGROUND ON MAXIMISING PREFERRED OUTCOMES

We review the preference based RL formulation common in RLHF (Ziegler et al., 2019; Rafailov et al., 2023) and show how methods from the literature on EM based policy optimization (Rawlik et al., 2013; Peters et al., 2010; Abdolmaleki et al., 2018) can be naturally applied to it.

In the following, x denotes the conditioning variable such as the state/observation in classical RL, or a document and query in the LLM finetuning setting. Providing this information to a model (or policy) produces $\pi(y|x)$, a probability distribution over outputs y; these would be actions in classical RL or responses/generations (sequences of tokens) in the LLM finetuning literature. We will also make use of the definition of a KL divergence between conditional distributions which we define as $\mathrm{KL}(p(\cdot|x) \parallel q(\cdot|x)) = \mathrm{KL}(p, q; x) = \mathbb{E}_{y \sim p(\cdot|x)}[\log p(y|x) - \log q(y|x)]$.

Objective. Define a binary random variable S, which takes a value of 1 in the event of a successful outcome and 0 otherwise. To lighten notation, we will use the shorthand $p(S)$ and $p(S')$ to mean $p(S = 1)$ and $p(S = 0)$, respectively, and similarly for the conditioned distributions. In words, our goal is to optimize the parameters θ of a parametric policy $\pi_\theta(y|x)$ to produce outcomes y that have a high likelihood of being preferred as measured by an unknown preference distribution $p(S|y, x)$, i.e. the event that y is a 'preferred' or 'successful' response to the condition x:

$$\max_{\theta} \mathop{\mathbb{E}}_{y \sim \pi_\theta} p(S|y, x) \tag{1}$$

Reference model. In addition to the above general formulation we assume access to a reference model π_{ref} that can either consist of a previous iteration of the model we would like to improve, or be the outcome of a pre-training or supervised fine-tuning phase (as routinely employed in RLHF for LLMs). We refer to this model as the reference policy and in general we use the terms model and policy interchangeably.

Preference information. In order to derive a practical sample based algorithm we have to assume some knowledge of the preference distribution $p(S|y, x)$; we distinguish two cases in this paper. In the first case we assume we have pointwise access function $f(y, x)$ that is proportional to the log-probability density function (PDF) of the preference distribution. That is, we have a Boltzmann distribution $p(S|y, x) \propto \exp(f(y, x)/\eta)$ where η is a temperature parameter. For cases where we can query a reward function r or a state-action value-function Q,[1] f can be set to be r or Q, respectively. In this case we only need samples (x, y) for optimization without explicit success/failure labels. In the second case we assume we only have access to a dataset of labelled examples:

$$\mathcal{D} = \left\{ x^{(i)},\ y^{(i,j)},\ s^{(i,j)} \right\}_{i,j=1}^{N,M_i}$$

where $y^{(i,j)} \sim \pi_{\text{ref}}(\cdot|x^{(i)})$ and the $s^{(i,j)}$ are binary preference labels, usually obtained from human feedback. In other words, in this second case we only have samples from the real preference distribution $P(S|y, x)$ as opposed to a Boltzmann model for it.

Policy optimization. Let us drop the superscripts (i) for now and only consider the objective on a per-condition basis, ultimately we average over the batch. Then for every conditioning $x = x^{(i)}$, the problem is finding a policy that achieves the highest marginal probability of preferred outcomes. This

---

[1] Defined as $Q(y, x) = \mathbb{E}[\sum_t \gamma^t r(y_t, x_t)|x_o = x, y_0 = y]$ for a timeseries of observation/action pairs.

amounts to optimizing

$$\log p_\pi(S|x) = \underbrace{\mathop{\mathbb{E}}_{y\sim q}\Big[\log \frac{\pi(y|x)p(S|y,x)}{q(y|x)}\Big]}_{\mathcal{J}(\pi;\,q,\,x)} + \mathrm{KL}(q(y|x)\|\, p_\pi(y|S,x)), \tag{2}$$

where we have used a standard formulation from the probabilistic inference literature (Kingma & Welling, 2013) to decompose the objective into an evidence lower bound $\mathcal{J}(\pi; q, x)$ and a KL term by introducing an auxiliary variational distribution q that we will use as a 'vehicle' to perform stable optimization via expectation maximization (Dayan & Hinton, 1997). The goal of EM is to iteratively find a tight lower bound given the current estimate π_{ref} by optimizing for q (E-Step) and improve the lower bound $\mathcal{J}$ by optimizing for π (M-Step). More concretely, in the E-step, we fix $\pi = \pi_{\text{ref}}$ and find the $\hat{q}$ which minimizes the KL; this tightens the bound. In the M-step, we fix $q = \hat{q}$ and maximize the lower bound $\mathcal{J}(\pi_\theta; \hat{q}, x)$ to update π_θ. This process of tightening the bound and improving the policy constitutes one iteration of policy improvement over π_{ref}.

E-step: Tighten the lower bound by fixing $\pi = \pi_{\text{ref}}$ and minimize $\mathrm{KL}(q(\cdot|x) \,\|\, p_{\pi_{\text{ref}}}(\cdot|S,x))$. Following prior work (Dayan & Hinton, 1997; Peters et al., 2010; Abdolmaleki et al., 2018), since the KL is minimized when both distributions are equal, the solution can be expressed in closed form as $\hat{q}(y|x) = p_{\pi_{\text{ref}}}(y|S,x)$. Then, according to Bayes rule:

$$p_{\pi_{\text{ref}}}(y|S,x) = \frac{1}{Z_x}\pi_{\text{ref}}(y|x)p(S|y,x), \tag{3}$$

where we used the normalization factor $Z_x = \int \pi_{\text{ref}}(y|x)p(S|y,x)\,\mathrm{d}y$. Recall that $p(S|y,x)$ is still a modelling choice discussed in the *Preference information* section.

M-Step: Optimize the lower bound $\mathcal{J}$ fixing $q = \hat{q}$ from the previous step. Since this problem does not have an analytic solution we use a parametric function approximator π_θ, usually a large neural network, and maximize the following objective via gradient ascent:

$$\mathcal{J}(\pi_\theta; \hat{q}, x) = \mathop{\mathbb{E}}_{y\sim\hat{q}}\Big[\log \frac{\pi_\theta(y|x)p(S|y,x)}{\frac{1}{Z_x}\pi_{\text{ref}}(y|x)p(S|y,x)}\Big] = \mathop{\mathbb{E}}_{y\sim\hat{q}}\Big[\log \pi_\theta(y|x)\Big] + K \tag{4}$$

$$\mathcal{J}(\pi_\theta; x) = \mathop{\mathbb{E}}_{y\sim\pi_{\text{ref}}}\Big[\frac{p(S|y,x)}{Z_x}\log \pi_\theta(y|x)\Big], \tag{5}$$

where K represents all constant terms that are independent of θ and are dropped from the final objective. Notice that this objective amounts to a weighted maximum likelihood with preferences determining the weights and samples coming from π_{ref}. Notice also that the final expression subsumes the closed form E-step solution such that we can safely consider only this objective and introduce the short-hand $\mathcal{J}(\pi_\theta; x)$, dropping the implicit dependence on the E-step solution. In practice, to optimize this objective we need to form a Monte-Carlo approximation of the expectation in Eq. (5). We distinguish the two cases mentioned in the *Preference information* section.

In the first case, we assume access to a function f that is proportional to the preference log-probability, and access to M responses $y^{(j)}$ for each x. We can then set $p(S|y,x) \approx w^{(j)} \propto \exp(f(y^{(j)},x)/\eta)$ in Eq. (5) (a softmax of f across the responses $y^{(j)}$ to x). This is the case commonly studied in the literature, e.g., in MPO where one uses $f = Q(y^{(j)}, x)$.

It is often unrealistic to assume access to a reliable model of preference labels. For example, preferences often come from human annotations and we thus only have access to samples or we might only have access to a learned and unreliable preference model.[2] To cover this case, let us partition our dataset of labeled examples $\mathcal{D} = \mathcal{D}_a \cup \mathcal{D}_r$ where $\mathcal{D}_a = \{y^{(j)} \ni (s^{(j)} = 1)\}_{i=1:M}$ and $\mathcal{D}_r = \{y^{(j)} \ni (s^{(j)} = 0)\}_{i=1:M}$, denote accepted (preferred) samples and rejected (dis-preferred) samples, respectively. In this case we can still use the objective from Eq. (5), using the binary preferences $s^{(j)}$ as weights:

$$\mathcal{J}(\pi_\theta; x) \approx \mathcal{J}_a(\pi_\theta; x) = \mathop{\mathbb{E}}_{y^{(j)}\sim\mathcal{D}}\big[s^{(j)}\log \pi_\theta(y^{(j)}|x)\big] = \mathop{\mathbb{E}}_{y^{(j)}\sim\mathcal{D}_a}\log \pi_\theta(y^{(i)}|x), \tag{6}$$

which effectively filters rejected generations $\mathcal{D}_r$ out, thus reverting back to the maximum likelihood objective on preferred data.

---

[2] A case studied in the offline RL literature where the authors realised that using binary weights often works better as in binary CRR (Wang et al., 2020).

3.2 USING DIS-PREFERRED OUTCOMES VIA REGULARISED MINIMUM LIKELIHOOD

We will now derive a simple way to incorporate negative (dis-preferred) samples into the optimization to address the shortcomings of naively applying the EM-based perspective from the previous section. We would like to incorporate these examples without changing the overall objective since it has well established policy improvement guarantees (Rawlik et al., 2013; Abdolmaleki et al., 2018). To accomplish this we take a second look at the non-parametric variational distribution $\hat{q}$ from Eq. (3) that is the solution to our E-step; since it determines the sampling distribution used for the M-step.

We can realise that the restriction to positive/preferred samples stems from the fact that we express $\hat{q}$ directly in terms of the preference distribution $p(S|y,x)$. A natural question then is: can we re-express $\hat{q}$ in terms of dis-preferences? It turns out the answer to this is positive. Recall that S' denotes the complement of the event S i.e. the event that y is not a successful action/response to a conditioning x. Then by definition, $p(S|y,x) = 1 - p(S'|y,x)$ we can equivalently write

$$\hat{q}(y|x) = \frac{1}{Z_x}\pi_{\text{ref}}(y|x)(1 - p(S'|y,x)). \tag{7}$$

We can plug this form of $\hat{q}$ into the evidence lower bound expressed in Eq. (4). After rearranging terms and re-writing in terms of two expectations over π_{ref} this gives the alternative form:

$$\mathcal{J}(\pi_\theta; x) = \mathop{\mathbb{E}}_{y\sim\pi_{\text{ref}}}\Big[-\frac{p(S'|y,x)}{Z_x}\log\pi_\theta(y|x)\Big] - \frac{1}{Z_x}\,\text{KL}(\pi_{\text{ref}}, \pi_\theta; x) + K, \tag{8}$$

where K again denotes terms independent of π_θ. This version of the objective now is expressed in terms of the dis-preference distribution and we can use it to define a Monte-Carlo estimate based on dis-preferred examples as (where the additional constant 1 results in KL regularization):

$$\mathcal{J}(\pi_\theta; x) \approx \mathcal{J}_r(\pi_\theta; x) = \mathop{\mathbb{E}}_{y^{(j)}\sim\mathcal{D}_r}\big[-\log\pi_\theta(y^{(j)}|x)\big] - \beta\,\text{KL}(\pi_{\text{ref}}, \pi_\theta; x), \tag{9}$$

where β is an additional tuning parameter that is typically set high enough to only remove the dis-preferred outcomes from the prior π_{ref}. As before, our use of samples $s^{(j)}$ (labelled data) filters out part of the dataset; in this case, it is the accepted responses which are filtered out, hence the expectation over $\mathcal{D}_r$. We refer to the appendix for a full derivation. This is a fairly intuitive objective to optimize. It tells us to *minimize* the likelihood of dis-preferred examples while staying close to the reference model. Interestingly, compared to the preferred data case, it has an additional KL term that appears as a result of the reparameterization of the variational distribution. We will see in the experiments that this term is required when learning from negative data to avoid arbitrary solutions where probability is assigned to random out-of-data responses. Intuitively, we can think of the objective as modifying the reference distribution such that the negative examples are removed. Interestingly such an additional KL for the M-step has previously been considered in the literature even for the case where we have access to the preference distribution but must perform optimization based on a limited set of sampled responses; as in MPO (Abdolmaleki et al., 2018). However, previous work used the additional KL term to prevent rapid entropy loss. In contrast, our motivation for incorporating the KL term is to learn from negative samples, as suggested by the derivations.

3.3 LEARNING FROM PREFERRED AND DIS-PREFERRED OUTCOMES

Finally, we can form a combined objective from our two M-step estimates – which both optimize the same quantity but can utilize different samples. That is, we combine Eq. (6) and Eq. (9):

$$\boxed{\mathcal{J}_{ar}(\pi_\theta; x) = \underbrace{\alpha \mathop{\mathbb{E}}_{y\sim\mathcal{D}_a}\big[\log\pi_\theta(y|x)\big]}_{\text{Learning From Accepted Samples}} - \underbrace{(1-\alpha)\mathop{\mathbb{E}}_{y\sim\mathcal{D}_r}\big[\log\pi_\theta(y|x)\big] - \beta\,\text{KL}(\pi_{\text{ref}}, \pi_\theta; x),}_{\text{Learning From Rejected Samples}}} \tag{10}$$

where α is a trade-off parameter between the two estimates. Recall that in practice, this objective will be aggregated over an entire dataset of conditions x and corresponding datasets $\mathcal{D}_a$ and $\mathcal{D}_r$. There are a few interesting things to note about this objective. First, we emphasize that our objective assumes categorization of samples into good/bad or preferred/dis-preferred datasets. As a result, it can be used *even when only positive or only negative samples are available* (this is in contrast to e.g. DPO (Rafailov et al., 2023) or IPO (Azar et al., 2023) which require relative scores of paired

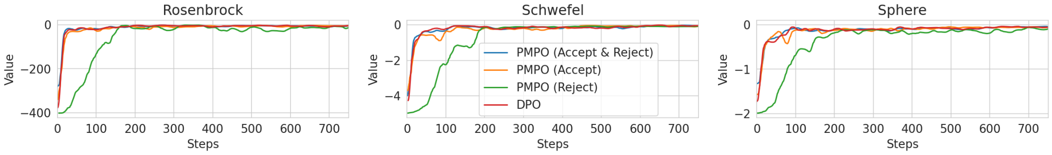

Figure 1: Performance of PMPO and DPO on Benchmark Functions - This figure illustrates the optimization progress of PMPO variants (PMPO-AR, PMPO-A, PMPO-R) on a selection of standard benchmark functions, showcasing their ability to leverage different types of preference feedback.

positive and negative examples for each query x). Furthermore, the objective has also no restriction on the number of positive / negative samples per query x and thus it automatically extends to the multi-sample case for fine-tuning language models. Finally, the objective is intuitive and simple to implement; it amounts simply to maximizing likelihood of good data while minimizing likelihood of bad data and staying close to the reference model.

PMPO introduces α and β hyperparameters; see Appendix A for guidance on tuning these parameters. Furthermore the KL term is implemented in closed form whenever possible. For example, for the autoregressive models used in LLMs, we use the sum of per-token closed-form KL divergences of categorical distributions. This enable us to learn only from a negative feedback without access to positive feedback as suggested by our derivations. See the details in appendix B for KL computation and more discussion on importance of KL computation in closed form.

4 EXTRACTING PREFERENCES FROM EVALUATION FUNCTIONS

Our algorithm requires access to preference information, which can come directly from human feedback or be extracted from an evaluation function. This section describes the latter. We consider improving policies within a traditional reinforcement learning (RL) setting; bandit optimization and optimization of language models via RLHF. In each setting our preference-based update rule can be used in the policy improvement step. For this we need to extract preference information from a (possibly learned) evaluation function. This can be achieved in the following way:

Generate Samples: For a given input or state x, sample one or multiple generations y from the current reference policy π_{ref}.

Evaluate Actions: Calculate the evaluation function $f(x, y)$ (e.g. a reward model in RLHF) for each input-generations pair (x, y).

Classify Actions: If $f(x, y) \geq b(x)$, classify the generation y as preferred in state x. Otherwise ($f(x, y) < b(x)$), classify it as dis-preferred.

5 EXPERIMENTS

We evaluate our algorithm in a variety of different settings, showcasing its utility as a general preference optimization algorithm that can deal with many different forms of preference feedback. We first test it in a Bandit setting (optimizing synthetic benchmark functions) then in a setting where we transform RL on control and robotics tasks into preference optimization. And finally we showcase strong performance for RLHF of large language models. To verify our derivations, we evaluate three different variants of the PMPO algorithm: learning only from accepted samples ($\alpha = 1$), learning only from rejected samples ($\alpha = 0$), and learning from both accepted and rejected samples ($\alpha = 0.5$). We also use MPO (Abdolmaleki et al., 2018) and DPO (Rafailov et al., 2023) as baselines. For all the experiments, we will use a beta value of 0.5 for learning from accept&reject, 0.0 for learning from accept only, and 2.0 for learning from reject only, unless stated otherwise. Furthermore, in all experiments except experiment 5.3, the reference policy for all baselines is updated every N steps to allow for multiple policy improvement steps and demonstrate that our algorithm can effectively optimize the underlying reward function until convergence. For experiment 5.3, we only have access to samples from the reference policy; therefore, we can make only one improvement step, which

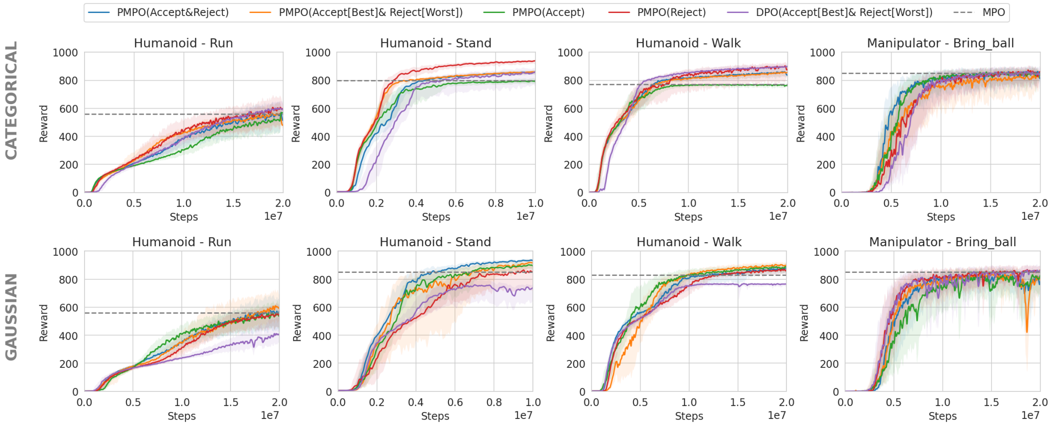

Figure 2: Comparison of PMPO/DPO/MPO for high-dimensional control tasks from the DeepMind Control Suite. We plot average reward over time of training (using 100 episodes for each evaluation).

means the reference policy is effectively fixed. Please note that experiment 5.3 is designed to have access to only positive or negative feedback for each state.

5.1 BANDIT RL: STANDARD FUNCTIONS

First, we evaluate our algorithm on a suite of well-known synthetic benchmark functions, including the Rosenbrock, Sphere and Schwefel functions (Hansen et al., 2003); all of which have an optimum value of zero. Each function optimization problem is framed as a preference optimization problem analogous to multi-armed bandit optimization (Auer et al., 2002); i.e. there is no state conditioning x. At each iteration, the policy proposes 4 samples within the function's domain and observes evaluations (function values) as feedback. The reference distribution used for sampling actions is a time lagged version of the policy being optimized (updated every 100 optimization steps). Subsequently, the algorithm utilizes the two top samples as preferred samples and the other two samples as dis-preferred samples. Figure 1 illustrates the performance of our PMPO algorithm under different feedback settings (PMPO-AR: uses all 4 samples and thus accepted and rejected samples, PMPO-A: uses only the accepted samples, PMPO-R: uses only the rejected samples). The results show that all three variants of our algorithm can successfully learn and optimize the objective functions. This demonstrates that we can effectively leverage diverse sources of preference information. It may seem surprising that even just using negative samples is enough to drive optimization towards the optimum but together with the KL constraint towards the slowly changing reference it can be effective. DPO, which uses the best and worst action samples among the 4 sample archives, exhibits similar performance to PMPO-AR.

5.2 FULL ONLINE RL: CONTROL SUITE

We evaluate our algorithm on a range of control tasks from the DeepMind Control Suite (Tunyasuvunakool et al., 2020). See appendix for details. We cast the setting of optimizing a policy for the control suite as a preference optimization problem by leveraging a learned action-value function (a Q-function)–represented by a separate network trained alongside the policy–to infer preferences for each observed state and action. This is analogous to the actor-critic setting in classical reinforcement learning. Similar to the bandit case, at each iteration, the reference policy proposes four actions for each state in the batch. The top two actions with the highest Q-values are considered preferred samples, while the two actions with the lowest Q-values are treated as dis-preferred samples. We consider two different cases, one where the output of the neural network are mean and standard deviation of a Gaussian control policy and one where the actions are discretized into bins (and the network outputs categorical logits over these bins).

Figure 2 demonstrates that, as in the bandit case, our algorithm can effectively learn from different types of available signals (accept/reject, accept-only, reject-only) to solve high-dimensional tasks, such as controlling humanoid agents to run, stand, and walk, as well as manipulating objects. In all of

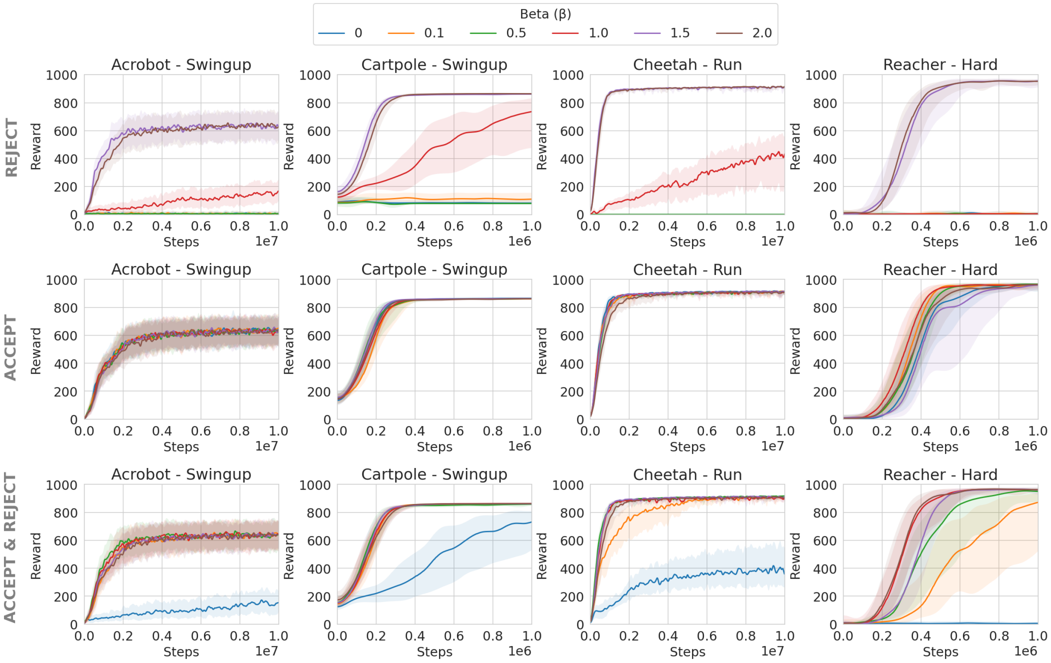

Figure 3: Impact of the KL weight 'beta' on the performance of PMPO. When learning solely from dispreferences across various Control Suite tasks (Reject, $\alpha = 0$), a sufficiently high beta value is required for effective learning. However, when learning from preferences only (Accept) PMPO is robustness to the KL weight 'beta' across different Control Suite tasks, confirming theoretical insights. When both both accept and reject signals are used (Accept & Reject), PMPO shows a partial sensitivity to KL Weight 'beta'. While learning is possible with a wider range of beta values, a beta higher than 0.5 is generally needed for optimal performance.

them PMPO matches or outperforms the strong MPO baseline. Notably, even with only reject signals (PMPO-R), the algorithm is capable of achieving good performance. As predicted by the theory, not using a KL can quickly lead to collapse when using only dis-preferred samples. We also compare to an implementation of DPO (Rafailov et al., 2023) which uses the best and worst action sample among the 4 samples. This still results in a strong algorithm that works well when using a discretized action representation. However, in the continuous Gaussian case, DPO requires a very high implicit regularization parameter ($\beta = 20$) which results in slow learning and suboptimal policies. For the sake of fair comparison with DPO that uses the worst and best generation, we also show results for PMPO when only the best is labeled as preferred and the worst is labeled as dispreferred, which is still competitive with DPO.

We further ablate the impact of the KL term on learning solely from dispreferences ($\alpha = 0$), solely from preferences ($\alpha = 1$), and from both ($\alpha = 0.5$). For each of these settings, we sweep over the β parameter in the range $(0.0, 0.5, 1.0, 1.5, 2.0)$. As depicted in Figure 3, when learning exclusively from dispreferences (PMPO-R), the performance is highly sensitive to β. To achieve effective learning, we need to set β sufficiently high (> 1.0), which aligns with our theoretical derivations. In contrast, Figure 3 shows that the algorithm is insensitive to the setting of β when learning only from preferred samples (PMPO-A), again confirming our theoretical insights. When learning from both types of signals (PMPO-AR), as shown in Figure 3, we observe a partial sensitivity to the KL weight β. While the algorithm can learn with a wider range of beta values, a β larger than 0.5 is still necessary to ensure optimal performance across all tasks.

5.3 OFFLINE RL USING ADVANTAGE FUNCTION

In a final set of experiment on control domains we want to show that our algorithm can also be applied to a setting where we have only access to one sample with either a reject or an accept label per state conditioning x. We consider the RGB Stacking benchmark (Lee et al., 2021), a pick-and-place manipulation task with image observations (see appendix for details). We investigate the effect of positive and negative feedback in the context of offline RL to exclude cascading effects from exploration. To this end we take a dataset of 140k episodes from a multi-task RL experiment trained to convergence (Lampe et al., 2024). We then train a value function on all data and use it to label the transitions in the first 40k episodes as accept (positive advantage) or reject (negative advantage). Different combinations of acceptance, rejection, and BC losses are then compared in order to understand their respective effects. In summary we use: i) the full 140k episodes to train a value function and label the first 40k episodes as accept or reject ; ii) the first 40k episodes to compute the positive weighted part of the loss if labeled as accept and to compute the negatively weighted part of the loss if labeled as reject. The KL part of the loss is calculated on all 140k episodes. Note that the value function is only used to transform the reward annotations into accept and reject labels.

Table 1 shows the achieved reward for different loss combinations. First we run BC on the full 140k episodes and we can observe that the performance is mediocre due to the data containing a significant amount of bad episodes. Using only the accepted transitions for BC training does not result in better performance; this is due to the limited number of positive examples contained in the first 40k episodes. When combining both BC and using the positive (accept) part of the loss, performance does not significantly improve as the large number of negative episodes is not compensated for. On the other hand, combining BC with the negative (reject) part of the loss does significantly improve performance. This is due to the rejection loss successfully pushing the policy away from the negative examples (while keeping close on all other data due to the KL constraint). Finally, best performance is achieved when combining all three losses; and thus effectively utilizing all data. While in this example we have constructed the dataset in a way that the effect is strong, and this might be less the case in more natural settings, it nevertheless shows that using a negative signal can have a significant effect on performance by masking the effect of bad data.

	BC	Accept+BC	Accept	Reject+BC	Accept+Reject+BC
Reward	24	26	27	77	93

Table 1: Comparing different mixtures of acceptance, rejection and BC losses. We measure average reward (over 100 evaluation episodes) across stacking of all 5 triplets. Training with BC is corrupted by bad examples. Training on only accepted examples lacks data. Only when integrating the rejection loss bad data can be masked and performance goes up. Best performance is achieved when combining acceptance, rejection and BC loss signals.

5.4 LANGUAGE ALIGNMENT EXPERIMENTS

We apply different versions of the PMPO algorithm to the task of aligning large language models. Specifically, we fine-tune a Gemma 2B pre-trained model using a trained reward model (Team, 2024b) using prompts from the LMSYS-chat-1M dataset (Zheng et al., 2023). The reward model has been trained on human preference data with a Bradley-Terry modelisation as explained in (Christiano et al., 2017). In these experiments, we perform one epoch of training, processing a dataset of 500k prompts in approximatively 4000 learner steps, meaning that each batch is composed of 128 prompts and 4 generations per prompt. Similar to the typical RLHF setting, at each iteration, for each prompt in a batch, we sample four generations from the model and rank them based on their reward values. The top two generations are labeled as preferred, and the bottom two as dis-preferred. For the sake of fair comparison with DPO that uses the top one (best) and bottom one (worst) generation, we also show results for PMPO when only the top one is labeled as preferred and the bottom one is labeled as dispreferred. Note that this particular choice could be refined further and tailored to the task. First, Fig. 4 showcases the best PMPO setting, leveraging both accept and reject signals (PMPO-AR) (and we compare to use either feedback signal in isolation). Notably, utilizing both types of feedback leads to faster learning compared to using either signal in isolation (PMPO-A or PMPO-R) and overall our approach is competitive to DPO, which is applicable in this setting by using only the best and worst sample respectively per prompt but would be more restrictive in general (i.e. it cannot naturally make use of unbalanced preference data). As shown on the right, when performing a side by side comparison using GPT-4 (OpenAI, 2024) to judge whether our model is preferred over the base Gemma model (using a set of held-out test prompts) the PMPO fine-tuned model wins over the base model. Note in Fig. 4 right, we see some drop indicating some exploitation of the imperfect reward model; known as reward hacking (Skalse et al., 2022). We can see that PMPO-AR is the quickest to "hack the reward"(see Fig. 4 left), it reaches a good performance but then in the middle of training its start hacking the reward and learns a pathological behaviour that makes it performs worse on the independent benchmark. This phenomenon has been observed consistently in RLHF. Overall, our language alignment experiments provide strong evidence for the effectiveness and versatility of PMPO. Finally, we illustrate in Figure 5 that, again, our algorithm demonstrates the ability to learn effectively from various preference signals, including scenarios with only accept (PMPO-A), only reject (PMPO-R), or both accept/reject (PMPO-AR) feedback. These results highlight the versatility of our approach to different preference acquisition settings. The results also underline the critical role of the KL term in enabling learning exclusively from dis-preferred generations (PMPO-R). As predicted by our derivation, a sufficiently high value $\beta > (1 - \alpha)$ is necessary to stabilize learning in this scenario. In contrast, when learning solely from preferred samples (PMPO-A), the algorithm is insensitive to the value of β in terms of stability.

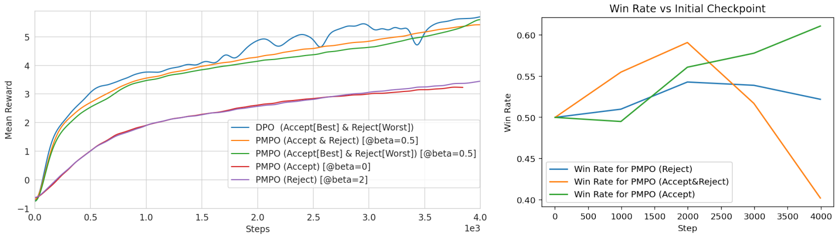

Figure 4: Left: Impact of Combining Accept and Reject Signals - The plot demonstrates the learning progress of PMPO-AR (using both accept and reject signals) compared to PMPO-A and PMPO-R, showcasing faster learning when leveraging both types of feedback in language alignment task and is competitive with DPO. Right: Win-rate when doing A/B comparisons on held-out prompts for PMPO against the base Gemma checkpoint as judged by GPT-4.

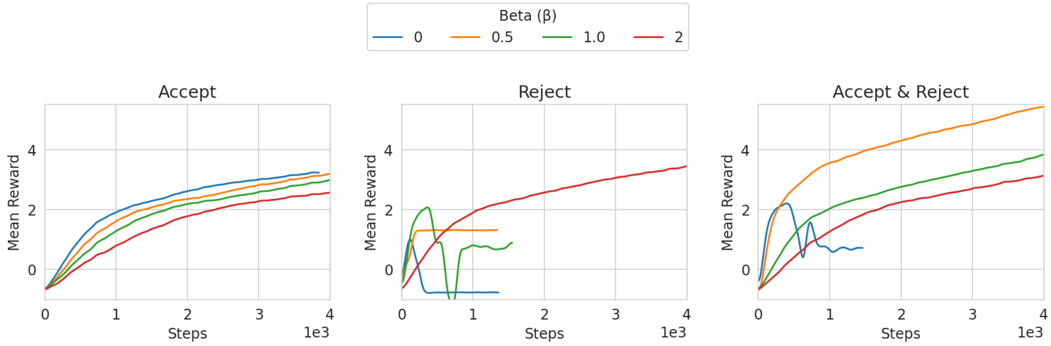

Figure 5: Rewards obtained by the policy at each training step, averaged over the batch and smoothed. Each curve corresponds to a configuration of β specified in the legend. This figure illustrates the ability of PMPO to learn effectively from various preference signals (accept-only, reject-only, or both) in language alignment tasks. highlighting its adaptability to different preference acquisition settings.

6 CONCLUSION

We propose a novel algorithm for policy optimization from preference feedback derived from the perspective of RL as probabilistic inference. Our policy improvement algorithm has a clear and intuitive objective: it maximizes the likelihood of preferred data while minimizing the likelihood of dis-preferred data. We show that doing the latter in a stable way requires a regularization term forcing the policy to stay close to a reference model. This regularization term follows naturally from the derivation. The main advantage of our algorithm over existing preference optimization algorithms such as DPO is that it does not rely on defining/fitting an explicit model of the preferences and can thus use data containing partial preference information; i.e. we can use data where instead of comparisons between samples we only have accept (or only reject) labels and make no further assumptions on their distribution. In a large number of settings, ranging from classical continuous control to modern LLM finetuning tasks, we show that our method is effective at training policies from such binary preference feedback, without requiring a balanced dataset of positive and negative examples.

REFERENCES

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256, 2002.

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. A general theoretical paradigm to understand learning from human preferences. *ArXiv*, abs/2310.12036, 2023.

Edwin V. Bonilla, Shengbo Guo, and Scott Sanner. Gaussian process preference elicitation. In *Neural Information Processing Systems*, 2010.

Craig Boutilier. A pomdp formulation of preference elicitation problems. *Proceedings of the National Conference on Artificial Intelligence*, 05 2002.

Paul Christiano, Jan Leike, Tom B Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *arXiv preprint arXiv:1706.03741*, 2017.

Peter Dayan and Geoffrey E Hinton. Using expectation-maximization for reinforcement learning. *Neural Computation*, 9(2):271–278, 1997.

Shengbo Guo and Scott Sanner. Real-time multiattribute bayesian preference elicitation with pairwise comparison queries. In Yee Whye Teh and Mike Titterington (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 289–296, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL `https://proceedings.mlr.press/v9/guo10b.html`.

Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1):1–18, 2003.

Haozhe Ji, Cheng Lu, Yilin Niu, Pei Ke, Hongning Wang, Jun Zhu, Jie Tang, and Minlie Huang. Towards efficient and exact optimization of language model alignment. *arXiv preprint arXiv:2402.00856*, 2024.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Thomas Lampe, Abbas Abdolmaleki, Sarah Bechtle, Sandy H Huang, Jost Tobias Springenberg, Michael Bloesch, Oliver Groth, Roland Hafner, Tim Hertweck, Michael Neunert, et al. Mastering stacking of diverse shapes with large-scale iterative reinforcement learning on real robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7772–7779. IEEE, 2024.

Alex X. Lee, Coline Devin, Yuxiang Zhou, Thomas Lampe, Konstantinos Bousmalis, Jost Tobias Springenberg, Arunkumar Byravan, Abbas Abdolmaleki, Nimrod Gileadi, David Khosid, Claudio Fantacci, José Enrique Chen, Akhil Raju, Rae Jeong, Michael Neunert, Antoine Laurens, Stefano Saliceti, Federico Casarini, Martin A. Riedmiller, Raia Hadsell, and Francesco Nori. Beyond pick-and-place: Tackling robotic stacking of diverse shapes. *CoRR*, abs/2110.06192, 2021. URL `https://arxiv.org/abs/2110.06192`.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review, 2018. URL `https://arxiv.org/abs/1805.00909`.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*, 2024.

OpenAI. Gpt-4 technical report, 2024. URL `https://arxiv.org/abs/2303.08774`.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL `https://arxiv.org/abs/2203.02155`.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *CoRR*, abs/1910.00177, 2019. URL `http://arxiv.org/abs/1910.00177`.

Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pp. 1607–1612, 2010.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://arxiv.org/abs/2305.18290`.

Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. In *R:SS 2021*, 2013.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. *Advances in Neural Information Processing Systems*, 35:9460–9471, 2022.

Jost Tobias Springenberg, Abbas Abdolmaleki, Jingwei Zhang, Oliver Groth, Michael Bloesch, Thomas Lampe, Philemon Brakel, Sarah Bechtle, Steven Kapturowski, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Offline actor-critic reinforcement learning scales to large models, 2024. URL `https://arxiv.org/abs/2402.05546`.

Gemini Team. Gemini: A family of highly capable multimodal models, 2024a. URL `https://arxiv.org/abs/2312.11805`.

Gemma Team. Gemma 2: Improving open language models at a practical size, 2024b. URL `https://arxiv.org/abs/2408.00118`.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, pp. 5026–5033. IEEE, 2012. ISBN 978-1-4673-1737-5. URL `http://dblp.uni-trier.de/db/conf/iros/iros2012.html#TodorovET12`.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. ISSN 2665-9638. doi: https://doi.org/10.1016/j.simpa.2020.100022. URL `https://www.sciencedirect.com/science/article/pii/S2665963820300099`.

Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, and Nando de Freitas. Critic regularized regression. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7768–7778. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/588cb956d6bbe67078f29f8de420a13d-Paper.pdf`.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric. P Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. Lmsys-chat-1m: A large-scale real-world llm conversation dataset, 2023.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A HYPER PARAMETER TUNING

PMPO introduces two hyperparameters, α and β. However, we found them relatively easy to tune in practice based on the following guidelines:

- $\boldsymbol{\alpha}$: This parameter reflects the relative importance of positive and negative feedback. When both are available, $\alpha = 0.5$ is a reasonable starting point if both types of feedback are equally reliable. Otherwise, can be adjusted to reflect the confidence in each feedback type. In cases with only one type of feedback, is naturally determined by the data.
- $\boldsymbol{\beta}$: This parameter controls the influence of the prior or reference policy. Our experiments suggest that a higher is generally beneficial when learning from negative feedback such that the more is contribution of negative feedback to the policy update (lower α) the more β should be. This can also be motivate from the derivations that KL term is appear as the result of learning from negative feedback. We also find that as long as β is large enough, the algorithm is fairly insensitive to the exact choice of the parameter.

B COMPUTING THE KULLBACK-LEIBLER DIVERGENCE

learning from negative feedback is only feasible with a correct KL term and access to a reference policy π_{ref}. When logits are available using an exact KL computation is ideal. In the absence of logits (Experiment 5.3), we relied on an abundance of unlabelled data to estimate it. This enables us to learn only from negative feedback without access to positive feedback as suggested by our derivations.

Subsequently, we present the derivations for computing KL divergence for LLMs with autoregressive policies in our experiments.

Let's consider a single two-token generation (y_1, y_2) sampled autoregressively from a prompt x. We can factorize the joint distribution associated with such a generation as follows:

$$\pi(y_1, y_2|x) = \pi^1(y_1|x)\pi^2(y_2|x, y_1) \tag{11}$$

Using this factorization for both π_θ and π_{ref} in our KL regularizer, we compute the following:

$$\begin{aligned}
\mathrm{KL}(\pi_{\text{ref}} \| \pi_\theta) &= \mathbb{E}_{\pi_{\text{ref}}} \log \frac{\pi_{\text{ref}}}{\pi_\theta} && (12)\\
&= \mathbb{E}_{\pi^1_{\text{ref}}} \mathbb{E}_{\pi^2_{\text{ref}}} \log \frac{\pi^1_{\text{ref}}\, \pi^2_{\text{ref}}}{\pi^1_\theta\, \pi^2_\theta} && (13)\\
&= \mathbb{E}_{\pi^1_{\text{ref}}} \mathbb{E}_{\pi^2_{\text{ref}}} \log \frac{\pi^1_{\text{ref}}}{\pi^1_\theta} + \mathbb{E}_{\pi^1_{\text{ref}}} \mathbb{E}_{\pi^2_{\text{ref}}} \log \frac{\pi^2_{\text{ref}}}{\pi^2_\theta} && (14)\\
&= \mathbb{E}_{\pi^1_{\text{ref}}} \log \frac{\pi^1_{\text{ref}}}{\pi^1_\theta} + \mathbb{E}_{\pi^1_{\text{ref}}} \mathbb{E}_{\pi^2_{\text{ref}}} \log \frac{\pi^2_{\text{ref}}}{\pi^2_\theta}, && (15)
\end{aligned}$$

where the first term can drop the expectation with respect to $\mathbb{E}_{\pi^2_{\text{ref}}}$ since the integrand $\log \frac{\pi^1_{\text{ref}}}{\pi^1_\theta}$ does not depend on y_2. The first term can be computed analytically because π^1_{ref} and π^1_θ are simply categorical distributions over the entire vocabulary of tokens. The second term, however, is problematic due to the outer expectation $\mathbb{E}_{\pi^1_{\text{ref}}}$, which requires us to integrate a KL over all possible values of y_1. While we can easily compute this KL for any one value of y^1, the integration would be unwieldy and certainly becomes intractable as we extend this to longer sequences y_3, y_4, etc. Luckily, during training we obtain samples $\tilde{y}_1, \tilde{y}_2 \sim \pi_{\text{ref}}$ and so $\tilde{y}_1 \sim \pi^1_{\text{ref}}$, which allows us to use a single-sample Monte Carlo unbiased estimate of the second term such that:

$$\approx \mathbb{E}_{\pi^1_{\text{ref}}} \log \frac{\pi^1_{\text{ref}}}{\pi^1_\theta} + \mathbb{E}_{\pi^2_{\text{ref}}} \log \frac{\pi^2_{\text{ref}}(\cdot|x, \tilde{y}_1)}{\pi^2_\theta(\cdot|x, \tilde{y}_1)}. \tag{16}$$

Extending this to sequences of length L, and dropping the superscripts on the policies as they are autoregressive, we obtain the following approximation:

$$\mathrm{KL}(\pi_{\text{ref}} \| \pi_\theta) \approx \mathbb{E}_{\pi_{\text{ref}}} \log \frac{\pi_{\text{ref}}(\cdot|x)}{\pi_\theta(\cdot|x)} + \sum_{i=1}^{L-1} \mathbb{E}_{\pi_{\text{ref}}} \log \frac{\pi_{\text{ref}}(\cdot|x, \tilde{y}_{1:i})}{\pi_\theta(\cdot|x, \tilde{y}_{1:i})}. \tag{17}$$

where $\tilde{y}_{1:L} \sim \pi_{\text{ref}}$.

C FUNDAMENTAL RESULTS

This section explains precisely why the EM method is a sound approach to optimization. We will present the case of a discrete function. This section relies on classical results in discrete optimization and shows how one can build a simple strictly improving algorithm that maximises a discrete function. For simplicity and clarity, we will optimize a function $f \in \mathbb{R}^{\mathcal{S}}$ mapping elements of $\mathcal{S}$ to real numbers, where $\mathcal{S}$ is a finite set. More precisely, our goal is to find a distribution $\delta \in \Delta_{\mathcal{S}}$ (parameterized policy in RL) that maximises the expectation of f under δ (expected value function in RL):

$$\mathbb{E}_{s\sim\delta}[f(s)] = \sum_{s\in\mathcal{S}} \delta(s) f(s).$$

We recall that a discrete probability distribution $\delta \in \Delta_{\mathcal{S}}$ can be identified as a positive real function $\delta \in \mathbb{R}_+^{\mathcal{S}}$ verifying:

$$\sum_{s\in\mathcal{S}} \delta(s) = 1.$$

To find a good distribution to maximise $\mathbb{E}_{s\sim\delta}[f(s)]$, the algorithm relies on the following results:

- Starting from a distribution $\eta \in \Delta_{\mathcal{S}}$, there exists a unique closed-form argmaximum δ^* of the regularised expectation $\mathbb{E}_{s\sim\delta}[f(s)] - \tau \mathrm{KL}(\delta \mid\mid \eta)$, where $\tau \in \mathbb{R}_+^*$ is a strictly positive real number. We have $\delta^* = \frac{\eta(\cdot)\exp(\tau^{-1} f(\cdot))}{\sum_{s'\in\mathcal{S}} \eta(s')\exp(\tau^{-1} f(s'))}$.
- Unless $\delta^* = \eta$, we have a strict improvement between δ^* and η:

$$\mathbb{E}_{s\sim\delta^*}[f(s)] > \mathbb{E}_{s\sim\eta}[f(s)].$$

We prove these results in the main paper. Those results implies that the following algorithm that starts at $\delta_0 = \eta$ and computes:

$$\forall k \in \mathbb{N}, \delta_{k+1} = \frac{\delta_k(\cdot)\exp(\tau^{-1} f(\cdot))}{\sum_{s'\in\mathcal{S}} \delta_k(s')\exp(\tau^{-1} f(s'))},$$

is strictly monotonically improving until there is $K \in \mathbb{N}$ such that $\delta_{K+1} = \delta_K$:

$$\mathbb{E}_{s\sim\delta_0}[f(s)] < \mathbb{E}_{s\sim\delta_1}[f(s)] < \cdots < \mathbb{E}_{s\sim\delta_K}[f(s)] = \mathbb{E}_{s\sim\delta_{K+1}}[f(s)].$$

In practice, we have a set of learnable weights θ to parameterize a distribution q_θ in order to fit δ_{k+1} and another set of fixed weights μ to parameterize a distribution $q_\mu = \delta_k$. Then, to fit δ_{k+1} the idea is to minimize the following KL divergence (this is the maximisation step):

$$\begin{aligned}
\mathcal{L}(\theta) &= \mathrm{KL}\Big(\frac{q_\mu(\cdot)\exp(\tau^{-1} f(\cdot))}{\sum_{s'\in\mathcal{S}} q_\mu(s')\exp(\tau^{-1} f(s'))} \mid\mid q_\theta\Big), \\
&= \mathbb{E}_{s\sim\frac{q_\mu(\cdot)\exp(\tau^{-1} f(\cdot))}{\sum_{s'\in\mathcal{S}} q_\mu(s')\exp(\tau^{-1} f(s'))}} \left[\log\left(\frac{\frac{q_\mu(\cdot)\exp(\tau^{-1} f(\cdot))}{\sum_{s'\in\mathcal{S}} q_\mu(s')\exp(\tau^{-1} f(s'))}}{q_\theta}\right)\right], \\
&= -\mathbb{E}_{s\sim\frac{q_\mu(\cdot)\exp(\tau^{-1} f(\cdot))}{\sum_{s'\in\mathcal{S}} q_\mu(s')\exp(\tau^{-1} f(s'))}} [\log(q_\theta)] + \mathbb{E}_{s\sim\frac{q_\mu(\cdot)\exp(\tau^{-1} f(\cdot))}{\sum_{s'\in\mathcal{S}} q_\mu(s')\exp(\tau^{-1} f(s'))}} \left[\log\left(\frac{q_\mu(\cdot)\exp(\tau^{-1} f(\cdot))}{\sum_{s'\in\mathcal{S}} q_\mu(s')\exp(\tau^{-1} f(s'))}\right)\right]
\end{aligned}$$

The term $\mathbb{E}_{s\sim\frac{q_\mu(\cdot)\exp(\tau^{-1} f(\cdot))}{\sum_{s'\in\mathcal{S}} q_\mu(s')\exp(\tau^{-1} f(s'))}} \left[\log\left(\frac{q_\mu(\cdot)\exp(\tau^{-1} f(\cdot))}{\sum_{s'\in\mathcal{S}} q_\mu(s')\exp(\tau^{-1} f(s'))}\right)\right]$ does not depend on θ so is irrelevant in the minimization. Using the re-weighting formula $\mathbb{E}_{s\sim\delta}[f(s)] = \mathbb{E}_{s\sim\eta}[\frac{\delta(s)}{\eta(s)} f(s)]$, the minimization problem is equivalent to:

$$\begin{aligned}\mathcal{L}(\theta) &= -\mathbb{E}_{s\sim\frac{q_\mu(\cdot)\exp(\tau^{-1}f(\cdot))}{\sum_{s'\in\mathcal{S}} q_\mu(s')\exp(\tau^{-1}f(s'))}}\left[\log\left(q_\theta\right)\right],\\ &= -\mathbb{E}_{s\sim q_\mu}\left[\frac{\exp(\tau^{-1}f(\cdot))}{\sum_{s'\in\mathcal{S}} q_\mu(s')\exp(\tau^{-1}f(s'))}\log\left(q_\theta\right)\right],\end{aligned}$$

As $\sum_{s'\in\mathcal{S}} q_\mu(s')\exp(\tau^{-1}f(s'))$ is a constant, this is equivalent to minimizing:

$$\mathcal{L}(\theta) = -\mathbb{E}_{s\sim q_\mu}[\exp(\tau^{-1}f(s))\log\left(q_\theta(s)\right)].$$

C.1 EXISTENCE AND UNIQUENESS OF THE REGULARIZED ARGMAXIMUM

For completeness, we briefly recall the proof of existence and uniqueness of the argmaximum of the following regularized criterion that can also be found in the work of Rafailov et al. (2023):

$$\begin{aligned}\mathcal{L}_\tau(\delta) &= \mathbb{E}_{s\sim\delta}[f(s)] - \tau\mathrm{KL}(\delta\,||\,\eta),\\ &= \sum_{s\in\mathcal{S}}\delta(s)f(s) - \tau\mathrm{KL}(\delta\,||\,\eta).\end{aligned}$$

Now, if we define the softmax probability $\delta^*\in\Delta_{\mathcal{S}}$ as:

$$\forall s\in\mathcal{S}, \delta^*(s) = \frac{\eta(s)\exp(\tau^{-1}f(s))}{\sum_{s'\in\mathcal{S}}\eta(s')\exp(\tau^{-1}f(s'))},$$

then, under the previous definitions, we have the following results:

$$\delta^* = \arg\max_{\delta\in\Delta_{\mathcal{S}}}\mathcal{L}_\tau(\delta)$$

Proof.

$$\begin{aligned}\frac{\mathcal{L}_\tau(\delta)}{\tau} &= \sum_{s\in\mathcal{S}}\delta(s)\frac{f(s)}{\tau} - \mathrm{KL}(\delta\,||\,\eta),\\ &= \sum_{s\in\mathcal{S}}\delta(s)\frac{f(s)}{\tau} - \sum_{s\in\mathcal{S}}\delta(s)\log\big(\frac{\delta(s)}{\eta(s)}\big),\\ &= \sum_{s\in\mathcal{S}}\delta(s)\big(\frac{f(s)}{\tau} - \log\big(\frac{\delta(s)}{\eta(s)}\big)\big),\\ &= \sum_{s\in\mathcal{S}}\delta(s)\big(\log\big(\exp(\tau^{-1}f(s))\big) - \log\big(\frac{\delta(s)}{\eta(s)}\big)\big),\\ &= \sum_{s\in\mathcal{S}}\delta(s)\big(\log\big(\frac{\eta(s)\exp(\tau^{-1}f(s))}{\delta(s)}\big)\big),\\ &= \sum_{s\in\mathcal{S}}\delta(s)\big(\log\big(\frac{\eta(s)\exp(\tau^{-1}f(s))\frac{\sum_{s'\in\mathcal{S}}\eta(s')\exp(\tau^{-1}f(s'))}{\sum_{s'\in\mathcal{S}}\eta(s')\exp(\tau^{-1}f(s'))}}{\delta(s)}\big)\big),\\ &= \sum_{s\in\mathcal{S}}\delta(s)\big(\log\big(\frac{\frac{\eta(s)\exp(\tau^{-1}f(s))}{\sum_{s'\in\mathcal{S}}\eta(s')\exp(\tau^{-1}f(s'))}}{\delta(s)}\big)\big) + \sum_{s\in\mathcal{S}}\delta(s)\log\big(\sum_{s'\in\mathcal{S}}\eta(s')\exp(\tau^{-1}f(s'))\big),\\ &= \sum_{s\in\mathcal{S}}\delta(s)\big(\log\big(\frac{\delta^*(s)}{\delta(s)}\big)\big) + \log\big(\sum_{s'\in\mathcal{S}}\eta(s')\exp(\tau^{-1}f(s'))\big),\\ &= -\mathrm{KL}(\delta\,||\,\delta^*) + \log\big(\sum_{s\in\mathcal{S}}\eta(s)\exp(\tau^{-1}f(s))\big).\end{aligned}$$

By definition of the KL, we now that $\delta^* = \arg\max_{\delta \in \Delta_{\mathcal{S}}} \left[- \text{KL}(\delta \mid\mid \delta^*) \right]$ and as:

$$- \text{KL}(\delta \mid\mid \delta^*) = \frac{\mathcal{L}_\tau(\delta)}{\tau} - \log\big(\sum_{s \in \mathcal{S}} \eta(s) \exp(\tau^{-1} f(s))\big) \tag{18}$$

where $\log\big(\sum_{s \in \mathcal{S}} \eta(s) \exp(\tau^{-1} f(s))\big)$ is a constant (does not depend on δ) and τ a positive multiplicative term, then $-\text{KL}(\delta \mid\mid \delta^*)$ and $\mathcal{L}_\tau(\delta)$ share the same argmaximum. This concludes the proof. $\square$

The fact that we have:

$$\delta^* = \frac{\eta(\cdot) \exp(\tau^{-1} f(\cdot))}{\sum_{s' \in \mathcal{S}} \eta(s') \exp(\tau^{-1} f(s'))} = \arg\max_{\delta \in \Delta_{\mathcal{S}}} \mathcal{L}_\tau(\delta),$$

implies by simply replacing δ by δ^* in equation (18) that:

$$-\text{KL}(\delta^* \mid\mid \delta^*) = \frac{\mathcal{L}_\tau(\delta^*)}{\tau} - \log\big(\sum_{s \in \mathcal{S}} \eta(s) \exp(\tau^{-1} f(s))\big).$$

As $\text{KL}(\delta^* \mid\mid \delta^*) = 0$, we have:

$$\mathcal{L}_\tau(\delta^*) = \max_{\delta \in \Delta_{\mathcal{S}}} \mathcal{L}_\tau(\delta) = \tau \log\big(\sum_{s \in \mathcal{S}} \eta(s) \exp(\tau^{-1} f(s))\big). \tag{19}$$

This result is often expressed in term of an inequality that says that the logsumexp is a majorant of the regularized expectation:

$$\forall \delta \in \Delta_{\mathcal{S}}, \tau \log\big(\sum_{s \in \mathcal{S}} \eta(s) \exp(\tau^{-1} f(s))\big) \geq \sum_{s \in \mathcal{S}} \delta(s) f(s) - \tau \text{KL}(\delta \mid\mid \eta). \tag{20}$$

C.2 PROOF OF IMPROVEMENT.

We use the same notations as in the previous section, and our goal is to show that using the distribution δ^* instead of η strictly increases the expected value of our function f:

$$\mathbb{E}_{s \sim \delta^*}[f(s)] > \mathbb{E}_{s \sim \eta}[f(s)],$$

unless $\delta^* = \eta$. This means that we can confidently replace η by δ^* for the next iteration of the algorithm.

Proof. From Eq.(19), we have:

$$\tau \log\big(\sum_{s \in \mathcal{S}} \eta(s) \exp(\tau^{-1} f(s))\big) = \mathcal{L}_\tau(\delta^*) = \mathbb{E}_{s \sim \delta^*}[f(s)] - \tau \text{KL}(\delta^* \mid\mid \eta).$$

Using Jensen inequality we have:

$$\begin{aligned} \tau \log\big(\sum_{s \in \mathcal{S}} \eta(s) \exp(\tau^{-1} f(s))\big) &\geq \tau \sum_{s \in \mathcal{S}} \eta(s) \log\big(\exp(\tau^{-1} f(s))\big), \\ &= \tau \sum_{s \in \mathcal{S}} \eta(s) \tau^{-1} f(s), \\ &= \sum_{s \in \mathcal{S}} \eta(s) f(s) = \mathbb{E}_{s \sim \eta}[f(s)]. \end{aligned}$$

This implies that:

$$\mathbb{E}_{s \sim \delta^*}[f(s)] - \tau \text{KL}(\delta^* \mid\mid \eta) \geq \mathbb{E}_{s \sim \eta}[f(s)].$$

As $\tau \text{KL}(\delta^* \mid\mid \eta)$ is strictly positive unless $\delta^* = \eta$, we conclude that we have strict improvement of the algorithm unless $\delta^* = \eta$ which means in this case that the method has converged. $\square$

D ADDITIONAL LARGE LANGUAGE MODEL EXPERIMENTS

The pre-trained reward model used for our LLM experiments is not perfect and is subject to reward hacking (Skalse et al., 2022). Generally, reward models only capture superficial features of the generation such as the length or the formatting. Therefore, instead of only looking at the reward scores, we need other ways to evaluate the quality of our policy, that are independent from the reward bias. We choose to use the LMSYS arena hard benchmark (Li et al., 2024) as an evaluation set. It is composed of 500 prompts. We decided to evaluate 3 different runs, PMPO-R (only reject), PMPO-AR (accept and reject) and PMPO-A (only accept) against the shared initial checkpoint. Each of the data-point in Fig. 6 corresponds to the win-rate ratio of a method at a given training step against the initial checkpoint. The win-rate ratio between two checkpoints is obtained by scraping the 500 prompts of the evaluation set to obtain 500 generations for each checkpoint. Then each prompt and couple of generations is sent to an LLM judge (gpt4 in this case) which decides what is the best generation for this given prompt. The win-ratio is simply the number of times one checkpoint wins over the other. In Fig. 6, we can see that PMPO-AR is the quickest to "hack the reward", it reaches a good performance but then in the middle of training its start hacking the reward and learns a pathological behaviour that makes it performs worse on the independent benchmark. This phenomenon has been observed consistently in RLHF.

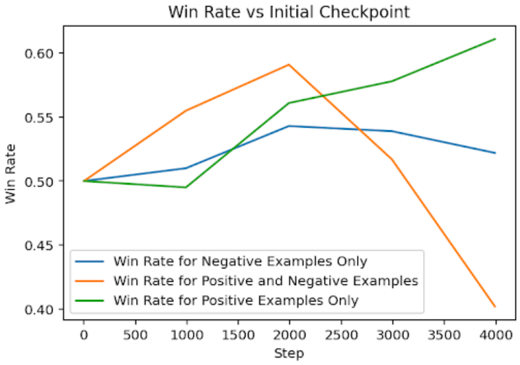

Figure 6

E LINK BETWEEN IPO AND PMPO

In this section, we draw a parallel between the IPO and the PMPO losses. For a dataset of triplets $(x^i, y_a^i, y_r^i)_{i=1}^N$, the IPO loss is:

$$\mathcal{L}_{\texttt{IPO}}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left[-\log\left(\pi_\theta(y_a^i|x^i)\right) + \log\left(\pi_\theta(y_r^i|x^i)\right) + \beta\left(\log\left(\frac{\pi_\theta(y_a^i|x^i)}{\pi_{\texttt{ref}}(y_a^i|x^i)}\right) - \log\left(\frac{\pi_\theta(y_r^i|x^i)}{\pi_{\texttt{ref}}(y_r^i|x^i)}\right)\right)^2\right].$$

The IPO loss is composed of two terms the policy optimisation term:

$$\mathcal{P}_{\texttt{IPO}}(\theta) = \frac{1}{N}\sum_{i=1}^{N} -\log\left(\pi_\theta(y_a^i|x^i)\right) + \log\left(\pi_\theta(y_r^i|x^i)\right),$$

and the policy regularisation term:

$$\mathcal{R}_{\texttt{IPO}}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left[\left(\log\left(\frac{\pi_\theta(y_a^i|x^i)}{\pi_{\texttt{ref}}(y_a^i|x^i)}\right) - \log\left(\frac{\pi_\theta(y_r^i|x^i)}{\pi_{\texttt{ref}}(y_r^i|x^i)}\right)\right)^2\right].$$

When $\alpha = \frac{1}{2}$, the PMPO loss can be written as:

$$\mathcal{L}_{\texttt{PMPO}}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left[-\frac{1}{2}\log\left(\pi_\theta(y_a^i|x^i)\right) + \frac{1}{2}\log\left(\pi_\theta(y_r^i|x^i)\right) + \beta\mathrm{KL}(\pi_{\texttt{ref}}(.|x^i)||\pi_\theta(.|x^i))\right].$$

The PMPO loss is also composed of two terms the policy optimisation term:

$$\mathcal{P}_{\texttt{PMPO}}(\theta) = \frac{1}{N}\sum_{i=1}^{N} -\frac{1}{2}\log\left(\pi_\theta(y_a^i|x^i)\right) + \frac{1}{2}\log\left(\pi_\theta(y_r^i|x^i)\right),$$

and the policy regularisation term:

$$\mathcal{R}_{\texttt{PMPO}}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left[\texttt{KL}(\pi_{\texttt{ref}}(.|x^i)||\pi_\theta(.|x^i))\right].$$

Therefore, the policy optimisation terms of the IPO $\mathcal{P}_{\texttt{IPO}}(\theta)$ and PMPO $\mathcal{P}_{\texttt{PMPO}}(\theta)$ losses are identical at a constant factor. Now we are going to create a connection between the IPO and PMPO regularisation terms when y_a^i and y_r^i are sampled from $\pi_{\texttt{ref}}(.|x^i)$. The first thing to remark is that $\left(\log\left(\frac{\pi_\theta(y_a^i|x^i)}{\pi_{\texttt{ref}}(y_a^i|x^i)}\right) - \log\left(\frac{\pi_\theta(y_r^i|x^i)}{\pi_{\texttt{ref}}(y_r^i|x^i)}\right)\right)^2$ is an unbiased estimate of $E_{Y,Y'\sim\pi_{\texttt{ref}}(.|x^i)}\left[\left(\log\left(\frac{\pi_\theta(Y|x^i)}{\pi_{\texttt{ref}}(Y|x^i)}\right) - \log\left(\frac{\pi_\theta(Y'|x^i)}{\pi_{\texttt{ref}}(Y'|x^i)}\right)\right)^2\right]$. Then, we remind the reader that the variance of a random variable X under distribution μ verifies:

$$\texttt{VAR}_{X\sim\mu}[X] = \frac{1}{2}\mathbb{E}_{X,X'\sim\mu}[(X-X')^2],$$

where X and X' are independent variables with distribution μ. This means that $\left(\log\left(\frac{\pi_\theta(y_a^i|x^i)}{\pi_{\texttt{ref}}(y_a^i|x^i)}\right) - \log\left(\frac{\pi_\theta(y_r^i|x^i)}{\pi_{\texttt{ref}}(y_r^i|x^i)}\right)\right)^2$ is an unbiased estimate of $2\texttt{VAR}_{Y\sim\pi_{\texttt{ref}}(.|x^i)}\left[\log\left(\frac{\pi_\theta(Y|x^i)}{\pi_{\texttt{ref}}(Y|x^i)}\right)\right]$.

Therefore the expectation (over $\pi_{\texttt{ref}}$) of the IPO regularization term is:

$$\mathbb{E}_{\pi_{\texttt{ref}}}\left[\mathcal{R}_{\texttt{IPO}}(\theta)\right] = \frac{2}{N}\sum_{i=1}^{N}\texttt{VAR}_{Y\sim\pi_{\texttt{ref}}(.|x^i)}\left[\log\left(\frac{\pi_\theta(Y|x^i)}{\pi_{\texttt{ref}}(Y|x^i)}\right)\right].$$

As $\texttt{VAR}_{X\sim\mu}[X] = \texttt{VAR}_{X\sim\mu}[-X]$, we also have:

$$\mathbb{E}_{\pi_{\texttt{ref}}}\left[\mathcal{R}_{\texttt{IPO}}(\theta)\right] = \frac{2}{N}\sum_{i=1}^{N}\texttt{VAR}_{Y\sim\pi_{\texttt{ref}}(.|x^i)}\left[\log\left(\frac{\pi_{\texttt{ref}}(Y|x^i)}{\pi_\theta(Y|x^i)}\right)\right].$$

This is in contrast with the regularization term of PMPO:

$$\mathcal{R}_{\texttt{PMPO}}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left[\texttt{KL}(\pi_{\texttt{ref}}(.|x^i)||\pi_\theta(.|x^i))\right] = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{Y\sim\pi_{\texttt{ref}}(.|x^i)}\left[\log\left(\frac{\pi_{\texttt{ref}}(Y|x^i)}{\pi_\theta(Y|x^i)}\right)\right].$$

So the difference between PMPO and IPO is that PMPO will minimize the expectation of the log ratio between the reference policy and the online policy whereas IPO will minimize the variance of the same quantity.

F DERIVATION FROM THE PERSPECTIVE OF RL AS INFERENCE

Consider a decision-making scenario characterized by the following elements:

Stationary State Distribution: A stationary distribution $\mu(x)$ describes the probability of encountering different states or contexts x. We typically have access to samples from this distribution.

Policy: A policy $\pi(y|x)$ dictates the probability of selecting an action/outcome y given a state x.

Preference Information: We have access to knowledge for driving the likelihood of preference or dis-preference toward actions taken by a policy $\pi(y|x)$ according to a known or latent objective. This information takes one of or these forms:

- **Preference Probabilities** $q(I=1|y\sim\pi, x\sim\mu)$, where I is a binary indicator with $I=1$ representing preference. when only a reward or q-function is available this probability can be derived as $q(I=1|y\sim\pi, x\sim\mu) \propto \exp(Q(y,x)/\eta)$.

- **Dis-preference Probabilities** $q(I=0|y \sim \pi, x \sim \mu)$, where $I=0$ denotes dis-preference. when only a reward or q-function is available this probability can be derived as $q(I=1|y \sim \pi, x \sim \mu) \propto \exp(-Q(y,x)/\eta)$.

Objective: derive an optimal policy to give a highest probability to the preferred outcomes:

$$\max_{\theta} \log p_{\theta}(I=1) = \log \int \mu(x) \int \pi_{\theta}(y|x) q(I=1|y,x)\, dy\, dx$$

G BACKGROUND:MAXIMISE PREFERRED OUTCOMES

We start with the stated objective:

$$\max_{\pi} \log p_{\pi}(S) = \log \int \mu(x) \int \pi_{\theta}(y|x) q(I=1|y,x)\, dy\, dx$$

This objective can be solved by EM algorithm to iterativly creating a tight lower bound on the current estimate of the objective $log p_{\texttt{ref}}(I=1)$ and then we optimise the objective. By repeating these two steps it's guaranteed that algorithm will converge. In order to do so we use the following identity:

$$\log p(X) = \int q(Z) \log \frac{p(X,Z)}{q(Z)} + \mathrm{KL}(q(Z)|p(Z|X))$$

where the first term is a lower bound because KL is always positive. Now considering the states x and outcome y as latent variables:

$$\log p_{\theta}(I=1) = \int \mu(x) \int q(y|x) \log \frac{p_{\theta}(I=1,y|x)}{q(y|x)} + \int \mu(x)\, \mathrm{KL}(q(y|x)|p_{\theta}(y|I=1,x))$$

G.1 E-STEP

In E-step our goal is to choose the variational distribution $q(y|x)$ such that lower bound on

$$\log p_{\texttt{ref}}(I=1)$$

is as tight as possible which is the case when KL term in equation above is zero at current estimate of policy $\pi_{\texttt{ref}}$. This simply leads to $q(y|x) = p_{\texttt{ref}}(y|I=1,x)$ or

$$q(y|x) = \frac{\pi_{\texttt{ref}}(y|x) p_{\texttt{ref}}(I=1|x,y)}{p_{\texttt{ref}}(I=1|x)}$$

where $p_{\texttt{ref}}(I=1|x) = \int \pi_{\texttt{ref}}(y|x) p_{\texttt{ref}}(I=1|x,y)$ is a normalizer for a given state x. Now in some cases such as data collected from human feedback we directly have access to sampled $q(y|x)$ according to equation above where if a y_a is accepted $q(y|x)$ would be a distribution that for a state x gives probability of 1 to y_a and zero to every other samples, i.e, $q(y_a \sim \pi_{\texttt{ref}}|x) = 1$ and $q(y_{!a} \sim \pi_{\texttt{ref}}|x) = 0$. In case we have multiple ranked outcomes for a x we again can simply assign probabilities accordingly and obtain $q(y|x)$ immediately.

Relation to Reinforcement Learning

E-step of preference objective coincides with KL regularized RL objective when a reward function or Q-function is given instead of preferences. In such case we can estimate preference function $p_{\texttt{ref}}(I=1,x,y)$ by using a transformation on reward function. Now if we choose $p_{\texttt{ref}}(I=1,x,y) \propto \exp(\frac{Q(x,y)}{\eta})$ and substituting this in expected KL term above we obtain the KL regularised RL objective, i.e,

$$\max_{q} \int \mu(x) \int q(y|x) Q(y,x) - \eta \int \mu(x)\, \mathrm{KL}(q(y|x)|\pi_{\texttt{ref}}(y|x))$$

This motivates the original preference optimization objective and implies that this objective can be seen as optimising for a latent Q-function or reward.

G.2 M-STEP

In E-step we found non-parametric variational distributions $q(y|x)$ for $x \sim \mu(x)$ that gives higher probability to preferred actions sampled from $\pi_{\texttt{ref}}$. E step can be seen as sample based policy improvement $\pi_{\texttt{ref}}$ wrt preferences. In M-step we optimise the lower bound to obtain a new distribution, i.e,

$$\max_{\theta} \int \mu(x) \int q(y|x) \log \frac{p_\theta(I=1, y|x)}{q(y|x)}$$

where

$$q(y|x) = \frac{\pi_{\texttt{ref}}(y|x) p(I=1|x,y)}{\int \pi_{\texttt{ref}}(y|x) p(I=1|y,x)}$$

according to derivations in E-step. after rearranging the terms and removing the terms that do not depends on θ we get

$$\min_{\theta} \int \mu(x) \operatorname{KL}(q(y|x)|\pi_\theta(y|x))$$

which effectively is a weighted maximum likelihood objective to fit improved non-parametric policies. In practice we also add a constraint to avoid moving too far from from the $\pi_{\texttt{ref}}$, i.e,

$$\min_{\theta} \int \mu(x) \operatorname{KL}(q(y|x)|\pi_\theta(y|x)) + \beta \int \mu(x) \operatorname{KL}(\pi_{\texttt{ref}}(y|x)|\pi_\theta|(y|x))$$

Note that this constraint can be motivated from different angles such as a prior on θ or partial EM and multi objective perspective. In terms of the direction of KL, this direction of KL is aligned with the direction of KL for fitting the distributions.

H INCORPORATING DIS-PREFERRED OUTCOMES FOR POLICY OPTIMIZATION

In the previous section we showed how optimising for preferred outcomes can lead us to useful policy update rules. In order to motivate this objective further, we showed that typical reinforcement learning objective coincides with the objective in E-Step. However we only incorporated preferred outcomes $q(I=1)$ to update the policy. Now we ask this question how we can incorporate dis-preferred outcomes to directly optimize the policy? In order to do that we first write down the policy optimisation objective we derived in terms of preferences:

$$\min_{\theta} \int \mu(x) \operatorname{KL}(q(y|x)|\pi_\theta(y|x))$$

where

$$q(y|x) = \frac{\pi_{\texttt{ref}}(y|x) p(I=1|x,y)}{\int \pi_{\texttt{ref}}(y|x) p(I=1|y,x)}.$$

We also know that $p(I=1|x,y) = 1 - p(I=0|x,y)$. This is correct as $p(I)$ is a probability function over a binary random variable I. Therefore the variational distribution can be re-written in terms of dis preferences, i.e,

$$q(y|x) = \frac{\pi_{\texttt{ref}}(y|x)(1 - p(I=0|x,y))}{\int \pi_{\texttt{ref}}(y|x)(1 - p(I=0|x,y))}.$$

substituting this in the KL minimization term above we get:

$$\min_{\theta} - \int \mu(x) \int \frac{1}{Z(x)} \pi_{\texttt{ref}}(y|x) \log \pi_\theta(y|x) + \int \mu(x) \int \frac{1}{Z(x)} \pi_{\texttt{ref}}(y|x) p(I=0|x,y) \log \pi_\theta(y|x)$$

where $Z(x) = \int \pi_{\texttt{ref}}(y|x)(1 - p(I = 0|x, y))$ is a normalizer term. ‘Now Now we can write the final policy update rule in terms of dis-preferences:

$$\min_\theta - \int \mu(x)\,\mathrm{KL}(t(y|x)|\pi_\theta(y|x)) + \beta \int \mu(x)\,\mathrm{KL}(\pi_{\texttt{ref}}(x|y)|\pi_\theta(x|y))$$

where $t(y|x) \propto \pi_{\texttt{ref}}(y|x)p(I = 0|x, y)$ and β is a positive tuning parameters that . β must be sufficiently large otherwise as we will show in the experiment, it wont work in practice. This update rule intuitively goes away from dis-preferences while stay close to the reference policy.

I FINAL UPDATE RULE: LEVERAGING BOTH PREFERENCES AND DIS-PREFERENCES

After putting things together we get the following update rule that maximises the preferred outcomes and minimizes the preferred outcomes, i.e,

$$\max_\theta \alpha \log q_\theta(I = 1) + (1 - \alpha)\log(1 - q_\theta(I = 0))$$

After repeating the EM steps to optimise this objective we get the following update rule:

$$\pi_{\text{imp}} = \arg\min_\theta \alpha \int \mu(x)\,\mathrm{KL}(q(y|x)|\pi_\theta(y|x)) + \beta \int \mu(x)\,\mathrm{KL}(\pi_{\text{ref}}|\pi_\theta) - (1-\alpha) \int \mu(x)\,\mathrm{KL}(t(y|x)|\pi_\theta(y|x))$$

where q is a distribution modified wrt preference probabilities resulting from $\pi_{\texttt{ref}}$ weighted by preference probabilities, i.e,

$$q(y|x) \propto \pi_{\texttt{ref}}(y|x)p(I = 1|x, y)$$

or in case of access to q-function or reward function

$$q(y|x) \propto \pi_{\texttt{ref}}(y|x)\exp(\frac{Q(x, y)}{\eta})$$

and t is a distribution modified wrt dis-preference probabilities resulting from $\pi_{\texttt{ref}}$ weighted by dis-preference probabilities, i.e,

$$t(y|x) \propto \pi_{\texttt{ref}}(y|x)p(I = 0|x, y)$$

or in case of access to q-function or reward function

$$t(y|x) \propto \pi_{\texttt{ref}}(y|x)\exp(\frac{-Q(x, y)}{\eta})$$

Intuitively this objective gets closer to preferred distribution $q(y|x)$ and gets away from dis-preferred distribution $t(y|x)$ while staying close to reference policy.

In simplest case when we have access to rejected and/or accepted actions $\{{y_a}^i \sim \pi_{\text{ref}}, {y_r}^i \sim \pi_{\text{ref}}, x^i \sim \mu(x)\}_{i=1...N}$ (In case of Q-function we simply sample 2 actions and the action with higher Q-value is accepted and the one with lower q-value is rejected). The above general update rule will simplify to:

$$\pi_{\text{imp}} = \arg\max_\theta \alpha\Sigma_i \log \pi_\theta(y_a^i|x^i) - (1-\alpha)\Sigma_i \log \pi_\theta(y_r^i|x^i) - \beta\Sigma_i\,\mathrm{KL}(\pi_{\text{ref}}|\pi_\theta)$$

where π_θ is initialized with π_{ref}.

Modality	Dimensions
Arm joint angles	7 x 3
Arm joint velocities	7 x 3
Arm joint torque	7 x 3
Gripper motor angle	1 x 3
Gripper motor velocity	1 x 3
Gripper grasp sensor	1 x 3
Cartesian tool center point pose	7 x 3
Cartesian wrist endpoint velocity	6 x 3
Cartesian wrist endpoint force	3 x 3
Cartesian wrist endpoint torque	3 x 3
Back left basket camera	80 x 80
Front left basket camera	80 x 80
Front right basket camera	80 x 80

Table 2: Observations given to the agent in the RGB Stacking benchmark. Note that for all observations except the camera images, a history of 3 time steps is provided, resulting in the x3 dimensionalities.

J BENCHMARKS

J.1 CONTROL SUITE

The DeepMind Control Suite (Tunyasuvunakool et al., 2020) is a collection of benchmark tasks implemented in the MuJoCo simulator (Todorov et al., 2012). The suite includes a variety of embodiments of different complexity and action dimensionality. For each of these embodiments, there are multiple tasks implemented, each of which defines a single reward function. Example images for some of the domains are shown in Figure 7.

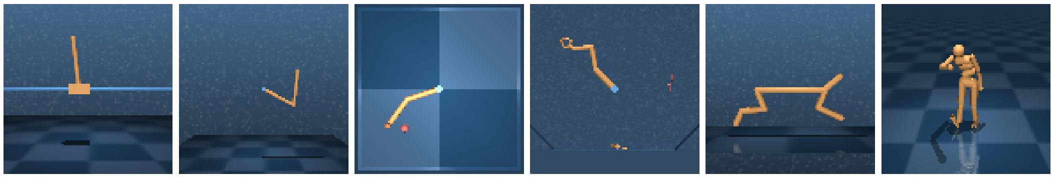

Figure 7: Example domains in the Control Suite. From left to right: Cartpole, Acrobot, Reacher, Manipulator, Cheetah, Humanoid

J.2 RGB STACKING

The RGB Stacking benchmark (Lee et al., 2021) is a robotics task involving a Rethink Sawyer robot arm outfitted with a Robotiq 2F-85 gripper, as well as a basket containing a number of parameterized geometric shapes in red, green and blue color. See Figure 8 for an illustration. The goal of this task is to have the robot arrange the shapes into varying arrangements, such as stacking one on top of another or building a tower. The policy only provides proprioception information and the images from three cameras surrounding the basket; there is no explicit tracking of the objects' relative positions, or their identities. Specifically, the agent is provided the observations listed in Table 2. Thus the task's challenge lies in forcing the agent to learn a control policy directly from vision, and recognizing which objects are in the workspace, since their different geometric properties demand different manipulation strategies.

The task is implemented in the MuJoCo physics simulator (Todorov et al., 2012). Using the ground truth positions of the objects' positions (which is not provided to the agent), several reward terms are calculated, including stacking two objects, stacking three objects, and building a pyramid, for each object permutation. Details of these rewards can be found in Springenberg et al. (2024).

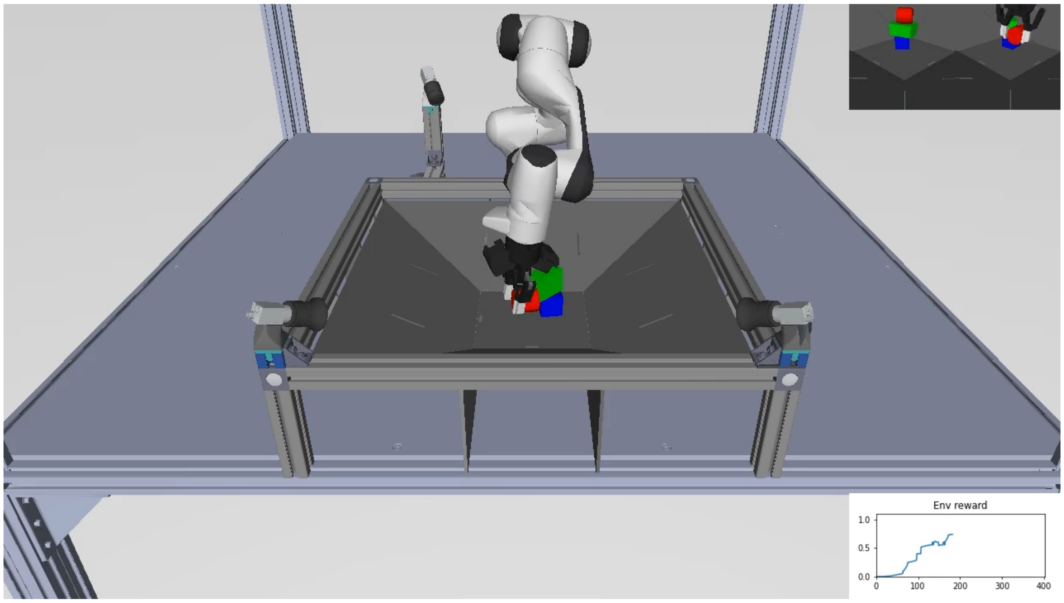

Figure 8: Illustration of the simulated RGB Stacking domain. Top right corner: goal image (left) and current observation (right) for the front left camera, both as provided to the agent. Bottom: reward trace for the dense triple-stacking objective.