A New Softmax Operator for Reinforcement Learning

Kavosh Asadi  
Brown University

Michael L. Littman  
Brown University

Abstract

A softmax operator applied to a set of values acts somewhat like the maximization function and somewhat like an average. In sequential decision making, softmax is often used in settings where it is necessary to maximize utility but also to hedge against problems that arise from putting all of one’s weight behind a single maximum utility decision. The Boltzmann softmax operator is the most commonly used softmax operator in this setting, but we show that this operator is prone to misbehavior. In this work, we study an alternative softmax operator that, among other properties, is both a non-expansion (ensuring convergent behavior in learning and planning) and differentiable (making it possible to improve decisions via gradient descent methods). We provide proofs of these properties and present empirical comparisons between various softmax operators.

1. Introduction

There is a fundamental tension in decision making between choosing the action that has highest expected reward estimate and avoiding “starving” the other actions. The issue arises in the context of the exploration–exploitation dilemma (Thrun, 1992), non-stationary decision problems (Sutton, 1990), and when interpreting observed decisions (Baker et al., 2007).

In the reinforcement learning setting, a typical approach to addressing this tension is the use of softmax operators for value function optimization and softmax policies for action selection. Examples of this commonly used approach include on-policy value function based methods such as SARSA (Rummery & Niranjan, 1994) or expected SARSA (Sutton & Barto, 1998; Van Seijen et al., 2009), and policy search methods such as REINFORCE (Williams, 1992).

An ideal softmax operator is a parameterized set of operators that:

1. has parameter settings that allow it to approximate maximization arbitrarily accurately (allowing for reward seeking behavior);
2. is a non-expansion for all parameter settings (guaranteeing convergence to a fixed point);
3. is differentiable (making it possible to improve via gradient descent); and
4. puts non-zero weight on non-maximizing actions (to avoid starving non-maximizing actions).

Let \( X = x_1, \ldots, x_n \) be a vector of values. We define the following operators:

\[
\text{max}(X) = \max_{i \in \{1, \ldots, n\}} x_i,
\]

\[
\text{mean}(X) = \frac{1}{n} \sum_{i=1}^{n} x_i,
\]

\[
\text{eps}_\epsilon(X) = \epsilon \text{mean}(X) + (1 - \epsilon) \text{max}(X),
\]

\[
\text{boltz}_\beta(X) = \frac{\sum_{i=1}^{n} x_i e^{\beta x_i}}{\sum_{i=1}^{n} e^{\beta x_i}}.
\]

The first operator, \( \text{max}(X) \), is known to be a non-expansion (Littman & Szepesvári, 1996). However, it is non-differentiable (Property 3), and ignores non-maximizing selections (Property 4).

The next operator, \( \text{mean}(X) \), computes the average of its inputs. It is differentiable and, like any operator that takes a fixed convex combination of its inputs, is a non-expansion.

However, it does not allow for maximization (Property 1).
The third operator $\epsilon_\beta(X)$, commonly referred to as epsilon greedy (Sutton & Barto, 1998), interpolates between max and mean. Though the operator is a non-expansion, it is non-differentiable (Property 3).

The Boltzmann operator $\text{boltz}_\beta(X)$, in contrast, is differentiable. It also approximates $\max$ as $\beta \to \infty$, and mean as $\beta \to 0$. However, it is not a non-expansion, as will be shown in a later section (Property 2).

In the following section we provide a simple example illustrating why the non-expansion property is important, especially in the context of planning and on-policy learning. We then present an alternative softmax operator that is similar to the Boltzmann operator yet is a non-expansion. We then prove several critical properties of this new operator and conclude with other possible applications.

## 2. Boltzmann Operator Is Prone to Misbehavior

We first show that $\text{boltz}_\beta$ can lead to problematic behavior in learning. To this end, we ran SARSA with Boltzmann softmax policy (Algorithm 1) on the MDP in Figure 1. The edges are labeled with a transition probability (unsigned) and a reward (signed). Note that in this MDP rewards are functions of states and actions, and not the next states. Also, state $s_2$ is a terminal state, so we only consider two action values, namely $\hat{Q}(s_1, a)$ and $\hat{Q}(s_2, b)$. Recall that Boltzmann softmax policy assigns the following probability to each action:

$$
\pi(a|s) = \frac{e^{\beta \hat{Q}(s,a)}}{\sum_a e^{\beta \hat{Q}(s,a)}}.
$$

In Figure 2, we plot state–action value estimates at the end of each episode of a single run (smoothed by averaging over ten consecutive points). We set $\alpha = .1$ and $\beta = 16.53$. The values and policy never converge.

### Algorithm 1 SARSA with Boltzmann softmax policy

```
Input: initial $Q(s, a) \forall s \in S \forall a \in A$, $\alpha$, and $\beta$

for each episode do
    Initialize $s$
    $a \sim$ Boltzmann softmax with parameter $\beta$
    repeat
        Take action $a$, observe $r, s'$
        $a' \sim$ Boltzmann softmax with parameter $\beta$
        $\hat{Q}(s, a) \leftarrow \hat{Q}(s, a) + \alpha [r + \gamma \hat{Q}(s', a') - \hat{Q}(s, a)]$
        $s \leftarrow s', a \leftarrow a'$
    until $s$ is terminal
end for
```

SARSA is known to converge in the tabular setting using $\epsilon$-greedy exploration (Littman & Szepesvári, 1996), under decreasing exploration (Singh et al., 2000), and to a region in the function approximation setting (Gordon, 2001). There are also variants of the SARSA update rule that converge more generally (Perkins & Precup, 2002; Baird & Moore, 1999; Van Seijen et al., 2009). However, this example is the first, to our knowledge, to show that SARSA fails to converge in the tabular setting with Boltzmann softmax. The next section provides background for our analysis of this example.

## 3. Background

A Markov decision process (Puterman, 1994), or MDP, is specified by the tuple $(S, A, R, P, \gamma)$, where $S$ is the set of states and $A$ is the set of actions. The functions $R$ and $P$ denote the reward and transition dynamics of the MDP. More precisely, the expected immediate reward following an action $a \in A$ in a state $s \in S$ before moving to a next state $s' \in S$ is specified by:

$$
R(s, a, s') = \mathbb{E}[R_{t+1}|S_t = s, A_t = a, S_{t+1} = s']
$$
and the probability of this transition is defined by:

$$\mathcal{P}(s, a, s') = \Pr(S_{t+1} = s' \mid S_t = s, A_t = a).$$

Finally, $\gamma \in [0, 1)$, the discount rate, determines the relative importance of immediate reward as opposed to the rewards received in the future.

A typical approach to finding a good policy is to define and estimate how good it is to be in a particular state—the state value function. The value of a particular state $s$ given a policy $\pi$ is formally defined to be:

$$V_\pi(s) = \mathbb{E}_\pi \left[ R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s \right].$$

It can also be useful to define the state–action value function, formally defined as:

$$Q_\pi(s, a) = \mathbb{E}_\pi \left[ R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s, A_t = a \right],$$

which is the expected sum of future discounted rewards upon taking an action $a$ in a state $s$ and committing to policy $\pi$ thereafter.

We define the optimal value of a state–action pair

$$Q^*(s, a) = \max_\pi Q_\pi(s, a).$$

It is possible to define $q^*(s, a)$ recursively and as a function of the optimal value of the other state action pairs:

$$Q^*(s, a) = \sum_{s' \in S} R(s, a, s') + \gamma \mathcal{P}(s, a, s') \max_{a'} Q^*(s', a').$$

Bellman equations, such as the above equation, are at the core of many reinforcement-learning algorithms.

Value iteration is an example of a fundamental planning algorithm for MDPs. It computes the value of the best policy in an iterative fashion using the update rule:

$$Q(s, a) \leftarrow \sum_{s' \in S} R(s, a, s') + \gamma \mathcal{P}(s, a, s') \max_{a'} Q(s', a').$$

Regardless of its initial value, $\hat{Q}$ will then converge to $Q^*$. $\hat{Q}$ can then be used for decision making.

(Littman & Szepesvári, 1996) generalized this approach by replacing the max operator by any arbitrary operator $\otimes$, resulting in the generalized value iteration (GVI) algorithm with the following update rule:

$$\hat{Q}(s, a) \leftarrow \sum_{s' \in S} R(s, a, s') + \gamma \mathcal{P}(s, a, s') \otimes \hat{Q}(s', a').$$

Regardless of its initial value, $\hat{Q}$ will then converge to $Q^*$. $\hat{Q}$ can then be used for decision making.

(Algorithm 2 GVI algorithm)

```latex
\begin{algorithm}
\caption{GVI algorithm}
\begin{algorithmic}
\State \textbf{Input:} initial $\hat{Q}(s, a)$ \forall $s \in S$ \forall $a \in A$ and $\delta \in \mathbb{R}^+$
\Repeat
\For {each $s \in S$}
\For {each $a \in A$}
\State $Q_{\text{copy}} \leftarrow \hat{Q}(s, a)$
\State $Q(s, a) \leftarrow \sum_{s' \in S} R(s, a, s')$
\State $Q(s, a) \leftarrow Q(s, a) + \gamma \mathcal{P}(s, a, s') \hat{Q}(s', a')$
\State $\text{diff} \leftarrow \max_a \{\text{diff}, |Q_{\text{copy}} - \hat{Q}(s, a)|\}$
\EndFor
\EndFor
\Until {\text{diff} < \delta}
\end{algorithmic}
\end{algorithm}
```

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{gvi_diagram.png}
\caption{An illustration of non-expansion property for the max operator. While max is shown to be a non-expansion (Littman, 1996), Boltzmann softmax operator is not a non-expansion.}
\end{figure}

Crucially, convergence of GVI to a unique fixed point follows if operator $\otimes$ is a non-expansion:

$$\max_a |\hat{Q}(s, a) - \hat{Q}'(s, a)| \leq \max_a |\hat{Q}(s, a) - Q_2(s, a)|,$$

for any $\hat{Q}$, $\hat{Q}'$ and $s$. As mentioned earlier, the max operator is shown to be a non-expansion, as illustrated by Figure 3. Also, mean, and $\text{eps}_t$ operators are non-expansions. Therefore, each of these operators causes GVI to converge to the corresponding unique fixed point. However, the Boltzmann softmax operator, $\text{boltz}_t$, is not a non-expansion (Littman, 1996), and so, its fixed point may not be unique.

Note that we can relate GVI to SARSA by noticing that SARSA update can be thought of as a stochastic implementation of GVI update. For example, under a Boltzmann softmax policy we have:
5. Mellowmax and its Properties

We advocate an alternative softmax operator defined as follows:

$$mm_\omega(X) = \log\left(\frac{1}{n} \sum_{i=1}^{n} e^{\omega x_i}\right) / \omega,$$

which can be viewed as a particular instantiation of the quasi-arithmetic mean (Beliakov et al., 2016).

We show that $mm_\omega(X)$, which we refer to as mellowmax, has all four of the desired properties and compares quite favorably to boltz$\beta$ in practice.

5.1. Mellowmax is a Non-Expansion

We prove that $mm_\omega$ is a non-expansion (Property 2), and therefore, GVI under $mm_\omega$ is guaranteed to converge to a unique fixed point.

Let $X = x_1, \ldots, x_n$ and $Y = y_1, \ldots, y_n$ be two vectors of values. Let $\Delta_i = x_i - y_i$ for $i \in \{1, \ldots, n\}$ be the difference of the $i$th components of the two vectors. Also, let $i^*$ be the index with the maximum component-wise difference, $i^* = \arg\max_i \Delta_i$. For simplicity, we assume that $i^*$ is unique. Also, without loss of generality, we assume that $x_{i^*} - y_{i^*} \geq 0$. It follows that:

$$|mm_\omega(X) - mm_\omega(Y)| = \left| \log\left(\frac{1}{n} \sum_{i=1}^{n} e^{\omega x_i}\right) / \omega - \log\left(\frac{1}{n} \sum_{i=1}^{n} e^{\omega y_i}\right) / \omega \right|$$

$$= |\log\left(\frac{1}{n} \sum_{i=1}^{n} e^{\omega x_i}\right) / \omega - \log\left(\frac{1}{n} \sum_{i=1}^{n} e^{\omega y_i}\right) / \omega|$$

$$\leq \log (\frac{1}{n} \sum_{i=1}^{n} e^{\omega y_i}) / \omega - \log (\frac{1}{n} \sum_{i=1}^{n} e^{\omega y_i}) / \omega$$

$$= |\log\left(\frac{1}{n} \sum_{i=1}^{n} e^{\omega y_i}\right) / \omega - \log\left(\frac{1}{n} \sum_{i=1}^{n} e^{\omega y_i}\right) / \omega|$$

$$= |\log(e^{\omega \Delta_{i^*}}) / \omega - |\Delta_{i^*}| = |x_{i^*} - y_{i^*}|,$$

allowing us to conclude that mellowmax is a non-expansion.

Experiments confirm that under mellowmax convergence is consistent and rapid.

5.2. Maximization

Mellowmax includes parameter settings that allow for maximization (Property 1) as well as for minimization. In
particular, as $\omega$ goes to infinity, $\text{mm}_\omega$ acts like $\max$.

Let $m = \max(X)$ and let $W = \{x_i = m|i \in \{1, \ldots, n\}\}$. Note that $W \geq 1$ is the number of maximum values ("winners") in $X$. Then:

\[
\lim_{\omega \to \infty} \text{mm}_\omega(X) = \lim_{\omega \to \infty} \frac{\log(\frac{1}{n} \sum_{i=1}^{n} e^{\omega x_i})}{\omega} = \lim_{\omega \to \infty} \frac{\log(\frac{1}{n} \omega^m \sum_{i=1}^{n} e^{\omega(x_i - m)})}{\omega} = \lim_{\omega \to \infty} \frac{\log(\frac{1}{n} \omega^m W)}{\omega} = \lim_{\omega \to \infty} \frac{\log(e^{\omega m}) - \log(n) + \log(W)}{\omega} = \frac{m + \lim_{\omega \to \infty} \frac{-\log(n) + \log(W)}{\omega}}{\omega} = m = \max(X).
\]

That is, the operator acts more and more like pure maximization as the value of $\omega$ is increased. Conversely, as $\omega$ goes to $-\infty$, the operator approaches the minimum.

\[
\lim_{\omega \to -\infty} \text{mm}_\omega(X) = \lim_{\omega \to -\infty} \text{mm}_{-\omega}(X) = \lim_{\omega \to -\infty} -\text{mm}_\omega(-X) = -\max(-X) = \min(X).
\]

5.3. Derivatives

We can take the derivative of mellowmax with respect to each one of the arguments $x_i$ and for any non-zero $\omega$:

\[
\frac{\partial \text{mm}_\omega(X)}{\partial x_i} = \frac{e^{\omega x_i}}{\sum_{i=1}^{n} e^{\omega x_i}} \geq 0.
\]

Note that the operator is non-decreasing in each component of $X$.

Moreover, we can take the derivative of mellowmax with respect to $\omega$. We define $n_\omega(X) = \log(\frac{1}{n} \sum_{i=1}^{n} e^{\omega x_i})$ and $d_\omega(X) = \omega$. Then:

\[
\frac{\partial n_\omega(X)}{\partial \omega} = \frac{\sum_{i=1}^{n} x_i e^{\omega x_i}}{\sum_{i=1}^{n} e^{\omega x_i}} \quad \text{and} \quad \frac{\partial d_\omega(X)}{\partial \omega} = 1,
\]

and so:

\[
\frac{\partial \text{mm}_\omega(X)}{\partial \omega} = \frac{n_\omega(X) d_\omega(X) - n_\omega(X) \omega d_\omega(X)}{d_\omega(X)^2},
\]

ensuring differentiability of the operator (Property 3).

5.4. Averaging

Because of the division by $\omega$ in the definition of $\text{mm}_\omega$, the parameter $\omega$ cannot be set to zero. However, we can examine the behavior of $\text{mm}_\omega$ as $\omega$ approaches zero and show that the operator computes an average in the limit.

Since both the numerator and denominator go to zero as $\omega$ goes to zero, we will use L'Hôpital's rule and the derivative given in the previous section to derive the value in the limit:

\[
\lim_{\omega \to 0} \text{mm}_\omega(X) = \lim_{\omega \to 0} \frac{\log(\frac{1}{n} \sum_{i=1}^{n} e^{\omega x_i})}{\omega} = \lim_{\omega \to 0} \frac{1}{n} \sum_{i=1}^{n} x_i e^{\omega x_i} = \frac{1}{n} \sum_{i=1}^{n} x_i = \text{mean}(X).
\]

That is, as $\omega$ gets closer to zero, $\text{mm}_\omega(X)$ approaches the mean of the values in $X$.

6. Maximum Entropy Mellowmax Policy

As described, $\text{mm}_\omega$ computes a value for a list of numbers somewhere between its minimum and maximum. However, it is often useful to actually provide a probability distribution over the actions such that (1) a non-zero probability mass is assigned to each action, and (2) the resulting expected value equals the computed value. Such a probability distribution can then be used for action selection in algorithms such as SARSA.

In this section, we address the problem of identifying such a probability distribution as a maximum entropy problem—over all distributions that satisfy the properties above, pick the one that maximizes information entropy (Cover & Thomas, 2006; Peters et al., 2010). We formally define the maximum entropy mellowmax policy of a state $s$ as:

\[
\pi_{\text{ME}}(s) = \arg\min_{\pi} \sum_{a \in A} \pi(a|s) \log \left( \pi(a|s) \right)
\]

subject to \( \sum_{a \in A} \pi(a|s) \hat{Q}(s,a) = \text{mm}_\omega(\hat{Q}(s,\cdot)) \)

Note that this optimization problem is convex and can be solved reliably using any numerical convex optimization library.

One way of finding the solution, which leads to an interesting policy form, is to use the method of Lagrange multipliers. Here, the Lagrangian is:
Taking the derivative of the Lagrangian with respect to each \( \pi(a|s) \) and setting them to zero, we obtain:

\[
\frac{\partial L}{\partial \pi(a|s)} = \log(\pi(a|s)) + 1 - \lambda_1 - \lambda_2 \hat{Q}(s,a) = 0 \quad \forall a \in A.
\]

These \(|A|\) equations, together with the two linear constraints, form \(|A| + 2\) equations to constrain the \(|A| + 2\) variables, \( \pi(a|s) \) \( \forall a \in A \) and the two Lagrangian multipliers \( \lambda_1 \) and \( \lambda_2 \).

Solving this system of equations, the probability of taking an action in the maximum entropy mellowmax policy has the form:

\[
\pi_{ME}(a|s) = \frac{e^{\beta \hat{Q}(s,a)}}{\sum_{a \in A} e^{\beta \hat{Q}(s,a)}} \quad \forall a \in A,
\]

where \( \beta \) is a value of \( y \) for which:

\[
\sum_{a \in A} e^{y \left( \hat{Q}(s,a) - \text{mm}_\omega(\hat{Q}(s,\cdot)) \right)} \left( \hat{Q}(s,a) - \text{mm}_\omega(\hat{Q}(s,\cdot)) \right) = 0.
\]

is zero. The value of \( \beta \) can be found easily using any root-finding algorithm. In particular, in our experiments we used Brent’s method (Brent, 2013) available in python’s numpy library.

It is simple to show that a unique root always exists. As \( \omega \) gets higher, the term corresponding to the best action dominates, and so, the function is positive. Conversely, as beta goes to \(-\infty\), the term corresponding to the action with lowest utility dominates, and so the function is negative. Finally, by taking the derivative it is clear that the function is monotonically increasing, allowing us to conclude that there exists only a single answer.

This policy has the same form as Boltzmann softmax, but with a parameter \( \beta \) whose value depends indirectly on \( \omega \). This mathematical form arose not from the structure of \( \text{mm}_\omega \), but from maximizing the entropy. One way to view the use of the mellowmax operator, then, is as a form of Boltzmann softmax policy with a parameter chosen adaptively to ensure that the non-expansion property holds.

Finally, note that the SARSA update under the maximum entropy mellowmax policy could be thought of as a stochastic implementation of the GVI update under \( \text{mm}_\omega \) operator:

\[
\mathbb{E}_\pi \left[ r + \gamma \hat{Q}(s',a') - \hat{Q}(s,a) \mid s,a \right] = \sum_{s' \in S} \mathcal{R}(s,a,s') + \\
\gamma \mathbb{P}(s,a,s') \sum_{a' \in A} \pi(a'|s') \hat{Q}(s',a') - \hat{Q}(s,a)
\]

deep due to the first constraint of the above optimization problem. As such, SARSA with the maximum entropy mellowmax policy is guaranteed to converge.

7. Experiments On MDPs

Next, we repeat the experiment from Figure 2 using SARSA with the maximum entropy mellowmax policy and with \( \omega = 16.53 \). The results, presented in Figure 6, show rapid convergence to the unique fixed point. Analogously to Figure 5, we provide a vector field for GVI updates under \( \text{mm}_\omega=16.53 \). As shown above, using \( \text{mm}_\omega \) ensures that GVI updates move estimates steadily to the fixed point. As a result, GVI under \( \text{mm}_\omega \) can terminate significantly faster.
Figure 8. Number of iterations before termination of GVI on the example MDP. GVI under mm$^\beta$ outperforms the alternatives.

Figure 9. Multi-passenger taxi domain consists of 336 states and 4 actions with slip probability 0.1. The discount rate $\gamma$ is 0.99. Reward is +1 for delivering one passenger, +3 for two passengers, and +15 for three passengers. Reward is zero for all the other transitions.

than GVI under boltz$^\beta$, as illustrated in Figure 8.

We now present two experiments on standard reinforcement learning domains. The first experiment compares softmax policies when used in SARSA with a tabular representation. The second experiment is a policy gradient experiment where a deep neural network is used to directly represent the policy.

7.1. Multi-passenger Taxi Domain

We evaluated SARSA with various policies on the multi-passenger taxi domain introduced by (Dearden et al., 1998). (See Figure 9.)

One challenging aspect of this domain is that it admits many locally optimal policies. Exploration needs to be set carefully to avoid either over-exploring or under-exploring the state space. Note also that Boltzmann softmax performs remarkably well on this domain, outperforming sophisticated bayesian reinforcement learning algorithms. (Dearden et al., 1998)

As shown in Figure 10, SARSA with the epsilon-greedy policy performs poorly. In fact, in our experiment, the algorithm rarely was able to deliver all the passengers. However, SARSA with Boltzman softmax and SARSA with the maximum entropy mellowmax policy achieved significantly higher average reward. Maximum entropy mellowmax policy is no worse than Boltzmann softmax here.

Figure 10. A comparison between various policies on the multi-passenger taxi domain. Results are shown for different values of $\epsilon$, $\beta$, and $\omega$. For each parameter setting, the stepsize is optimized. Results are also averaged over 25 independent runs each consisting of 300000 time steps.

Figure 11. A screenshot of the lunar lander domain.

7.2. Lunar Lander Domain

In this section we evaluate maximum entropy mellowmax policy in the context of the policy gradient algorithms. Specifically, we represent policy by a deep neural network (we discuss the details of the network below). Usually the activation function of the last layer of the network is a softmax function, and typically Boltzmann softmax boltz$^\beta$ is used. Alternatively, we use maximum entropy mellowmax policy, presented in section 6, by treating the input of the activation function as the $\hat{Q}$ values.

We used the lunar lander domain, from OpenAI Gym (Brockman et al., 2016), as our benchmark. A screenshot of the domain is presented in Figure 11, and the code of the domain is publicly available. This domain has a continuous state space with 8 dimensions, namely x-y coordinates, x-y velocities, angle and angular velocities, and leg-touchdown sensors. There are 4 discrete actions to control 3 engines. The reward is +100 for a safe landing in the designated area, and -100 for a crash. There is a small shaping reward for approaching the landing area. Using the engines will also result in a negative reward. Episode finishes when the
A New Softmax Operator for Reinforcement Learning

Figure 12. A comparison of Boltzmann softmax (top) and maximum entropy mellowmax (middle) policies in the lunar lander domain. We also plot mean return over all episodes (bottom). Results are averaged over 400 independent runs.

spacecraft crashes or lands. Solving the domain is defined as maintaining mean episode return higher than 200 in 100 consecutive episodes.

The policy is represented by a neural network with a hidden layer comprised of 16 units with RELU activation function, followed by a second layer with 16 units and softmax activation functions. We used REINFORCE to train the network. A batch episode size of 10 is used, as we had stability issues with a batch size of 1. We used Adam algorithm (Kingma & Ba, 2014) with $\alpha = 0.005$ and set the other parameters as suggested by the paper. We used Keras (Chollet, 2015) and Theano (Team et al., 2016) to implement the neural network.

For each softmax policy, we present in Figure 12, the learning curves for different values of their free parameter. We further plot average return over all 40000 episodes. Mellowmax is indeed less sensitive to the choice of its free parameter and outperforms Boltzmann.

8. Related Work

Softmax operators play an important role in sequential decision-making algorithms.

In model-free reinforcement-learning, they can help strike a balance between exploration (mean) and exploitation (max). Decision rules based on epsilon-greedy and Boltzmann softmax, while very simple, often perform surprisingly well in practice, even outperforming more advanced exploration techniques (Kuleshov & Precup, 2014).

When learning “on policy”, exploration steps can (Rummery & Niranjan, 1994) and perhaps should (John, 1994) become part of the value-estimation process itself. On-policy algorithms like SARSA can be made to converge to optimal behavior in the limit when the exploration rate and the update operator is gradually moved toward $\max$ (Singh et al., 2000). Our use of softmax operators in learning updates reflects this point of view.

Analyses of the behavior of human subjects in choice experiments very frequently use softmax. Sometimes referred to in the literature as logit choice (Stahl & Wilson, 1994), it forms an important part of the most accurate predictor of human decisions in normal-form games (Wright & Leyton-Brown, 2010), quantal level-k reasoning (QLk). Softmax-based fixed points play a crucial role in this work. As such, mellowmax could potentially make a good replacement resulting in better behaved solutions.

Algorithms for inverse reinforcement learning (IRL), the problem of inferring reward functions from observed behavior (Ng & Russell, 2000), frequently use a Boltzmann operator to avoid assigning zero probability to non-optimal actions and hence assessing an observed sequence as impossible. Such methods include Bayesian IRL (Ramachandran & Amir, 2007), natural gradient IRL (Neu & Szepesvári, 2007), and maximum likelihood IRL (Babes et al., 2011). Given the recursive nature of value defined in these problems, mellowmax could be a more stable and efficient choice.

9. Conclusion and Future Work

We proposed the mellowmax operator as an alternative for the Boltzmann operator. We showed that mellowmax has several desirable properties and that it works favorably in practice. Arguably, mellowmax could be used in place of Boltzmann throughout reinforcement-learning research.

Important future work is to expand the scope of investigation to the function approximation setting in which the state space or the action space is large and
ABSTRACT

abstraction techniques are used. We expect mellowmax operator and its non-expansion property to behave more consistently than the Boltzmann operator when estimates of state–action values can be arbitrarily inaccurate.

Another direction is to analyze the fixed point of planning, reinforcement-learning, and game-playing algorithms when using softmax and mellowmax operators. In particular, an interesting analysis could be one that bounds the suboptimality of fixed points found by value iteration under each operator.

Finally, due to the convexity (Boyd & Vandenberghe, 2004) of mellowmax, it is compelling to use this operator in a gradient ascent algorithm in the context of sequential decision making. Inverse reinforcement-learning algorithms is a natural candidate given the popularity of softmax in these settings.

10. Acknowledgments

The authors gratefully acknowledge the assistance of George Konidaris.

References

Babes, Monica, Marivate, Vukosi N., Littman, Michael L., and Subramanian, Kaushik. Apprenticeship learning about multiple intentions. In International Conference on Machine Learning, pp. 897–904, 2011.

Baird, Leemon and Moore, Andrew W. Gradient descent for general reinforcement learning. In Advances in Neural Information Processing Systems, pp. 968–974, 1999.

Baker, Chris L, Tenenbaum, Joshua B, and Saxe, Rebeca R. Goal inference as inverse planning. In Proceedings of the 29th Annual Meeting of the Cognitive Science Society, 2007.

Beliakov, Gleb, Sola, Humberto Bustince, and Sánchez, Tomasa Calvo. A Practical Guide to Averaging Functions. Springer, 2016.

Boyd, S.P. and Vandenberghe, L. Convex optimization. Cambridge University Press, 2004.

Brent, Richard P. Algorithms for minimization without derivatives. Courier Corporation, 2013.

Brockman, Greg, Cheung, Vicki, Pettersson, Ludwig, Schneider, Jonas, Schulman, John, Tang, Jie, and Zaremba, Wojciech. Openai gym, 2016.

Chollet, François. Keras. https://github.com/fchollet/keras, 2015.

Cover, T.M. and Thomas, J.A. Elements of Information Theory. John Wiley and Sons, 2006.

Dearden, Richard, Friedman, Nir, and Russell, Stuart. Bayesian Q-learning. In Fifteenth National Conference on Artificial Intelligence (AAAI), pp. 761–768, 1998.

Gordon, Geoffrey J. Reinforcement learning with function approximation converges to a region, 2001. Unpublished.

John, George H. When the best move isn’t optimal: Q-learning with exploration. In Proceedings of the Twelfth National Conference on Artificial Intelligence, pp. 1464, Seattle, WA, 1994.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

Kuleshov, Volodymyr and Precup, Doina. Algorithms for multi-armed bandit problems. arXiv preprint arXiv:1402.6028, 2014.

Littman, Michael L. and Szepesvári, Csaba. A generalized reinforcement-learning model: Convergence and applications. In Saitta, Lorenza (ed.), Proceedings of the Thirteenth International Conference on Machine Learning, pp. 310–318, 1996.

Littman, Michael Lederman. Algorithms for Sequential Decision Making. PhD thesis, Department of Computer Science, Brown University, February 1996. Also Technical Report CS-96-09.

Neu, Gergely and Szepesvári, Csaba. Apprenticeship learning using inverse reinforcement learning and gradient methods. In UAI, 2007.

Ng, Andrew Y. and Russell, Stuart. Algorithms for inverse reinforcement learning. In International Conference on Machine Learning, pp. 663–670, 2000.

Perkins, Theodore J and Precup, Doina. A convergent form of approximate policy iteration. In Advances in Neural Information Processing Systems, pp. 1595–1602, 2002.

Peters, Jan, Mülling, Katharina, and Altun, Yasemin. Relative entropy policy search. In AAAI. Atlanta, 2010.

Puterman, Martin L. Markov Decision Processes—Discrete Stochastic Dynamic Programming. John Wiley & Sons, Inc., New York, NY, 1994.

Ramachandran, Deepak and Amir, Eyal. Bayesian inverse reinforcement learning. In IJCAI, 2007.
Rumery, G. A. and Niranjan, M. On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department, 1994.

Singh, Satinder, Jaakkola, Tommi, Littman, Michael L., and Szepesvári, Csaba. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 39:287–308, 2000.

Stahl, Dale O. and Wilson, Paul W. Experimental evidence on players’ models of other players. *Journal of Economic Behavior and Organization*, 25(3):309–327, 1994.

Sutton, Richard S. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*, pp. 216–224, Austin, TX, 1990. Morgan Kaufmann.

Sutton, Richard S. and Barto, Andrew G. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

Team, The Theano Development, Al-Rfou, Rami, Alain, Guillaume, Almahairi, Amjad, Angermueller, Christof, Bahdanau, Dzmitry, Ballas, Nicolas, Bastien, Frédéric, Bayer, Justin, Belikov, Anatoly, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.

Thrun, Sebastian B. The role of exploration in learning control. In White, David A. and Sofge, Donald A. (eds.), *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, pp. 527–559. Van Nostrand Reinhold, New York, NY, 1992.

Van Seijen, Harm, Van Hasselt, Hado, Whiteson, Shimon, and Wiering, Marco. A theoretical and empirical analysis of expected sarsa. In *2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pp. 177–184. IEEE, 2009.

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

Wright, James R. and Leyton-Brown, Kevin. Beyond equilibrium: Predicting human behavior in normal-form games. In *AAAI*, 2010.