IMPROVING EXPLORATION IN POLICY GRADIENT SEARCH: APPLICATION TO SYMBOLIC OPTIMIZATION

Mikel Landajuela Larma[*], **Brenden K. Petersen**[*†], **Soo K. Kim, Claudio P. Santiago, Ruben Glatt, T. Nathan Mundhenk, Jacob F. Pettit, & Daniel M. Faissol**
Lawrence Livermore National Laboratory
Livermore, CA 94550, USA

ABSTRACT

Many machine learning strategies designed to automate mathematical tasks leverage neural networks to search large combinatorial spaces of mathematical symbols. In contrast to traditional evolutionary approaches, using a neural network at the core of the search allows learning higher-level symbolic patterns, providing an informed direction to guide the search. When no labeled data is available, such networks can still be trained using reinforcement learning. However, we demonstrate that this approach can suffer from an *early commitment* phenomenon and from *initialization bias*, both of which limit exploration. We present two exploration methods to tackle these issues, building upon ideas of entropy regularization and distribution initialization. We show that these techniques can improve the performance, increase sample efficiency, and lower the complexity of solutions for the task of symbolic regression.

1 INTRODUCTION

The application of machine learning to symbolic optimization (SO) tasks such as symbolic regression (SR), automatic equation solving, or program synthesis involves combinatorial search spaces that are vast and complex. In such tasks, the goal is to find a sequence of actions (i.e. symbols) that, upon semantic interpretation, optimizes a desired criteria. For these tasks, uninformed search can suffer from prohibitively large time and memory requirements. Thus, researchers have turned to heuristic or local search strategies, and much attention has been directed to genetic algorithms (Koza, 1992; Schmidt & Lipson, 2009; Bäck et al., 2018). However, the effectiveness of genetic algorithms for these tasks is still debatable (Russell & Norvig, 2002), as they are known to suffer from premature convergence (Lu et al., 2021), especially when the optimization landscape suffers from non-locality (Rothlauf & Oetzel, 2006). Approaches based on Monte Carlo Tree Search (MCTS)

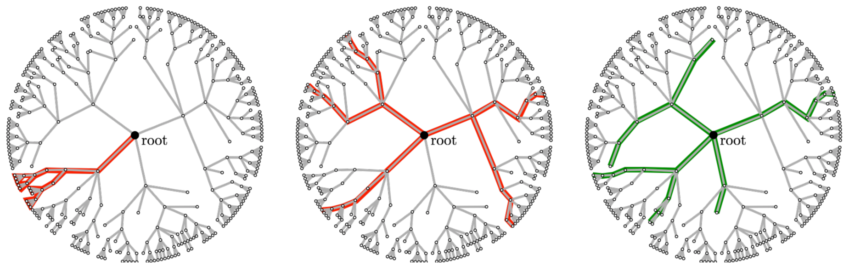

Figure 1: Schematic view of a symbolic search space. Any path from root to a leaf node represents a mathematical expression. **Left.** Without hierarchical entropy, the search tends to choose paths with the same initial branches. **Center.** Without the soft length prior, the search tends to reach the deepest levels of the tree, i.e. it avoids shorter expressions. **Right.** With both exploration techniques, the search explores various initial branches and expression lengths.

[*]Equal contribution. [†]Corresponding author: bp@llnl.gov.

have also been proposed (Li et al., 2019). However, efficient storage of the tree structure can become prohibitively large for spaces with high depth and high branching factor (Anthony et al., 2019).

Recently, a new paradigm has emerged to use deep learning to guide search without the need to store a search tree or have access to labeled data. These approaches are based on deep reinforcement learning, using an autoregressive recurrent neural network (RNN) optimized via policy gradients. Example applications range from automated machine learning (AutoML) to other SO tasks, including neural architecture search (Zoph & Le, 2016), standard combinatorial optimization problems (Bello et al., 2016), neural optimizer search (Bello et al., 2017), searching for activation functions (Ramachandran et al., 2017), program synthesis (Liang et al., 2018), molecular optimization (Popova et al., 2019), and SR (Petersen et al., 2021). Software frameworks for SO include Tree-Based Pipeline Optimization Tool (Olson & Moore, 2016) and PyGlove (Peng et al., 2021).

Given the large search space, intelligent and guided exploration is critical. Leveraging the deep symbolic regression algorithm by Petersen et al. (2021), we demonstrate that standard approaches to exploration (e.g. standard entropy regularizer) can still lead to premature convergence, and that naive weight initialization can result in large initial biases of the distribution over sequence lengths. To address these issues, we introduce two novel exploration techniques: (1) a *hierarchical entropy regularizer* to avoid premature convergence on the first few tokens of newly generated sequences, and (2) a *soft length prior* to promote exploration over sequence lengths. We demonstrate that each of these techniques improves both performance and sample efficiency in benchmark SR tasks.

2 Related Work

Entropy exploration. Haarnoja et al. (2018) demonstrate remarkable success using an entropy regularizer to aid exploration in benchmark control tasks. However, when applied to symbolic search, this approach disproportionately encourages exploration in later search steps. Emphasis of exploration in early steps of the tree search is considered in Anthony et al. (2019) using a bandit rule. However, only the first action of a trajectory is treated differently, with subsequent actions sampled normally. In contrast, our hierarchical entropy regularizer promotes exploration across all tokens based on their position within the search tree.

Priors. The use of priors is an *informed search strategy* that uses knowledge about the problem to guide search and find solutions more efficiently (Russell & Norvig, 2002). Constraints have been used in policy gradient search applications: in generating molecular structures, Popova et al. (2019) use valency constraints to follow rules of organic chemistry; in generating mathematical expressions, Petersen et al. (2021) include various constraints to prevent unrealistic expressions (e.g. nested trigonometric operators) or redundancies (e.g. multiplying two constants together). These constraints are based on domain-specific knowledge; here, in contrast, our soft length prior helps guide the search in a data-driven manner. In other neural-guided search, Li et al. (2019) identify asymptotic constraints of leading polynomial powers and use those constraints to guide Monte Carlo tree search.

3 Policy Gradient Search in Symbolic Spaces

An example symbolic search space is illustrated in Fig. 1. To navigate this space, we consider *policy gradient search* methods such as those discussed above. These works employ a *search policy* $\pi(\tau;\theta)$ with policy parameters θ to generate candidate solutions τ. Each candidate comprises a variable-length sequence of T "tokens" τ_i from a library $\mathcal{L}$, such that $\tau = [\tau_1, \dots, \tau_T]$. For example, τ may represent a neural network architecture (Zoph & Le, 2016) or a mathematical expression (Petersen et al., 2021). In the latter case, the library comprises binary operators (e.g. $+$ or $\div$), unary operators (e.g. $\sin$ or $\log$), and terminals (input variables or constants). Candidates are evaluated using a black-box reward function $R(\tau) \in \mathbb{R}$. The policy itself commonly assumes a RNN architecture. During sampling, the RNN emits logits $\psi_i \in \mathbb{R}^{|\mathcal{L}|}$, which parameterize a categorical distribution from which the ith token τ_i is sampled. Thus, the likelihood over tokens in the library at the ith position in the sequence is given by $\pi(\cdot|\tau_{1:(i-1)};\theta) = \text{softmax}(\psi_i)$. Note that token τ_i is conditionally independent given the previous tokens $\tau_{1:(i-1)}$; this sampling procedure is known as *autoregressive* sampling. The search space can effectively be pruned by penalizing *in situ* the emission ψ_i to avoid undesirable symbolic patterns (see section 4.2 below).

The policy is typically trained to maximize expected reward using policy gradients (Williams, 1992). Specifically, we collect a batch of M candidate solutions (or trajectories in the search space) $\{\tau^{(j)}\}_{j=1}^{M}$ sampled from the current RNN and update the parameters according to:

$$\theta \leftarrow \theta + \alpha \frac{1}{M} \sum_{j=1}^{M} \left(\sum_{i=1}^{|\tau^{(j)}|} \left(\left(R(\tau^{(j)}) - b \right) \nabla_\theta \log \pi(\tau_i^{(j)} | \tau_{1:(i-1)}^{(j)}; \theta) \right) + \eta \nabla_\theta \mathcal{R}_H(\tau^{(j)}) \right), \quad (1)$$

where α is the learning rate, b is a control variate or "baseline", $\mathcal{R}_H$ is an entropy regularizer and η is the entropy weight (see section 4.1 below). We use the *risk-seeking policy gradient* proposed in Petersen et al. (2021) to promote best-case over expected performance. This method uses $b = R_\epsilon$, where R_ϵ is the empirical $(1 - \epsilon)$-quantile of $\{R(\tau^{(j)})\}_{j=1}^{M}$, and filters the batch to contain only the top ϵ fraction of samples. The training procedure presented in (1) encodes knowledge about the most promising paths in the search space in the weights of the RNN, similarly to the knowledge encoded in search trees in MCTS methods.

4 METHODS

The RNN-based policy gradient setup described above is a powerful way to search combinatorial spaces (Bello et al., 2017). However, we show that this framework can suffer from an *early commitment* phenomenon and from *initialization bias* in symbolic spaces, both of which limit exploration. To address these issues, we present two novel exploration techniques for neural-guided SO.

4.1 HIERARCHICAL ENTROPY REGULARIZER

Policy gradient methods typically include an entropy regularizer $\mathcal{R}_H$ in (1) that is proportional to the entropy at each step of autoregressive sampling (Abolafia et al., 2018). This standard entropy regularizer is given by:

$$\mathcal{R}_H(\tau) = \sum_{i=1}^{|\tau|} H[\pi(\cdot|\tau_{1:(i-1)}; \theta)], \quad (2)$$

where $H[\pi(\cdot)] = -\sum_{x \in X} \pi(x) \log \pi(x)$ is the entropy. This entropy regularizer promotes exploration and helps prevent the RNN from converging prematurely to a local optimum. Notably, (2) is simply a sum across time steps i; thus, all time steps contribute equally.

When optimizing discrete sequences (or navigating RL environments with deterministic transition dynamics), each sequence can be viewed as a path through a search tree, as depicted in Fig. 1. As learning progresses (i.e., the entropy around promising paths is reduced), trajectories $\{\tau^{(j)}\}_{j=1}^{M}$ sampled from the RNN tend to satisfy

$$\frac{1}{M} \sum_{j=1}^{M} H[\pi(\cdot|\tau_{1:(i-1)}^{(j)}; \theta)] \leq \frac{1}{M} \sum_{j=1}^{M} H[\pi(\cdot|\tau_{1:(k-1)}^{(j)}; \theta)], \ \forall\, 0 < i < k,$$

by virtue of the sequential sampling process. This causes the sum in (1) corresponding to the entropy regularizer (2) to be concentrated in the later terms, while the earliest terms can quickly approach zero entropy. When this happens, the RNN stops exploring early tokens entirely (and thus entire branches of the search space), which can greatly hinder performance. This phenomenon, which we refer to as the "early commitment problem," can be observed empirically, as shown in Figure 2 (top). Note that the entropy of the first token of the sequence drops toward zero early on in training.

We propose a simple change to combat the early commitment problem: replacing the sum in (2) with a weighted sum whose weights decay exponentially with a factor $\gamma < 1$. Thus, our hierarchical entropy regularizer is given by:

$$\mathcal{R}_H(\tau) = \sum_{i=1}^{|\tau|} \gamma^{i-1} H[\pi(\cdot|\tau_{1:(i-1)}; \theta)]. \quad (3)$$

Intuitively, using (3) encourages the RNN to perpetually explore even the earliest tokens.

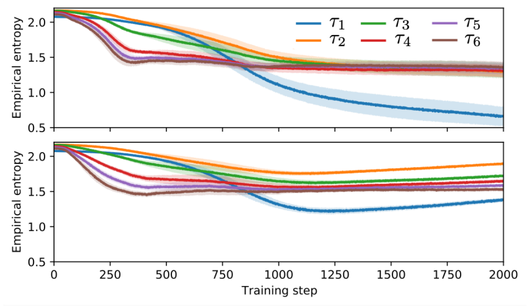

Figure 2: Empirical entropy of tokens τ_1 to τ_6 with standard (top) or hierarchical (bottom) entropy regularizer on the Nguyen-7 SR benchmark.

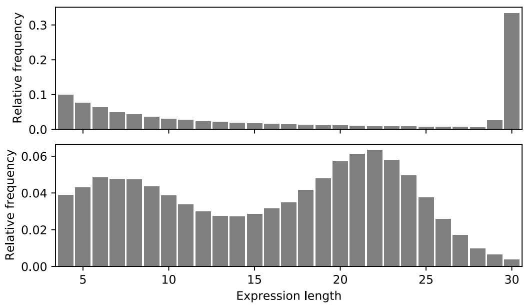

Figure 3: Initial distributions over expression lengths under the RNN without (top) or with (bottom) the soft length prior ($\lambda = 10, \sigma^2 = 20$).

4.2 Soft length prior

Before training begins, using zero-weight initializers, the RNN emissions ψ_i are zero. Thus, the probability distribution over tokens is uniform at all time steps: $\pi(\tau_i) = \frac{1}{|\mathcal{L}|} \forall \tau_i \in \mathcal{L}$. We can inform this starting distribution by including a *prior*, i.e. by adding a logit vector $\psi_\circ \in \mathbb{R}^{|\mathcal{L}|}$ to each RNN emission ψ_i. This transforms the likelihoods over tokens to $\pi(\cdot|\tau_{1:(i-1)}; \theta) = \text{softmax}(\psi_i + \psi_\circ)$. For example, we can ensure that the prior probability of choosing a binary, unary, or terminal token is equal (regardless of the number of each type of token in the library) by solving for $\psi_\circ^{\text{eq}}$ below:

$$\text{softmax}(\psi_\circ^{\text{eq}}) = \left(\frac{1}{3n_2}\right)_{n_2} \| \left(\frac{1}{3n_1}\right)_{n_1} \| \left(\frac{1}{3n_0}\right)_{n_0},$$

where $(\cdot)_n$ denotes that element $(\cdot)$ is repeated n times, $\|$ denotes vector concatenation, and n_2, n_1, n_0 are the number of binary, unary, and terminal tokens in $\mathcal{L}$, respectively. The solution is:

$$\psi_\circ^{\text{eq}} = (-\log n_2)_{n_2} \| (-\log n_1)_{n_1} \| (-\log n_0)_{n_0} + c,$$

where c is an arbitrary constant.

Under the prior $\psi_\circ^{\text{eq}}$, the distribution over expression lengths is heavily skewed toward longer expressions. In particular, the expected expression length is $\mathbb{E}_{\tau \sim \psi_\circ^{\text{eq}}}[|\tau|] = \infty$ (see Feller (1957) for an analogous discussion on random walks). In practice, a *length constraint* is applied; however, empirically this results in the vast majority of expressions having length equal to the maximum allowable length. We show this empirically in Figure 3 (top). This strong initialization bias makes it very difficult for the distribution to *learn* an appropriate expression length. To provide this capability, we introduce an additional *soft length prior* to the RNN's ith emission:

$$\psi_i^{\text{SLP}} = \left(\frac{-(i-\lambda)^2}{2\sigma^2} \mathbf{1}_{i>\lambda}\right)_{n_2} \| (0)_{n_1} \| \left(\frac{-(i-\lambda)^2}{2\sigma^2} \mathbf{1}_{i<\lambda}\right)_{n_0},$$

where λ and σ are hyperparameters. Note that the values of this prior depend on the positional index i in the sequence. In probability space, ψ_i^{SLP} is a multiplicative Gaussian function applied to either binary tokens ($i > \lambda$) or terminal tokens ($i < \lambda$). Thus, ψ_i^{SLP} discourages expressions from being either too short ($i < \lambda$, where probabilities of terminal tokens are reduced) or too long ($i > \lambda$, where probabilities of binary tokens are reduced).

5 Results and Discussion

We demonstrate the benefits of our proposed exploration techniques on the standard Nguyen suite of SR benchmark problems proposed by Uy et al. (2011). The goal is to exactly recover benchmark symbolic expressions from a small synthetic dataset and limited set of tokens. We build upon the open-source implementation of *deep symbolic regression* given in Petersen et al. (2021). We refer to this baseline approach as standard entropy (SE), then introduce the following additional variants: hierarchical entropy (HE), soft length prior with SE (SLP; $\lambda = 10, \sigma^2 = 5$), and SLP with HE (SLP+HE). For each variant, we perform a small hyperparameter grid search over $\eta \in \{0.001, 0.005, 0.01, 0.02, 0.03\}$ and $\gamma \in \{0.7, 0.75, 0.8, 0.85, 0.9\}$. All other hyperparameters are fixed, using values from Petersen et al. (2021). We found best results with $\eta = 0.005$ for SE, $\eta = 0.02$ and $\gamma = 0.85$ for HE, $\eta = 0.005$ for SLP, and $\eta = 0.03$ and $\gamma = 0.7$ for SLP+HE.

In Figure 2 (bottom), we demonstrate the effectiveness of the hierarchical entropy regularizer in ensuring that the RNN maintains diversity across time steps during the course of training, mitigating the tendency to choose paths with the same initial branches (i.e. τ_1). Figure 3 (bottom) shows that including the soft length prior results in a much smoother a priori distribution over expression lengths, reducing the bias toward long expressions. In contrast to using a length constraint (i.e. *hard* length prior) that *forces* each expression to fall between a pre-specified minimum and maximum length, the soft length prior affords the RNN the ability to *learn* the optimal length.

In Table 1, we report recovery rate (fraction of runs in which the exact symbolic expression is found), average number of steps to solve the benchmark (up to 2,000), and the average length of the best found expression. All metrics are averaged over 100 independent runs. We observe improvements in each metric with both HE and SLP techniques separately: recovery rate improves, the search is more sample efficient, and the final expression is more compact. The combination of both provides the best results. Finally, it is worth noting that both contributions improve upon the state-of-the-art results for SR reported in Petersen et al. (2021), which already outperformed several commercial SR software solutions based on genetic algorithms and simulated annealing.

Table 1: Average recovery rate, steps to solve, and length of best found expression across the 12 Nguyen SR benchmarks using SE, HE, and SLP.

Metric	SE	HE	SLP	SLP+HE
Recovery	83.8%	86.0%	84.0%	**88.4%**
Steps	592.6	545.6	516.2	**429.1**
Length	19.86	19.16	16.17	**14.14**

6 Conclusion

The combinatorial nature of symbolic spaces renders exploration a key component to the success of search strategies. Here, we identified that existing policy gradient approaches for SO suffer from an early commitment phenomenon and from initialization bias, and we proposed two novel exploration techniques to address these limitations: a hierarchical entropy regularizer and a soft length prior. We demonstrate that these techniques improve the state-of-the-art in neural-guided search for SR.

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC. LLNL-CONF-820015.

References

Daniel A Abolafia, Mohammad Norouzi, Jonathan Shen, Rui Zhao, and Quoc V Le. Neural program synthesis with priority queue training. *arXiv preprint arXiv:1801.03526*, 2018.

Thomas Anthony, Robert Nishihara, Philipp Moritz, Tim Salimans, and John Schulman. Policy gradient search: Online planning and expert iteration without search trees. *arXiv preprint arXiv:1904.03646*, 2019.

Thomas Bäck, David B Fogel, and Zbigniew Michalewicz. *Evolutionary Computation 1: Basic Algorithms and Operators*. CRC press, 2018.

Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

Irwan Bello, Barret Zoph, Vijay Vasudevan, and Quoc V Le. Neural optimizer search with reinforcement learning. In *Proc. of the International Conference on Machine Learning*, 2017.

William Feller. An introduction to probability theory and its applications. 1957.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv:1801.01290*, 2018.

John R Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, volume 1. MIT press, 1992.

Li Li, Minjie Fan, Rishabh Singh, and Patrick Riley. Neural-guided symbolic regression with asymptotic constraints, 2019.

Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc Le, and Ni Lao. Memory augmented policy optimization for program synthesis and semantic parsing. *arXiv preprint arXiv:1807.02322*, 2018.

Qiang Lu, Fan Tao, Shuo Zhou, and Zhiguang Wang. Incorporating actor-critic in monte carlo tree search for symbolic regression. *Neural Computing and Applications*, pp. 1–17, 2021.

Randal S Olson and Jason H Moore. Tpot: A tree-based pipeline optimization tool for automating machine learning. In *Workshop on automatic machine learning*, pp. 66–74. PMLR, 2016.

Daiyi Peng, Xuanyi Dong, Esteban Real, Mingxing Tan, Yifeng Lu, Hanxiao Liu, Gabriel Bender, Adam Kraft, Chen Liang, and Quoc V Le. Pyglove: Symbolic programming for automated machine learning. *arXiv preprint arXiv:2101.08809*, 2021.

Brenden K Petersen, Mikel Landajuela, T Nathan Mundhenk, Claudio P Santiago, Soo K Kim, and Joanne T Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. *Proc. of the International Conference on Learning Representations*, 2021.

Mariya Popova, Mykhailo Shvets, Junier Oliva, and Olexandr Isayev. Molecularrnn: Generating realistic molecular graphs with optimized properties. *arXiv preprint arXiv:1905.13372*, 2019.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv:1710.05941*, 2017.

Franz Rothlauf and Marie Oetzel. On the locality of grammatical evolution. In *European Conference on Genetic Programming*, pp. 320–330. Springer, 2006.

Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 2002.

Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

Nguyen Quang Uy, Nguyen Xuan Hoai, Michael O'Neill, Robert I McKay, and Edgar Galván-López. Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, 12(2):91–119, 2011.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv:1611.01578*, 2016.