Convergence of reinforcement learning with general function approximators

Vassilis A. Papavassiliou and Stuart Russell
Computer Science Division, U. of California, Berkeley, CA 94720-1776
{vassilis,russell}@cs.berkeley.edu

Abstract

A key open problem in reinforcement learning is to assure convergence when using a compact hypothesis class to approximate the value function. Although the standard temporal-difference learning algorithm has been shown to converge when the hypothesis class is a linear combination of fixed basis functions, it may diverge with a general (nonlinear) hypothesis class. This paper describes the Bridge algorithm, a new method for reinforcement learning, and shows that it converges to an approximate global optimum for any agnostically learnable hypothesis class. Convergence is demonstrated on a simple example for which temporal-difference learning fails. Weak conditions are identified under which the Bridge algorithm converges for any hypothesis class. Finally, connections are made between the complexity of reinforcement learning and the PAC-learnability of the hypothesis class.

1 Introduction

Reinforcement learning (RL) is a widely used method for learning to make decisions in complex, uncertain environments. Typically, an RL agent perceives and acts in an environment, receiving rewards that provide some indication of the quality of its actions. The agent's goal is to maximize the sum of rewards received. RL algorithms work by learning a *value function* that describes the long-term expected sum of rewards from each state; alternatively, they can learn a *Q-function* describing the value of each action in each state. These functions can then be used to make decisions.

Temporal-difference (TD) learning [Sutton, 1988] is a commonly used family of reinforcement learning methods. TD algorithms operate by adjusting the value function to be locally consistent. When used with *function approximators*, such as neural networks, that provide a compact parameterized representation of the value function, TD methods can solve real-world problems with very large state spaces. Because of this, one would like to know if such algorithms can be guaranteed to work—i.e., to converge and to return optimal solutions.

The theoretical study of RL algorithms usually divides the problem into two aspects: *exploration policies* that can guarantee complete coverage of the environment, and *value determination* to find the value function that corresponds to a given policy. This paper concentrates on the second aspect. Prior work [Jaakkola *et al.*, 1995] has established convergence of TD-learning with probability 1 when the value function is represented as a table where each state has its own entry. For large state spaces, however, compact parametric representations are required; for such representations, we are interested in whether an algorithm will converge to the function that is closest, by some metric, to the true value function (a form of *agnostic learning*). Gordon [1995] proved that TD converges in this sense for representations called "averagers" on which the TD update is a max-norm contraction (see Section 2). Tsitsiklis and van Roy [1996] proved convergence and established error bounds for TD(λ) with linear combinations of fixed basis functions.

With nonlinear representations, such as neural networks, TD has been observed to give suboptimal solutions [Bertsekas and Tsitsiklis, 1996] or even to diverge. This is a serious problem since most real problems require nonlinearity. Baird [1995] introduced residual algorithms, for which convergence can be proved when combined with a gradient descent learning method (such as used with neural networks). Unfortunately the error in the resulting approximation can be arbitrarily large and furthermore the method requires two independent visits to each sampled state.

This paper describes the Bridge algorithm, a new RL method for which we establish convergence and error bounds with any agnostically learnable representation.

Section 2 provides the necessary definitions and notation. Section 3 explains the problem of nonconvergence and provides examples of this with TD. Section 4 outlines the Bridge algorithm, sketches the proof of convergence, and shows how it solves the examples for which TD fails. Section 5 briefly covers additional results on convergence to local optima for *any* representation and on the use of PAC-learning theory. Section 6 mentions some alternative techniques one might consider. The paper is necessarily technically dense given the space restrictions. The results, however, should be of broad interest to the AI and machine learning communities.

2 Definitions

2.1 MDP

A Markov decision process $\mathbf{M} = (\mathbf{S}, \mathbf{A}, \mathbf{p}, \mathbf{r}, \gamma)$ is a set of states $\mathbf{S}$, a set of actions $\mathbf{A}$, transition probability distributions $\mathbf{p}(\cdot|x, a)$ that define the next state distribution given a current state x and action a, reward distributions $\mathbf{r}(\cdot|x, a)$ that define the distribution of real-valued reward received upon executing a in x, and a discount factor $\gamma \in (0, 1)$. Since we are interested in the problem of value determination, we assume we are given a fixed policy (choice of action at each state). When executing only this fixed policy, the MDP actually becomes a Markov chain, and we may therefore also write the transition probabilities as $\mathbf{p}(\cdot|x)$ and the reward distributions as $\mathbf{r}(\cdot|x)$. We assume that we are able to define the stationary distribution π of the resulting Markov chain and also that the rewards lie in the range $[-R_{max}, R_{max}]$.

Let $(X_1, X_2, X_3, ...)$ be a random trace in the Markov chain starting from x, i.e. $X_1 = x$, X_2 has distribution $\mathbf{p}(\cdot|X_1)$ and X_k has distribution $\mathbf{p}(\cdot|X_{k-1})$. Let $(R_1, R_2, R_3, ...)$ be the observed random rewards, i.e. R_k has distribution $\mathbf{r}(\cdot|X_k)$. Define the *true value function* $\mathbf{V}^*$ at a state x to be the expected, discounted reward to go from state x:

$$\mathbf{V}^*(x) = \mathbf{E}[R_1 + \gamma R_2 + \gamma^2 R_3 + ...]$$

The problem of value determination is to determine the true value function or a good approximation to it. Classical TD solutions have made use of the *backup operator* $\mathbf{T}$, which takes an approximation $\mathbf{V}$ and produces a better approximation

$$\begin{aligned} \mathbf{TV}(x) &= \mathbf{E}[R_1 + \gamma \mathbf{V}(X_2)] \\ &= \mathbf{E}[R_1] + \gamma \sum_{x_2 \in S} \mathbf{p}(x_2|x)\mathbf{V}(x_2) \end{aligned}$$

An operator $\mathbf{A}$ is said to be a *contraction* with factor $\zeta < 1$ under some norm $||\cdot||$ if

$$\forall \mathbf{V}, \mathbf{W} \quad ||\mathbf{AV} - \mathbf{AW}|| \leq \zeta ||\mathbf{V} - \mathbf{W}||$$

If $\zeta = 1$, $\mathbf{A}$ is said to be a *nonexpansion*. If we define the max-norm to be $||\mathbf{V}||_{max} = \max_{x \in \mathbf{S}} \mathbf{V}(x)$ and the π-norm to be $||\mathbf{V}||_\pi = [\sum_{x \in S} \mathbf{V}(x)^2 \pi(x)]^{1/2}$, then $\mathbf{T}$ is a contraction with factor γ and fixed point $\mathbf{V}^*$ under both max-norm and π-norm [Tsitsiklis and Van Roy, 1996]. Therefore repeated application of $\mathbf{T}$ (i.e. the iterative process $\mathbf{V}_{n+1} = \mathbf{TV}_n$) converges to $\mathbf{V}^*$. We will use $\mathbf{T}^j$ to represent the operator $\mathbf{T}$ applied j times.

If the transition probabilities $\mathbf{p}$ and reward distributions $\mathbf{r}$ are known, then it is possible to compute $\mathbf{TV}$ directly from its definition. However, if $\mathbf{p}$ and $\mathbf{r}$ are not known, then it is not possible to compute the expectation in the definition of $\mathbf{T}$. In this case, by observing a sequence of states and rewards in the Markov chain, we can form an unbiased estimate of $\mathbf{TV}$. Specifically, if we observe state x and reward r followed by state x_2, then the observed backed-up value, $r + \gamma \mathbf{V}(x_2)$, is an unbiased estimate of $\mathbf{TV}(x)$. Formally we define $\mathcal{P}_T^V(\cdot|x)$ to be the conditional probability density of observed backed-up values from state x:

$$\mathcal{P}_T^V(y|x) = \mathbf{Pr}[R_1 + \gamma \mathbf{V}(X_2) = y|x]$$

where R_1 and X_2 are, as defined above, random variables with distributions $\mathbf{r}(\cdot|x)$ and $\mathbf{p}(\cdot|x)$ respectively. Thus if a random variable Y has associated density $\mathcal{P}_T^V(\cdot|x)$ then $\mathbf{E}[Y] = \mathbf{TV}(x)$. Similarly, we define $\mathcal{P}_{T^j}^V(\cdot|x)$ to be the conditional probability density of j-step backed-up values observed from state x.

2.2 Function Approximation

As state spaces becomes large or even infinite, it becomes infeasible to tabulate the value function for each state x, and we must resort to function approximation. Our approximation scheme consists of a hypothesis class $\mathcal{H}$ of representable functions and a learning operator $\Pi_\mathcal{H}$ which maps arbitrary value functions to functions in $\mathcal{H}$.

The standard $\mathbf{T}$ based approaches that use function approximation essentially compute or approximately compute the iterative process $\mathbf{V}_{n+1} = \Pi_\mathcal{H} \mathbf{TV}_n$. In practice, the $\Pi_\mathcal{H} \mathbf{T}$ mapping usually cannot be performed exactly because, even if we have access to the necessary expectation to compute $\mathbf{TV}(x)$ exactly, it is infeasible to do so for all states x. Thus we perform an approximate mapping using samples. We will take the state sample distribution to be the stationary distribution π. In general when we cannot compute $\mathbf{TV}(x)$ exactly, we approximate $\Pi_\mathcal{H} \mathbf{TV}$ by generating samples (x, y) with sample distribution

$$\mathcal{P}_T^V(x, y) = \pi(x) \mathcal{P}_T^V(y|x)$$

and passing them to a learning algorithm for $\mathcal{H}$. The joint probability density $\mathcal{P}_T^V$ (from which we generate the samples) simply combines π (from which we sample the state x) with the conditional probability density $\mathcal{P}_T^V(\cdot|x)$ (from which we generate an estimate y for $\mathbf{TV}(x)$).

In this paper we focus on *agnostic learning*. In this case, the learning operator $\Pi_\mathcal{H}$ seeks the hypothesis h that best matches the target function $\mathbf{V}$, even though typically the target function is not in the hypothesis class. If we measure distance using the π-norm, then we can define the learning operator for agnostic learning to be:

$$\Pi_\mathcal{H} \mathbf{V} = \text{argmin}_{h \in \mathcal{H}} ||h - \mathbf{V}||_\pi$$

As already mentioned, in the typical case we do not have access to the exact function $\mathbf{V}$ to be learned, but rather we can draw samples (x, y) from a sample distribution $\mathcal{P}$ such that the expected value of the conditional distribution $\mathcal{P}(\cdot|x)$ is $\mathbf{V}(x)$. If, in addition, $\mathcal{P}$ samples the states according to π (or whatever distribution was used to measure distance in the previous definition) then an equivalent definition for agnostic learning is based on minimizing risk:

$$\Pi_\mathcal{H} \mathbf{V} = \text{argmin}_{h \in \mathcal{H}} \mathbf{r}(h, \mathcal{P})$$

where we define the *risk* of a hypothesis h with respect to a distribution $\mathcal{P}$ to be:

$$\mathbf{r}(h, \mathcal{P}) = \int_{(\mathbf{S}, \Re)} (h(x) - y)^2 d\mathcal{P}(x, y)$$

In practice, $\Pi_\mathcal{H}$ is approximately performed by generating enough samples (x, y) from the sample distribution $\mathcal{P}$ so as to be able to estimate risk well, and thus to be able to output the hypothesis in $\mathcal{H}$ that has minimal risk with respect to this distribution. In the algorithm we present, we assume the ability to compute $\Pi_\mathcal{H}$ exactly for the given hypothesis class $\mathcal{H}$. This is certainly not a trivial assumption. In a later section, we briefly discuss an extension to our algorithm for the case where $\Pi_\mathcal{H}^{\epsilon,\delta}$ is a PAC-agnostic learning step rather than an exact agnostic learning step.

Finally, let us define our goal. $\mathbf{V}_\mathcal{H}^*$ is defined to be the best approximation to $\mathbf{V}^*$ possible using $\mathcal{H}$:

$$\mathbf{V}_\mathcal{H}^* = \Pi_\mathcal{H} \mathbf{V}^* = \text{argmin}_{h \in \mathcal{H}} ||h - \mathbf{V}^*||_\pi$$

We seek techniques that return a value function $\mathbf{V}$ that minimizes a relative or absolute error bound:

$$\frac{||\mathbf{V} - \mathbf{V}^*||_\pi}{||\mathbf{V}_\mathcal{H}^* - \mathbf{V}^*||_\pi} \quad \text{or} \quad ||\mathbf{V} - \mathbf{V}^*||_\pi$$

3 Nonconvergence of TD

In this section, we examine the non-convergence problem of TD when used with non-linear function approximators. We present simple examples which we will reconsider with the Bridge algorithm in the next section.

As mentioned above, standard TD with function approximation is based on the iterative process $\mathbf{V}_{n+1} = \Pi_\mathcal{H} \mathbf{TV}_n$. If

$\Pi_{\mathcal{H}}$ is a non-expansion under the same norm that makes $\mathbf{T}$ a contraction, then the composite operator $\Pi_{\mathcal{H}}\mathbf{T}$ is a contraction and this process will converge to some error bound relative to $\mathbf{V}^*_{\mathcal{H}}$. For example, Tsitsiklis and Van Roy [1996] consider a linear hypothesis class, for which $\Pi_{\mathcal{H}}$ is simply a projection. If one uses a nonlinear hypothesis class $\mathcal{H}$ for which $\Pi_{\mathcal{H}}$ is not a nonexpansion then this iterative process can either diverge or get stuck in a local minimum arbitrarily far from $\mathbf{V}^*_{\mathcal{H}}$.

We now give simple examples demonstrating ways in which TD can fail when it is used with a nonlinear hypothesis class. Consider an MDP with two states where the probability of going from one state to the other is 1, and the rewards are also deterministic. The stationary distribution π is 0.5 for each state, the discount factor is .8 and the hypothesis class $\mathcal{H}$ is the subset of the value function space $\mathcal{R}^2$ given by the v_1-axis, the v_2-axis and the two diagonals $v_1 = v_2$ and $v_1 = -v_2$. Formally, $\mathcal{H} = \{(v_1, v_2) \in \mathcal{R}^2 : v_1 = 0 \text{ or } v_2 = 0 \text{ or } |v_1| = |v_2|\}$. The learning operator $\Pi_{\mathcal{H}}$ projects a function onto the nearest of the 4 axes of $\mathcal{H}$. For example, $\Pi_{\mathcal{H}}(6,4) = (5,5)$, $\Pi_{\mathcal{H}}(6,-4) = (5,-5)$ and $\Pi_{\mathcal{H}}(-7,-1) = (-7,0)$.

We first consider the case where the rewards of the two states are $(r_1, r_2) = (10, -8)$. The true value function $\mathbf{V}^*$ turns out to be $(10, 0)$. Note that in this case, the true value function is actually in the hypothesis class $\mathcal{H}$. Starting from $\mathbf{V}_0 = (0,0)$ the result of repeated applications of $\Pi_{\mathcal{H}}\mathbf{T}$ is shown in Figure 1(a). For example, the first step is $\Pi_{\mathcal{H}}\mathbf{T}\mathbf{V}_0 = \Pi_{\mathcal{H}}(r_1 + \gamma\mathbf{V}(2), r_2 + \gamma\mathbf{V}(1)) = \Pi_{\mathcal{H}}(10,-8) = (9,-9)$. This process converges to $\mathbf{V} = (5,-5)$ which is a fixed point because $\Pi_{\mathcal{H}}\mathbf{T}(5,-5) = \Pi_{\mathcal{H}}(6,-4) = (5,-5)$. Remember that $\mathbf{V}^* \in \mathcal{H}$, so the relative error bound $\frac{\|\mathbf{V}-\mathbf{V}^*\|_\pi}{\|\mathbf{V}^*_{\mathcal{H}}-\mathbf{V}^*\|_\pi}$ is infinite. Thus $\mathbf{T}$ can converge arbitrarily far (in terms of relative error bound) from the best representation in $\mathcal{H}$ of the true value function.

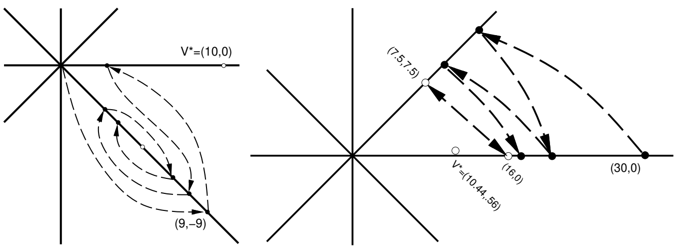

Figure 1: (a) Suboptimal Fixed Point and (b) Oscillation

If we modify the rewards slightly to be $(r_1, r_2) = (10, -7.8)$ then the true value function $\mathbf{V}^* = (10\frac{4}{9}, \frac{5}{9})$ is no longer in $\mathcal{H}$. The best representation of $\mathbf{V}^*$ is $\mathbf{V}^*_{\mathcal{H}} = \Pi_{\mathcal{H}}\mathbf{V}^* = (10\frac{4}{9}, 0)$. If we start from $(0,0)$ as above, we will again reach a suboptimal fixed point around $(5,-5)$. However, starting from $\mathbf{V}_0 = (30,0)$ (or even $\mathbf{V}_0 = (15,0)$) the result of repeated applications of $\Pi_{\mathcal{H}}\mathbf{T}$ as shown in Figure 1(b) displays a different type of failure — oscillation between points approaching $(7.5, 7.5)$ and $(16,0)$. As in the previous example, $\|\mathbf{V}^*_{\mathcal{H}} - \mathbf{V}^*\|_\pi$ is small, so the relative error bound is large.

4 The Bridge Algorithm

We begin with a high level description of the algorithm (details are in the Appendix). This is followed by the convergence results and another look at the examples from the previous section.

The main algorithm $\mathbf{BridgeValueDet}$, determines the value function within some error bound by making repeated calls to $\mathbf{BridgeStep}$. We will now describe the first invocation of $\mathbf{BridgeStep}$.

Metaphorically it consists of throwing a bridge across the treacherous terrain that is the hypothesis class $\mathcal{H}$, towards a point on the far side of the optimal solution. If the bridge lands somewhere close to where we aimed it, we will be able to walk along it in a productive direction (achieve a contraction). If the bridge lands far from our target, then we know that there isn't any $\mathcal{H}$-expressible value function near our target on which the bridge could have landed (hence an error bound). This is made precise by Lemma 2 in the next section.

We are given an old approximation $\mathbf{V}$ from which we try to create a better approximation $\mathbf{V}_{new}$. We basically have two tools to work with: $\mathbf{T}$ and $\Pi_{\mathcal{H}}$. As can be seen in Figure 2 (and in the example in the previous section), if we combine these two operators in the standard way, $\mathbf{V}_{new} = \Pi_{\mathcal{H}}\mathbf{T}\mathbf{V}$, we can get stuck in a local minimum. We will instead use them more creatively to guarantee progress or establish an error bound.

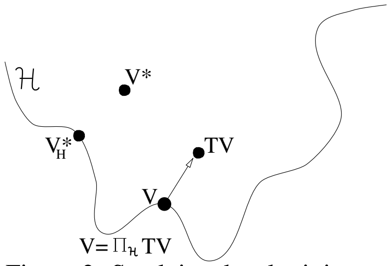

Figure 2: Stuck in a local minimum

We begin by using not $\mathbf{T}$ but rather $\mathbf{T}^j$ where j is determined by the main algorithm $\mathbf{BridgeValueDet}$ before it calls $\mathbf{BridgeStep}$. We can then ask the question, given we know where $\mathbf{V}$ and $\mathbf{T}^j\mathbf{V}$ are, what does that tell us about the location of $\mathbf{V}^*$? It turns out that $\mathbf{V}^*$ is restricted to lie in some hypersphere whose position can be defined in terms of the positions of $\mathbf{V}$ and $\mathbf{T}^j\mathbf{V}$. This is made precise by Lemma 1 in the next section. The hypersphere is depicted in Figure 3 and as required, $\mathbf{V}^*$ lies inside it.

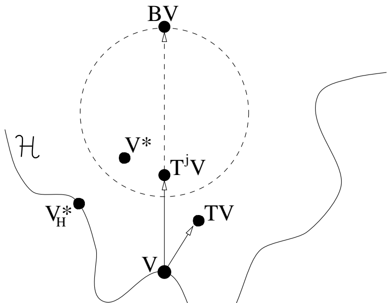

Figure 3: The bridge is aimed

We now define a new operator $\mathbf{B}$ based on $\mathbf{T}^j$ and the identity operator $\mathbf{I}$.

$$\mathbf{B} = \mathbf{I} + \frac{1}{1-\gamma^j}(\mathbf{T}^j - \mathbf{I})$$

$\mathbf{B}$ simply amplifies the Bellman residual by a factor of $\frac{1}{1-\gamma^j}$. As can be seen in Figure 3, $\mathbf{BV}$ is the point on the far side of the hypersphere from $\mathbf{V}$. This operator is what we use to throw a bridge. We aim the bridge for $\mathbf{BV}$, which is beyond anywhere where our goal might be, i.e.the true value function lies somewhere between $\mathbf{V}$ and $\mathbf{BV}$. The motivation for using $\mathbf{B}$ is in a sense to jump over all local minima between $\mathbf{V}$ and $\mathbf{V}^*$.

Ideally we would be able to represent $\mathbf{BV}$ (just as in the standard approach we would want to represent $\mathbf{TV}$) but this function is most likely not in our class of representable functions. Therefore we must apply the operator $\Pi_{\mathcal{H}}$ to map it into $\mathcal{H}$. The result, $\mathbf{W} = \Pi_{\mathcal{H}}\mathbf{BV} \in \mathcal{H}$, is shown in Figure 4. The bridge is supported by $\mathbf{V}$ and $\mathbf{W}$ and is shown as a line between them. In summary we throw the bridge aiming for $\mathbf{BV}$, but $\Pi_{\mathcal{H}}$ determines the point $\mathbf{W}$ on which it actually lands.

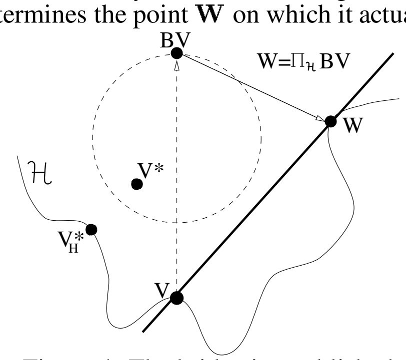

Figure 4: The bridge is established

In practice we perform the mapping $\Pi_{\mathcal{H}}$ by generating samples from an appropriate distribution and passing them to a learning algorithm for $\mathcal{H}$. In particular to compute $\Pi_{\mathcal{H}}\mathbf{BV}$, we generate samples (x, y) according to the distribution:

$$\begin{aligned}\mathcal{P}_B^V(x,y) &= \pi(x)\mathcal{P}_B^V(y|x)\\ &= \pi(x)\mathcal{P}_{T^j}^V((y-\mathbf{V}(x))(1-\gamma^j)+\mathbf{V}(x)|x)\end{aligned}$$

The key feature of this distribution is that if a random variable Y has associated density $\mathcal{P}_B^V(\cdot|x)$ then $\mathbf{E}[Y] = \mathbf{BV}(x)$.

The final step is to walk along the bridge. The bridge is a line between $\mathbf{V}$ and $\mathbf{W}$ and our new approximation $\mathbf{V}_{new}$ will be some point on this line (see Figure 5). This point is determined by projecting a point $\eta \leq 1$ of the way from $\mathbf{V}$ to $\mathbf{T}^j\mathbf{V}$ onto the line, where η is a function of the input parameters. (We could just project $\mathbf{T}^j\mathbf{V}$, but using η is a refinement that yields a better guaranteed effective contraction factor.)

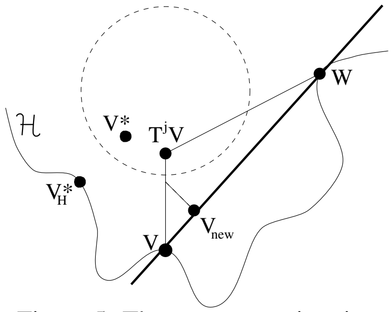

Figure 5: The new approximation

Thus the new approximation $\mathbf{V}_{new}$, which is not necessarily in $\mathcal{H}$, is a weighted average of the old approximation $\mathbf{V}$ and $\mathbf{W} \in \mathcal{H}$. Calculating the weights (ρ and $1-\rho$) in this average requires the ability to measure distance and risk. In particular we need to measure the distance between $\mathbf{V}$ and $\mathbf{W}$ and the risk of $\mathbf{V}$ and $\mathbf{W}$ with respect to the distribution $\mathcal{P}_{T^j}^V$. These three lengths (and η) determine the relative position of $\mathbf{V}_{new}$ with respect to $\mathbf{V}$ and $\mathbf{W}$ (See Figure 5). In practice we estimate the true risk with the *empirical risk* [Haussler, 1992], which we calculate using samples drawn from the distribution $\mathcal{P}_{T^j}^V$.

We have just described a single invocation of BridgeStep that represents the first iteration of the main algorithm. Each iteration builds a new bridge based on the previous one, so a generic iteration would begin with a $\mathbf{V}$ that was the $\mathbf{V}_{new}$ of the previous iteration (see Figure 6). In particular, the input $\mathbf{V}$ of a generic iteration is not in $\mathcal{H}$, but is rather a linear combination of the initial approximation $\mathbf{V}_0$ and all previous $\mathbf{W}$ functions. Thus the final result is a tall weighted tree whose leaves are in $\mathcal{H}$. If we insist on a final result that is in $\mathcal{H}$, then we can apply a final $\Pi_{\mathcal{H}}$ mapping at the very end.

Just as the standard TD algorithm was summarized as $\mathbf{V}_{n+1} = \Pi_{\mathcal{H}}\mathbf{TV}_n$, the Bridge algorithm can be essentially summarized as

$$\begin{aligned}\mathbf{V}_{n+1} &= (1-\rho_n)\mathbf{V}_n + \rho_n\Pi_{\mathcal{H}}\mathbf{BV}_n\\ &= ((1-\rho_n)\mathbf{I} + \rho_n\Pi_{\mathcal{H}}(\mathbf{I} + \frac{1}{1-\gamma^j}(\mathbf{T}^j - \mathbf{I})))\mathbf{V}_n\end{aligned}$$

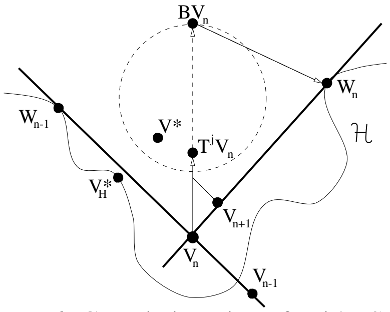

Figure 6: Generic iteration of BridgeStep

4.1 Convergence of the Bridge algorithm

We will state the main convergence theorem for the Bridge algorithm, but space limitations allow us to state only the two most important Lemmas used in the proof. We begin with a very useful observation about the geometric relationship between $\mathbf{V}$, $\mathbf{TV}$ and $\mathbf{V}^*$.

Lemma 1 *Let* $\mathbf{A}$ *be a contraction with contraction factor* ζ *under some norm. Let* $\mathbf{V}^*$ *be the fixed point of* $\mathbf{A}$. *For any point* $\mathbf{V}$ *let* $\mathbf{O} = \mathbf{V} + \frac{1}{1-\zeta^2}(\mathbf{AV} - \mathbf{V})$. *Then,*

$$||\mathbf{V}^* - \mathbf{O}|| \leq \frac{\zeta}{1-\zeta^2}||\mathbf{AV} - \mathbf{V}||$$

In words, given the positions of $\mathbf{V}$ and $\mathbf{AV}$, let $c = ||\mathbf{AV} - \mathbf{V}||$. Then we know that the position of $\mathbf{V}^*$ has to be on or inside the hypersphere of radius $\frac{c\zeta}{1-\zeta^2}$ centered at $\mathbf{O}$ (see Figure 7). This hypersphere is simply the set of points that are at least a factor of ζ closer to $\mathbf{AV}$ than to $\mathbf{V}$. Note that the distance from $\mathbf{V}$ to the furthest point on the hypersphere is $\frac{c}{1-\zeta}$.

We apply Lemma 1 using $\mathbf{T}^j$ for $\mathbf{A}$ and γ^j for ζ. This defines a hypersphere inside of which the true value function must lie. Lemma 1 is used mainly to prove Lemma 2, which characterizes the behavior of $\mathbf{BridgeStep}$ and provides most of the meat of the convergence proof.

Lemma 2 *Given an approximation* $\mathbf{V}$ *and parameters* $\alpha > 0$ *and* $j \geq 1$, $\mathbf{BridgeStep}(\mathbf{V}, \alpha, j)$ *returns a new approxima-*

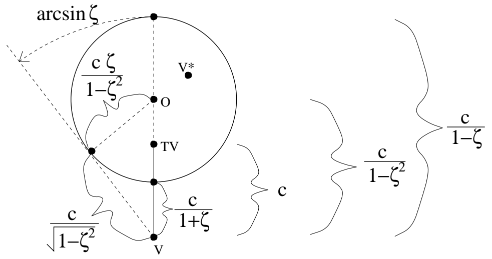

Figure 7: Hypersphere containing $\mathbf{V}^*$

tion $\mathbf{V}_{new}$ that satisfies at least one of the following two conditions, where the error bound $\kappa = \mathbf{errBound}(\alpha, \gamma, j)$ is defined in the Appendix

$$||\mathbf{V}_{new} - \mathbf{V}^*||_\pi \leq \alpha ||\mathbf{V} - \mathbf{V}^*||_\pi \qquad \textit{(Contraction)}$$

$$||\mathbf{V}_{new} - \mathbf{V}^*||_\pi \leq \kappa ||\mathbf{V}^*_{\mathcal{H}} - \mathbf{V}^*||_\pi \qquad \textit{(Error Bound)}$$

Intuitively, if the bridge lands close to where we aimed it, we will achieve a contraction towards the goal. If the bridge lands far away, we will prove a relative error bound for the result. The key quantity that determines which of these two events happens, is the angle θ formed between the bridge and the line from $\mathbf{V}$ to $\mathbf{BV}$. If $\mathbf{W} = \Pi_{\mathcal{H}} \mathbf{BV}$ is close to $\mathbf{BV}$, then θ will be small, the bridge will lie close to the hypersphere, and we will be able to walk along the bridge and make progress. If instead $\mathbf{W}$ is far from $\mathbf{BV}$, then θ will be large and walking along the bridge will not take us closer to the goal, but we will be able to prove that we are already close enough.

Figure 8 shows the case where the angle θ is small. As described previously, the small hypersphere represents the set of points that are at least a factor of γ^j closer to $\mathbf{T}^j\mathbf{V}$ than they are to $\mathbf{V}$. This follows from applying Lemma 1 to the operator $\mathbf{T}^j$. Now think of $\mathbf{BridgeStep}$ as an operator that takes $\mathbf{V}$ and returns $\mathbf{V}_{new}$, and ask the question, what set of points are at least a factor of α (which is an input parameter to $\mathbf{BridgeStep}$) closer to $\mathbf{V}_{new}$ than to $\mathbf{V}$? Applying Lemma 1 to this question defines another, much larger hypersphere which is depicted in Figure 8 with center at $\mathbf{O}$ for the case $\theta = \arcsin \alpha - \arcsin \gamma^j$. Note that this larger hypersphere completely contains the smaller hypersphere which contains $\mathbf{V}^*$. Thus $\mathbf{V}^*$ also lies inside the larger hypersphere and so $\mathbf{V}_{new}$ is at least a factor of α closer to $\mathbf{V}^*$ than $\mathbf{V}$ is. This holds for $\theta = \arcsin \alpha - \arcsin \gamma^j$. If θ is smaller than this, the achieved contraction is even better.

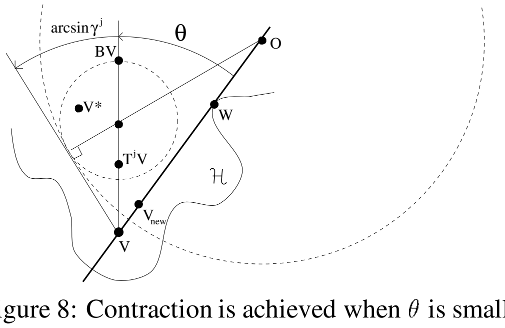

Figure 8: Contraction is achieved when θ is small

Figure 9 shows the case where the angle θ is large. θ is large when it is not possible to find a hypothesis in $\mathcal{H}$ close to $\mathbf{BV}$. In fact we choose $\mathbf{W} = \Pi_{\mathcal{H}} \mathbf{BV}$ to be the closest such hypothesis, so the rest of $\mathcal{H}$ must lie further away. In particular $\mathcal{H}$ must lie completely outside the big hypersphere depicted in Figure 9 with center at $\mathbf{BV}$, for otherwise $\mathbf{W}$ would not be the closest hypothesis to $\mathbf{BV}$. Furthermore we know that $\mathbf{V}^*$ must lie on or inside the small hypersphere in Figure 9. Thus there is a separation between $\mathbf{V}^*$ and $\mathcal{H}$ and this separation allows us to prove, for any possible position of $\mathbf{V}^*$, an upper bound on the relative error $\frac{||\mathbf{V}_{new} - \mathbf{V}^*||_\pi}{||\mathbf{V}^*_{\mathcal{H}} - \mathbf{V}^*||_\pi}$.

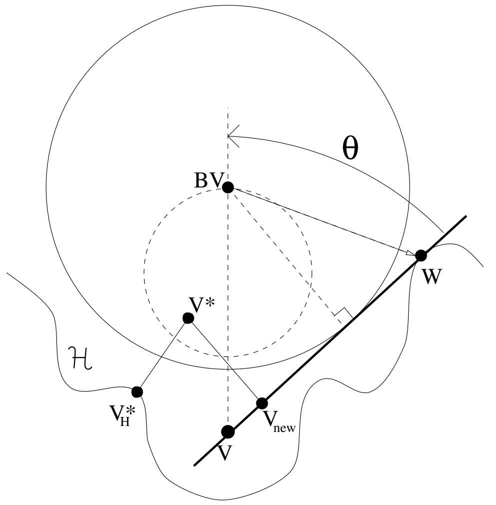

Figure 9: Relative error bound is established when θ is large

It should be noted that in general we do not know and we cannot measure θ to determine which of the two conditions of Lemma 2 $\mathbf{V}_{new}$ satisfies. We only know that it satisfies at least one of them.

By Lemma 2, if $\mathbf{V}$ already satisfies the relative error bound then so will $\mathbf{V}_{new}$, because if $\mathbf{V}_{new}$ achieves a contraction over $\mathbf{V}$, its error decreases. Thus each successive approximation is better than the one before, until we achieve the relative error bound from which point every subsequent approximation will also achieve that bound.

We now give the main result, which is guaranteed convergence to a relative or absolute error bound. Moreover, the maximum number of invocations of $\mathbf{BridgeStep}$, and thus the maximum number of hypotheses in the linear combination, can be specified.

Theorem 1 *Let $\nu > 1$ and $\epsilon_0 > 0$ be the desired relative and absolute error bounds respectively. Let N be an upper bound on the desired number of iterations. Then the algorithm $\mathbf{BridgeValueDet}(\nu, \epsilon_0, N)$ produces an approximation $\tilde{\mathbf{V}}$, consisting of a linear combination of at most $N+1$ hypotheses from $\mathcal{H}$, that satisfies at least one of either the relative error bound ν or the absolute error bound ϵ_0:*

$$\frac{||\tilde{\mathbf{V}} - \mathbf{V}^*||_\pi}{||\mathbf{V}^*_{\mathcal{H}} - \mathbf{V}^*||_\pi} \leq \nu \qquad \textit{or} \qquad ||\tilde{\mathbf{V}} - \mathbf{V}^*||_\pi \leq \epsilon_0$$

The proof of the theorem follows directly from Lemma 2; rewards are bounded, so the true value function is bounded, so the absolute error of the initial approximation can be bounded. If all N iterations achieve a contraction, then the absolute error will be smaller than requested. If at least one of the iterations failed to achieve a contraction, then it achieved a relative error bound and all subsequent iterations, including the last one, will achieve the requested relative error bound. Again, since we do not know which of the two conditions of Lemma 2 each iteration satisfies, we do not know whether the final answer $\tilde{\mathbf{V}}$ satisfies the relative or the absolute error bound. We know only that it satisfies at least one of them.

Corollary 1 *Let ν, ϵ_0 and N be as defined in Theorem 1. Let $\tilde{\mathbf{V}} = \mathbf{BridgeValueDet}(\nu, \epsilon_0, N)$, a linear combination of hypotheses from $\mathcal{H}$. Then $\tilde{\mathbf{V}}_{\mathcal{H}} = \Pi_{\mathcal{H}}\tilde{\mathbf{V}}$, the result of mapping $\tilde{\mathbf{V}}$ back into $\mathcal{H}$, satisfies at least one of either the relative error bound $2\nu + 1$ or absolute error bound $\epsilon_0(2 + \frac{1}{\nu})$.*

4.2 The Examples Revisited

We now reconsider the examples from Section 3. The main algorithm $\mathbf{BridgeValueDet}$ takes parameters ν, ϵ_0, and N, from which it computes the number of lookahead steps j to use to achieve the requested error bounds. Also for each iteration, it chooses a parameter α_n which determines the contraction factor achieved or relative error bound established for that iteration. These two parameters, j and α_n, are passed to $\mathbf{BridgeStep}$ at each iteration. In this section, we examine the effect of repeated applications of $\mathbf{BridgeStep}$, using $j = 3$ and $\alpha = .99$ for every iteration.

For the first example, with initial $\mathbf{V}_0 = (0, 0)$, the results of repeated applications of $\mathbf{BridgeStep}$ are shown in Figure 10(a). Because for this example $\|\mathbf{V}^*_{\mathcal{H}} - \mathbf{V}^*\|_\pi = 0$ (i.e. the true value function is in $\mathcal{H}$), the relative error bound is always infinite. Therefore, by Lemma 2, every step achieves at least a contraction α and so the algorithm converges to the true value function.

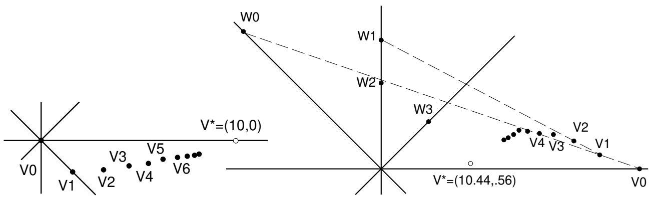

Figure 10: Examples revisited with Bridge

For the second example, with $\mathbf{V}_0 = (30, 0)$, the results of repeated applications of $\mathbf{BridgeStep}$ are shown in Figure 10(b). Looking at the first step more closely, $\mathbf{T}^j\mathbf{V}_0 = (10.2, 10.6)$, $\mathbf{BV}_0 = (-10.7, 21.7)$ and $\mathbf{W}_0 = \Pi_{\mathcal{H}}\mathbf{BV}_0 = (-16.2, 16.2)$. The dotted line between $\mathbf{V}_0$ and $\mathbf{W}_0$ is the bridge. $\mathbf{V}_1$, being a weighted average of $\mathbf{V}_0$ and $\mathbf{W}_0$, lies on this bridge. Similarly, $\mathbf{V}_2$ lies on the bridge between $\mathbf{V}_1$ and $\mathbf{W}_1$.

	relative error bound	effective contraction factor
V0	35.2	---
V5	13.5	.87
V10	7.4	.89
V15	4.1	.89
V20	2.3	.89
V25	1.3	.90
V30	0.8	.91
V35	0.5	.93
V40	0.4	.97
V45	0.4	.99

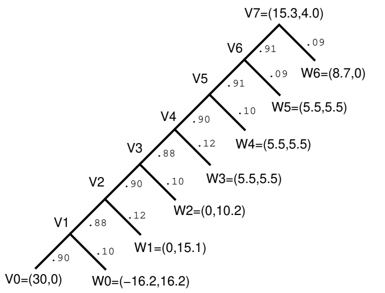

Figure 11: (a) Lemma 2 applied to second example
(b) Linear combination

Figure 11(a) demonstrates Lemma 2 on every fifth application of $\mathbf{BridgeStep}$. In particular, note that the effective contraction factor only exceeds α after the desired relative error bound $\mathbf{errBound}(\alpha, \gamma, j) = 4.3$ has been achieved. In fact on this example, the algorithm performs far better than the theory guarantees. Figure 11(b) shows the weights of the averages and the structure of the resulting linear combination after 7 steps.

5 Extensions

It is possible to extend the algorithm in many ways. In particular relying on an exact, agnostic learning operator $\Pi_{\mathcal{H}}$ is not practical. Here we briefly discuss the use of two other learning operators and we hope in the future to consider others still.

5.1 PAC(ϵ, δ) learning

Most significantly we have extended our algorithm to the case where the learning step $\Pi_{\mathcal{H}}$ cannot be done exactly but is instead a PAC learning step $\Pi_{\mathcal{H}}^{\epsilon,\delta}$ (see [Papavassiliou and Russell, 1998]). We actually use the same ϵ and δ for every iteration, so the learning step $\Pi_{\mathcal{H}}^{\epsilon,\delta}$ has the same complexity for every iteration. This is simple but most likely not optimal. One appealing aspect of considering PAC agnostic learning is the potential availability of sample complexity results based on some measure of the complexity of $\mathcal{H}$. Unfortunately it is necessary to learn and estimate risk under the stationary distribution of the Markov chain. Simply running the chain to generate samples will only generate them correctly in the steady-state limit. Therefore computing sample complexity results for the risk estimation and agnostic learning steps requires extending the current state of the theory to the case where samples are generated from a Markov chain, rather than i.i.d. One would expect the sample complexity to depend on the mixing time of the Markov chain and the variance of the sample distribution. The form of these theorems will also determine the extent to which samples can be reused between the different risk estimation steps within an iteration or even across iterations.

5.2 Suboptimal learning

Previous algorithms for this problem have been shown to converge for learning operators $\Pi_{\mathcal{H}}$ that are non-expansions

$$\|\Pi_{\mathcal{H}}\mathbf{V} - \Pi_{\mathcal{H}}\mathbf{W}\| \leq \|\mathbf{V} - \mathbf{W}\|$$

The convergence results for Bridge hold for learning operators that perform agnostic learning. Unfortunately there is a general lack of useful agnostic learning algorithms (the risk minimization step is typically intractable), so it would be beneficial to extend the results to learning systems that are not optimal.

It is possible to weaken the conditions on the learning operator and give convergence results for Bridge that hold when $\Pi_{\mathcal{H}}$ satisfies the banana-fudge condition

$$\frac{\|\mathbf{V} - \Pi_{\mathcal{H}}\mathbf{V}\|}{\mathbf{k_1}(\|\mathbf{V} - \mathbf{W}\|)} - \mathbf{k_2}(\|\mathbf{V} - \mathbf{W}\|) \leq \|\mathbf{W} - \Pi_{\mathcal{H}}\mathbf{W}\|$$

for some nondecreasing functions $\mathbf{k_1} > \mathbf{0}$ and $\mathbf{k_2}$. The only modification necessary to Bridge is to include $\mathbf{k_1}$ and $\mathbf{k_2}$ in the calculation of the relative error bound $\mathbf{errBound}(\alpha, \gamma, j)$. Note that for $\mathbf{k_1}(z) = 1$ and $\mathbf{k_2}(z) = z$, this condition reduces to

$$\|\mathbf{V} - \Pi_{\mathcal{H}}\mathbf{V}\| - \|\mathbf{V} - \mathbf{W}\| \leq \|\mathbf{W} - \Pi_{\mathcal{H}}\mathbf{W}\|$$

which is in fact the property of agnostic learning that is used to derive the results in this paper.

Intuitively, the nonexpansion condition for $\Pi_{\mathcal{H}}$ requires that two points that are close to each other, are mapped close to each other. The banana-fudge condition requires that two points that are close to each other, are mapped a similar *distance* away, but they can be mapped in opposite directions and so end up very far from each other. The banana-fudge condition is obviously the weaker one, requiring only similarity in level of success and not similarity in outcome. It disallows the case where one function is learned very well, but another function very close to the first is learned very poorly. We are currently searching for learning algorithms that satisfy the banana-fudge condition, but unfortunately it seems most common practical learning algorithms do not.

6 Other Approaches

We briefly discuss other known alternatives to Bridge as well as mention some of the new directions one might consider.

6.1 Alternatives

If $\mathcal{H}$ is convex and $\Pi_{\mathcal{H}}$ is the agnostic learning operator, then $\Pi_{\mathcal{H}}$ is a nonexpansion and $\mathbf{V}_{n+1} = \Pi_{\mathcal{H}}\mathbf{T}\mathbf{V}_n$ converges. For nonconvex $\mathcal{H}$, an alternative approach to Bridge is to PAC-agnostically learn the convex hull of $\mathcal{H}$ using $\Pi_{\mathcal{H}}$ at each iteration [Lee *et al.*, 1995]. The resulting iterated procedure $\mathbf{V}_{n+1} = \Pi_{convex-hull(\mathcal{H})}\mathbf{T}\mathbf{V}_n$ converges since $\Pi_{convex-hull(\mathcal{H})}$ is a nonexpansion. Unfortunately, this algorithm requires many more agnostic learning steps per iteration than seems practical.

A noniterative method that returns the optimal answer is to reduce the value determination problem to a single instance of supervised learning by using the operator $\mathbf{T}^{\infty}$ (otherwise known as TD($\lambda = 1$)). It does unlimited lookahead, has contraction factor $\gamma^{\infty} = 0$ and so it generates $\mathbf{V}^*$ after just one iteration. Looked at another way, the distribution $\mathcal{P}^{V}_{T^{\infty}}(\cdot|x)$ has mean $\mathbf{V}^*(x)$ and so $\Pi_{\mathcal{H}}\mathbf{T}^{\infty}\mathbf{V} = \mathbf{V}^*_{\mathcal{H}}$. Unfortunately there is empirical evidence that suggests the sample distribution $\mathcal{P}^{V}_{T^{\infty}}$ is very hard to learn and requires very many samples (perhaps because it can have high variance).

Strictly speaking, it is not necessary to backup values beyond the ϵ-horizon which is $\log_{\gamma}\frac{\epsilon(1-\gamma)}{R_{max}}$. Even this, however, may yield sample distributions with too much variance for practical use, although it is offset by the need to perform only a single learning step.

Finally it may be possible, using Lemma 1, to establish convergence rates and error bounds for the iterated procedure $\mathbf{V}_{n+1} = \Pi_{\mathcal{H}}\mathbf{T}^m\mathbf{V}_n$ where m is less than the ϵ-horizon. However, m would probably have to be much larger than j, the number of lookahead steps used by Bridge, and so again we would expect bad sample complexity.

6.2 New Directions

There are many ways in which the basic tools used in constructing this algorithm might be used in constructing more powerful methods. Specifically the geometric relationship between $\mathbf{V}, \mathbf{TV}$, and $\mathbf{V}^*$ established in Lemma 1 is very useful in (1) providing geometric intuition to design new methods and (2) proving performance guarantees for these methods.

One can think of many different ways to throw a bridge and many different kinds of bridges to throw. For example, we establish $\mathbf{W}$, the other end of the bridge by learning the point $\mathbf{V} + \frac{1}{1-\gamma^j}(\mathbf{T}^j\mathbf{V} - \mathbf{V})$. This choice is rather arbitrary, picked to simplify the error bound analysis. One might try instead learning a point further or a little closer to $\mathbf{V}$.

Once we establish $\mathbf{W}$, we throw a one-dimensional, linear bridge from $\mathbf{V}$ to $\mathbf{W}$ and learn a point close to $\mathbf{T}^j\mathbf{V}$ on this line (learning is equivalent to projection in linear hypothesis classes). One might try establishing more than two points with which to support the bridge. For example, given $\mathbf{V}$ and after establishing $\mathbf{W}(1)$, we could try establishing $\mathbf{W}(2)$ by learning a point strategically located far from both $\mathbf{V}$ and $\mathbf{W}(1)$ on the other side of the hypersphere defined by Lemma 1. Then we could throw a two-dimensional, planar bridge across these three points and project $\mathbf{T}^j\mathbf{V}$ (or a point close by) onto this plane. We can continue in this way, considering methods that establish $\mathbf{W}(1), ..., \mathbf{W}(n)$ and use an n-dimensional hyperplane to learn $\mathbf{T}^j\mathbf{V}$. In the logical limit this method looks like a local version of [Lee *et al.*, 1995] which learns the full convex hull. It is local in that it only closes under weighted averaging those points of $\mathcal{H}$ that are closest to some point of the hypersphere defined by Lemma 1. Our current method which only uses one-dimensional bridges is effectively a light version of these convex-hull methods, in that before learning $\mathbf{T}^j\mathbf{V}$ it closes under linear combinations only two points from $\mathcal{H}$, namely $\mathbf{V}$ and $\mathbf{W}$.

7 Conclusion

We have developed a method that reduces the value determination problem to the agnostic learning problem. Requesting that our algorithm halt in fewer iterations or with better error bounds pushes more of the complexity into the learning step and in the limit effectively forces it to consider infinite lookahead which is $\mathbf{T}^{\infty}$. Similarly, if we were to extend our algorithm to use more and more supports for the bridge, we suspect it would approximate the performance of the convex hull learning algorithm. Thus our method can be thought of as a more versatile and hopefully more efficient alternative to these aggressive methods.

The key features that characterize our approach are (1) the complication of learning is abstracted into a learning operator $\Pi_{\mathcal{H}}$, (2) we use a new operator $\mathbf{B}$ rather than being restricted to the backup operator $\mathbf{T}$, (3) we form linear combinations of hypotheses from a class $\mathcal{H}$ rather than being limited to just $\mathcal{H}$, and (4) we use Lemma 1 to prove convergence and error bound results. These techniques can be applied or modified to develop endless variations on Bridge as well as completely new algorithms.

A big missing ingredient in justifying one method over another is sample complexity. In particular, we do not know how sample complexity depends on the lookahead j, or, in the case of $\Pi^{\epsilon,\delta}_{\mathcal{H}}$, how it depends on ϵ, and so we cannot properly trade off these parameters to achieve the best performance.

Our results are stated for the problem of value determination, but they apply to any situation with an operator that is a contraction with respect to a norm defined by a samplable distribution. For the problem of value determination, the operator is $\mathbf{T}$, the one-step backup operator, and it is a contraction under the norm defined by the stationary distribution of the Markov chain. As stated previously, this distribution can only be sampled exactly in the steady-state limit, so improvements in the theory are necessary. Finally, a big hurdle to a practical, implementable algorithm is the lack of useful, well-behaved (agnostic or not) learning algorithms. By applying the techniques used in developing Bridge, we hope to bridge the gap between the available supervised learning algorithms and those needed by theoretically justified reinforcement learning methods.

References

[Baird, 1995] Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on Machine Learning*, Tahoe City, CA, Proceedings of the Twelfth International Conference on Machine Learning 1995. Morgan Kaufmann.

[Bertsekas and Tsitsiklis, 1996] D. C. Bertsekas and J. N. Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, Belmont, Mass., 1996.

[Gordon, 1995] Geoffrey J. Gordon. Stable function approximation in dynamic programming. In *Proceedings of the Twelfth International Conference on Machine Learning*, Tahoe City, CA, July 1995. Morgan Kaufmann.

[Haussler, 1992] David Haussler. Decision theoretic generalizations of the pac model for neural net and other learning applications. *Information and Computation*, 100(1):78–150, 1992.

[Jaakkola *et al.*, 1995] Tommi Jaakkola, Satinder P. Singh, and Michael I. Jordan. Reinforcement learning algorithm for partially observable Markov decision problems. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 345–352, Cambridge, Massachusetts, 1995. MIT Press.

[Lee *et al.*, 1995] W.S. Lee, P.L. Bartlett, and R.C. Williamson. On efficient agnostic learning of linear combinations of basis functions. In *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, pages 369–376, 1995.

[Papavassiliou and Russell, 1998] V. Papavassiliou and S. Russell. Convergence of reinforcement learning with pac function approximators. Technical Report UCB//CSD-98-1005, University of California, Berkeley, 1998.

[Sutton, 1988] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, August 1988.

[Tsitsiklis and Van Roy, 1996] John N. Tsitsiklis and Benjamin Van Roy. An analysis of temporal-difference learning with function approximation. Technical Report LIDS-P-2322, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, 1996.

A Detailed Algorithm

Here we give details of the algorithm as well as define the relative error bound $\mathbf{errBound}(\alpha, \gamma, j)$. $\mathbf{BridgeValueDet}$ first calculates the necessary number of backup steps j in order to achieve the desired error bounds within the desired number of iterations. For each iteration it intelligently selects the parameter α and calls the subroutine BridgeStep with the current approximation. Finally it detects when it can halt successfully and returns a weighted tree of hypotheses.

$\mathbf{BridgeValueDet}(\nu, \epsilon_0, N)$

$$\epsilon_{max} = \frac{2R_{max}}{1-\gamma}; \quad \alpha_{goal} = \frac{\epsilon_0}{\epsilon_{max}}; \quad \alpha_{max} = \alpha_{goal}^{1/N}$$

Choose smallest j such that $\mathbf{errBound}(\alpha_{max}, \gamma, j) \leq \nu$

$n = 0$; $\alpha_{total} = 1$; $\mathbf{V}_0 =$ some initial hypothesis

LOOP UNTIL $\alpha_{total} \leq \alpha_{goal}$ {

Choose smallest α_n s.t.

$$\alpha_{max} \geq \alpha_n \gtrapprox \frac{\alpha_{goal}}{\alpha_{total}} \frac{\mathbf{errBound}(\alpha_n, \gamma, j)}{\nu}$$

$\mathbf{V}_{n+1} = \mathbf{BridgeStep}(\mathbf{V}_n, \alpha_n, j)$

$\alpha_{total} = \alpha_{total}\alpha_n$; $n = n + 1$}

RETURN $\mathbf{V}_n$

$\mathbf{BridgeStep}(\mathbf{V}, \alpha, j)$

$$\mathbf{B} = \mathbf{I} + \frac{1}{1-\gamma^j}(\mathbf{T}^j - \mathbf{I})$$

$$\mathbf{W} = \Pi_{\mathcal{H}}\mathbf{BV}; \qquad u = \|\mathbf{V} - \mathbf{W}\|_\pi$$

$$v = \mathbf{r}_\pi(\mathbf{V}, \mathcal{P}^V_{Tj}); \qquad w = \mathbf{r}_\pi(\mathbf{W}, \mathcal{P}^V_{Tj})$$

$$\beta = \alpha\sqrt{1-\gamma^{2j}} - \gamma^j\sqrt{1-\alpha^2}$$

$$\eta = \sqrt{\frac{(1-\alpha^2)}{(1-\beta^2)(1-\gamma^{2j})}}$$

$$\rho = \frac{v - w + u}{2u}\eta$$

RETURN $(1-\rho)\mathbf{V} + \rho\mathbf{W}$

$\mathbf{errBound}(\alpha, \gamma, j)$

$$\beta = \alpha\sqrt{1-\gamma^{2j}} - \gamma^j\sqrt{1-\alpha^2}$$

$$c_1 = -\frac{1}{\gamma^j}[1 + \sqrt{(1-\gamma^{2j})}\sqrt{(1-\alpha^2)}(\beta - \sqrt{1-\beta^2})]$$

$$c_2 = 2\sqrt{2}\beta\frac{\sqrt{1-\alpha^2}}{\sqrt{1-\gamma^{2j}}}; \quad c_3 = (\frac{\alpha}{1-\gamma^j})^2$$

$$c_4 = \frac{\beta}{1-\gamma^j}; \quad c_5 = \frac{2\gamma^j}{1-\gamma^{2j}}; \quad s = -\frac{2c_3 + c_2c_4}{c_2 + 2c_1c_4}$$

IF $c_4 \leq c_5$ THEN RETURN $+\infty$

$$\text{ELSE RETURN} \begin{cases} \sqrt{\frac{4c_1c_3 - c_2^2}{4(c_3 + c_2c_4 + c_1c_4^2)}} & \text{if} \quad s \leq c_5 \\ \sqrt{\frac{c_1c_5^2 + c_2c_5 + c_3}{(c_4 - c_5)^2}} & \text{if} \quad s > c_5 \end{cases}$$