# Convex Functions for Reinforcement Learning

**Introduction:** Reinforcement learning (RL) is a subfield of artificial intelligence (AI) wherein an agent learns to optimally interact with a simulated environment by attempting to maximize a reward signal. RL has been used to learn the dynamics of many difficult problems including scheduling for radiation therapy in cancer treatment, optimizing chemical reactions, and playing humans in complex games like "Go".

Unlike "supervised learning", RL does not require humans to label large amounts of data for the system to train on. Therefore, creating practical RL algorithms with guarantees of success are of great importance to the greater scientific community wishing to apply AI to tasks where expert data collection is expensive and **performance critical**. However, popular RL techniques utilizing non-convex neural networks (NNs) suffer from a lack of guarantees. We seek a drop-in replacement for NNs, **potentially allowing for provable guarantees of success** within a popular RL framework called Q-learning.

**Background and Motivation:** Many RL problems can be mathematically described by a Markov Decision Process (MDP), which specifies the dynamics of an environment through a set of states and actions the simulated agent may take to transition between these states, and a reward function which returns a real valued reward for each state-action pair. The agent interacting with the MDP attempts to maximize the *cumulative discounted reward* over a number of time steps. The Q-value of a state-action pair, *Q(s,a)*, represents the relative long term value of taking action a from state *s*. In addition, the optimal Q-value function *Q*(s,a)* has a special property called the Bellman optimality condition given in (1), where *Q*(s,a)* is a function of the returned reward, discount factor γ, and Q*-value of the best action possible in the subsequent state *s'*. Deep Q-learning is a popular RL paradigm where NNs are used to approximate the optimal Q-value function *Q**. During training, the NN is successively updated using gradients of the Bellman condition (1).

$$Q^*(s,a) = r + \gamma \max_{a'} Q^*(s',a') \qquad (1)$$

In practice, it is not clear that using NNs as Q-value function approximators is optimal since the maximization procedure for choosing the best possible action in a given state *s'* in (1) is non-concave. Motivated by this, Amos et al. [1] introduce *input convex* NNs (ICNNs) defined as the composition of convex activation functions and NN layers with non-negative weights. ICNNs are thus piecewise-linear functions which are convex in their inputs. In using such ICNNs to model the Q-function, the authors are able to exploit this property in the Bellman optimality condition (1) to obtain precisely the best action for a given state. This ensures that they arrive at the maximal Q-value during training and inference.

However, training ICNNs as Q-value function approximators in the context of RL is practically challenging and does not have convergence guarantees: we observed that the fully learned ICNN Q-value function approximator can diverge to *arbitrarily bad fits* depending on choices of the hyperparameters.

**Objective:** We seek a new convex function approximation scheme for Q-learning independent of ICNNs which simultaneously **obtains high performance** and features **guarantees of successful learning**.

**Approximate Convex Hulls:** There are two phases to a single *episode* of the Q-learning algorithm. In **Phase I**, the agent collects data from an MDP using a fixed action selection strategy. **Phase II** has the agent analyze data collected during the first phase and modify its choices of actions according to the Bellman condition (1). These phases repeat over many episodes until satisfactory performance is achieved. Experience replay is a common technique in deep RL by Lin [2] which modifies the standard Q-learning algorithm to instead *store* all observations during phase I. The agent then samples from the set of all stored observations to learn from in phase II.

We propose maintaining an *approximate convex hull* of all sampled observations from phase II of Q-learning. The benefit is twofold: we **retain geometric information** between episodes, and can **fit a convex function over the hull**. In phase II of each episode in Q-learning, we have a set of points which currently make up our hull. We then sample a number of additional points and run an algorithm from Buskirk et al. [3] to obtain a *sparse approximate convex hull* of this combined set. What remains is a convex set approximating the convex hull of all sampled points, including points from previous episodes.

**Convex Functions Over Convex Hulls:** By the previous algorithm, we are given a set of *representative points H* and now need an appropriate function for phase I of Q-learning. A natural first approach to emulate

ICNNs would have us fit the best convex piecewise-linear function over *H*. However, we observed that this optimization problem is ill-posed since the function is not well defined for points outside of *H*. Instead, we describe a new optimization procedure for fitting a convex function defined even outside the hull. Our proposed function is the “bounded lower envelope” given by the hyperplanes of bounded slope most closely approximating the lower bound of the convex hull of *H*. This naturally extends the function outside of *H*.

Our goal is to **prove that our method is approximately optimal and has guarantees of success during learning**. We aim to show this by using the approximation guarantees for the approximate convex hull procedure obtained by Buskirk et al. [3], given an assumption that the *true* Q-function is convex with bounded slope. This assumption may allow us to bound the error between our Q-function approximator and the correct solution *Q**. We seek an error bound that is a polynomial function of the number of Q-learning episodes, which would represent a **theoretically efficient guarantee of learning success.**

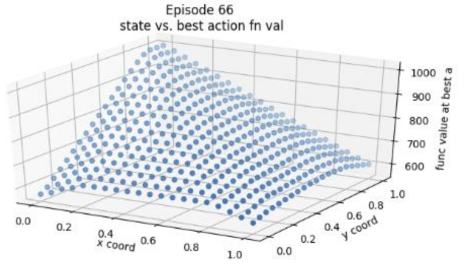

**Figure 1:** 2-D state vs. Q-value of best action within the state. Learned “envelope” function is concave and maximized at top left corner.

**<u>Preliminary Results and Evaluation:</u>** We now have the required pieces to implement our new Q-function estimator into the learning algorithm. We begin by testing our algorithm in a simple environment. The agent starts somewhere within a 1x1 Euclidean grid, taking 2-dimensional steps as actions. It receives 1000 reward for arriving near the top left corner. Otherwise, the agent is penalized. Therefore, the optimal Q-function is concave with a maximum value of 1000 near the grid point [0,1]. **Figure 1** verifies that the learned Q-function is optimal by plotting the value of the best action across the entire grid.

We will comprehensively evaluate our method within various OpenAI gym RL simulations. With hyperparameter fine-tuning, we believe we can perform competitively with ICNNs on difficult RL tasks such as having simulated robots learn to grasp objects or walk on uneven ground. Unlike ICNNs, we may also be able to **provide guarantees of learning success**.

**<u>Adding Expressivity with Kernels:</u>** Although we have promising initial results, we have made the strong assumption that the optimal *Q*(s,a)* is concave over both states and actions. However, we have in fact designed our procedure with *kernels* in mind. Kernels transform the space that we are learning our convex function in by decoupling the space that the hypothesis is represented in from the space that the data resides in. This enables the learning and optimization of a simple, convex function in the transformed space that corresponds to a complex Q-function fitting the data in the original space. For example, Gaussian kernels map states into an *arbitrary* dimensional space. Kernels generalize our procedure to a wider class of Q-functions, while retaining our ability to exactly optimize for an action *a’* within (1).

**<u>Intellectual Merit:</u>** State-of-the-art approaches to RL utilizing NNs do not have learning or correctness guarantees. We may be able to achieve high performance *and* show correctness guarantees which *do not exist for all NN approaches*, as our function behaves well everywhere (unlike NNs). We also determine a set of representative points *H* which may **succinctly** and **completely** describe a solution to a particular MDP. This advances theoretical and practical knowledge by revealing an interpretable set of anchors determining the entire learning environment, which could potentially be useful to *any* learning algorithm.

**<u>Broader Impacts:</u>** Effective, provable methods like ours will contribute to the adaptation of RL in critical domains such as healthcare or defense. We will **release an open source software package** allowing our method to be dropped into *any interdisciplinary RL environment*. For example, clinicians could use our complex, parametrized Q-function in drug scheduling for patients (similar to Padmanabhan et al. [4]) by creating a virtual RL environment based on patient feedback and applying our solver. This represents a step towards creating nuanced, well behaved AI systems for sensitive domains with scarce expert data.

**<u>References:</u>** [1] B. Amos, L. Xu, and J. Z. Kolter. (2017). Input convex neural networks. *ICML*. [2] Lin, L. (1992). Reinforcement learning for robots using neural networks. *Machine Learning*. [3] Buskirk, G.V., Raichel, B., & Ruozzi, N. (2017). Sparse Approximate Conic Hulls. *NeurIPS*. [4] Padmanabhan R. et al. (2017). Reinforcement learning-based control of drug dosing. *Mathematical Biosciences*.