Convex Functions for Reinforcement Learning

Siddartha Devic, Yuqiao Chen, Gregory Van Buskirk, Benjamin Raichel*, Nicholas Ruozzi*
Computer Science Dept., The University of Texas at Dallas

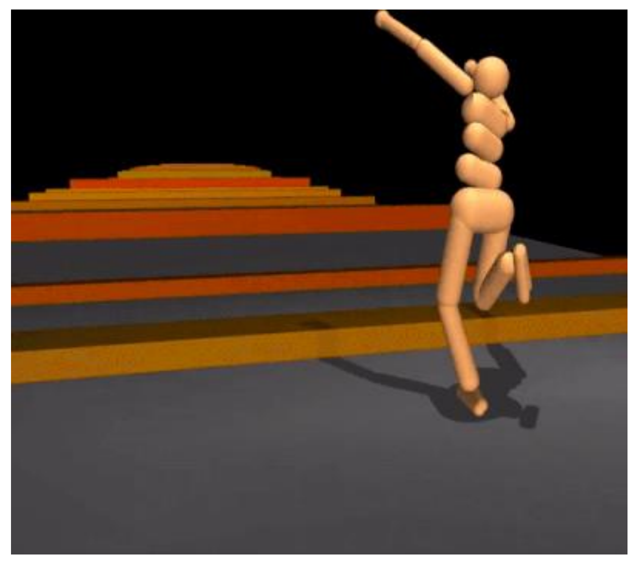
I. Introduction

We introduce a convex function approximator that has provable convergence in Q-learning, a highly applicable model for reinforcement learning. We present our results on a variety of reinforcement learning tasks, and show promising initial results when compared to naively-built "Input Convex Neural Networks".

II. Reinforcement Learning

Generally, reinforcement learning concerns an agent interacting with an environment to complete a task. Surprisingly many problems may fall into the reinforcement learning paradigm. Some notable examples include Google DeepMind training a simulated human how to run around obstacles (Figure 1), a reinforcement learning agent learning to optimize chemical reactions and outperform state of the art methods, and news recommendation systems.

In practice, agents are part of the state, action, reward cycle presented in Figure 2. For example, in the case of the running human problem (Figure 1), the state description would be the position and velocity of each joint. An agent applies an action, in this case torque to each joint, and will then receive a reward proportional to how far the human moves and how little energy was expended. The agent receives a new state, and the process repeats. Eventually, the agent learns to maximize the sum of the rewards at each time step over a finite "episode".



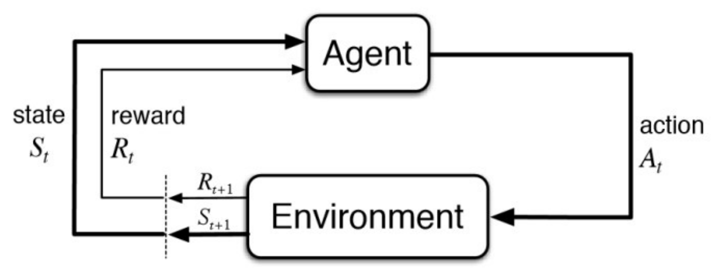


Figure 1 (Left): DeepMind's simulated human agent learns to run and avoid obstacles/falling (source: https://deepmind.com/blog/producing-flexible-behaviours-simulated-environments/). Figure 2 (Right): The agent-environment feedback loop (source: https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html)

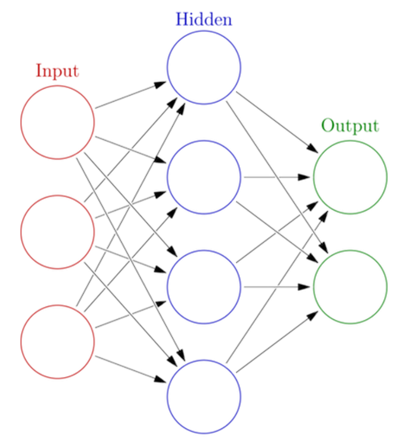
III. Neural Networks

Neural networks are comprised of thousands of nodes and weighted connections between these nodes that attempt to accurately predict an output given a certain input (Figure 3). Neural networks in computer science are loosely modeled after those in the human brain, where each portion of the input is weighted and considered differently. However, in computer science they are simply trained to perform well on a single prediction task and do not have any general intelligence.

Recently, computational advancements have made neural networks tractable for a large subset of machine learning problems, including certain types of reinforcement learning.

IV. Convex Function Fitting

Convex functions are functions where the tangent line lies below the function at every point. They have the special property that any critical point of the function is guaranteed to be a global minimum or maximum. Unfortunately, training neural networks requires "optimizing" (finding the minimum) of non-convex functions, but we cannot guarantee that any local minimum we arrive at will be a global minimum (Figure 4).



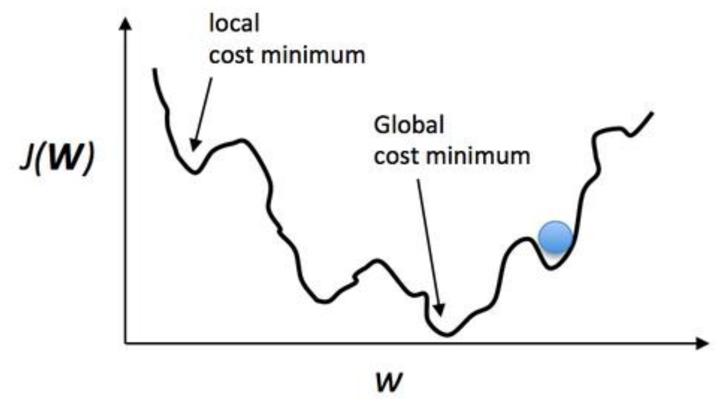


Figure 3 (Left): A simple neural network. Figure 4 (Right): Example non-convex loss function of a neural network.

Piecewise-linear functions are both interpretable and fast, removing the black box nature surrounding neural networks. Boyd [1] gives us a quadratic program to fit the best convex piecewise-linear function,

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{m}(y_i - \hat{y}_i)^2 \\ \text{subject to} & \hat{y}_j \geq \hat{y}_i + g_i^T(u_j - u_i), \quad i,\ j = 1,\dots,m, \end{array}$$

however this is intractable for reinforcement learning. We instead develop a stochastic method to learn the best convex piecewise-linear function to a given set of data. Furthermore, our method provably* converges in the Q-learning setting ([2], V). We begin by computing the convex lower hull, and projecting all points onto it. We then iteratively move our hull upwards, maintaining convexity at each step (Figure 5).

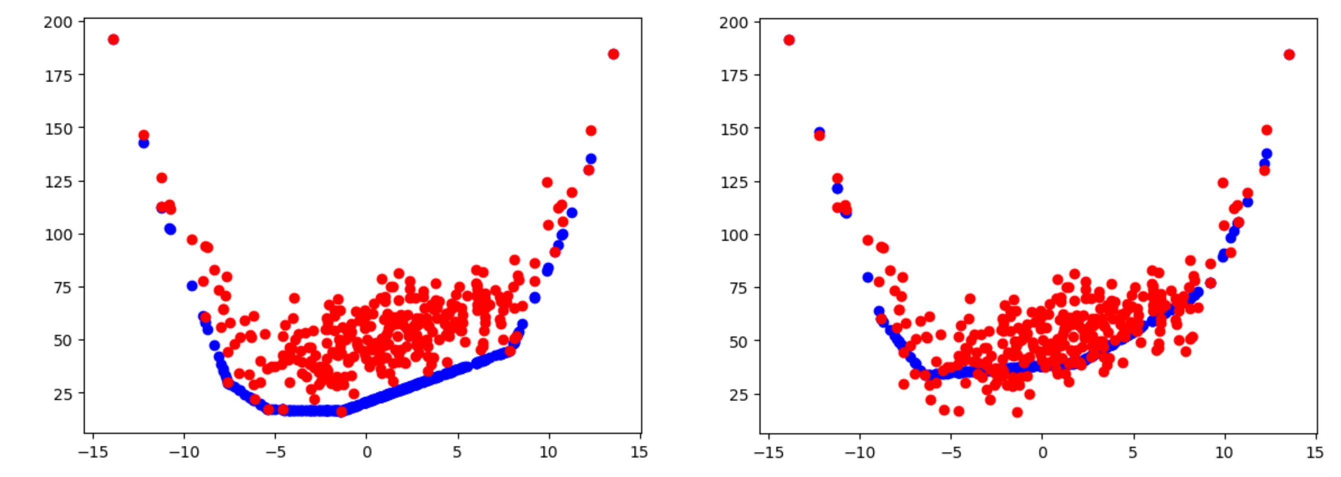


Figure 5: We begin with the convex lower hull (Left), and iteratively move our fit upwards to obtain the best convex-piecewise linear function (Right, 750 iterations).

V. Deep Q-Learning

Q-learning is an approach to reinforcement learning where the agent attempts to learn the long term value (Q-function) of a particular state-action pair. The Q-update rule is defined as:

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

We iterate hundreds of thousands of times in an attempt to learn regions of high and low Q-values. Once training is complete, the agent simply takes actions, maximizing Q-values at each step.

VI. Evaluation and Results

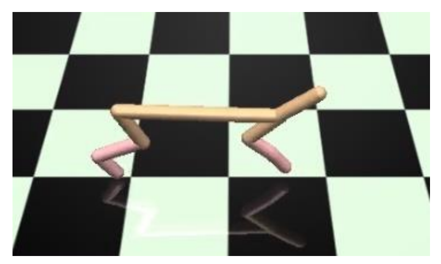
We first compare an iterative version of Boyd's algorithm (cvx-fit) against a single layer neural network (NN) on certain regression tasks:

Dataset	cvx-fit MSE	NN MSE
Malignant tumor radius prediction	0.8815	2.4514
Airfoil noise dataset	38.2339	92.0625
MPG prediction dataset	111.2250	12.8408

The high performance of the cvx-fit algorithm implies that our stochastic variant may perform well on more complex tasks, where cvx-fit is intractable.

We implement the convex-neural network analogue, "fully input convex neural networks (ICNNs)" [3]. Although ICNNs are a good first step in exploring convexity in deep reinforcement learning, we believe that we can outperform them in the Q-learning setting using our simple convex function approximator. Our results are obtained from a reinforcement learning simulation library called openai-gym. The tasks we evaluate our method on are inverted pendulum and half-cheetah (Figure 6).

Task	ICNN Reward
Inverted-Pendulum	1000
Half-Cheetah	**NaN**



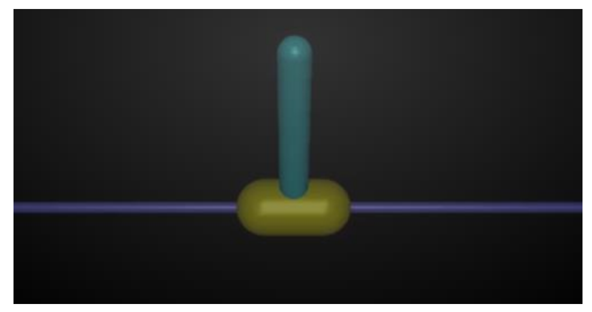


Figure 6: (Left, Inverted Pendulum) The agent can choose to move the platform left or right at each state, attempting to balance the pole without falling. (Above, Half-cheetah) We apply torque to joints and maximize distance traveled.

Note: NaN means non-convergent. The ICNN cannot learn to perform well due to the assumption of convexity in the Q-Function.

Our next step is to apply our stochastic convex piecewise-linear function fitting variant to these same reinforcement learning problems. Given the relatively good results on supervised learning tasks, we expect to be able to outperform ICNNs.

VII. Acknowledgements

Thank you to Prof. Ruozzi, who graciously takes the time to meet with me each week, and points me in new directions with each failure. I would also like to thank Yuqiao, Greg, and Prof. Raichel for the many informative and interesting conversations that have defined my year.

VIII. Citations

[1]: Stephen Boyd and Lieven Vandenberghe. Convex Optimization. Cambridge University Press, New York, NY, USA, 2004.
[2]: G. J. Gordon, "Stable Function Approximation in Dynamic Programming," *Machine Learning Proceedings 1995*, pp. 261–268, 1995.
[3]: Brandon Amos, Lei Xu, and J. Zico Kolter. Input convex neural networks. CoRR, abs/1609.07152, 2016.