

Metrics for Temporal Difference Learning

Mance E. Harmon

*Wright Laboratory, Wright-Patterson AFB,
Dayton, OH 45433*

Leemon C. Baird, III

*Department of Computer Science, United States Air Force Academy,
Colorado Springs, CO 80840*

For an absorbing Markov chain with a reinforcement on each transition, Bertsekas (1995a) gives a simple example where the function learned by TD(λ) depends on λ . Bertsekas showed that for $\lambda=1$ the approximation is optimal with respect to a least-squares error of the value function, and that for $\lambda=0$ the approximation obtained by the TD method is poor with respect to the same metric. With respect to the error in the values, TD(1) approximates the function better than TD(0). However, with respect to the error in the differences in the values, TD(0) approximates the function better than TD(1). TD(1) is only better than TD(0) with respect to the former metric rather than the latter. In addition, direct TD(λ) weights the errors unequally, while residual gradient methods (Baird, 1995, Harmon, Baird, & Klopf, 1995) weight the errors equally. For the case of control, a simple Markov decision process is presented for which direct TD(0) and residual gradient TD(0) both learn the optimal policy, while TD(1) learns a suboptimal policy. These results suggest that, for this example, the differences in state values are more significant than the state values themselves, so TD(0) is preferable to TD(1).

1 Introduction

Bertsekas (1995a) proposes a counterexample to the use of temporal difference methods for approximating value functions in the context of Markov chains and suggests that his results extend to the domain of Markov decision processes as well. Bertsekas uses a least-squares error with respect to the value function as the metric for evaluating the functions learned by TD(1) and TD(0). In the counterexample, the function learned by TD(0) is inferior to that learned by TD(1) when using this metric. However, we observe that other metrics produce different results.

Bertsekas' examples are Markov chains with states $0, 1, 2, \dots, n$. Each transition returns a reinforcement with the exception of state 0, which is cost-free and absorbing and is eventually reached from every other state. For each initial state x the objective is to estimate the expected total return $V^*(x)$ received when following a series of transitions from state x to the terminal state.

Like Bertsekas, we use a linear function approximator of the form $V(x, w) = xw$ to approximate the optimal value function $V^*(x)$, where x is the state and w is a weight vector. Sutton's TD(λ) method (1988) is a gradient-descent-like algorithm for obtaining a suitable vector w after observing a large number of simulated trajectories of the Markov chain. As Bertsekas (1995a) and Sutton (1988) point out, TD(1) can be considered a stochastic gradient descent method for minimizing an expected value of the square of the error $V^*(x) - V(x, w)$.

On the other hand, TD(0) can be viewed as a stochastic gradient-descent-like method for minimizing an expected value of the temporal difference error $[r(x, x_{t+1}) + V(x_{t+1}, w)] - V(x_t, w)$. The TD(1) algorithm attempts to approximate V^* by finding a function that is a direct approximation of the value function for all x , while the TD(0) algorithm attempts to approximate V^* by finding a function whose differences in values approximates the differences in values of the value function for all x , while simultaneously approximating the value of the terminal state. For this discussion, it is useful to define an operator that represents the difference in value of two adjacent states. We define this operator δf in equation (0). This operator is somewhat like a derivative or slope, particularly when $\gamma=1$ (as is the case for the examples presented here), and is equivalent to the difference between the two sides of the Bellman equation (Bertsekas, 1995b).

$$\delta f(x_t) \equiv \gamma f(x_{t+1}) - f(x_t) \quad (0)$$

The TD(λ) update equation is defined as follows:

$$\begin{aligned} w = w + \alpha [r(x_{N+1}, x_N) + \gamma V(x_{N+1}, w) - V(x_N, w)] \\ [\lambda^{N-1} \nabla V(x_t, w) + \lambda^{N-2} \nabla V(x_{t+1}, w) + \Lambda + \nabla V(x_N, w)] \end{aligned} \quad (1)$$

Note that if $\lambda=0$ then equation (1) reduces to equation (2) which is the update equation for *value iteration*:

$$w = w + \alpha [r(x_t, x_{t+1}) + \gamma V(x_{t+1}, w) - V(x_t, w)] \nabla V(x_t, w) \quad (2)$$

By solving the temporal difference error for r one can see that TD(0) attempts to find a function in which δV approximates δV^* . In a given state, there will be no change in the weight vector w if $\delta V = \delta V^*$, assuming $\gamma=1$.

Noting that TD(1) finds a function that approximates the value function V^* directly and TD(0) finds a function V in which δV approximates δV^* , we can better evaluate the empirical results Bertsekas presented.

2 Markov chains

The following two examples are identical to the examples presented in Bertsekas (1995a). We use a linear function approximator of the form $V(x, w) = xw$. The weight w was calculated exactly rather than approximated in simulation. The state transitions and associated reinforcements are deterministic. From state x we move to state $x-1$ with a given reinforcement r_x . All simulation runs start at state n and end at state 0 after visiting all the states $n-1, n-2, \dots, 1$ in succession. The temporal difference associated with the transition from x to $x-1$ is $r_x + V(x-1, w) - V(x, w) = r_x - w$ and the gradient is $\nabla V(x, w) = x$.

Example 1

Figure 1 shows the value function V^* and the functions learned by TD(1) and TD(0) for $n=50$ and for $r_1=1$, $r_x=0$ for all $x \neq 1$.

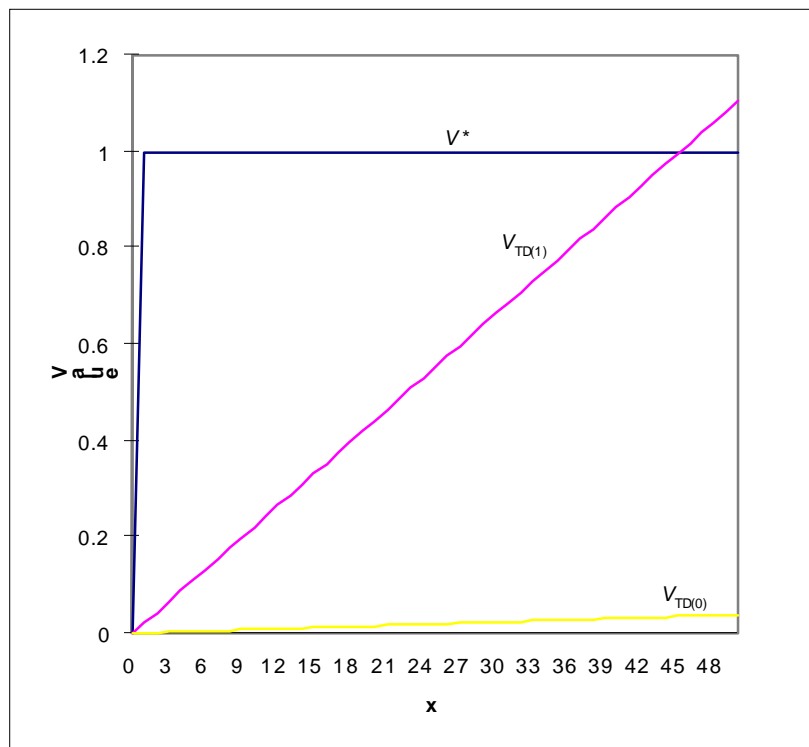


Figure 1: The optimal value function V^* and the functions learned by TD(1) and TD(0) for the case $r_1=1$, $r_x=0$ for all $x \neq 1$. The function learned by TD(1) is a better approximation to V^* than is TD(0) according to the 2-norm.

In Figure (1), we can see that TD(0) yields a poor approximation to the value function. However, TD(0) is not trying to directly approximate the value function. Rather, TD(0) generated a function $V_{TD(0)}$ for which $\delta V_{TD(0)}$ approximates δV^* . If one uses the error in the values as a metric, TD(1) learns the better function. However, if one uses the error in the *weighted* differences as a metric, TD(0) learns the better function. TD(λ) does not weight the errors (differences) equally in all states. The TD(λ) algorithm weights the error in each state proportional to the frequency with which that state is trained and the magnitude of $\nabla_w V(x)$ for the given state. TD(0) finds the

best weighted 2-norm fit to δV^* . It is not obvious which metric is the most appropriate to use. Figure (2) shows δV^* , $\delta V_{TD(1)}$, and $\delta V_{TD(0)}$.

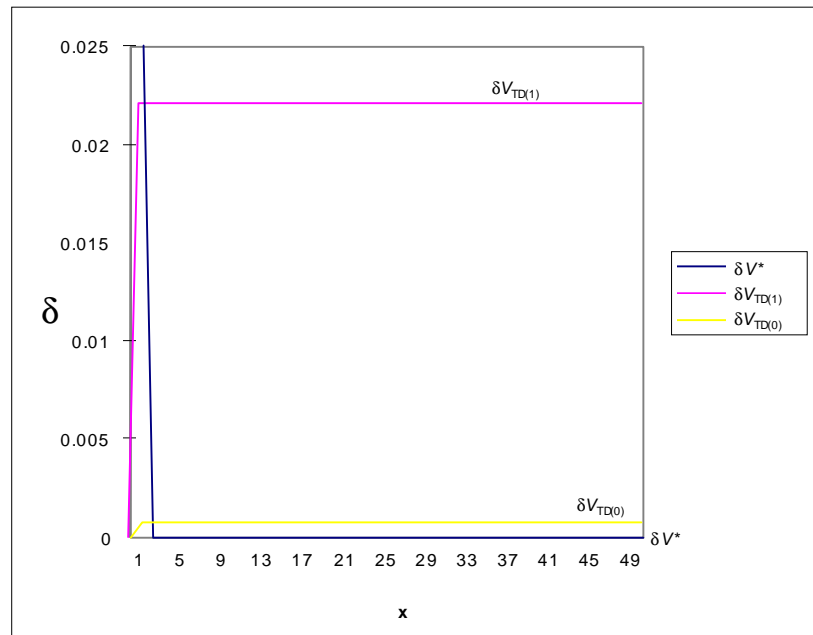


Figure 2: $\delta V_{TD(0)}$ is a better approximation to δV^* than is $\delta V_{TD(1)}$ according to the weighted 2-norm. In this graph, $\delta V^*=1$ for $x=1$, and is 0 for $\forall x \neq 1$.

Example 2

Figure (3) shows the value function V^* as well as the functions learned by TD(1) and TD(0), respectively, for $n=50$ and for $r_n=-(n-1)$, $r_x=1$ for all $x \neq n$.

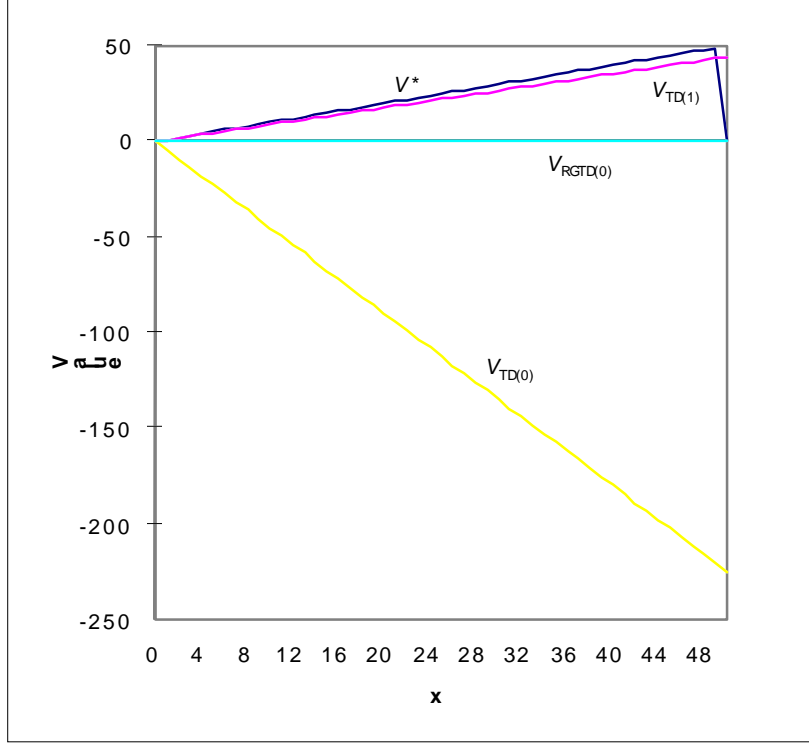


Figure 3: The optimal value function V^* , and the functions learned by TD(1), direct TD(0), and residual gradient TD(0) for the case $r_n = -(n-1)$, $r_x = 1$ for all $x \neq n$. $V_{TD(1)}$ is the best approximation to V^* according to the weighted or unweighted 2-norm.

In Figure (3) the TD(0) algorithm finds a function $V_{TD(0)}$ for which $\partial V_{TD(0)}$ approximates δV^* and for which earlier states (50,49,...) are given more weight than later states (... ,2,1,0). As previously stated, TD(λ) does not weight the errors equally in all states. The TD(λ) algorithm weights the error in each state proportional to the frequency with which that state is trained and the magnitude of $\nabla_w V(x)$ for the given state. The temporal difference error multiplied by the derivative of $V(x)$ when $x=50$ yields a value that is much larger than in any other state and influences the approximation by a commensurably disproportionate amount. We stated earlier that TD(λ) is a gradient-descent-like method. TD(λ) is not a true gradient descent algorithm for $\lambda < 1$ because no single error function, E , exists whose derivative, $\nabla_w E^2$, is that found in the weight vector update equation used by the TD(λ) algorithm given in equation (1). A class of algorithms that does perform gradient descent on a single error function, *residual algorithms* (Baird, 1995, Harmon, Baird, and Klopf, 1995), weights each temporal difference error based solely on the frequency with which it is trained and can be viewed as stochastic gradient descent on the mean squared Bellman residual.

Residual algorithms should be used if one prefers the weighting of states be a function of only the frequency with which a state is trained, and not a function of both the frequency with which a state is trained and the derivative of the value function with respect to the given state. Residual gradient TD(0) is an algorithm that weights the temporal error in a given state based solely on how often that state is visited. This update is presented in equation (6) and is the equivalent of residual gradient value iteration. The function learned by residual gradient TD(0), for the case where $n=50$, $r_n = -(n-1)$, and $r_x = 1$ for all $x \neq n$, is also presented in Figure (3).

$$w = w + \alpha [r(x, x_{t+1}) + V(x_{t+1}, w) - V(x_t, w)] [\nabla V(x_{t+1}, w) - \nabla V(x_t, w)] \quad (6)$$

Figure (4) shows δV^* as well as $\delta V_{TD(0)}$, $\delta V_{TD(1)}$, and $\delta V_{RGTD(0)}$ for this example. In Figure (4) we can see that the function learned by direct TD(0) is disproportionately influenced by the δ s in the higher (earlier) states, while residual gradient TD(0) weights the temporal difference errors equally.

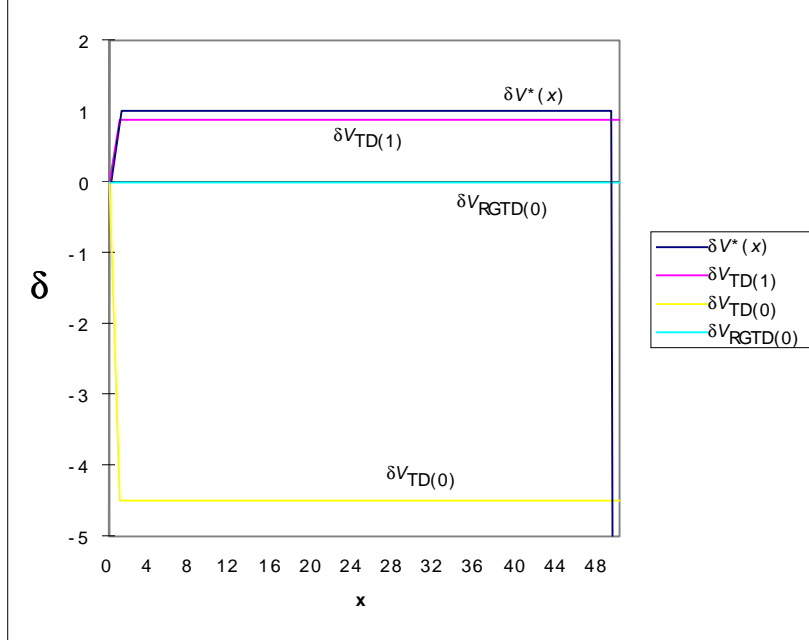


Figure 4: δV^* and the functions learned by direct TD(0) and residual gradient TD(0) respectively for the case $r_n=-(n-1)$, $r_x=1$ for all $x \neq n$. $\delta V^*=-49$ when $x=50$. $\delta V_{RGTD(0)}$ is the best approximation to δV^* according to the unweighted 2-norm and $\delta V_{TD(0)}$ is the best approximation according to the weighted 2-norm.

3 Markov decision processes

In section (1) we demonstrated that TD(1) finds a function that approximates the value function directly and direct TD(0) finds a function V for which δV approximates δV^* and V approximates V^* for the terminal states. In section (2) we demonstrated that TD(λ) weights the temporal difference errors based on the derivative of the value function with respect to the weights for the given state and the frequency with which that state is trained. We also showed that residual gradient TD(λ) weights the errors based only on the frequency with which they are trained. Bertsekas used a least-squares error criterion with respect to the value function for evaluating the utility of TD(λ) for $\lambda < 1$. Observing Figures (1) and (3), the learned functions with respect to the optimal value function V^* , we see that TD(1) learns a better approximation of V^* than direct TD(0) or residual gradient TD(0) according to the 2-norm. Observing Figures (2) and (4), the differences in the values of successive states, we see that residual gradient TD(0) and direct TD(0) learn functions that are better approximations with respect to the differences than the function learned by TD(1). It is not obvious which metric to use when evaluating the utility of the various TD methods. In the domain of Markov decision processes another metric can be defined: the sum of the reinforcements when following a learned policy from a given state x . TD(1) finds a function that approximates the value function V^* directly and TD(0) finds a function V for which δV approximates δV^* . Which of these two methods is best suited for maximizing the sum of the reinforcements in a control context? Here we show that, in at least one case, both the direct TD(0) and residual gradient TD(0) algorithms learn functions that yield optimal control policies, while TD(1) learns a function that yields a suboptimal control policy.

Figure (5) depicts the following MDP. The state \mathbf{x} is a two element vector $\{x_1, x_2\}$. In state $\{0,0\}$ are two possible actions: transition to state $\{0,1\}$, or transition to state $\{1,0\}$. Each action leads to a Markov chain. The states succeeding state $\{0,0\}$ are labeled $\{\{0,1\}, \{0,2\}, \dots, \{0,n\}\}$ and $\{\{1,0\}, \{2,0\}, \dots, \{n,0\}\}$. We assume that states $\{0,n\}$ and $\{n,0\}$ yield a return of 0 and are absorbing.

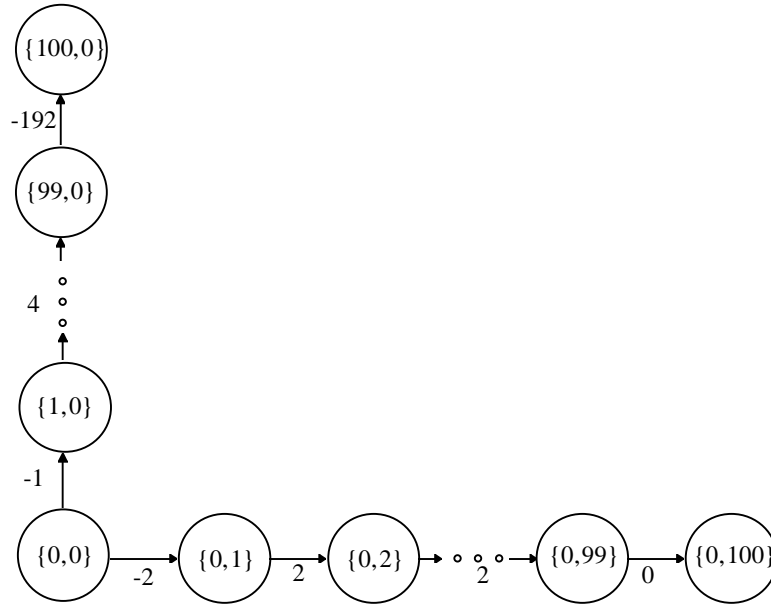


Figure 5: MDP with reinforcements $r_{\{0,0\}\{1,0\}}=-1$, $r_{\{0,0\}\{0,1\}}=-2$, $r_{\{99,0\}}=-192$, $r_{\{0,99\}}=0$, $r_{\{x,0\}}=4$, $\forall x \notin \{\{0,0\},\{99,0\}\}$, and $r_{\{0,x\}}=2$, $\forall x \notin \{\{0,0\},\{0,99\}\}$.

We use a linear approximation of the form $V(\mathbf{x}, \mathbf{w})=x_1w_1+x_2w_2$. The optimal weights were calculated exactly. For this MDP, direct TD(0) and residual gradient TD(0) learned functions that yield the optimal policy: when in state $\{0,0\}$, transition to state $\{1,0\}$. TD(1) learned a function that yields the wrong policy: when in state $\{0,0\}$, transition to state $\{0,1\}$.

In Figures (6) and (7) it can be seen that TD(1) learned a function that is a better approximation of the value function V^* than did either direct TD(0) or residual gradient TD(0), yet TD(1) yields a policy that is inferior to the policies generated when using either direct TD(0) or residual gradient TD(0). Also, in Figure (6) it can be seen that the function learned by TD(0) was affected by the disproportionately large derivative in state $\{99,0\}$, while residual gradient TD(0) equally weighted the temporal difference errors in all states. In Figure (7) it can be seen that the functions provided by direct TD(0) and residual gradient TD(0) are almost identical. In this case, the derivative in state $\{0,99\}$ is 0 for direct TD(0), and therefore had little effect on the approximation. In Figure (7), it is clear that direct TD(0) and residual gradient TD(0) generated functions $\delta V_{TD(0)}$ and $\delta V_{RGTDX(0)}$ that closely approximated δV^* .

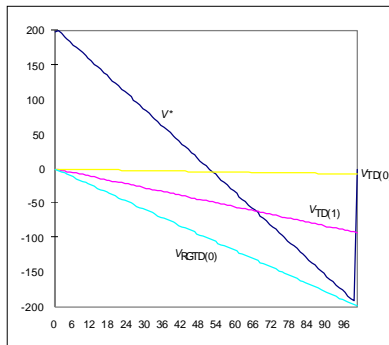


Figure 6: The value function V^* for states $\{n,0\}$ where $n=0\dots100$, and the functions learned by TD(1), direct TD(0), and residual gradient TD(0). TD(1) finds the best 2-norm fit to V^* , but doesn't find the optimal policy.

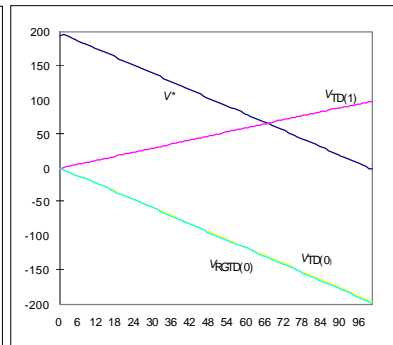


Figure 7: The value function V^* for states $\{0,n\}$ where $n=0\dots100$, and the functions learned by TD(1), direct TD(0), and residual gradient TD(0). TD(1) finds the best 2-norm fit to V^* , but doesn't find the optimal policy.

4 Conclusion

In Bertsekas' example TD(0) appeared worse than TD(1), but that is only the case when considering the 2-norm of the value function error. With respect to the difference in values TD(0) appears better than TD(1). These "differences" represent the degree to which the value function fails to satisfy the Bellman equation. It is not clear which metric should be used. For our example, the salient information associated with a state is not the accuracy of the approximation to the value function, but the accuracy of the approximation of the difference in the values of adjacent states. In other words, the information contained in the difference in the values of adjacent states is most relevant for making control decisions. TD(1) attempts to approximate the absolute value of each state. Both direct and residual gradient TD(0) attempt to find approximations of the optimal value function by learning a function whose differences approximate the differences of the optimal value function while simultaneously approximating the value of the terminal states. However, the function learned by direct TD(0) is also affected by a disproportional weighting of the temporal errors in different states. Direct TD(0) weights each state proportional to the frequency with which it is trained and the magnitude of $\nabla_w V(x)$. TD(1) and residual gradient TD(0) weight each state based solely on the frequency with which it is trained.

Acknowledgment

This research was supported under Task 2312 R1 by the United States Air Force Office of Scientific Research.

References

- Baird, L.C. (1995). Residual Algorithms: Reinforcement Learning with Function Approximation. In Armand Prieditis & Stuart Russell, eds. *Machine Learning: Proceedings of the Twelfth International Conference*, 9-12 July, Morgan Kaufman Publishers, San Francisco, CA.
- Bertsekas, D.P. (1995a). A counterexample to temporal differences learning. *Neural Computation*, **7**, 270-279.
- Bertsekas, D.P. (1995b). *Dynamic Programming And Optimal Control Vol. 1 & 2*. Belmont, Massachusetts: Athena Scientific.
- Harmon, M. E., Baird, L. C., & Klopf, A. H. (1996). Reinforcement learning applied to a differential game. *Adaptive Behavior*, **4**(1), 3-28.
- Sutton, R.S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*. **3**, 9-44.