Replacing Implicit Regression with Classification in Policy Gradient Reinforcement Learning

Josiah P. Hanna, Brahma S. Pavse, and Abhinav Harish
Computer Sciences Department
The University of Wisconsin – Madions
jphanna@cs.wisc.edu, pavse@wisc.edu, aharish2@wisc.edu

Abstract

Stochastic policy gradient methods are a fundamental class of reinforcement learning algorithms. When using these algorithms for continuous control, it is common to parameterize the policy using a Gaussian distribution. In this paper, we show that the policy gradient with Gaussian policies can be viewed as the gradient of a weighted least-squares objective function. That is, policy gradient algorithms are implicitly implementing a form of regression. Several recent works have shown that reformulating regression problems as classification problems can improve learning. Inspired by these works, we investigate whether replacing this implicit regression with classification can improve the data efficiency and stability of policy learning. We introduce a novel policy gradient surrogate objective for softmax policies over a discretized action space. This surrogate objective uses a form of cross-entropy loss to replace the implicit least-squares loss found in the surrogate loss for Gaussian policies. We extend prior theoretical analysis of this loss to our policy gradient surrogate objective and provide experiments showing that this novel loss improves the data efficiency of stochastic policy gradient learning.

1 Introduction

Stochastic policy gradient algorithms are a fundamental class of reinforcement learning (RL) algorithms. In their simplest form, the learning agent runs its current policy to collect data in the form of state, action, and reward transitions to produce a dataset of $(s_i, a_i, \hat{A}_i)$ where $\hat{A}_i$ is an estimate of the *advantage* of taking action a_i in state s_i. The learning agent then updates its parameterized and differentiable stochastic policy with a step of gradient ascent on the expected cumulative reward objective. The gradient update increases the log-likelihood of each observed action in proportion to the advantage of that action.

Following Peters and Schaal (2007), we observe that the policy gradient update can be viewed as implicitly optimizing a weighted supervised learning loss function. We particularly focus on the case of continuous control with Gaussian policies, in which case we will show that the policy gradient matches the gradient of a weighted least-squares loss function. In this sense, we say that policy gradient algorithms are implicitly implementing (weighted) regression.

A growing body of research (discussed in Appendix C) supports the claim that reformulating regression problems as classification problems can boost task performance in supervised regression. Of particular relevance to this work, Imani and White (2018) introduced a form of cross-entropy loss for regression and showed that it boosts regression accuracy compared to the commonly used squared loss. Subsequently, Farebrother et al. (2024) adopted this *histogram loss* in place of the squared loss for value-based RL algorithms and found that the approach unlocked new levels of scalability in a wide variety of RL benchmarks. Motivated by these prior works, in this paper, we reformulate the

implicit regression of stochastic policy gradient RL for continuous action domains as classification. Specifically, this work aims to answer the question:

Does replacing the least-squares loss and Gaussian policies with a cross-entropy loss and softmax policies over discretized actions improve the data efficiency of stochastic policy gradient algorithms?

In answering this question, we make the following contributions: 1) we introduce a novel policy gradient surrogate loss that recasts the implicit regression toward continuous actions as classification of discrete actions, 2) building on (Imani and White, 2018), we show that the loss we introduce will have a smaller bound on the gradient norm compared to the surrogate loss for Gaussian policies, implying that the new loss is easier to optimize, 3) we empirically investigate the use of cross-entropy losses and softmax policies as an alternative to widely-used Gaussian policies within stochastic policy gradient algorithms and find that our reformulation leads to increased data-efficiency, more stable learning, and increased final performance.

2 Preliminaries

In this section, we introduce RL notation, stochastic policy gradient learning, and introduce the histogram regression loss.

2.1 Reinforcement Learning

We formalize an RL agent's task environment as a finite-horizon, episodic *Markov decision process* (MDP) with state set $\mathcal{S}$, action set $\mathcal{A}$, transition function, $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, discount factor γ, and initial state distribution d_0 (Puterman, 2014). The agent follows a *policy*, $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, which is a function mapping states to probability distributions over possible actions. Given a policy and task environment, interaction begins at time $t = 0$ in some initial state ($s_0 \sim d_0$) and then proceeds with the agent selecting actions according to its policy ($a_t \sim \pi(\cdot|s_t)$) and the environment responding with a reward, $r_t = r(s_t, a_t)$, and transitioning to a next state ($s_{t+1} \sim p(\cdot|s_t, a_t)$). Interaction continues until the agent reaches a terminal state, at which point, the agent returns to a new initial state and the process begins again. The result of this interaction is a *trajectory*, $h \coloneqq (s_0, a_0, r_0, s_1, ..., s_T, a_T, r_T)$.

We measure policy performance by the expected discounted return in a given MDP:

$$J(\pi) \coloneqq \mathbf{E}[\sum_{t=0}^{T} \gamma^t R_t | H \sim \pi]$$

where $H = (S_0, A_0, R_0, ...S_T, A_T, R_T)$ is a random variable representing a trajectory and $H \sim \pi$ denotes sampling H by running π for one episode. In RL, the transition and reward functions of the task MDP are unknown. RL algorithms are designed to collect trajectory data from the task MDP and use this data to return a policy, $\pi^* \in \arg\max_\pi J(\pi)$.

2.2 Policy Gradient Reinforcement Learning

In policy gradient reinforcement learning, the agent's policy is parameterized by a vector, θ, and the policy is differentiable with respect to these parameters. Policy gradient RL algorithms optimize the policy through gradient ascent over θ to maximize $J(\pi_\theta)$. The gradient of the $J(\theta)$ with respect to θ, or *policy gradient*, is typically expressed as:

$$\nabla_\theta J(\pi_\theta) \propto \mathbb{E}_{\mathbf{s}\sim d_{\pi_\theta}, \mathbf{a}\sim\pi_\theta(\cdot|\mathbf{s})} \left[A^{\pi_\theta}(\mathbf{s}, \mathbf{a}) \nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s}) \right],$$

where $A^{\pi_\theta}(\mathbf{s}, \mathbf{a})$ is the *advantage* of choosing action $\mathbf{a}$ in state $\mathbf{s}$ and quantifies the extra expected reward that will be obtained when taking $\mathbf{a}$ instead of sampling an action from π_θ in $\mathbf{s}$, and $d_{\pi_\theta} : \mathcal{S} \to [0, 1]$ is the expected distribution of states that will be seen when running π_θ in the task MDP.[1]

2.3 Histogram Losses for Regression

Supervised regression problems are usually formulated using the least-squares loss function:

$$\mathcal{L}_{\texttt{LS}}(\theta) \coloneqq \frac{1}{m} \sum_{j=1}^{m} ||f_\theta(x_j) - y_j||_2^2,$$

for predictor $f_\theta : \mathcal{X} \to \mathbb{R}^d$ that maps inputs $x \in \mathcal{X}$ to labels $y \in \mathbb{R}^d$ and m is the number of training examples. Though minimizing the squared distance to a desired target is a natural choice of the loss function, an alternative is to discretize the label space and reformulate regression as classification with a cross-entropy loss function. Perhaps counterintuitively, this reformulation has been shown to be beneficial in practice (Farebrother et al., 2024) and theory (Imani et al., 2024).

In this section, we will only consider the case $d = 1$. Let $y_{\min}$ and $y_{\max}$ be a minimum and maximum bound on the predicted value from $f_\theta(x)$. Since classification requires a discrete label set, we discretize the interval $[y_{\min}, y_{\max}]$ uniformly into k bins and the predictor $f_\theta(x)$ outputs k logits that parameterize a softmax distribution over the k bins. Let $\tilde{y}_i$ be the center of the ith bin and $\hat{p}_i(x)$ be the probability of the ith bin output by $f_\theta(x)$. The scalar-valued prediction for y given x is the expected value of $\tilde{y}_i$ under $\hat{p}(x)$ or $\sum_i \tilde{y}_i \hat{p}_i(x)$.

We train f_θ using a cross-entropy loss between $f_\theta(x)$ and a target distribution that is specified from y. Following the notation of Imani and White (2018), we denote this target distribution as q_y. We then train f_θ by minimizing the cross-entropy loss:

$$\mathcal{L}_{\texttt{CE}}(\theta) \coloneqq \frac{1}{m} \sum_{j=1}^{m} \sum_{i}^{k} q_{y_j}(i) \log \hat{p}_i(x_j).$$

In this work, we will consider two choices for the target distribution q_y. A straightforward choice is a 1-hot distribution with the bin corresponding to y receiving probability 1. However, this choice potentially discards information about the spatial structure and ordinality of the label space that the least-squares loss preserves. Imani and White (2018) and Farebrother et al. (2024) found that a histogram approximation to a Gaussian distribution with a mean of y and the standard deviation chosen as a hyper-parameter was a better choice for this reason. Using a histogram approximation of a Gaussian in Equation (2.3) results in a loss that Imani and White (2018) called HL-Gauss. Optimizing HL-Gauss for input x_j increases the probability of outputting the target label y_j the most while also increasing the probability of values close to y_j.

3 Implicit Regression in Policy Gradient RL

We first show how policy gradient RL updates are equivalent to weighted regression updates. First, we note that policy gradient algorithms are often implemented to maximize the surrogate loss:

$$\mathcal{L}_{\texttt{surr}}(\theta) \coloneqq \frac{1}{m} \sum_{j=1}^{m} A_{\pi_\theta}(\mathbf{s}_j, \mathbf{a}_j) \log \pi_\theta(\mathbf{a}_j | \mathbf{s}_j),$$

[1] Note that in reality Equation (2.2) is *not* the gradient of $J(\pi_\theta)$ but is a widely used and biased approximation of it (Thomas, 2014; Nota and Thomas, 2020).

where we have obtained m state-action pairs by running some policy. Assuming that the data was collected on-policy (i.e., by running π_θ) then the gradient of $\mathcal{L}_{\texttt{surr}}(\theta)$ is an unbiased estimator of $\nabla_\theta J(\pi_\theta)$ (Foerster et al., 2018). We interpret $\mathcal{L}_{\texttt{surr}}$ as a weighted supervised learning loss with states as inputs, actions as labels, and the weight on each sample given by the advantage function.

When the task MDP has continuous actions, the most common policy parameterization is a multivariate Gaussian where the mean and covariance are given as functions of the state that are parameterized by θ. That is $\pi_\theta(\mathbf{a}|\mathbf{s}) = \mathcal{N}(\mathbf{a}; \mu_\theta(\mathbf{s}), \Sigma_\theta(\mathbf{s}))$. For the sake of exposition, we will treat $\Sigma_\theta(s)$ as a constant identity matrix and focus on $\mu_\theta(s)$.[2] Under a Gaussian parameterization, the surrogate objective becomes a weighted least-squares regression problem:

$$\arg\max_\theta \mathcal{L}_{\texttt{surr}}(\theta) = \arg\min_\theta \mathcal{L}_{\texttt{PG-LS}}(\theta) := \frac{1}{m}\sum_{j=1}^{m} A_{\pi_\theta}(\mathbf{s}_j, \mathbf{a}_j)\frac{1}{2}||\mathbf{a}_j - \mu_\theta(\mathbf{s}_j)||_2^2 + \texttt{const}$$

It can now be seen that the policy gradient surrogate loss for Gaussian policies is a weighted least-squares problem that resembles Equation (2.3).

The connection between policy optimization and supervised learning has been previously made by Peters and Schaal (2007); Peng et al. (2019); Abdolmaleki et al. (2018) in the context of formulating policy optimization with KL-divergence constraints. Under the formulation in these past works, policy optimization can also be cast as weighted regression. The key difference between these works and ours is that the KL-divergence constraint results in the weighting function being $\exp(\frac{1}{\tau}A_{\pi_\theta}(\mathbf{s}, \mathbf{a}))$ rather than $A_{\pi_\theta}(\mathbf{s}, \mathbf{a})$.[3]

4 Replacing Implicit Regression with Classification

Empirical evidence in supervised learning and RL suggests there is an empirical benefit to reformulating regression problems as classification problems. Our goal in this work is to understand if this benefit translates to policy gradient learning if we reformulate the implicit regression in policy gradient methods as classification. Toward this understanding, we first describe how we can represent continuous action policies as policies over discrete actions and then introduce a new policy gradient surrogate objective for training these policies.

4.1 Policy Representation

To recast regression as classification we first need to parameterize the policy we are learning as a distribution over a finite set $\mathcal{Z}$ where each continuous $\mathbf{a} \in \mathcal{A}$ maps to an element of $z \in \mathcal{Z}$. The naive way to accomplish this mapping is to discretize the continuous space using a multi-dimensional grid with k bins along each dimension. The grid representation is useful in that it can learn policies in which different action dimensions are correlated. The downside of this representation is that the number of discrete actions will be exponential in the native action space dimensionality.

To make discrete action policies tractable, we make the simplifying assumption that each action dimension is selected independently of the others. This assumption is reasonable as it is already standard practice when using Gaussian policies to use a diagonal covariance matrix. Thus, in comparison to such Gaussian policies, the policy representation that we introduce is only limited in terms of the granularity of the discretization. This simplification means that after discretization, each dimension has k bins and the policy network only needs to output kd values instead of k^d. The

[2] In our experiments, we will learn a state-independent covariance matrix when considering Gaussian policies. A non-identity covariance matrix means that the policy gradient method is implicitly implementing heteroscedastic regression.

[3] We informally experimented with using the exponentiated advantage at the start of our investigation. We found that the non-exponentiated advantage tended to give better results and had fewer hyper-parameters to tune.

limitation of this assumption is that the space of possible distributions over actions that the policy can represent is now limited. In applications where d is small, it may be preferable to keep the added expressivity or to investigate alternative factorizations over the action dimensions.

Formally, we learn a policy that outputs d softmax distributions where each distribution is over a finite set $\mathcal{Z}_l$ for $l \in \{1, ..., d\}$. We map each continuous value in $[a_{\min}, a_{\max}]$ to an element of $\mathcal{Z}_l$. In this work, we use a uniform discretization with k bins and let $c \coloneqq \frac{a_{\max} - a_{\min}}{k}$ be the width of each bin. The elements of $\mathcal{Z}_l$ form an ordered set where the ith element, z_l^i, represents the range $[a_{\min} + c \cdot (i-1), a_{\min} + c \cdot i]$ for $i \in \{1, ...k\}$. Let a_l^i be the center of this range.

We denote ${\pi_\theta}^l(\cdot|\mathbf{s})$ as the policy distribution over action dimension l. To sample from this policy representation, we first sample $z_l^i \sim {\pi_\theta}^l(\cdot|\mathbf{s})$ for each dimension l. We then return the associated a_l^i as the value of the action for that dimension.[4]

4.2 Policy Gradient Learning as Classification

Now, to replace the implicit regression in Equation (3) with classification, we replace the least-squares portion of $\mathcal{L}_{\texttt{PG-LS}}$ with a cross-entropy loss. By doing so, we obtain the loss function:

$$\mathcal{L}_{\texttt{PG-CE}}(\theta) \coloneqq \frac{1}{m} \sum_{j=1}^{m} A_{\pi_\theta}(\mathbf{s}_j, \mathbf{a}_j) \sum_{l=1}^{d} \sum_{i=1}^{k} q_{a_{j,l}}(i) \log \pi(\tilde{a}_l^i|\mathbf{s}),$$

where q_{a_l} is a target probability distribution over action dimension l that is defined in terms of the sampled action $a_{j,l}$. We consider two choices for the target distribution: the 1-hot distribution that places all probability mass on the observed action and a histogram approximation to a Gaussian distribution centered at dimension l of action $\mathbf{a}$. We call these two instantiations of our new loss HL-1-Hot, and HL-Gauss, respectively. For the latter, the standard deviation, σ, of the approximated Gaussian is a method hyper-parameter; we follow Farebrother et al. (2024) by tuning $\eta \coloneqq \frac{\sigma}{c}$.

4.3 Bound on Gradient Norm

Imani and White (2018) found that stable gradients were a potential benefit of the HL-Gauss loss compared to either a 1-hot cross-entropy loss or a least-squares loss for regression. Here, increased stability means that the norm of the loss gradient has a smaller upper bound compared to the gradient of $\mathcal{L}_{\texttt{PG-LS}}$. Imani and White (2018) attribute the utility of a small gradient norm to prior theoretical work showing that a loss with small Lipschitz constant provides an improved upper bound on generalization performance in supervised-learning. We extend this analysis to our loss.

For conciseness, we will only consider the case that $d = 1$. Define our loss at a given state-action pair to be $\mathcal{L}_{\texttt{PG-HL}}(\theta, \mathbf{s}, \mathbf{a}) \coloneqq A_{\pi_\theta}(\mathbf{s}, \mathbf{a}) \sum_{i=1}^{k} q_{\mathbf{a},i} \log \hat{p}_i(s)$.

Proposition 1. *Let $\mu_\phi(\mathbf{s})$ be the feature representation of $\mathbf{s}$ at the penultimate layer of the policy network. Assume that the policy's logits are l-Lipschitz, i.e $\|\mathbf{w}_j^\top \mu_\phi(\mathbf{s})\| \leq l$ with respect to ϕ. Then, we have that:*

$$\|\nabla_\theta \mathcal{L}_{\texttt{PG-CE}}(\theta, \mathbf{s}, \mathbf{a})\| \leq \|A_{\pi_\theta}(\mathbf{s}, \mathbf{a})\| \left(\sum_{i=1}^{k} |q_{\mathbf{a},i} - \pi_\theta(a_i|\mathbf{s})| \left(l + \|\mu_\phi(\mathbf{s})\|\right) \right)$$

See Appendix A for the proof. In comparison to Imani et al. (2024), we see that the bound on the gradient norm at any state-action pair is multiplied by the magnitude of the advantage which is expected as $\mathcal{L}_{\texttt{PG-CE}}$ is equal to $\mathcal{L}_{\texttt{CE}}$ multiplied by the advantage. Following their argument, we can also expect the gradient of $\mathcal{L}_{\texttt{PG-LS}}$ to have a greater upper bound on its norm, particularly when the range of the action-space is large.

[4] We choose to deterministically return the center of the range for simplicity but alternative choices could be made. For instance, we could uniformly sample from the range.

5 Empirical Analysis

We next conduct an empirical study designed to answer the question, "does replacing the implicit regression of the policy gradient surrogate loss with a cross-entropy loss increase the data efficiency of stochastic policy gradient methods?"

5.1 Empirical Set-up

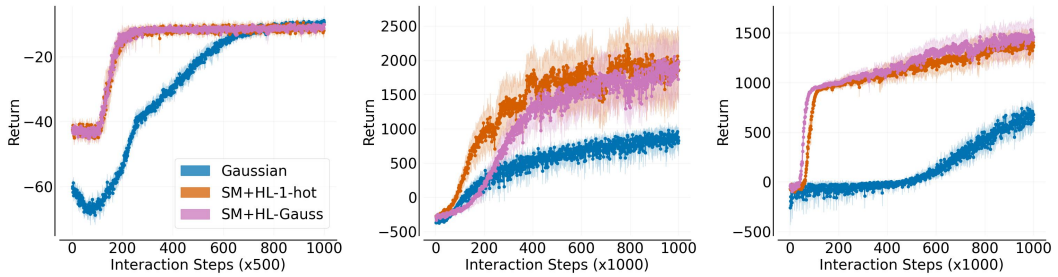
To investigate these questions, we run learning trials in the Reacher, HalfCheetah, and Ant domains (Towers et al., 2023). Please refer to Appendix B.1 for detailed descriptions of each environment.

For simplicity, we use a stochastic actor-critic algorithm as the base policy gradient algorithm (Sutton and Barto, 2018). We use n-step returns to estimate the advantage function, the Adam optimizer (Kingma and Ba, 2015), and clip gradients during training. For advantage estimation, we fit a state-dependent value function using an MSE loss with observed returns as targets.

Our primary point of comparison is between Gaussian policies with the standard policy gradient surrogate loss, softmax policies with the HL-Gauss loss, and softmax policies with the 1-hot cross-entropy loss. We refer the reader to Appendix B.2 for training details of all the algorithms such hyperparameters, batch sizes, policy architectures, etc. The default hyper-parameters for each method are chosen using a sweep; the sensitivity of each method is discussed in Appendix B.3. We generally find that the cross-entropy losses are more robust to hyper-parameters (Appendix B.4 gives performance profiles for each method across all hyper-parameters tested).

5.2 Empirical Results

In general, we find that recasting the regression loss as a cross-entropy loss significantly boosts learning efficiency. In almost all instances, we observe that agents that minimize a cross-entropy loss learn faster and achieve a higher return at the end of the training period.



(a) Reacher ($d = 2$) (b) HalfCheetah ($d = 6$) (c) Ant ($d = 8$)

Figure 1: Highest undiscounted training returns achieved by each algorithm as a function of environment interaction steps after a hyperparameter sweep. SM is a softmax policy and HL is the histogram loss. Results are the mean averaged over 20 trials and the shaded region represents the 95% confidence interval. Higher is better.

We have found that the cross-entropy-based policy gradient surrogate loss that we introduced in this work generally leads to more data-efficient policy gradient learning across the continuous control domains where we evaluated it. Results showed that performance improved even as the action-space dimensionality increased which shows the viability of simply selecting each action dimension independently. These results suggest that reformulating the weighted regression found in policy gradient learning for Gaussian policies as weighted classification for softmax policies can be an effective strategy in continuous control RL applications.

We found that the performance difference between the HL-1-Hot and HL-Gauss surrogate losses was often small and the ordering of the methods changed across domains. This result was somewhat surprising to us as HL-1-Hot discards spatial information about the action-space when reinforcing actions whereas HL-Gauss makes use of this information through the Gaussian target distribution. We do find in some cases that HL-1-Hot may be more prone to find local optima (i.e., it converges to a discretized action that is adjacent to the optimal discretized action), however, the loss in performance from these cases is small in the benchmarks we considered.

Perhaps the principle limitation of the HL-Gauss loss is the need to discretize the action space so that a softmax policy can be used. The result is that the true deterministic, optimal policy may not be representable, e.g., if the optimal action in some state is not a bin center. The degree of this limitation depends upon the properties of a domain and how necessary it is for the policy to output precise actions for acceptable performance. In our experiments, we did not observe adverse effects from discretizing the action-space. This result could be because the domains we considered do not require precise control for high returns. It could also indicate that learning with Gaussian policies is sufficiently slow that we never reach the point where their improved representation power becomes useful. Further small-scale studies on carefully designed toy problems could help understand when discretization is *not* a viable strategy.

6 Conclusion

This paper has studied the degree to which stochastic policy gradient algorithms can be improved for continuous action domains by replacing an implicit least-squares loss term with a cross-entropy loss term in the policy gradient surrogate objective. We first derived the connection between the policy gradient for Gaussian policies and a certain weighted least-squares optimization problem. We then introduced a novel loss function that replaces the implicit weighted regression loss for Gaussian policies with a weighted cross-entropy loss for softmax policies. We showed theoretically that this loss enjoys a smaller bound on gradient norms and then showed empirically that this novel loss improves the data efficiency of a prototypical stochastic actor-critic method for continuous control.

Acknowledgements

The authors thank Nicholas Corrado and the workshop reviewers for comments that improved the paper.

References

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a Posteriori Policy Optimisation, June 2018. URL http://arxiv.org/abs/1806.06920. arXiv:1806.06920 [cs, math, stat].

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *CoRR*, abs/2108.13264, 2021. URL https://arxiv.org/abs/2108.13264.

Yuanzhouhan Cao, Zifeng Wu, and Chunhua Shen. Estimating Depth From Monocular Images as Classification Using Deep Fully Convolutional Residual Networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(11):3174–3182, November 2018. ISSN 1051-8215, 1558-2205. doi: 10.1109/TCSVT.2017.2740321. URL https://ieeexplore.ieee.org/document/8010878/.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision Transformer: Reinforcement Learning via Sequence Modeling, June 2021. URL http://arxiv.org/abs/2106.01345. arXiv:2106.01345 [cs].

Po-Wei Chou, Daniel Maturana, and Sebastian Scherer. Improving Stochastic Policy Gradients in Continuous Control with Deep Reinforcement Learning using the Beta Distribution. In *Proceedings of the 34th International Conference on Machine Learning*, pages 834–843. PMLR, July 2017. URL https://proceedings.mlr.press/v70/chou17a.html. ISSN: 2640-3498.

Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taïga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, Aviral Kumar, and Rishabh Agarwal. Stop Regressing: Training Value Functions via Classification for Scalable Deep RL, March 2024. URL http://arxiv.org/abs/2403.03950. arXiv:2403.03950 [cs, stat].

Jakob Foerster, Gregory Farquhar, Maruan Al-Shedivat, Tim Rocktäschel, Eric Xing, and Shimon Whiteson. DiCE: The Infinitely Differentiable Monte Carlo Estimator. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1529–1538. PMLR, July 2018. URL https://proceedings.mlr.press/v80/foerster18a.html. ISSN: 2640-3498.

Yasuhiro Fujita and Shin-ichi Maeda. Clipped Action Policy Gradient. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1597–1606. PMLR, July 2018. URL https://proceedings.mlr.press/v80/fujita18a.html. ISSN: 2640-3498.

Ehsan Imani and Martha White. Improving Regression Performance with Distributional Losses. In *Proceedings of the 35th International Conference on Machine Learning*, pages 2157–2166. PMLR, July 2018. URL https://proceedings.mlr.press/v80/imani18a.html. ISSN: 2640-3498.

Ehsan Imani, Kai Luedemann, Sam Scholnick-Hughes, Esraa Elelimy, and Martha White. Investigating the Histogram Loss in Regression, February 2024. URL http://arxiv.org/abs/2402.13425. arXiv:2402.13425 [cs, stat].

Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-To-End Learning of Geometry and Context for Deep Stereo Regression. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 66–75, 2017. URL https://openaccess.thecvf.com/content_iccv_2017/html/Kendall_End-To-End_Learning_of_ICCV_2017_paper.html.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*. arXiv, 2015. doi: 10.48550/arXiv.1412.6980. URL http://arxiv.org/abs/1412.6980. arXiv:1412.6980 [cs].

Yanjie Li, Sen Yang, Peidong Liu, Shoukui Zhang, Yunxiao Wang, Zhicheng Wang, Wankou Yang, and Shu-Tao Xia. SimCC: A Simple Coordinate Classification Perspective for Human Pose Estimation. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 89–106, Cham, 2022. Springer Nature Switzerland. ISBN 978-3-031-20068-7. doi: 10.1007/978-3-031-20068-7_6.

Luke Metz, Julian Ibarz, Navdeep Jaitly, and James Davidson. Discrete Sequential Prediction of Continuous Actions for Deep RL, June 2019. URL http://arxiv.org/abs/1705.05035. arXiv:1705.05035 [cs, stat].

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. AWAC: Accelerating Online Reinforcement Learning with Offline Datasets, April 2021. URL http://arxiv.org/abs/2006.09359. arXiv:2006.09359 [cs, stat].

Chris Nota and Philip S. Thomas. Is the Policy Gradient a Gradient? In *International Joint Conference on Autonomous and Multi-agent Systems (AAMAS)*, 2020. doi: 10.48550/arXiv.1906.07073. URL http://arxiv.org/abs/1906.07073. arXiv:1906.07073 [cs, stat].

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703, 2019. URL http://arxiv.org/abs/1912.01703.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-Weighted Regression: Simple and Scalable Off-Policy Reinforcement Learning, October 2019. URL http://arxiv.org/abs/1910.00177. arXiv:1910.00177 [cs, stat].

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 745–750, New York, NY, USA, June 2007. Association for Computing Machinery. ISBN 978-1-59593-793-3. doi: 10.1145/1273496.1273590. URL https://doi.org/10.1145/1273496.1273590.

Silvia L. Pintea, Yancong Lin, Jouke Dijkstra, and Jan C. van Gemert. A step towards understanding why classification helps regression. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 19972–19981, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Pintea_A_step_towards_understanding_why_classification_helps_regression_ICCV_2023_paper.html.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Rasmus Rothe, Radu Timofte, and Luc Van Gool. Dex: Deep expectation of apparent age from a single image. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015.

Juergen Schmidhuber. Reinforcement Learning Upside Down: Don't Predict Rewards – Just Map Them to Actions, June 2020. URL http://arxiv.org/abs/1912.02875. arXiv:1912.02875 [cs].

Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

Chen Tessler, Guy Tennenholtz, and Shie Mannor. Distributional Policy Optimization: An Alternative Approach for Continuous Control. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/72da7fd6d1302c0a159f6436d01e9eb0-Abstract.html.

Philip Thomas. Bias in Natural Actor-Critic Algorithms. In *Proceedings of the 31st International Conference on Machine Learning*, pages 441–448. PMLR, January 2014. URL https://proceedings.mlr.press/v32/thomas14.html. ISSN: 1938-7228.

Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023. URL https://zenodo.org/record/8127025.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992698. URL https://doi.org/10.1007/BF00992698.

Shihao Zhang, Linlin Yang, Michael Bi Mi, Xiaoxu Zheng, and Angela Yao. Improving Deep Regression with Ordinal Entropy. In *Proceedings of the International Conference on Learning Representations (ICLR)*. arXiv, 2023. doi: 10.48550/arXiv.2301.08915. URL http://arxiv.org/abs/2301.08915. arXiv:2301.08915 [cs].

A Proof of Proposition 1

Proposition 1. *Let $\mu_\phi(\mathbf{s})$ be the feature representation of $\mathbf{s}$ at the penultimate layer of the policy network. Assume that the policy's logits are l-Lipschitz, i.e $\|\mathbf{w}_j^\top \mu_\phi(\mathbf{s})\| \leq l$ with respect to ϕ. Then, we have that:*

$$\|\nabla_\theta \mathcal{L}_{\texttt{PG-CE}}(\theta, \mathbf{s}, \mathbf{a})\| \leq \|A_{\pi_\theta}(\mathbf{s}, \mathbf{a})\| \left(\sum_{i=1}^{k} |q_{\mathbf{a},i} - \pi_\theta(a_i|\mathbf{s})| \left(l + \|\mu_\phi(\mathbf{s})\|\right) \right)$$

Proof. The proof for this proposition follows largely from Imani and White (2018) with the key difference being that the advantage estimate for a given state action pair is weighted by $A_\pi(\mathbf{s}, \mathbf{a})$. However, we list out the proof for completeness for the reader.

Let us represent $\theta = [\phi, \mathbf{w}]^T$ where ϕ is network parameters up to and including the penultimate layer and $\mathbf{w} = \{\mathbf{w}_i\}_{i=1}^k$ represent the parameters of the policy belonging to the last layer. The unnormalized softmax logit for the ith bin is given as $b_i = e^{\mu_\theta(\mathbf{s})^T \mathbf{w}_i}$. Then $\forall j \neq i$ and $j \in \{1, 2, \ldots k\}$,

$$\frac{\partial}{\partial b_i} \pi_\theta(a_j|\mathbf{s}) = \frac{\partial}{\partial b_i} \frac{e_j}{\sum_{l=1}^k e_l} = -\frac{e_j}{\left(\sum_{l=1}^k e_l\right)^2} e_i = -\pi_\theta(a_j|\mathbf{s})\pi_\theta(a_i|\mathbf{s})$$

Similarly, for $j = i$, we can write,

$$\frac{\partial}{\partial b_i} \pi_\theta(a_j|\mathbf{s}) = \frac{e_i}{\sum_{l=1}^k e_l} - \frac{e_i}{\left(\sum_{l=1}^k e_l\right)^2} e_i = \pi_\theta(a_i|\mathbf{s}) \left[1 - \pi_\theta(a_i|\mathbf{s})\right]$$

Using the above expressions we can compute the partial derivative of the histogram loss *without the advantage weighting*:

$$\begin{aligned}
\frac{\partial}{\partial b_i} \sum_{j=1}^{k} q_{a_l,j} \log \pi_\theta(a_j|\mathbf{s}) &= \sum_{j=1, j\neq i}^{k} \frac{q_{a_l,j}}{\pi_\theta(a_i|\mathbf{s})} \pi_\theta(a_j|\mathbf{s})\pi_\theta(a_i|\mathbf{s}) + \frac{q_{a_l,i}}{\pi_\theta(a_i|\mathbf{s})} \pi_\theta(a_i|\mathbf{s})) \left[1 - \pi_\theta(a_i|\mathbf{s})\right] \\
&= q_{a_l,i} - \pi_\theta(a_i|\mathbf{s}) \sum_{j=1, j\neq i}^{k} q_{a_l,j} - q_{a_l,i}\pi_\theta(a_i|\mathbf{s}) \\
&= q_{a_l,i} - \pi_\theta(a_i|\mathbf{s})
\end{aligned}$$

By applying the chain rule, we can use the above to show that,

$$\begin{aligned}
\left\| \nabla_\phi \left(\sum_{j=1}^{k} q_{a_l,j} \log \pi_\theta(a_j|\mathbf{s}) \right) \right\| &= \left\| \sum_{j=1}^{k} \frac{\partial}{\partial b_j} \left(q_{a_l,j} \log \pi_\theta(a_j|\mathbf{s})\right) \frac{\partial b_j}{\partial \phi} \right\| \\
&= \left\| \sum_{j=1}^{k} \left(q_{a_l,j} - \pi_\theta(a_j|\mathbf{s})\right) \nabla_\phi w_j^\top \mu_\theta(\mathbf{s})) \right\| \\
&\overset{(a)}{\leq} \sum_{j=1}^{k} \|q_{a_l,j} - \pi_\theta(a_j|\mathbf{s})\| \, l,
\end{aligned}$$

where (a) follows by the assumption that $\|\nabla_\phi \mathbf{w}_j^\top \mu_\phi(\mathbf{s})\| \leq l$.

$$\begin{aligned}
\left\|\nabla_{w_i}\left(\sum_{j=1}^{k} q_{a_l,j}\log\pi_\theta(a_j|\mathbf{s})\right)\right\| &= \left\|\sum_{j=1}^{k}\frac{\partial}{\partial b_j}\left(q_{a_l,j}\log\pi_\theta(a_j|\mathbf{s})\right)\frac{\partial b_j}{\partial w_i}\right\| \\
&= \left\|\sum_{j=1}^{k}\left(q_{a_l,j}-\pi_\theta(a_j|\mathbf{s})\right)\frac{\partial}{\partial w_i}w_j^\top\mu_\phi(\mathbf{s})\right)\right\| \\
&\overset{(b)}{\leq} \|q_{a_l,i}-\pi_\theta(a_i|\mathbf{s})\|\,\|\mu_\phi(\mathbf{s})\|
\end{aligned}$$

Now, the gradient of the histogram loss, $\|\nabla\mathcal{L}_{\texttt{PG-CE}}(\theta,\mathbf{s},\mathbf{a})\|$, can be expressed as,

$$\begin{aligned}
\left\|\nabla_\theta A_{\pi_\theta}(s,a)\left(\sum_{j=1}^{k} q_{a_l,j}\log p_{a_l,j}\right)\right\| &\leq \|A_{\pi_\theta}(s,a)\|\sum_{j=1}^{k}\left\|\nabla_{w_i}\left(\sum_{j=1}^{k} q_{a_l,j}\log p_{a_l,j}\right)\right\| \\
&\quad + \|A_{\pi_\theta}(s,a)\|\left\|\nabla_\phi\left(\sum_{j=1}^{k} q_{a_l,j}\log p_{a_l,j}\right)\right\| \\
&\overset{(c)}{\leq} \|A_{\pi_\theta}(s,a)\|\left(\sum_{j=1}^{k}|q_{a_l,j}-\pi_\theta(a_j|\mathbf{s})|(\|\mu_\theta(\mathbf{s})\|+l)\right)
\end{aligned}$$

Here (c) follows directly by adding inequalities (a) and (b). The full gradient that we use is actually an expectation over $\nabla\mathcal{L}_{\texttt{PG-CE}}(\theta,\mathbf{s},\mathbf{a})$ under the state-action distribution of the current policy. We thus upper bound the full gradient as:

$$\begin{aligned}
\|\mathbb{E}_{\mathbf{s}\sim d_{\pi_\theta},\mathbf{a}\sim\pi_\theta(\cdot|\mathbf{s})}\left[\nabla\mathcal{L}_{\texttt{PG-CE}}(\theta,\mathbf{s},\mathbf{a})\right]\| &\leq \mathbb{E}_{\mathbf{s}\sim d_{\pi_\theta},\mathbf{a}\sim\pi_\theta(\cdot|\mathbf{s})}\left[\|A_\pi(\mathbf{s},\mathbf{a})\|\left(\sum_{i=1}^{k}|q_{a_l}(i)-\pi_\theta(a_i|\mathbf{s})|(\|\mu_\theta(\mathbf{s})\|+l)\right)\right] \\
&\leq \max_{(\mathbf{s},\mathbf{a})\in\mathcal{S}\times\mathcal{A}}\left\{\|A_\pi(\mathbf{s},\mathbf{a})\|\left(\sum_{i=1}^{k}|q_{a_l}(i)-\pi_\theta(a_i|\mathbf{s})|(\|\mu_\theta(\mathbf{s})\|+l)\right)\right\}
\end{aligned}$$

□

B Empirical Details

In this section, we provide additional details about the experiments that were deferred from the main section.

B.1 Environment Details

In this section, we provide details of the evaluated environments.

1. **Continuous Bandit:** This domain has a single state and the reward is an unknown quadratic function of a d-dimensional action. The range for possible actions in each dimension is $[a_{\texttt{min}}, a_{\texttt{max}}]$.
2. **Linear quadratic regulator:** LQR is a fundamental control problem in control theory. In this domain, the transition dynamics are a linear function of the state and action with Gaussian noise added. The reward is a quadratic function of the state and action. The action space is 2 dimensional where each dimension is bounded between $[-1, 1]$.

3. **Continous Mountain Car:** In this domain, a toy car attempts to reach the top of a mountain. The action space is 1 dimensional and is bounded between $[-1, 1]$.

4. **Continuous Acrobot:** In this domain, an agent attempts to swing itself above a certain height. The action space is 1 dimensional.

5. **Reacher:** In this domain, a robotic arm tries to reach a goal location. The action space is 2 dimensional and each dimension is bounded between $[-1, 1]$.

6. **HalfCheetah:** In this domain, a cheetah-like robotic agent attempts to run as fast as possible. The action space is 6 dimensional and each dimension is bounded between $[-1, 1]$.

7. **Ant:** In this domain, an ant-like robotic agent attempts to run as fast as possible. The action space is 8 dimensional and each dimension is bounded between $[-1, 1]$.

B.2 Training Details

For the continous bandit environment, all the algorithms (Gaussian regression, softmax + 1-hot, softmax + HL-Gauss) used a linear policy, used a batch size of 5, and all used a value function baseline. Each algorithm was trained for 2000 interaction steps and the policy was evaluated every interaction step. We tuned the following hyperparameters. For all algorithms, we swept over the following values for the learning rate: $\{10^{-2}, 5 \cdot 10^{-2}, 10^{-1}, 1\}$. For the two classification methods, we swept over the following number of bins: $\{50, 100, 200\}$. For softmax + HL-Gauss, we swept over the following width multipliers: $\{0.1, 0.25, 0.5, 0.75, 1\}$.

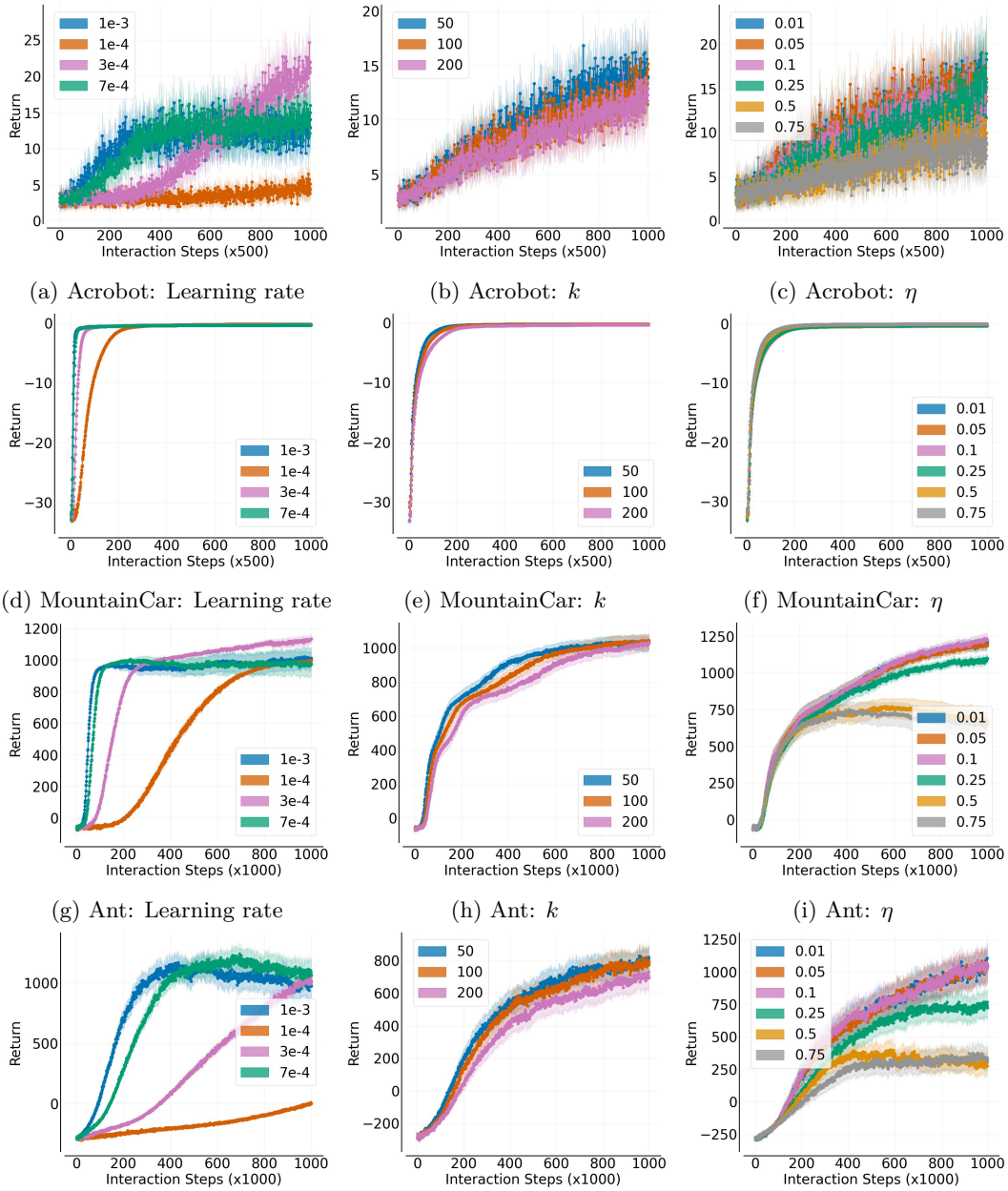
For all the other environments, all the algorithms used the default neural network policy in stablebaseline3 (Raffin et al., 2021), used a batch size of 20, and all used a value function baseline. On LQR, Continuous MountainCar, Reacher, and Acrobot, all algorithms were trained for 500K interaction steps and the policy was evaluated every 500 steps. On the HalfCheetah and Ant domain, all algorithms were trained for 1M interaction steps and the policy was evaluated every 1000K steps. For all these domains and algorithms, we swept over the following learning rate values: $\{10^{-4}, 3 \cdot 10^{-4}, 7 \cdot 10^{-4}, 10^{-3}\}$. For the two classification methods, we swept over the following number of bins: $\{50, 100, 200\}$. For softmax + HL-Gauss, we swept over the following width multipliers: $\{0.01, 0.05, 0.1, 0.25, 0.5, 0.75\}$.

For all algorithms, our hyperparameter sweep is based on picking the hyperparameter combination that led to the highest average undiscounted return (averaged across all trials) on the final step.

B.3 Hyperparameter Sensitivity Experiments

In this section, we show the sensitivity of the softmax + HL-Gauss algorithm when varying the learning rate, number of bins (k), and width mutliplier (η). From Figure 2, we find that softmax + HL-Gauss is sensitive to the learning rate, and performance can widely differ based on the value of the learning rate. It is also sensitive to η, which determines the standard deviation of the histogram Gaussian distribution, where if the standard deviation is too large (larger η), performance tends to degrade, which is expected since the agent has a challenging time to converge to the optimal actions. The method is quite robust to number of bins (k), where performance is generally stable across all k values.

With regards to the baselines, we see in Figure 3 and Figure 4 that the Gaussian regression method and softmax with 1-hot histogram loss are also sensitive to the learning rates. And softmax 1-hot is similarly robust to number of bins as softmax + HL-Gauss.



(a) Acrobot: Learning rate (b) Acrobot: k (c) Acrobot: η

(d) MountainCar: Learning rate (e) MountainCar: k (f) MountainCar: η

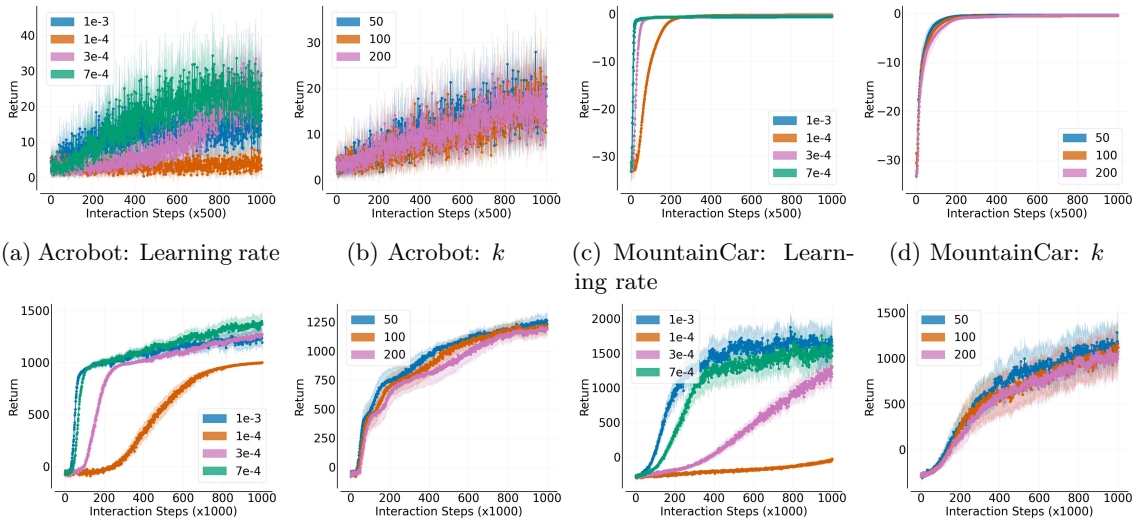
(g) Ant: Learning rate (h) Ant: k (i) Ant: η

(j) HalfCheetah: Learning rate (k) HalfCheetah: k (l) HalfCheetah: η

Figure 2: Hyperparameter sensitivity of softmax + HL-Gauss. Undiscounted training returns achieved by softmax + HL-Gauss as a function of environment interaction steps for different hyperparameters. Results are averaged over 20 trials and the shaded region represents the 95% confidence interval. Higher is better.

B.4 Performance Profiles

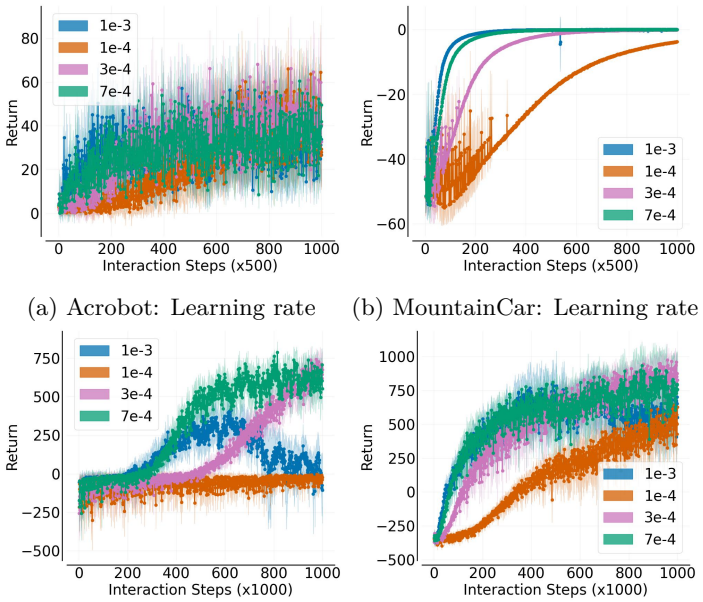
In this section, we report the performance profiles of each algorithm across *all* trials and hyperparameters. These plots illustrate what fraction of the total runs of an algorithm led to a return of greater than τ. In general, we find that the classification losses have a higher fraction of the runs that achieve a high return than the regression loss.



(a) Acrobot: Learning rate (b) Acrobot: k (c) MountainCar: Learning rate (d) MountainCar: k

(e) Ant: Learning rate (f) Ant: k (g) HalfCheetah: Learning rate (h) HalfCheetah: k

Figure 3: Hyperparameter sensitivity of softmax + HL-1-hot. Undiscounted training returns achieved by softmax + HL-Gauss as a function of environment interaction steps for different hyperparameters. Results are averaged over 20 trials and the shaded region represents the 95% confidence interval. Higher is better.



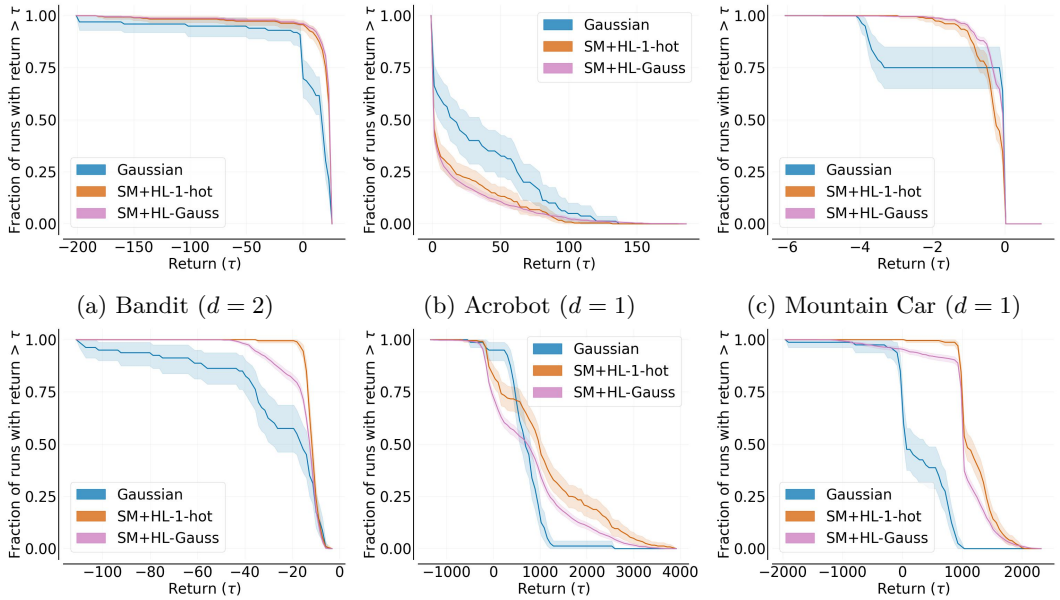
(a) Acrobot: Learning rate (b) MountainCar: Learning rate

(c) Ant: Learning rate (d) HalfCheetah: Learning rate

Figure 4: Hyperparameter sensitivity of the Gaussion regression method. Undiscounted training returns achieved by softmax + HL-Gauss as a function of environment interaction steps for different hyperparameters. Results are averaged over 20 trials and the shaded region represents the 95% confidence interval. Higher is better.

B.5 Evaluation Returns

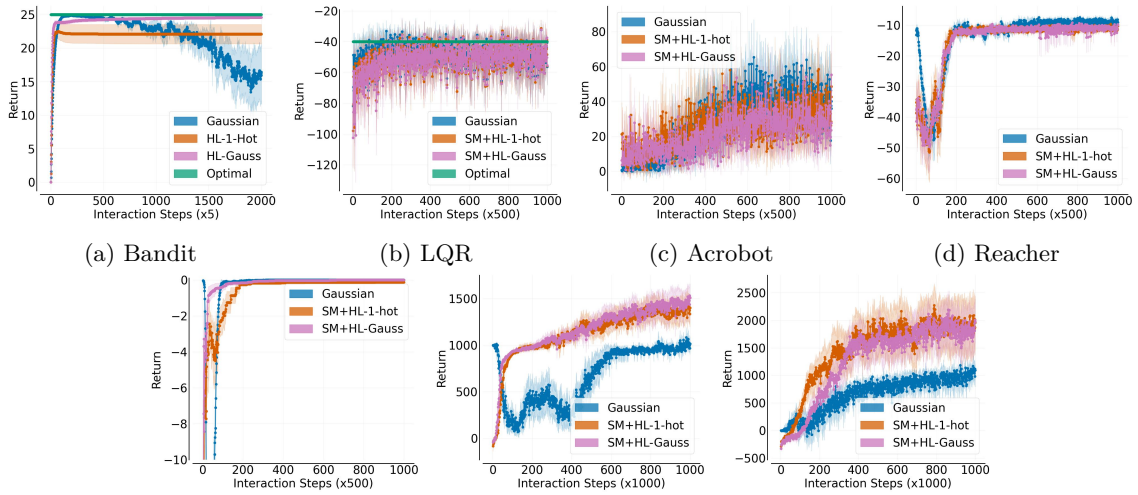
In this section, we report (Figure 6) the undiscounted return achieved by the deterministic policy as a function of environment interaction steps.



(a) Bandit ($d = 2$) (b) Acrobot ($d = 1$) (c) Mountain Car ($d = 1$)

(d) Reacher ($d = 2$) (e) HalfCheetah ($d = 6$) (f) Ant ($d = 8$)

Figure 5: Performance profiles of all the algorithms across all 20 trials and hyperparameter combinations.. SM is a softmax policy and HL is the histogram loss. Higher is better. For each domain, we also give the action-dimensionality, d.



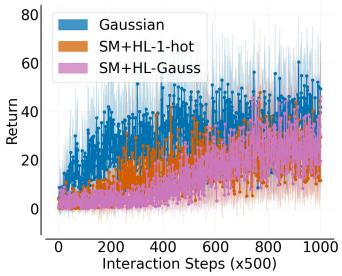
(a) Bandit (b) LQR (c) Acrobot (d) Reacher

(e) Mountain Car (f) Ant (g) HalfCheetah

Figure 6: Highest undiscounted *evaluation* returns achieved by each algorithm as a function of environment interaction steps after a hyperparameter sweep. Results are averaged over 20 trials and the shaded region represents the 95% confidence interval. Higher is better. The optimal return can be computed exactly in the Bandit and LQR settings.

B.6 Remaining Training Return Result

Due to space limitations, we include the training returns achieved in Acrobot in this section.

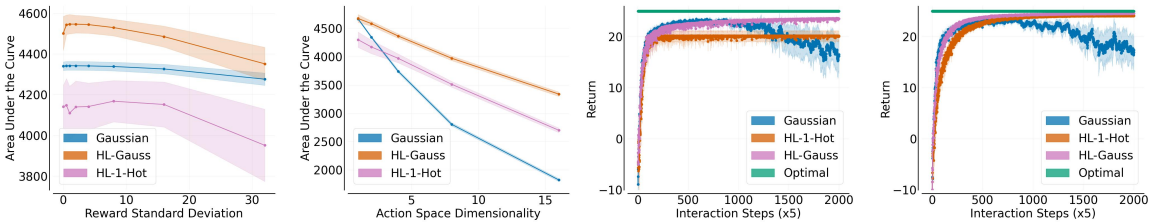


(a) Acrobot ($d = 1$)

Figure 7: Highest undiscounted training returns achieved by each algorithm as a function of environment interaction steps after a hyperparameter sweep. SM is a softmax policy and HL is the histogram loss. Results are the mean averaged over 20 trials and the shaded region represents the 95% confidence interval. Higher is better.

B.7 Sensitivity to Advantage Noise and Action Dimensionality

This subsection examines the sensitivity of different methods to noise in the advantage function estimate as well as the dimensionality of the agent's action space. We use the stateless continuous bandit domain for these experiments and keep all hyper-parameters fixed at their default values that were tuned for the experiments in the preceding subsection. In this domain, the reward is $r(\mathbf{a}) \leftarrow 25 - \frac{1}{d}(\mathbf{a} - \mathbf{5})^2 + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma)$ and the default values of d and σ are 2 and 1 respectively. To determine sensitivity to advantage noise and dimensionality, we independently vary the standard deviation, σ, of the reward received following each action and the dimensionality respectively.



(a) Reward Noise (b) Dimensionality (c) Gaussian Initialization (d) Gaussian Exploration

Figure 8: Environment sensitivity and exploration ablation studies. Error bars give a 95% confidence interval over 50 trials. For the sensitivity experiments, we report area under the curve where higher indicates faster and more stable learning, though it may hide nuance in the final performance.

Figure 8a shows that HL-Gauss is generally more insensitive to reward noise whereas Gaussian policies are strongly affected by it. Figure 8b shows that HL-Gauss is generally less sensitive to the dimensionality of the action space. Gaussian policies are strongly affected by action space dimensionality.

B.8 Exploration vs Optimization

The use of histogram losses for policy gradient learning is qualitatively different than past studies because the softmax representation does not just affect policy optimization but also affects the data distribution of the learning agent. This observation motivates our second empirical question on disentangling the benefits of exploration from ease of optimization.

To answer our question, we use the Bandit domain and repeat our main experiment under two additional conditions.

1. **Init Gaussian.** We initialize softmax policies to approximate the same initial Gaussian distribution that Gaussian policies use.
2. **Gaussian Exploration** At each iteration, we transform the softmax policy into a Gaussian policy and sample actions from this policy for exploration. To compute the Gaussian exploration policy, we compute the mean and variance of bin centers under the softmax distribution. This mean and variance then parameterize the Gaussian exploration distribution.

Figure 8c and Figure 8d show learning curves under **Init Gaussian** and **Gaussian Exploration** respectively. Both cross-entropy methods perform slightly worse with the Gaussian initialization suggesting that these methods have wider initial exploration in our base setting. With **Gaussian Exploration**, we observe that Gaussian and HL-Gauss learn at a similar rate for the first approximately 200 time-steps, however, whereas Gaussian becomes unstable, HL-Gauss smoothly converges close to the optimal action. We understand this result to indicate that 1) gains in the base experiment setting can be partially ascribed to improved initial exploration as the gains vanish when exploration is controlled, and 2) the HL-Gauss loss also increases stability which helps learning beyond just exploration benefits. Finally, we observe that HL-1-Hot lags slightly behind other methods but actually converges closer to optimal than it does with softmax exploration. We suspect that this result is due to the fact that the Gaussian exploration adds information about how close different actions are to one another.

B.9 Assets and Software

We implement REINFORCE with the three loss functions using the stabelebaselines-3 framework (Raffin et al., 2021), which uses pytorch for auto-differentiation (Paszke et al., 2019). The HL-Gauss code was built upon the code by Farebrother et al. (2024). All the environments were implemented within the gymnasium framework (Towers et al., 2023). We use the rlliable code for plotting (Agarwal et al., 2021).

B.10 Hardware For Experiments

For all experiments, we used the following compute infrastructure:

- Distributed cluster on HTCondor framework
- Intel(R) Xeon(R) CPU E5-2470 0 @ 2.30GHz
- RAM: 5GB
- Disk space: 5GB

C Related Work

Several related ideas to the contributions of our paper have been previously explored in the literature.

Policy Search as Supervised Learning We have shown how stochastic policy gradient algorithms with Gaussian policy representations can be viewed as implicitly solving a regression problem. A number of prior works have tried to recast RL as supervised learning. Peters and Schaal (2007) derive the reward-weighted regression algorithm in order to cast policy search in continuous control spaces as a weighted regression problem. Recent works such as MPO (Abdolmaleki et al., 2018), advantage-weighted regression (Peng et al., 2019), and advantage-weighted actor-critic (Nair et al.,

2021) use similar derivations to develop policy search methods that implicitly solve least-squares optimization problems when using Gaussian policies. An alternative approach to RL as SL is upside-down RL that proposes to learn a policy $\pi(s, g)$ by regressing state-return pairs to the action taken in state s before ultimately receiving return g. The optimal action for state s is then predicted as $\pi(s, g^\star)$ (Schmidhuber, 2020). Upside-down RL is also the basis for the decision-transformer approach to offline and online RL (Chen et al., 2021). Our study is complementary to these prior works in its focus on recasting implicit regression as classification in policy search.

Recasting Regression as Classification In the supervised learning literature, several works have studied the empirical and theoretical benefits of replacing the least-squares regression loss with a cross-entropy classification loss. Our approach most closely follows the approach of Imani and White (2018); Imani et al. (2024) due to our use of the HL-Gauss loss as a means to preserve the spatial structure of the action space. Zhang et al. (2023) found that the cross-entropy loss encouraged better representations in regression problems and Pintea et al. (2023) found that casting regression as classification helped with class imbalances. While these findings are focused on the supervised-learning case and 1-hot cross-entropy losses, it would be interesting to see whether similar benefits can be found in policy gradient learning. Lastly, we note that a number of applied works, primarily in computer vision, have found classification formulations of regression to give superior empirical performance (Cao et al., 2018; Kendall et al., 2017; Li et al., 2022; Rothe et al., 2015, e.g.,). The policy gradient learning setting is quite different from supervised learning as the function we are learning also directly determines the data distribution being learned over.

Alternatives to Gaussian Policies in Continuous Action Domains In order to recast continuous action policy gradient learning as a classification problem, we discretized each dimension of the action-space and learned softmax policies over each dimension. As mentioned in the introduction, naive discretization can lead to an exponentially sized action space that would make it intractable to represent the policy. One prior work has addressed this increase in the size of the action space by sequentially selecting the discretized action for each dimension (Metz et al., 2019). Though Metz et al. (2019) introduced this approach to enable q-learning (Watkins and Dayan, 1992) in continuous action domains, it would be interesting to consider new policy parameterizations based on this approach that could be trained with classification losses. Alternative policy distributions such as truncated Gaussians (Fujita and Maeda, 2018), Beta distributions (Chou et al., 2017), and non-parametric distributions Tessler et al. (2019) have also been explored as alternatives to the Gaussian representation. Continuous actions can be directly sampled from these distributions and it would be interesting to investigate if our findings pertain in some form to these alternative representations.