Belief Reward Shaping in Reinforcement Learning

Ofir Marom[1], **Benjamin Rosman** [1, 2]
[1]University of the Witwatersrand, Johannesburg, South Africa
[2]Council for Scientific and Industrial Research, Pretoria, South Africa

Abstract

A key challenge in many reinforcement learning problems is delayed rewards, which can significantly slow down learning. Although reward shaping has previously been introduced to accelerate learning by bootstrapping an agent with additional information, this can lead to problems with convergence. We present a novel Bayesian reward shaping framework that augments the reward distribution with prior beliefs that decay with experience. Formally, we prove that under suitable conditions a Markov decision process augmented with our framework is consistent with the optimal policy of the original MDP when using the Q-learning algorithm. However, in general our method integrates seamlessly with any reinforcement learning algorithm that learns a value or action-value function through experience. Experiments are run on a gridworld and a more complex backgammon domain that show that we can learn tasks significantly faster when we specify intuitive priors on the reward distribution.

1 Introduction

Delayed and sparse rewards present a key challenge in many reinforcement learning (RL) problems, as this can significantly slow down learning through long, uninformed exploration trajectories. For example, when learning to play chess, typically the only reward signal that the agent receives is at the end of an episode, which occurs after many timesteps. A naïve approach to overcome this problem is to provide additional reward signals to the agent for achieving certain subgoals. For example, the environment may provide a reward to the agent for taking the opponent's queen. However, this is generally undesirable because (a) it forces the agent to play according to a specific strategy rather than letting it learn novel strategies, and (b) this may not be optimal.

Various techniques have been proposed to overcome this problem. In Hierarchical Reinforcement Learning we decompose a task into subtasks and learn to solve each subtask individually (Dietterich 1998; Sutton, Precup, and Singh 1999). Specifically, the MAXQ framework uses pseudo-rewards to learn the optimal policy for a subtask, which is analogous to providing more frequent rewards to the agent. Other techniques proposed include biasing the initial values of the action-value function using prior knowledge (Hailu and Sommer 1999; Matignon, Laurent, and le Fort-Piat 2006), using progress estimators to supply the agent with partial, goal-specific advice (Mataric 1994; Matignon, Laurent, and le Fort-Piat 2006), or using action priors so that the agent can positively bias action selection in the initial stages of learning (Rosman and Ramamoorthy 2012).

Most closely related to the work in this paper is reward shaping, where an additional reward from a reward shaping function is added at each step of learning. In the most general case, this reward shaping function can depend on the full transition information: the current state the agent is in, the action taken by the agent and the resulting next state. Reward shaping has proven to be a powerful method for speeding up many RL tasks. However, one of the issues discovered is that the agent may learn to enter cycles to optimize the shaping reward and forget about the true underlying RL problem (Randløv and Alstrøm 1998) and so may learn an optimal policy that is not consistent with the optimal policy learned without reward shaping. To address this, potential-based reward shaping (PBRS) was introduced, where the reward shaping function is restricted to a difference of potential functions, where the potential function is defined over states (Ng, Harada, and Russell 1999).

While PBRS has the advantage of ensuring consistency to the optimal policy, the restriction on the form of the reward shaping signal limits its expressiveness. This limitation has been alleviated to an extent with a framework called potential-based advice (PBA) which allows the potential function to also include actions (Wiewiora, Cottrel, and Elkan 2003), although the form of the shaping reward is still restricted with PBA since the shaping reward is still a difference of potential functions. It would be ideal if we could use the full transition information directly to define shaping rewards while still having consistency, as the additional expressive power can speed up convergence to the optimal policy.

More recently, it has been shown that PBA can be combined with dynamic potential-based reward shaping (Devlin and Kudenko 2012) to express shaping rewards that can use the full transition information, although this requires the introduction of a second action-value function that learns the required potential function concurrently with the main RL task (Harutyunyan et al. 2015).

In this paper we present a novel approach to reward shaping using a Bayesian framework, which we call belief reward shaping (BRS). BRS allows us to use the full transition information to provide shaping rewards directly, without having to learn it through interactions with the environment.

The major departure between BRS and standard RL is that we argue that rewards should not come only from an environment but should also incorporate prior beliefs. Some authors have addressed the need to define different types of rewards when solving an RL problem. For example, RL tasks where an agent received both intrinsic and extrinsic rewards have been introduced (Singh, Barto, and Chentanez 2004). Furthermore, in this view, the reward that an agent receives is considered to come from an internal critic that provides a primary reward to an agent, which may not come solely from the external environment sensation.

We update this view slightly in this paper to account for beliefs. Specifically, in our view the critic still receives a sensation from the environment. Now though, the sensation updates the critic's belief system and only thereafter the critic provides a reward (which we call a belief reward) to the agent that is an integration of the critic's prior beliefs and the current sensation. Whereas in traditional reward shaping methods the shaping reward is augmented to the environment reward distribution through the addition of a shaping function, in our framework we augment the shaping reward through a prior distribution on the environment reward distribution itself. These prior beliefs can be provided to the critic as a form of intuition or advice for the new task. Figure 1 illustrates our view.

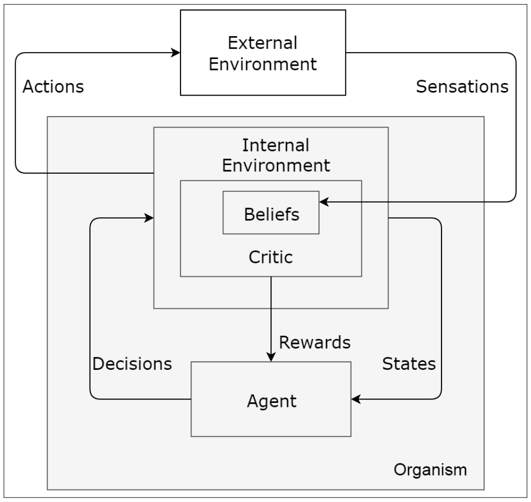

Figure 1: Agent / environment relationship with BRS.

2 Preliminaries

2.1 Standard Reinforcement Learning Framework

We restrict our attention to RL problems that can be represented as a discrete-time, finite-state and finite-action Markov decision process (MDP) whose reward distributions have finite variance. Such an MDP is defined by a tuple $M = (S, A, P, R, \gamma)$, where S is a finite set of states, and A is a finite set of all actions, while A_s is the actions available in state s. $P : S \times A \times S \to [0, 1]$ are the transition probabilities for transitioning to state s' conditional on the agent being in state s and taking action a. $R : S \times A \times S \to \mathbb{R}$ is the (stationary) reward distribution that governs the reward signal when in state s, taking action a and moving to state s'. In most RL problems R is a deterministic function. However, for our formalisation we use the more general case and treat it as a distribution. In section 3 we show that we can treat deterministic reward functions as a special case using degenerate distributions. $\gamma \in [0, 1]$ is a discount rate.

A policy is defined by $\pi : S \times A \to [0, 1]$ and represents the probability of taking action a conditional on being in state s. The goal of an RL agent is to find an optimal policy, π_*, that maximises the expected future discounted rewards. Given a fixed policy, π, we can define the action-value function for $n \geq 0$ as

$$Q_\pi(s, a) = \mathbf{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{n+k+1} | S_n = s, A_n = a \right],$$

where n is an index over time steps. This represents the expected discounted future rewards of following the policy π given that the agent is currently in state s and takes action a. The optimal action-value function is defined by $Q_*(s, a) = \sup_\pi Q_\pi(s, a)$ for all $s \in S$ and $a \in A_s$. Given Q_* we can derive the action to take in state s under π_* by choosing $argmax_{a \in A_s} Q_*(s, a)$. Note that in the undiscounted case ($\gamma = 1$) we assume that S contains a self-transitioning absorbing state that emits a reward of zero and further that for any policy and any starting state we will reach this absorbing state with probability 1.

In most practical RL applications, R and P are not known. This has led to a collection of algorithms that learn the optimal action-value function of an MDP from experience. As we further elaborate in section 3 BRS can integrate with any such algorithm by replacing the observed environment reward with a belief reward that, under suitable conditions, converges to the environment reward in the limit.

2.2 Reinforcement Learning with Reward Shaping

With reward shaping, the agent is provided with additional shaping rewards that come from a deterministic function, $F : S \times A \times S \to \mathbb{R}$. However, there is no guarantee that an MDP with arbitrary reward shaping will have an optimal policy that is consistent with the original MDP. Potential-based reward shaping (PBRS) addresses this problem by restricting the shaping reward function to $F(s, a, s') = \gamma\Phi(s') - \Phi(s)$ where Φ is an arbitrary function called a potential function (Ng, Harada, and Russell 1999). Using PBRS is a necessary and sufficient condition for the original MDP and the MDP with reward shaping to have consistent optimal policies.

One of the limitations of PBRS is that the potential function depends only on the state. However, PBRS has been extended by potential-based advice (PBA) to include actions in the potential function (Wiewiora, Cottrel, and Elkan 2003). In this framework we still have consistency, however there is an additional dependency of the potential function on the policy being followed that needs to be incorporated. With PBA we have look-ahead advice defined as $F(s,a,s',a') = \gamma\Phi(s',a') - \Phi(s,a)$ where we now choose the action, a, under a policy π as $\sup_{a \in A_s}(Q_\pi(s,a) + \Phi(s,a))$.

3 Belief Reward Shaping

3.1 Framework

Our departure point from standard RL is that we allow for prior beliefs to be augmented onto R. We hereafter refer to the standard RL reward as the "environment reward" and the reward that the critic gives the agent as the "belief reward".

We define a transition by the tuple (s,a,s') where s is the source state, a is the action taken when in s, and s' is the destination state, and denote by T the set of all possible transitions in the MDP. We denote by D_n^τ the history of environment rewards received by the critic for the transition $\tau \in \mathrm{T}$ up to and including time n. For each τ the (unknown) environment reward distribution is given by $p^\tau(r)$. We make the following assumptions in our framework:

- For each τ the critic hypothesises some models of the environment reward distribution, $F^\tau = \{\hat{p}^\tau(r|h) : h \in \mathcal{H}^\tau\}$, where $\mathcal{H}^\tau$ denotes the hypothesis space for transition τ. The $\hat{}$ notation indicates that $\hat{p}^\tau$ is an hypothesis for the true environment reward distribution.
- For each τ and at every $n \geq 0$ the critic knows its prior beliefs represented as a distribution over $\mathcal{H}^\tau$, $q^\tau(h|D_{n-1}^\tau)$.
- The environment reward received by the critic at time n after τ is identically and independently distributed (iid) from the previous environment rewards received in τ, D_{n-1}^τ.

The intuition behind our framework follows Bayesian reasoning. Each $\hat{p}^\tau(.|h) \in F^\tau$ is a possible reward distribution that may be representative of the true environment reward distribution. The critic's belief over possible reward distributions is captured by a prior distribution $q^\tau(h|D_{n-1}^\tau)$ which depends on the environment rewards the critic has observed so far. Before we start learning the RL task, q^τ represents the critic's beliefs without having seen any environment rewards, therefore at this point the distribution is influenced only by prior knowledge.

This knowledge may bias certain transitions and hence shape the behaviour of the agent. During learning each time the agent visits transition τ, data is collected from the environment and the critic refines its beliefs over the possible reward distributions and gets closer to the environment reward distribution.

We now formalise this intuition mathematically. At time n, the agent passes through a transition τ, and the critic receives some environment reward r_n. The critic updates its beliefs using Bayes' rule, $posterior \propto likelihood \times prior$, as

$$q^\tau(h|D_n^\tau) \propto \hat{p}^\tau(r_n|h)q^\tau(h|D_{n-1}^\tau). \qquad (1)$$

The maximum a posteriori (MAP) estimate, $h_{MAP}^\tau = argmax_{h \in \mathcal{H}^\tau} q^\tau(h|D_n^\tau)$ is derived. Lastly, the critic generates an unbiased estimate, $\hat{\mu}_n^B$, of $\mathbf{E}_{\hat{p}^\tau(r|h_{MAP}^\tau)}[R]$, the expected value of $\hat{p}^\tau(r|h_{MAP}^\tau)$, and provides this belief reward to the agent.

In practice, if $\hat{p}^\tau(r|h_{MAP}^\tau)$ is known in closed form we can compute the mean of the distribution directly, otherwise we may sample observations from the distribution and use the sample mean as an unbiased estimate of the true mean.

By convention we define $q^\tau(h|D_{-1}^\tau) = q^\tau(h)$, that is the prior distribution with no evidence. We note that because of the data invariance property of Bayes' rule, together with our iid assumption, the procedure described only requires us to keep track of $q^\tau(h|D_{n-1}^\tau)$ for all $\tau \in \mathrm{T}$ at time n and not the full histories, D_{n-1}^τ.

To illustrate how BRS can augment existing RL algorithms, the well-known Q-learning algorithm for episodic tasks with BRS is shown in algorithm 1 (we introduce $\alpha(s,a)$ in step 11 which is a step-size parameter that influences the rate of learning).

The framework proposed in this section extends naturally to the case where functional approximations are used to represent the action-value function. For example, in algorithm 1 we would use the same procedure except for step 11, where we would update the parameters of the function using $\hat{\mu}_n^B$, whereas in standard RL we would have used r_n.

Algorithm 1: Q-learning algorithm augmented with BRS for episodic tasks.

```
1  Initialise $Q(s,a)$ for all $s \in S$ and $a \in A_s$ and define
     $Q(s,a) = 0$ if $s$ is terminal
2  For each $\tau \in \mathrm{T}$ select $F^\tau = \{\hat{p}^\tau(r|h) : h \in \mathcal{H}^\tau\}$ and
     $q^\tau(h)$
3  foreach episode do
4      Initialise $s$
5      while $s$ is not terminal do
6          Choose $a$ from $A_s$ using policy from Q
7          Take action $a$. Observe next state $s'$ and
             environment reward $r$
8          Update $q^\tau(h|D^\tau)$ using equation 1
9          Compute $h_{MAP}^\tau$ from $q^\tau(h|D^\tau)$
10         Generate an unbiased estimate, $\hat{\mu}^B$, of
             $\mathbf{E}_{\hat{p}^\tau(r|h_{MAP}^\tau)}[R]$
11         $Q(s,a) \leftarrow Q(s,a) + \alpha(s,a)[\hat{\mu}^B +$
             $\gamma \sup_{b \in A_{s'}} Q(s',b) - Q(s,a)]$
12         $s \leftarrow s'$
13     end
14 end
```

In many RL problems the environment rewards are known to be deterministic. We can adopt this into the BRS framework by using degenerate distributions. Suppose the environment reward for τ is a constant μ. Define $p^\tau(r) = \mathcal{N}(\mu, \sigma^2)$ where $\sigma > 0$. We impose a conjugate normal prior on μ, $q^\tau(\mu) = \mathcal{N}(\mu_0, \frac{\sigma^2}{\lambda})$ where $\lambda > 0$. Applying Bayes' rule to compute $q^\tau(\mu|r_1 = \mu, r_2 = \mu, ..., r_n = \mu)$ and letting $\sigma \to 0$ converges to a degenerate distribution with point mass at

$$\frac{\lambda}{\lambda + n}\mu_0 + \frac{n}{\lambda + n}\mu. \tag{2}$$

The parameter λ is the pseudo-count for the prior mean and controls the rate at which the critic shifts from the prior mean to the true environment reward.

In the deterministic reward setting one may consider an analogous procedure using traditional reward shaping methods, whereby we define some reward shaping function F and multiply it by a decay parameter so as to ensure that the product converges to 0 in the limit. BRS is more general however as it extends naturally to cases where the environment reward is a distribution and it may be advantageous to set prior beliefs on the various parameters of the distribution.

3.2 Theoretical Results

In this section we prove that under suitable conditions Q-learning augmented with BRS has an optimal policy that is consistent with the original MDP.

Theorem 1. *Let $M = (S, A, P, R, \gamma)$ be an MDP and T the set of all possible transitions, (s, a, s'), in M. Suppose that for each $\tau \in \mathrm{T}$ we have a set of hypothesised models for the environment reward distribution, $F^\tau = \{\hat{p}^\tau(r|h) : h \in \mathcal{H}^\tau\}$, where $\mathcal{H}^\tau$ denotes the hypothesis space for transition τ and further we have a prior distribution over the hypothesis space, $q^\tau(h)$. Suppose further that $p^\tau(r) \in F^\tau$ where $p^\tau(r)$ is the true environment reward distribution. Then the Q-learning algorithm augmented with BRS described in figure 1 is consistent with the optimal policy of M provided that:*

- *The technical conditions required for Q-learning convergence to the optimal action-value function hold in M. Most notably, $\sum_{n=0}^{\infty} \alpha_n(s,a) = \infty$ and $\sum_{n=0}^{\infty} \alpha_n(s,a)^2 < \infty$ for all $(s,a) \in S \times A$. For full details of conditions see (Jaakkola, Jordan, and Singh 1994).*
- *For each $\tau \in \mathrm{T}$ the environment rewards received by the critic are iid.*
- *The technical conditions required for the MAP estimate of the posterior distribution to be consistent with the correct hypotheses hold for each $\tau \in \mathrm{T}$. Most notably that the likelihood is a continuous function of the hypothesis space and that the correct hypothesis is not on the boundary of the hypothesis space. For full details of conditions see (Gelman et al. 1995).*

Proof. We have assumed that $p^\tau(r) \in F^\tau$ and that the technical conditions required for the MAP estimate of the posterior distribution to be consistent with the correct hypothesis hold for all $\tau \in \mathrm{T}$. Therefore, by Bayesian asymptotic consistency, $\lim_{n\to\infty} \hat{p}^\tau(r|h^\tau_{n,MAP}) = p^\tau(r)$ for all r in the support of the distribution $p^\tau(r)$. Equivalently for $\epsilon > 0$, there exists some integer N such that for all $n > N$, $|\hat{p}^\tau(r|h^\tau_{n,MAP}) - p^\tau(r)| < \epsilon$ for all r in the support of the distribution $p^\tau(r)$. The conditions for Q-learning convergence are assumed to hold in M. Therefore

$$\sum_{n=0}^{\infty} \alpha_n(s,a) = \infty \quad \text{and} \quad \sum_{n=0}^{\infty} \alpha_n(s,a)^2 < \infty \tag{3}$$

for all $(s,a) \in S \times A$. Equation 3 implies that $\alpha_n(s,a)$ is finite for all $(s,a) \in S \times A$ and all $n \geq 0$. We can rewrite equation 3 as

$$\begin{gathered}\sum_{n=0}^{N} \alpha_n(s,a) + \sum_{n=N+1}^{\infty} \alpha_n(s,a) = \infty \quad \text{and} \\ \sum_{n=0}^{N} \alpha_n(s,a)^2 + \sum_{n=N+1}^{\infty} \alpha_n(s,a)^2 < \infty \end{gathered} \tag{4}$$

for all $(s,a) \in S \times A$. The left summations in equation 4 are finite because they are finite summations of finite terms. Therefore $\sum_{n=N+1}^{\infty} \alpha_n(s,a) = \infty$ and $\sum_{n=N+1}^{\infty} \alpha_n(s,a)^2 < \infty$ for all $(s,a) \in S \times A$. So we have that at $N+1$ the conditions required for Q-learning convergence hold while at the same time, the difference between the environment reward distribution and the critic's hypothesised distribution when using the MAP estimate are arbitrarily small.

□

It is straightforward to show that consistency still holds when only a subset of the transitions are augmented with BRS. Specifically, the only modification required is to define the scope of transitions to be $\mathrm{T}' \subseteq \mathrm{T}$ and apply standard Q-learning for all other transitions. This is useful because it may be cumbersome to define prior beliefs on all transitions and we can focus on specific transitions that we believe are important for our learning task.

A natural question to ask is what happens if the assumption that the environment reward distribution is in the critic's hypothesised set of models is false. Unfortunately, in such cases we lose the consistency guarantees of Q-learning with BRS. Nevertheless, we do have a guarantee that the MAP estimate will converge so as to make the hypothesised distribution as close as possible to the true distribution in terms of Kullback-Leibler divergence. For further details on this asymptotic guarantee see (Gelman et al. 1995).

3.3 Belief Clusters

In many real-world applications the state-space is very large. This makes it impractical to define a prior for each transition. However, we can work around this by defining a set of transitions that share some common structure. This is a form of state abstraction within the BRS framework.

Formally, we define a belief cluster as any subset, $\mathrm{T}' \subseteq \mathrm{T}$, where the intersection of the hypothesised models induced by T' is not empty. That is $F^{\mathrm{T}'} = \cap_{\tau \in \mathrm{T}'} F^\tau \neq \phi$. The critic may now define a prior on all the models in $F^{\mathrm{T}'}$, $q^{\mathrm{T}'}(h|D_n^{\mathrm{T}'})$, where the hypothesis space is now $\cap_{\tau \in \mathrm{T}'} \mathcal{H}^\tau$ and $D_n^{\mathrm{T}'} = \cup_{\tau \in \mathrm{T}'} D_n^\tau$ is the history of environment rewards received by the critic for any transition $\tau \in \mathrm{T}'$. The critic may now update its beliefs using Bayes' rule,

$$q^{\mathrm{T}'}(h|D_n^{\mathrm{T}'}) \propto \hat{p}^{\mathrm{T}'}(r_n|h)q^{\mathrm{T}'}(h|D_{n-1}^{\mathrm{T}'}). \tag{5}$$

It is straightfoward to update theorem 1 and show that if for any $\tau_1, \tau_2 \in \mathrm{T}'$ we have $p^{\tau_1}(r) \in F^{\mathrm{T}'}$ and $p^{\tau_1}(r) = p^{\tau_2}(r)$ and further, each $\tau \in \mathrm{T}$ belongs to exactly one belief cluster then the Q-learning algorithm augmented with belief clusters still has the consistency property. In words, these conditions ensure that all transitions in the belief cluster share the same true environment reward distribution and furthermore that the true environment reward distribution exists in the hypothesis space induced by the belief cluster. If we have that a transition belongs to more than one belief cluster, then we can enhance our framework slightly by allowing the critic to generate multiple belief rewards in each transition, one for each belief cluster that contains the transition, and take the average of these as the final reward given to the agent. Note that theorem 1, where we treat each transition separately, is a special case of belief clusters where we assign each transition to its own belief cluster.

4 Experiments

4.1 Cliff-Jump Gridworld

The purpose of this experiment is to show that we can obtain better performance with BRS over competing methods because we can use the full transition information directly to provide shaping rewards to the agent. The experiment is conducted on a 10 × 10 gridworld. The agent's task is to get from a fixed start coordinate, (1,1), to a fixed end coordinate, (10,10). In order to achieve this, it needs to jump over a cliff edge so it can get to the side where the end coordinate is located. A jump only succeeds with some probability, p_{jump}. This domain is inspired from various video games where a player needs to make a difficult jump which they may fail many times, but that is necessary to complete the level.

The state-space is (x, y), where x and y represents the agent's current x- and y-coordinates respectively ($x, y \in \{1, 2, ..., 10\}$). The agent can take the actions up, down, left and right which move the agent in those directions. In addition, there is a jump action. If the jump action is taken when the agent is on the edge of a cliff then the agent is transported to the other side with probability p_{jump}, otherwise the agent "falls off" the cliff and is transported back to the start coordinate. If the jump action is taken in any other state, the agent remains in the current state. Furthermore, if the agent takes an up or down action that moves it off the cliff then it is also transported back to the start coordinate. Finally, any action that should take the agent off the gridwold leaves the state unchanged. The gridworld is illustrated in figure 2 where the start coordinate is denoted by S, the end coordinate by E, and the cleft that the agent must jump over is denoted by C (i.e. the cliff edges are at $y = 2$ and $y = 4$).

The environment rewards are -1 for every step that is non-terminal and 100 for the terminal state. For learning to solve this RL problem we use an ϵ-greedy policy that explores random actions with some probability ϵ and acts greedily otherwise. We use $\gamma = 1$, a constant learning-rate $\alpha = 0.05$ and $\epsilon = 0.1$. We set $p_{jump} = 0.2$ and apply an early termination criterion of 100 steps per episode.

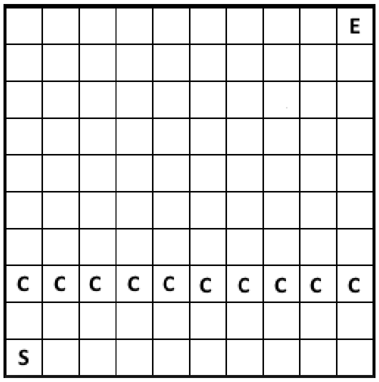

Figure 2: First domain: cliff-jump gridworld.

We use equation 2 to model the environment rewards as they are deterministic. Let $d(s, c)$ represent the distance from state s to coordinate $c = (x, y)$ for some arbitrary distance metric. Specifically, in our experiments we use the Manhattan distance. Intuitively, we want our agent to get to the cliff edge as quickly as possible, take the jump and then get to the end coordinate as quickly as possible. Define $s.x$ and $s.y$ to be the x- and y- coordinates in s respectively, $c_1 = (1, 2)$ (the bottom cliff edge coordinate) and $c_2 = (10, 10)$ (the end coordinate). Then to encode this advice we first define the following belief clusters:

- $B_1 = \{(s, a, s') \in \mathrm{T} : \ s.y \leq 2, \ d(s', c_1) < d(s, c_1)\}$
- $B_2 = \{(s, a, s') \in \mathrm{T} : \ s.y \leq 2, \ d(s', c_1) \geq d(s, c_1)\}$
- $B_3 = \{(s, a, s') \in \mathrm{T} : d(s, c_1) = 0, \ a = jump\}$
- $B_4 = \{(s, a, s') \in \mathrm{T} : \ s.y > 2, \ d(s', c_2) < d(s, c_2)\}$
- $B_5 = \{(s, a, s') \in \mathrm{T} : \ s.y > 2, \ d(s', c_2) \geq d(s, c_2)\}$

We will set priors on B_1 and B_2 to encourage the agent to get to the edge as quickly as possible, on B_3 to encourage the agent to take a jump at the edge and on B_4 and B_5 to encourage the agent to get to the end coordinate as quickly as possible once it has reached the other side of the cliff. We set a positive prior mean for belief clusters B_1, B_3 and B_4 as we want to encourage these transitions and negative prior mean for B_2 and B_5 as we want to discourage these transitions.

We compare our results to PBRS and PBA frameworks. While early work on these methods used negative potential functions for distance-based shaping rewards, recent work has shown that positive potentials may improve performance (Grzes and Kudenko 2009). We conducted experiments with both negative and positive potentials and found positive potentials had better performance in our domain, hence we report these results in the paper.

We note that for our comparisons we have not biased any of the frameworks as they have access to the same level of information and exploit that information to the best of their capabilities. In total we run the following algorithms on this domain:

- QL: standard Q-learning.

- BRS(μ_0,λ): Q-learning with BRS using belief clusters B_1, B_2, B_3, B_4 and B_5. We set the sign of μ_0 positive for belief clusters B_1, B_3 and B_4 and negative for B_2 and B_5 while the same λ is used for all belief clusters.
- PBRS(μ_0,μ_1): Q-learning with PBRS, $\Phi(s) = \mu_0(\max_{s' \in S} d(s', c_1) - d(s, c_1))\mathbb{1}_{s.y \leq 2} + \mu_1(\max_{s' \in S} d(s', c_2) - d(s, c_2))\mathbb{1}_{s.y > 2}$.
- PBA(μ_0,μ_1,μ_2): Q-learning with PBA, $\Phi(s, a) = \mu_0(\max_{s' \in S} d(s', c_1) - d(s, c_1))\mathbb{1}_{s.y \leq 2} + \mu_1(\max_{s' \in S} d(s', c_2) - d(s, c_2))\mathbb{1}_{s.y > 2} + \mu_2 \mathbb{1}_{d(s,c_1)=0,\ a=jump}$.

Each algorithm is tested on a grid of parameter values. BRS(μ_0,λ) is run for $\mu_0 \in \{0.5, 1, 5, 10, 100\}$ and $\lambda \in \{100, 500, 1000, 5000, 10000\}$. PBRS($\mu_0$,$\mu_1$) and PBA($\mu_0$,$\mu_1$,$\mu_2$) are run where each parameter takes values in $\{0.5, 1, 5, 10, 100\}$. We run each algorithm / parameter value over 1000 episodes and average over 50 independent runs. Plots are then averaged over 10 consecutive points with error bars included.

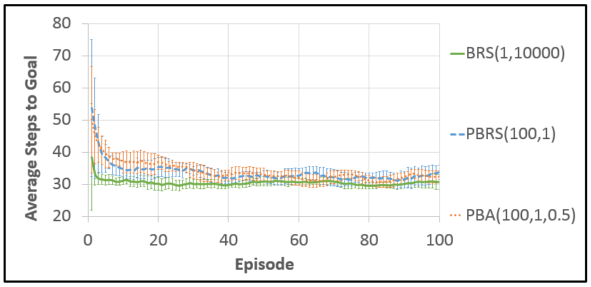

Figure 3: Cliff-jump optimal performance for each algorithm.

We see from figure 3 that using reward shaping significantly outperforms standard Q-learning for optimal parameter values (Q-learning converges in approximately 700 episodes as shown in figure 5). BRS outperforms other reward shaping methods. In fact, it converges to the optimal policy almost immediately whereas PBRS and PBA converge in approximately 50 episodes. This performance gain when using BRS is unsurprising given that we can be specific with our reward structures and provide shaping rewards using the full transition information directly. The fact that we can do this while still having convergence guarantees that we don't have with general reward shaping is quite appealing.

Now consider figure 4. When using BRS with deterministic rewards, we have two parameters – μ_0, which is the prior mean and λ which is the pseudo-count of the prior mean. In practice, λ controls the rate at which we shift our beliefs from the prior mean to the true environment reward, and the lower λ the faster we shift from the prior mean. In most cases μ_0 is more intuitive to set than λ. In figure 4 we show how BRS performs when we set an intuitive prior $\mu_0 = 1$ for the different values of λ. We see that even when we set $\lambda = 10^2$, small, we get improvements over standard Q-learning. This is because the agent has the benefit of learning from optimal actions early in the learning process, albeit for a short time. When $\lambda = 10^3$ we have similar convergence to the optimal parameter settings although we have a period of divergence between 150 and 200 episodes. The reason for this is that at the start of learning, the agent's trajectories are controlled most strongly by the priors we set and when this reduces the agent starts exploring based on the environment dynamics. In this domain, the agent initially is encouraged by the positive prior μ_0 to jump at the cliff edge but once the environment dynamics start to dominate the agent starts incurring negative rewards for jumping, as the jump fails with high probability, and this leads to exploration of unexplored states. Since these states are not on the trajectory of the optimal policy the agent returns over time to the optimal policy. Overall we see that using BRS with small λ may still provide significant performance improvements and that BRS is robust to λ in this domain when we set a reasonable value for the prior mean.

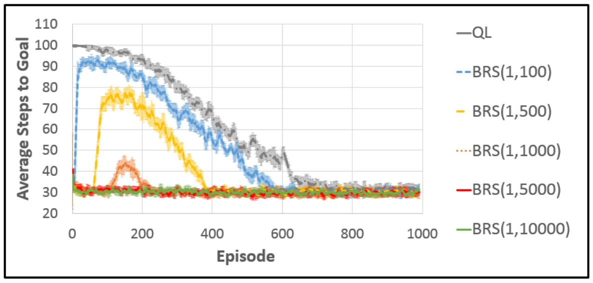

Figure 4: Cliff-jump sensitivity to λ.

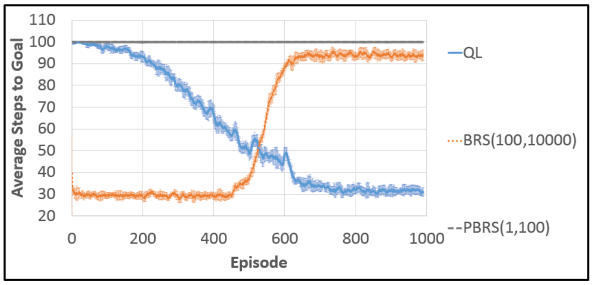

Figure 5: Cliff-jump examples of poor performance performance.

In figure 5 we examine results for parameter settings that result in poor performance. Consider first BRS(100,10000) where we set $\mu_0 = 100$ high and $\lambda = 10^4$ high. In this case BRS converges to the optimal policy almost immediately but then starts to diverge at around 500 episodes. The reason for this divergence is that the agent enters a cycle whereby it moves in and out of c_1 repeatedly. This is because the prior mean is set so high and the rate at which we shift to the environment reward so slow that the agent learns to prefer obtaining the shaping rewards over solving the underlying RL task. This issue with cycles is similar to that described in the introduction when using general reward shaping, however we note that with BRS we have convergence in the limit. Next consider PBRS(1,100). At 1000 episodes we see the agent is still behaving as poorly as when it started. We conclude from this that poorly specified shaping rewards result in slower convergence to the optimal policy regardless of the algorithm chosen. Furthermore one may argue that in both the cases of BRS(100,10000) and PBRS(1,100) it is not immediately obvious that the chosen parameter values would produce poor performance. Therefore, care must be taken when choosing shaping rewards in RL tasks to ensure they improve performance.

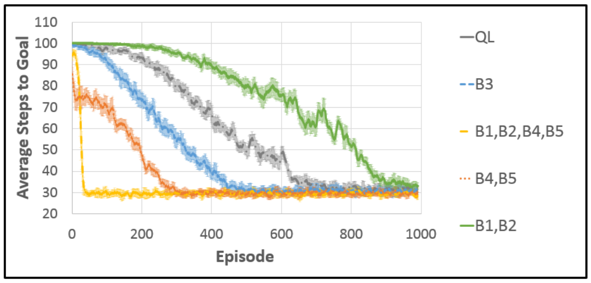

Figure 6: Cliff-jump examples with incomplete information.

In figure 6 we analyse how BRS performs with incomplete information by running experiments where we do not use the full set of belief clusters. In all these experiments we use $\mu_0 = 1$ and $\lambda = 10^4$. We see that in most cases, we obtain improved performance even in the setting of incomplete information. For example, when only using belief cluster B_3 where our priors incentivise the agent to jump at the cleft, we get significant improvements over standard Q-learning. Interestingly, when using only belief clusters B_1 and B_2 we get worse performance over standard Q-learning. Recall that the priors on these clusters incentivises the agent to get to the cleft edge as quickly as possible however since there is no immediate incentive to reach the goal the agent learns to optimize the shaping reward by walking in and out of coordinate c_1.

4.2 Backgammon

We now illustrate the benefits of BRS on a more complex backgammon domain. In this section we do not report results for PBRS as it would be difficult to draw any objective conclusion on which method performs better due to the domain's complexity and hence the number of ways we might construct our shaping rewards.

The backgammon domain has previously been solved using TD(λ) neural networks (Tesauro 2004). The first iteration of the TD-Gammon algorithm (TDG0.0) achieved strong intermediate level of play and further refinements to the algorithm achieved master-level play. In this paper we focus on the first iteration due to its ease of implementation and replication – the solution design is clearly described in the literature (Tesauro 1995; Sutton and Barto 1998). We use an identical setup to TDG0.0 for our experiments. Note that in this domain using techniques that bias the initial value function are difficult to implement because the value function is now parameterized, however, reward shaping is well-suited to such tasks.

We first train a baseline neural network (BL) over 50,000 games. We then train a new neural network repeatedly over 500 games and let it play 10 batches of 50 games against BL. We do this 100 times so that at the end of the process both neural networks have been trained on the same number of games. We repeat this procedure for 3 different neural networks: one standard network (SNN), one network that uses a simple set of prior beliefs (SPNN) and one network that uses a complex set of prior beliefs (CPNN). In brief, with SPNN we set priors on states that are "improvements" over the starting game state in the sense that we reward having no blot exposure, more than 4 points, more than 1 inner point and a maximum prime of more than 1 – these rewards are only given when degree of contact is not 0. In our setup, the states for each of these concepts are grouped into a separate belief cluster and we use $|\mu_0| = 0.5$ and $\lambda = 3000$ for all belief clusters – the sign of μ_0 depends on whether we want to incentivise or disincentivise the states in that cluster.

With CPNN we scale the reward based on the position (i.e. higher prior mean for having 5 inner points than 2 inner points) and we restrict rewards to situations when the position is beneficial (i.e. having a 6-point prime is most beneficial when there is an opponent checker trapped behind the prime). We note that these strategies are quite crude and in many cases not optimal – for example there are situations when it is preferable to take risks and increase blot exposure. However, these states can be thought of as beginner level strategies that will beat a random player but are outperformed by a good player. To illustrate this, we also train two additional neural networks, SCNN and CCNN, that use the same structure as SPNN and CPNN respectively but that provide a *constant environment reward* that is equal to the prior mean.

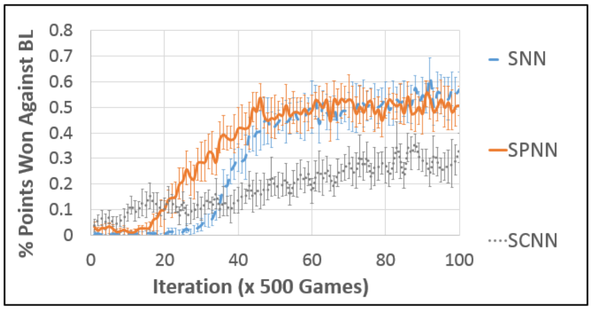

Figure 7: Backgammon with simple prior beliefs.

Figures 7 and 8 show how each of these networks performs against BL by averaging the number of points won across the batches played at each stage of learning. The standard deviations are also shown in this figure.

We see from the results that SPNN performs better at the initial stages of learning than the standard network, SNN. At 10,000 games this network is performing as well as SNN is after 17,000 games and gets to a similar level of play as BL by the end of training. Meanwhile SCNN (simple constant rewards) initially performs better than SPNN but is never able to win more than 35% of the points against

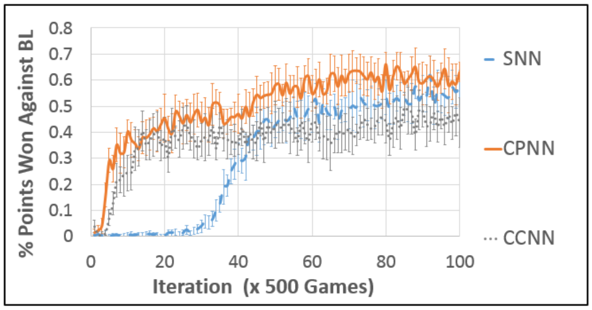

Figure 8: Backgammon with complex prior beliefs.

BL. When using CPNN, we get significantly improved performance. At 10,000 games this network wins almost 45% of the points against BL, which took SNN 24,000 training games to achieve and wins 60% of the points by the end of training. CCNN performs better than SCNN did in the previous experiment, but is never able to win 50% of the points or more against BL.

While we have not proved any consistency theorems for TD(λ) neural networks it is reasonable to expect that SPNN and CPNN converge to identical policies as SNN because after enough training iterations the belief rewards obtained will be arbitrarily close to the environment rewards. They converge much faster to the optimal policy because of the shaping rewards provided by BRS. Meanwhile SCNN and CCNN will perform worse than SNN as they indefinitely reward strategies that are not always optimal.

5 Concluding Remarks

This paper has introduced a new reward shaping framework based on Bayesian methods that specifies prior beliefs on the environment reward distribution and generates belief rewards that integrate prior beliefs with environment rewards. We also provided theoretical guarantees of our method's consistency when augmenting Q-learning. Our experiments illustrate that using BRS we can specify rich reward structures that can improve performance over competing methods and that BRS can be integrated with general RL algorithms to improve performance in more complex domains.

6 Acknowledgements

The authors wish to thank the anonymous reviewers for their thorough feedback and helpful comments.

References

Devlin, S., and Kudenko, D. 2012. Dynamic potential-based reward shaping. *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems* 433–440.

Dietterich, T. 1998. The maxq method for hierarchical reinforcement learning. *Proceedings of the 15th International Conference on Machine Learning*.

Gelman, A.; Carlin, J. B.; Stern, H. S.; and Rubin, D. B. 1995. *Bayesian Data Analysis*. Chapman and Hall.

Grzes, M., and Kudenko, D. 2009. Theoretical and empirical analysis of reward shaping in reinforcement learning. *Proceedings of the IEEE International Conference on Machine Learning and Applications* 337–344.

Hailu, G., and Sommer, G. 1999. On amount and quality of bias in reinforcement learning. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics* 1491–1495.

Harutyunyan, A.; Devlin, S.; Vrancx, P.; and Nowé, A. 2015. Expressing arbitrary reward functions as potential-based advice. *Proceedings of the 29th AAAI Conference on Artificial Intelligence*.

Jaakkola, T.; Jordan, M. I.; and Singh, S. 1994. On the convergence of stochastic iterative. dynamic programming algorithms. *Neural Computation* 1185–1201.

Mataric, M. 1994. Reward functions for accelerated learning. *Proceedings of the 11th International Conference on Machine Learning* 181–189.

Matignon, L.; Laurent, G. J.; and le Fort-Piat, N. 2006. Reward function and initial values: Better choices for accelerated goal-directed reinforcement learning. *Lecture Notes in Computer Science* 4131:840–849.

Ng, Y.; Harada, D.; and Russell, S. J. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. *Proceedings of the 16th International Conference on Machine Learning* 278–287.

Randløv, J., and Alstrøm, P. 1998. Learning to drive a bicycle using reinforcement learning and shaping. *Proceedings of the 15th International Conference on Machine Learning* 463–471.

Rosman, B., and Ramamoorthy, S. 2012. What good are actions? accelerating learning using learned action priors. *Development and Learning and Epigenetic Robotics 2012 IEEE International Conference*.

Singh, S.; Barto, A.; and Chentanez, N. 2004. Intrinsically motivated reinforcement learning. *18th Annual Conference on Neural Information Processing Systems*.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

Sutton, R.; Precup, D.; and Singh, S. 1999. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* 112:181–211.

Tesauro, G. J. 1995. Temporal difference learning and td-gammon. *Communication of the ACM* 38:58–68.

Tesauro, G. J. 2004. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation* 6(2):215–219.

Wiewiora, E.; Cottrel, G. W.; and Elkan, C. 2003. Principled methods for advising reinforcement learning agents. *Proceedings of the 20th International Conference on Machine Learning* 792–799.