Hybrid Least-Squares Algorithms for Approximate Policy Evaluation

Jeff Johns, Marek Petrik, and Sridhar Mahadevan
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003
{johns, petrik, mahadeva}@cs.umass.edu
Technical Report No. UM-CS-2008-045

Abstract

The goal of approximate policy evaluation is to "best" represent a target value function according to a specific criterion. Temporal difference methods and Bellman residual methods differ in the choice of the optimization criterion. So-called *residual algorithms*, which we refer to as *hybrid algorithms*, effectively combine these two solution methods. We propose two least-squares implementations of hybrid algorithms. This improves the previous incremental algorithm by making more efficient use of data. Furthermore, we provide a geometric interpretation of hybrid algorithms and demonstrate on a simple problem why a combination of temporal difference methods and Bellman residual methods may be useful. Experimental results in both small and large domains suggest hybrid algorithms can find solutions that lead to better policies when performing policy iteration.

1 Introduction

Solving Markov decision processes (MDPs) [1] with large or infinite state spaces requires some form of function approximation. Algorithms that use value functions, such as approximate value iteration and approximate policy iteration, encounter the problem of how best to represent a target function. Given a policy π, the *approximate policy evaluation* problem is to compute an approximate value function $\hat{V}$ that represents that policy's value function V^π. Temporal difference methods [2] and Bellman residual methods [3] offer different solutions to this problem. These methods, which behave very differently in practice, are similar in that they both stem from functions of the Bellman equation. Baird [4] proposed a combination of the two methods using a single parameter that at one extreme defaults to the temporal difference method and at the other extreme defaults to the Bellman residual method. Intermediate values combine the objective functions of the two methods. Baird termed this a residual algorithm, which for clarity we refer to as a hybrid algorithm.

The original hybrid algorithm [4] was incremental in nature as updates to the value function either occur after each observed transition or after each epoch. Incremental algorithms have the

disadvantage that they make inefficient use of data and require appropriately setting a learning rate. Least-squares algorithms have been proposed to remedy these issues. The least-squares temporal difference (LSTD) algorithm was proposed by Bradke and Barto [5] and generalized by Boyan [6] to learn a value function for a fixed policy. Least-squares methods have also been applied to control problems. Lagoudakis and Parr proposed least-squares policy iteration (LSPI) [7] for learning an action-value function. There has also been theoretical work along this line [8]. To take advantage of the benefits of least-squares algorithms, we propose two such implementations of hybrid algorithms.

We provide an analysis of hybrid algorithms in terms of a projection of the target function V^π. The difference amongst the various algorithms can also be understood from a geometric perspective. Hybrid algorithms offer more active control over the geometry of the Bellman equation. We illustrate why this control may be useful using a small MDP. We also compare the algorithms empirically.

2 Background and Terminology

We consider Markov decision processes (MDPs) [1] with a finite state space S of N states, a finite action space A, a state transition function $P(s, a, s')$ yielding the probability of moving from state $s \in S$ to state $s' \in S$ given action $a \in A$, and a reward function $R(s, a, s')$ giving the expected reward under the transition from $s \in S$ to $s' \in S$ given $a \in A$. Let P^π be a $N \times N$ matrix with $P^\pi(i, j) = P(i, \pi(i), j)$ and let R^π be a length N vector with $R^\pi(i) = \sum_j P^\pi(i, j)R(i, \pi(i), j)$. The value function V^π is a length N vector that solves the Bellman equation

$$V^\pi = R^\pi + \gamma P^\pi V^\pi \tag{1}$$

for policy π which maps states to actions. We consider infinite horizon problems where the discount factor γ is in the range $[0, 1)$.

Policy evaluation involves computing V^π for an arbitrary policy π. When V^π can be represented exactly, this is achieved by either direct computation $V^\pi = (I - \gamma P^\pi)^{-1} R^\pi$ or through iteration

$$V_{k+1} = T^\pi(V_k) = R^\pi + \gamma P^\pi V_k \tag{2}$$

where $T^\pi : \mathbb{R}^N \to \mathbb{R}^N$ is the Bellman operator. This method converges to V^π.

When an exact representation of V^π is infeasible, the value function must be approximated. We consider *linear* function approximation where a value function V is expressed as a linear combination of basis functions. This is written $V = \Phi w$ where $w \in \mathbb{R}^K$ is an adjustable parameter vector and $\Phi = [\Phi_1 | \dots | \Phi_K] \in \mathbb{R}^{N \times K}$ with each column Φ_i being a basis function. We assume without loss of generality that the basis functions are linearly independent. It is also typical for the number of free parameters to be much smaller than the number of states ($K \ll N$).

Approximate policy evaluation involves computing an approximate value function $\hat{V} = \Phi w$ that represents V^π. We describe the following four techniques for solving this problem.[1]

[1] The first three techniques were similarly described in [9].

1. **Optimal Approximate Solution (OPT)**
   If the target value function V^π were known, then it is easy to find an approximation $\hat{V}$ simply by projecting V^π onto the space spanned by the basis functions. This directly minimizes $\|\hat{V} - V^\pi\|_\rho$, where the errors for each state are weighted according to distribution ρ. Thus, the solution is $\hat{V} = \Phi w = \Pi_\rho V^\pi$ where $\Pi_\rho = \Phi(\Phi^T D_\rho \Phi)^{-1}\Phi^T D_\rho$ is a projection matrix and D_ρ is a diagonal matrix $D_\rho(i,i) = \rho(i)$. The difficulty of this method is in computing V^π, which can in principle be done using Monte Carlo methods.

2. **Bellman Residual Minimization (BR)**
   This technique computes a solution by minimizing the magnitude of the Bellman residual, $\|T^\pi(\hat{V}) - \hat{V}\|_\rho$, where the errors for each state are weighted according to distribution ρ.

$$\begin{aligned} \min_w \quad & \|T^\pi(\hat{V}) - \hat{V}\|_\rho \\ \min_w \quad & \|T^\pi(\Phi w) - \Phi w\|_\rho \\ \min_w \quad & \|R^\pi + \gamma P^\pi \Phi w - \Phi w\|_\rho \end{aligned} \tag{3}$$

   The least-squares solution to this problem is to find w such that $A_{BR} w = b_{BR}$ where:

$$\begin{aligned} A_{BR} &= \Phi^T (I - \gamma P^\pi)^T D_\rho (I - \gamma P^\pi)\Phi \\ b_{BR} &= \Phi^T (I - \gamma P^\pi)^T D_\rho R^\pi. \end{aligned}$$

   This technique, proposed in [3], has also been referred to as the residual-gradient method [4, 10], the quadratic residual method [9], and as the Bellman residual method [7, 8].

3. **Fixed Point Solution (FP)**
   This technique computes a solution by forcing $\hat{V}$ to be a fixed point of the Bellman equation. Since the Bellman operator can back up values out of the space spanned by the basis functions, it must be followed by a projection onto the column space of Φ (written $[\Phi]$) to ensure $\hat{V}$ is a fixed point. Thus, the solution is to minimize $\|\Pi_\rho T^\pi(\hat{V}) - \hat{V}\|_\rho$.

$$\begin{aligned} \min_w \quad & \|\Pi_\rho T^\pi(\hat{V}) - \hat{V}\|_\rho \\ \min_w \quad & \|\Pi_\rho T^\pi(\Phi w) - \Phi w\|_\rho \\ \min_w \quad & \|\Pi_\rho \left(T^\pi(\Phi w) - \Phi w\right)\|_\rho \\ \min_w \quad & \|\Pi_\rho \left(R^\pi + \gamma P^\pi \Phi w - \Phi w\right)\|_\rho \end{aligned} \tag{4}$$

   In the third step above, note that $\Phi w = \Pi_\rho \Phi w$. The least-squares solution to this problem is to find w such that $A_{FP} w = b_{FP}$ where:

$$\begin{aligned} A_{FP} &= \Phi^T D_\rho (I - \gamma P^\pi)\Phi \\ b_{FP} &= \Phi^T D_\rho R^\pi. \end{aligned}$$

We refer to this technique as the FP solution to be consistent with previous work [7], but it has also been referred to as the temporal difference method [10, 9].

4. **Hybrid Minimization (H)**
The BR solution minimizes the Bellman residual (Eq. 3) and the FP solution minimizes the projected Bellman residual (Eq. 4). Baird proposed *residual algorithms* [4] as a way to combine these techniques. The term "residual" algorithm was used to emphasize that it was different from a "residual-gradient" algorithm (his terminology for BR). To avoid any confusion, we refer to residual algorithms as *hybrid algorithms*. This name also emphasizes the fact that it is a combination of BR and FP. Baird's original version was an incremental algorithm. An update to the weight vector was computed by linearly combining the updates due to the BR and FP: $\Delta w_{H} = \beta \Delta w_{BR} + (1-\beta)\Delta w_{FP}$. We present two ways to formulate the hybrid technique using least-squares.

3 Hybrid Least-Squares Algorithms

The hybrid approach accounts for both the Bellman residual (which is minimized by the BR) *and* the projection of the Bellman residual onto $[\Phi]$ (which is minimized by the FP). Our first algorithm H_1 starts from the Bellman equation from which we derive a least-squares equation. The second algorithm H_2, instead of starting from first principles, begins directly from the BR and FP least-squares solutions. Both H_1 and H_2 when used with an exact representation (i.e. $\Phi = I_N$) produce the target value function V^π. When using an approximate representation, H_1 and H_2 produce different results and have different storage and computational requirements.

3.1 Algorithm H_1

We combine the BR and FP terms in the Bellman equation with a parameter $\beta \in [0,1]$. The problem is to minimize $\|\beta T^\pi(\hat{V}) + (1-\beta)\Pi_\rho T^\pi(\hat{V}) - \hat{V}\|_\rho$.

$$
\begin{aligned}
\min_w \quad & \left\| \beta T^\pi(\hat{V}) + (1-\beta)\Pi_\rho T^\pi(\hat{V}) - \hat{V} \right\|_\rho \\
\min_w \quad & \left\| \beta \left(T^\pi(\hat{V}) - \hat{V} \right) + (1-\beta)\Pi_\rho \left(T^\pi(\hat{V}) - \hat{V} \right) \right\|_\rho \\
\min_w \quad & \left\| (\beta I + (1-\beta)\Pi_\rho) \left(T^\pi(\hat{V}) - \hat{V} \right) \right\|_\rho \\
\min_w \quad & \left\| (\beta I + (1-\beta)\Pi_\rho) \left(R^\pi + \gamma P^\pi \Phi w - \Phi w \right) \right\|_\rho
\end{aligned} \tag{5}
$$

A least-squares equation of the form $A_{H_1} w = b_{H_1}$ can be derived from the minimization problem in Equation 5. To simplify the derivation, let $F = \beta I + (1-\beta)\Pi_\rho$ and let $G = (I - \gamma P^\pi)$:

$$
\begin{aligned}
FG\Phi w &= FR^\pi \\
(FG\Phi)^T D_\rho FG\Phi w &= (FG\Phi)^T D_\rho FR^\pi \\
\Phi^T G^T F^T D_\rho FG\Phi w &= \Phi^T G^T F^T D_\rho FR^\pi .
\end{aligned}
$$

It is easy to show that $F^T D_\rho F = D_\rho(\beta^2 I + (1-\beta^2)\Pi_\rho)$. The final result is therefore:

$$\begin{aligned} A_{H_1} &= \Phi^T(I-\gamma P^\pi)^T D_\rho(\beta^2 I + (1-\beta^2)\Pi_\rho)(I-\gamma P^\pi)\Phi \\ b_{H_1} &= \Phi^T(I-\gamma P^\pi)^T D_\rho(\beta^2 I + (1-\beta^2)\Pi_\rho)R^\pi. \end{aligned}$$

When the model is unknown or is too large, the matrix A_{H_1} and vector b_{H_1} must be estimated from samples. To achieve an *unbiased* estimate, two samples from each state are required [11]. This constraint has been removed in recent work by Antos, Szepesvári, and Munos [8]. They proposed a new Bellman residual algorithm that avoids double sampling by adding an auxiliary function which must be optimized. Unfortunately, it was shown that with linear function approximation their algorithm produces the same result as the FP solution. Therefore, we cannot simultaneously use their technique for avoiding double samples and search for hybrid solutions between the BR and FP.

To estimate A_{H_1} and b_{H_1} from samples, it is necessary to store three $K \times K$ matrices and two length K vectors. This can be seen by rewriting the equations:

$$\begin{aligned} A_{H_1} &= \beta^2 A_{BR} + (1-\beta^2)A_{FP}^T C^{-1} A_{FP} \\ b_{H_1} &= \beta^2 b_{BR} + (1-\beta^2)A_{FP}^T C^{-1} b_{FP} \end{aligned}$$

where $C = \Phi^T D_\rho \Phi$. Given double samples $\langle s, a, r', s' \rangle$ and $\langle s, a, r'', s'' \rangle$, the updates are

$$\begin{aligned} A_{BR} &= A_{BR} + \rho(s)(\phi(s) - \gamma\phi(s''))(\phi(s) - \gamma\phi(s'))^T \\ A_{FP} &= A_{FP} + \rho(s)\phi(s)(\phi(s) - \gamma\phi(s'))^T \\ C &= C + \rho(s)\phi(s)\phi(s)^T \\ b_{BR} &= b_{BR} + \rho(s)(\phi(s) - \gamma\phi(s''))r'' \\ b_{FP} &= b_{FP} + \rho(s)\phi(s)r'' \end{aligned}$$

where $\phi(s)$ is a column vector of length K associated with the s^{th} row of Φ (i.e. $\phi(s) = \Phi(s,:)^T$). Both the BR and FP least-squares problems only need to store one $K \times K$ matrix and one length K vector, whereas H_1 requires three matrices and two vectors. Moreover, the matrix C must be inverted when computing A_{H_1}. These issues motivated our second implementation.

3.2 Algorithm H_2

Rather than starting from the Bellman equation, we consider a direct combination of the BR and FP least-squares solutions:

$$\begin{aligned} A_{H_2} &= \beta A_{BR} + (1-\beta)A_{FP} &&= \Phi^T(I-\beta\gamma P^\pi)^T D_\rho(I-\gamma P^\pi)\Phi \\ b_{H_2} &= \beta b_{BR} + (1-\beta)b_{FP} &&= \Phi^T(I-\beta\gamma P^\pi)^T D_\rho R^\pi. \end{aligned}$$

By definition, this technique returns the BR solution when $\beta = 1$ and the FP solution when $\beta = 0$. Only one $K \times K$ matrix and one length K vector are required for H_2. The incremental update for each double sample $\langle s, a, r', s' \rangle$ and $\langle s, a, r'', s'' \rangle$ has the form:

$$\begin{aligned} A_{H_2} &= A_{H_2} + \rho(s)(\phi(s) - \beta\gamma\phi(s''))(\phi(s) - \gamma\phi(s'))^T \\ b_{H_2} &= b_{H_2} + \rho(s)(\phi(s) - \beta\gamma\phi(s''))r''. \end{aligned}$$

3.3 Difference Between H_1 and H_2

Aside from the difference in data structures used by H_1 and H_2, it is useful to elucidate any other differences. To make this comparison more obvious, the least-squares equations can be rewritten as follows

$$\begin{aligned} A_{H_1} &= \Phi^T \left(\beta G^T G + (1-\beta) G^T \Pi G\right) \Phi \\ A_{H_2} &= \Phi^T \left(\beta G^T G + (1-\beta) G\right) \Phi \\ b_{H_1} &= \Phi^T \left(\beta G^T + (1-\beta) G^T \Pi\right) R^\pi \\ b_{H_2} &= \Phi^T \left(\beta G^T + (1-\beta) I\right) R^\pi \end{aligned}$$

where we again use the abbreviation $G = (I - \gamma P^\pi)$. For simplicity, the weighting ρ was assumed uniform and the parameter β was used for H_1 (instead of β^2). Consider the extreme values of β. Both H_1 and H_2 clearly produce the BR solution when $\beta = 1$. When $\beta = 0$, it is obvious that FP and H_2 are identical. It is less obvious that H_1 produces the same solution, but this can in fact be shown. The interesting case occurs when $0 < \beta < 1$ because the H_1 and H_2 solutions differ. For most typical problems where function approximation is required (i.e. when Π is far from equaling the identity matrix), $\|A_{H_1}\| < \|A_{H_2}\|$. This in turn means that the H_2 solution will be weighted more toward the FP component whereas the H_1 solution will be more weighted toward the BR component. This relationship did indeed hold up in the experiments. Lastly, in terms of practical implications, we note that A_{H_2} has a smaller condition number than A_{H_1} which means the H_2 solution can potentially be more robust depending on the particular problem.

3.4 Other Possible Algorithms

The two proposed hybrid algorithms *implicitly* constrain the Bellman residual by the choice of the parameter β. This constraint could be made explicit. The problem would be to minimize the projection of the Bellman residual subject to an inequality constraint on the Bellman residual (either on its magnitude or component-wise).

$$\min_w \quad \|A_{FP} w - b_{FP}\|_\rho$$
$$\text{subject to:} \quad \|A_{BR} w - b_{BR}\|_\rho \leq \delta \quad \text{or:} \quad \pm (A_{BR} w - b_{BR}) \leq \Delta$$

The parameters δ (a positive scalar) and Δ (a positive vector) must be set appropriately based on the minimal value of the Bellman residual magnitude attained with the BR. We point out the possibility of explicitly controlling the Bellman residual to be thorough. However, since this increases the computational complexity, we limit our discussion to the two simple least-squares algorithms H_1 and H_2.

4 Analysis

4.1 Projections of the Target Function

The first three approximate policy evaluation techniques were shown to be images of the target function V^π under different projection operations [10]. More specifically, each method $X =$

$\{OPT, BR, FP\}$ produces an approximate value function with the following form: $\hat{V} = \Phi w = \Phi A_X^{-1} b_X = \Phi(\Phi^T D_X \Phi)^{-1} \Phi^T D_X V^\pi$.[2] The matrix D_X controls the weighting of the projection and takes on the following values [10]:

$$\begin{aligned} D_{OPT} &= D_\rho \\ D_{BR} &= (I - \gamma P^\pi)^T D_\rho (I - \gamma P^\pi) \\ D_{FP} &= D_\rho (I - \gamma P^\pi) \end{aligned}$$

The hybrid methods have a similar characterization.

$$\begin{aligned} D_{H_1} &= (I - \gamma P^\pi)^T D_\rho (\beta^2 I + (1 - \beta^2)\Pi_\rho)(I - \gamma P^\pi) \\ D_{H_2} &= (I - \beta\gamma P^\pi)^T D_\rho (I - \gamma P^\pi) \end{aligned}$$

4.2 Geometry of the Bellman Equation

Each approximate policy evaluation algorithm uses the Bellman equation in different ways to compute a value function. There is an intuitive geometric perspective to the algorithms when using linear function approximation. We expand on the original presentation of this perspective in [7].

The Bellman equation with linear function approximation has three components: $\hat{V}$, $T^\pi(\hat{V})$, and $\Pi_\rho T^\pi(\hat{V})$. These components geometrically form a triangle where $\hat{V}$ and $\Pi_\rho T^\pi(\hat{V})$ reside in the space spanned by Φ while $T^\pi(\hat{V})$ is, in general, outside this space. This is illustrated in the leftmost triangle of Figure 1. The three-dimensional space in the figure is the space of exact value functions while the two-dimensional plane represents the space of approximate value functions in $[\Phi]$. The angle between subspace $[\Phi]$ and the vector $T^\pi(\hat{V}) - \hat{V}$ is denoted θ. The different approximate policy evaluation methods compute solutions that minimize the length of different sides of the triangle. The second triangle in Figure 1 shows the BR solution, which minimizes the length of $T^\pi(\hat{V}) - \hat{V}$. The third (degenerative) triangle shows the FP solution, which minimizes the length of $\Pi_\rho T^\pi(\hat{V}) - \hat{V}$. This length is 0 which means $\theta_{FP} = 90^\circ$. The fourth triangle shows the H, which minimizes a combination of the lengths of the two sides. In general, θ_H lies between θ_{BR} and 90°. The hybrid solution allows for controlling the shape of this triangle.

Empirical evidence suggests the BR is a more stable method, but the FP finds better policies [7]. This effect was also analyzed theoretically [9, 12]. We have also observed the same behavior. The BR tends to compute weights w whose greedy policies derived from $\hat{V}$ do not change much from iteration to iteration. The FP can compute weights and policies that vary greatly between iterations. One motivation for the H is to combine the stability of the BR with the FP's ability to find better policies. In this light, the BR component of the H solution can be thought of as implicitly providing regularization. The hybrid algorithms have the flexibility of finding solutions that are almost fixed points but have more desirable properties (smaller Bellman residuals).

4.3 Chain MDP

To illustrate the differences in stability between the BR and FP, consider the simple six state MDP described in Figure 2 with discount factor $\gamma = 0.99$. The optimal policy is to move right in the first

[2] The V^π term comes from the equality $R^\pi = (I - \gamma P^\pi)V^\pi$.

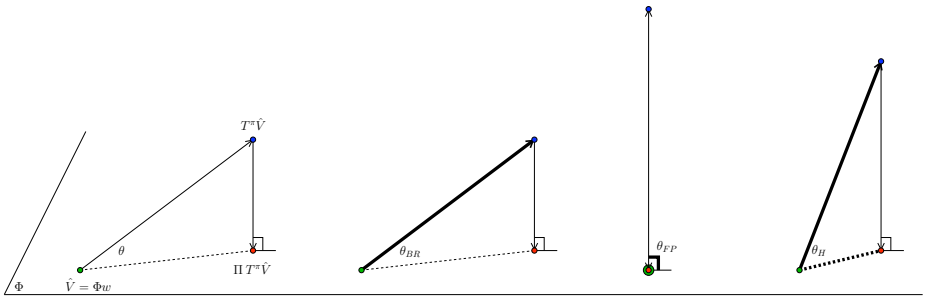


Figure 1: The triangle on the left shows the general form of the Bellman equation. The three triangles on the right correspond to the different approximate policy evaluation methods (BR, FP, and H) where the bold lines indicate what is being optimized by each method.

three states and left in the last three states ($\pi^* = RRRLLL$). Let the initial policy be $\pi_0 = LLLLLL$ and assume there are three basis functions corresponding to the first three proto-value functions, or eigenvectors of the graph Laplacian [13]. These basis functions are symmetric and are rich enough to represent an approximate value function whose corresponding greedy policy is π^*. We also assume the distribution ρ is uniform ($D_\rho = I$), which is appropriate when doing policy iteration [14]. The approximate value functions $\hat{V}^{\pi_0}_{BR}$, $\hat{V}^{\pi_0}_{FP}$, and $\hat{V}^{\pi_0}_{H_1}$ were computed according to the least-squares solutions described in Sections 2 and 3. Then the model was used to determine a policy π_1 that is greedy with respect to $\hat{V}$. The BR produces a policy $\pi_1 = LLLLLL$ while the FP produces a policy $\pi_1 = RRRRRR$. Thus, after one round of policy iteration, the BR converges on the initial policy and the FP completely flips the policy. Moreover, since the model and basis functions are symmetric, the FP oscillates forever between $LLLLLL$ and $RRRRRR$. This example demonstrates the stability of the BR and the FP's potential instability. The hybrid method produces a policy π_1 in between the two extremes. The exact policy depends on β and is shown in Figure 3(c). The trade-off between the Bellman residual and the projection of the Bellman residual is shown in Figure 3(a) and the angle θ_{H_1} is shown in Figure 3(b).

While this result clearly holds when the basis functions are symmetric, this behavior also occurs with asymmetric bases. For example, consider a polynomial basis with $\phi_1(s) = 1$, $\phi_2(s) = s$, and $\phi_3(s) = s^2$, appropriately orthonormalized. The BR still converges to the initial policy, the FP oscillates between the two extreme policies, and the H finds policies between the two extremes.

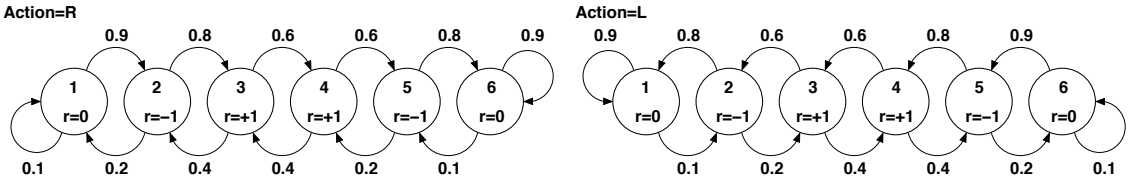
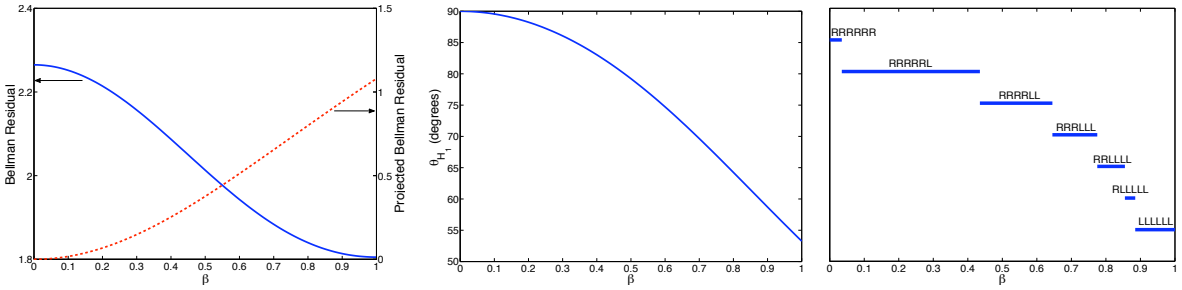


Figure 2: Reward and transition functions for a six state MDP with two possible actions.



(a) Bellman residual (left) and projected Bellman residual (right).

(b) Angle θ_{H_1}.

(c) Greedy policy w.r.t. $\hat{V}^{\pi_0}_{H_1}$.

Figure 3: Results of approximate policy evaluation for H_1 while varying β.

5 Experiments

5.1 Grid MDP

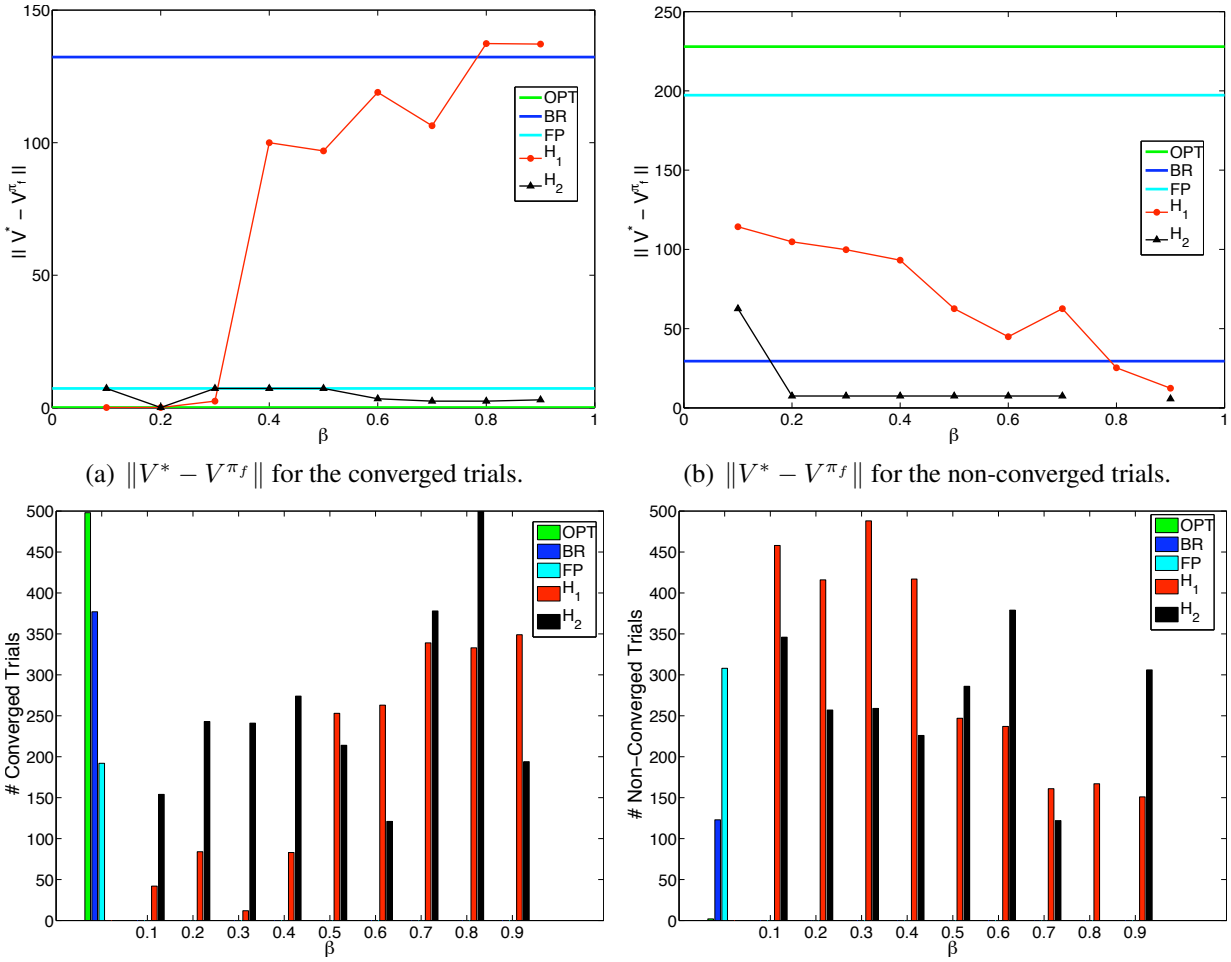
We compared all methods on a 10×10 grid MDP. The MDP has 100 states, 4 actions that have probability 0.9 of success, a 0.95 discount factor, and a reward of +1 in one corner and +2 in the diagonal corner. Fifteen Laplacian eigenvectors [13] were used as basis functions.

We ran 500 trials. Each trial began with a randomly initialized policy, then policy iteration was run using each policy evaluation method until the weight vector converged or 500 iterations were reached. The model was used during policy iteration to avoid any difficulty comparing the various methods due to sampling. The result of policy iteration is a final policy π_f. We evaluate these policies by computing V^{π_f} exactly and comparing it with V^*. The results, which are broken into the trials that converged and those that did not converge, are shown in Figure 4. The same results, along with information about the variance over the 500 trials, are also reported in Table 1.

The results show that BR is more stable but FP finds better policies when it converges (smaller $\|V^* - V^{\pi_f}\|_2$). The non-converged trials for BR produced better policies than the converged trials. Since BR tends to make small changes to the value function between rounds of policy iteration, it is not surprising that early convergence (starting from a random policy) leads to very suboptimal policies. This same phenomenon occurred for H_1, $\beta = 0.5$-0.9. H_1's performance transitions between FP at small values of β and BR at large values of β. It does not achieve improved performance at intermediate values. The most interesting aspect of this experiment is the excellent performance of H_2 and the method's robustness across all β values.

5.2 Tetris

We have presented hybrid least-squares algorithms for approximating value functions, but the same idea holds for approximating action-value functions. We omit the equations for lack of space, but they are very similar to those in Section 3. We tested all policy evaluation methods on the problem of learning an approximate action-value function for Tetris. Ten basis functions over state-action pairs (s, a) were used. The first four are for the current state s: maximum height, number of holes, sum of absolute height differences between adjacent columns, and the mean height. The next four basis functions are the change in the value of the first four features after taking action a from s.



(a) $\|V^* - V^{\pi_f}\|$ for the converged trials.

(b) $\|V^* - V^{\pi_f}\|$ for the non-converged trials.

(c) Number of converged trials.

(d) Number of non-converged trials.

Figure 4: Results of policy iteration for the grid MDP. The median value of $\|V^* - V^{\pi_f}\|$ is plotted versus β for the converged trials (a) and non-converged trials (b). The number of trials that converged and did not converged (out of the 500 total) is shown in (c) and (d) respectively.

The last two are the change in the score and a constant 1. This feature set was proposed in [15].

Forty episodes of data ($\sim$30,000 samples) were generated using an initial policy w^{π_0} = [-1, -10, -1, -1, -2, -11, -2, -2, 50, 10]^T. We ran policy iteration starting from w^{π_0} until the weight vector converged or 100 iterations were reached. Instead of generating double samples (to form unbiased estimates of A and b), we used the model to compute the expectation over next-states and actions. For Tetris, each action results in seven equally likely next-states corresponding to the seven Tetris pieces. This method of using the model instead of samples is described in [7] (see **LSTDQ-Model**).

We tested the learned policies 50 times. Each time, we generated a random ordered set of pieces that all policies were forced to place to make the comparison more accurate. The average score over the 50 trials is shown in Table 2. The initial policy w^{π_0} scored 310 on average. Policy iteration converged in less than 7 iterations for the FP and H_2, whereas the BR and H_1 experiments did not converge. The performance split along this division. The final policy computed using

Table 1: Results of policy iteration for the grid MDP. The triplets of numbers are the first, second, and third quartiles to give a sense of the variance.

Technique	Converged Trials			Non-converged Trials	
	# Trials	# Iterations	$\|V^* - V^{\pi_f}\|_2$	# Trials	$\|V^* - V^{\pi_f}\|_2$
OPT	498	9 / 11 / 16	0.1 / 0.1 / 0.1	2	- / 227.9 / -
BR	377	7 / 8 / 10	78.2 / 132.3 / 159.3	123	7.3 / 29.5 / 136.0
FP	192	17.5 / 69 / 123	0.1 / 7.3 / 7.5	308	152.6 / 197.3 / 225.4
H_1, β=0.1	42	7 / 9 / 13	0.1 / 0.1 / 0.1	458	65.8 / 114.3 / 202.6
H_1, β=0.2	84	7 / 8 / 11	0.1 / 0.1 / 7.3	416	65.8 / 104.8 / 141.9
H_1, β=0.3	12	7 / 8.5 / 9	2.5 / 2.5 / 2.5	488	65.8 / 99.8 / 142.9
H_1, β=0.4	83	8 / 10 / 11	100.0 / 100.0 / 100.0	417	16.8 / 93.2 / 148.8
H_1, β=0.5	253	8 / 9 / 10	6.6 / 96.9 / 143.7	247	25.3 / 62.6 / 177.7
H_1, β=0.6	263	7 / 9 / 10	25.3 / 119.0 / 144.4	237	12.4 / 44.9 / 158.7
H_1, β=0.7	339	7 / 9 / 10	6.6 / 106.4 / 150.9	161	13.4 / 62.6 / 154.8
H_1, β=0.8	333	7 / 8 / 9	90.4 / 137.4 / 169.3	167	7.3 / 25.3 / 135.2
H_1, β=0.9	349	7 / 8 / 9	96.6 / 137.2 / 164.2	151	6.6 / 12.4 / 111.8
H_2, β=0.1	154	14 / 52 / 163	0.1 / 7.3 / 7.3	346	7.5 / 62.6 / 156.0
H_2, β=0.2	243	10 / 45 / 121	0.1 / 0.1 / 7.3	257	7.5 / 7.5 / 95.4
H_2, β=0.3	241	8 / 22 / 69.5	0.1 / 7.3 / 7.3	259	7.5 / 7.5 / 7.5
H_2, β=0.4	274	7 / 9 / 22	7.3 / 7.3 / 7.3	226	7.5 / 7.5 / 7.5
H_2, β=0.5	214	8 / 9 / 11	7.3 / 7.3 / 7.3	286	7.3 / 7.5 / 7.5
H_2, β=0.6	121	7 / 8 / 9	3.0 / 3.4 / 5.7	379	5.7 / 7.5 / 7.5
H_2, β=0.7	378	10 / 11 / 12	2.5 / 2.5 / 2.5	122	7.5 / 7.5 / 7.5
H_2, β=0.8	500	12 / 14 / 15	2.5 / 2.5 / 2.5	0	-
H_2, β=0.9	194	8 / 17 / 20	3.0 / 3.0 / 3.0	306	5.7 / 5.7 / 5.7

the BR rarely removed a line. This was also the case for policies learned using H_1 except when $\beta = 0.4$. On the other hand, the FP and H_2 policies performed at least as well as the initial policy and in some cases significantly better. The best policy was computed using H_2 with $\beta = 0.1$.

Table 2: Results of policy iteration for Tetris. An asterisk indicates policy iteration converged.

Technique	Score	Technique	Score	Technique	Score	Technique	Score
BR	0	H_1, β=0.4	295	H_1, β=0.9	0	H_2, β=0.5*	455
FP*	630	H_1, β=0.5	60	H_2, β=0.1*	800	H_2, β=0.6*	395
H_1, β=0.1	15	H_1, β=0.6	5	H_2, β=0.2*	580	H_2, β=0.7*	370
H_1, β=0.2	0	H_1, β=0.7	5	H_2, β=0.3*	645	H_2, β=0.8*	405
H_1, β=0.3	80	H_1, β=0.8	0	H_2, β=0.4*	515	H_2, β=0.9*	330

6 Conclusions

The FP and BR solutions can be combined to form a hybrid approximate policy evaluation algorithm. We proposed two ways to implement a hybrid algorithm using least-squares methods, thus improving efficiency over the original incremental algorithm [4]. The first implementation is

derived directly from the Bellman equation. The second implementation is a linear combination of the fixed point FP and Bellman residual methods. We analyzed the algorithms in terms of projections of the target function and we showed that hybrid algorithms have an intuitive geometric interpretation.

Policy iteration experiments were conduced on a simple grid MDP so that the quality of the learned policies could be determined analytically. Experiments were also run on the challenging task of learning to play Tetris where learned policies were evaluated empirically. In both domains, the hybrid algorithm H_2 discovered policies that performed much better than BR and as well as (and in some instances better than) FP. A surprising finding was H_2's robustness for a wide range of β values. One would expect that for β values close to 1, the difference between BRM and H_2 would be minimal. A theoretical explanation of this effect would be useful. Another interesting area for future work is to provide a mechanism for automatically setting β.

References

[1] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, New York, NY, 1994.

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

[3] P. Schweitzer and A. Seidmann. Generalized polynomial approximations in Markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110:568–582, 1985.

[4] L. Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Machine Learning*, pages 30–37, 1995.

[5] S. Bradke and A. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1-3):33–57, 1996.

[6] J. Boyan. Least-squares temporal difference learning. In *Proceedings of the 16th International Conference on Machine Learning*, pages 49–56, 1999.

[7] M. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

[8] A. Antos, C. Szepesvári, and R. Munos. Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71(1):89–129, 2008.

[9] R. Munos. Error bounds for approximate policy iteration. In *Proceedings of the 20th International Conference on Machine Learning*, pages 560–567, 2003.

[10] R. Schoknecht. Optimality of reinforcement learning algorithms with linear function approximation. In *Advances in Neural Information Processing Systems 15*, pages 1555–1562, 2003.

[11] R. Sutton and A. Barto. *Reinforcement Learning*. MIT Press, Cambridge, MA, 1998.

[12] L. Li. A worst-case comparison between temporal difference and residual gradient with linear function approximation. In *Proceedings of the 25th International Conference on Machine Learning*, pages 560–567, 2008.

[13] S. Mahadevan. Representation policy iteration. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, pages 372–379, 2005.

[14] D. Koller and R. Parr. Policy iteration for factored MDPs. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 326–334, 2000.

[15] M. Lagoudakis, R. Parr, and M. Littman. Least-squares methods in reinforcement learning for control. In *Proceedings of the Second Hellenic Conference on Artificial Intelligence*, pages 249–260, 2002.