The Estimation of Reward and Value in Reinforcement Learning

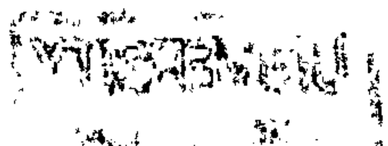
Michael James, BSc, FIMA

Thesis submitted for the degree of Doctor of Philosophy

Department of Computer Science

University of York

January 2003



Abstract

TD(λ) is a sophisticated estimator of value functions encountered as part of reinforcement learning. This research examines the behaviour of TD(λ) as a statistical estimator of the value function. The relationship between the eligibility forms of TD(λ) and the various ways that recurrent states can be used is explored. The reason why some forms of TD(λ) diverge for some choice of α is investigated. A simple condition is presented that ensures that TD-like estimators converge in the mean for all α and this is used to propose a non-divergent form of Accumulating traces TD(λ).

The reason why TD(λ) shows a small sample advantage over simpler estimators is examined. The form of the estimator in the first few steps of estimation is shown to be close to a weighted reward estimator or WR(λ) estimator which does not use any "temporal difference" principles. By comparing the behaviour of the various forms of TD(λ) to the behaviour of the corresponding WR(λ) estimator in a range of models, it is proposed that the action of weighted rewards accounts for TD(λ)'s small sample advantage.

Estimates of the value function are often used to derive the optimum policy. An additional measure of success, the concordance coefficients is used to compare WR(λ), TD(λ) and simpler estimators for their suitability in this role. It is shown that the WR(λ) and TD(λ) estimators behave in a very different way to simpler estimators and this behaviour is largely parameter insensitive. An experiment that compares the estimates of optimal state values reveals that WR(λ) and TD(λ) retain their distinct behaviour. This proves to be an advantage when the initial estimates are optimistic but a disadvantage when they are pessimistic.

Contents

List of Tables

List of Figures

Acknowledgements

I am indebted to my supervisor Professor James Austin whose support, encouragement and many hours of patience have made this work possible and enjoyable.

My grateful thanks are due to Professor Peter Dayan for his many constructive suggestions, in particular for Chapter 12 of this thesis.

I would also like to thank my wife Suzan for devoting many hours to the task of proof reading.

Author's declaration

Unless otherwise stated in the text, the ideas presented in this thesis are solely the work of the author.

Chapter One
Reinforcement Learning

This chapter introduces the basic ideas of reinforcement learning in the most general terms. TD(λ) is then described as the best known technique specific to reinforcement learning and its spectacular success when applied to the problem of playing backgammon is outlined. The question of whether the success of the backgammon playing program is due to TD(λ) or something else is considered. As TD(λ) is such an important method within reinforcement learning its relationship to a broader philosophy of temporal difference learning is considered.

Reinforcement learning

An autonomous system that has to learn to improve its behaviour without the help of a supervisor or teacher has little choice but to attempt to characterise the quality or value of its behaviour in terms of the reward or benefit that accrues to it. Such situations are generally referred to as "reinforcement learning", (Sutton & Barto 1998).

The exact details of the assumed situation vary but the constants are that the autonomous system, or agent, has to evaluate and modify its behaviour using nothing but knowledge of a reward gained at some time in the future – a delayed reward. This can be contrasted with the easier problem of supervised learning, where more detailed information concerning the direction in which to modify the behaviour is supplied. That is, in supervised learning an "error vector" is supplied to the agent which indicates both the distance that the behaviour is from the optimum, or target, behaviour and the direction to "move in" to get closer to that target.

Examples of reinforcement learning are very common in the natural world. In psychology reinforcement learning is supposed to work at a more primitive and basic level than logic or "reasoning" and the behaviourists have argued that all behaviour, even the most sophisticated, is due to conditioning. With regard to a more synthetic environment, AI researchers have invented many artificial tests of reinforcement learning, including pole balancing and playing games such as Backgammon, Chess

and Go. In these artificial situations expert knowledge has often been incorporated into the design of the unsupervised learning mechanisms. For example, Tesauro's Backgammon program (Tesauro, 1992) was given a predefined set of "features" to use to judge the quality of a position. There is nothing inherently different about such hybrid approaches as the final stage of learning involves improving play or behaviour using nothing but the reward eventually obtained.

Value estimation

A key idea in reinforcement learning is that the worth of any given state or action is judged by the reward it eventually produces, (Sutton & Barto, 1998). In many cases it is not enough to select the state or action which maximises the immediate reward because states subsequent to this might well be sub-optimal. In other words, it may be worth accepting a lower immediate reward than possible at a given state in order to reap the benefit of higher rewards later. Thus the task of maximising the eventual reward is likely to be a global, i.e. involving all the possible future states, rather than a local search or the optimisation of the states immediately reachable from the current state.

The estimation problem may well be global but the agent can only use local information such as the nature of the current and immediately accessible subsequent states. This leads to the idea of an "optimal value function" which gives the expected best reward each state produces in the long run. Knowledge of the optimal value function reduces the global search for a maximum reward to a local search. The reason is that, with knowledge of the optimal value function, at each stage the agent simply has to select the action or next state with highest value. That is, the optimal behaviour is simply "greedy" with respect to the optimal value function, i.e. the agent simply selects the state or action that produces the maximum value function at the next stage.

Clearly complete knowledge of the optimal value function makes the reinforcement-learning task trivial but obtaining knowledge of this is far from trivial. As a result part of the theory of reinforcement learning is concerned with finding ways of efficiently estimating value functions. Alternative approaches such as estimating model parameters and direct policy methods can also be applied and these sidestep the problem of direct value function estimation. However any policy, no matter how

derived, can be used to infer a value function and this can be used as a value function estimate.

Backgammon and temporal difference learning

The most successful direct application of reinforcement learning theory to date is Tesauro's backgammon program, which used Temporal Difference (TD) learning to estimate the state value function for the game using a neural network (Tesauro, 1992). TD is a value estimation method introduced by Sutton (Sutton, 1988) and its role in reinforcement learning is the main topic of this study. In the case of backgammon the sequence of board states $x_1, x_2, x_3 \ldots x_f$ results in a final reward z, which is 1 for a win and 0 for a defeat, and the output of a suitably trained neural network is simply $P(x_t)$, the probability of winning from position x_t.

The training method employed, for the neural network, was to start from an initial configuration and to use the network to select moves for both sides. The move selection consisted of presenting the network with all possible legal moves and selecting the one that maximised its output. That is, the next move, x_i is given by:

$$i = \arg\max_{k \in \text{legalmoves}}(P(x_k, w))$$

where $P(x_k,w)$ is the output of the neural network with weight vector w when presented with the board state x_k derivable from the current board state by a legal move. The same procedure is used for the opponent's move but in this case the output of the network is minimised. That is, the neural network was being used to predict the evaluation function for the game.

The weights were adjusted using the TD(λ) method (see Chapter Two for a detailed explanation) employing the outputs from the network on the moves it selected, i.e. taking P as the error signal. As the network was being used to play itself, there was no input of information from a trainer, other than the win or lose signalled at the final move. Tesauro also states that only raw features were used as input and not features constructed to have a relevance to the game.

In several tests based on particular phases of the game of backgammon, it was found that the TD algorithm lagged behind a traditional supervised learning method at

selecting the best moves using the same configuration of network. However, unlike the expert-trained network, its performance continued to increase as the number of hidden units was increased. It was suggested that the TD algorithm is not highly subject to overfitting the data because of the way that new training data is generated during self-play.

A more surprising result is that, when pitted against an algorithm-based backgammon-playing program, the TD algorithm won more games than the expert-trained network. This seems to suggest that differences in performance at predicting the best move may have more to do with a similarity between the test data and the expert-trained network's training data. Tesauro goes on to state that the TD network is as good, or slightly better, than Neurogammon 1, the best backgammon program in the 1989 Computer Olympiad. This is all the more surprising because Neurogammon 1 uses hand-crafted features and the TD network does not.

A later paper (Tesauro, 1994), describes how adding the feature set used by Neurogammon 1 gave the newly named TD-Gammon the ability to greatly surpass Neurogammon 1. Indeed, with the new program installed as a move evaluator within a program that used a heuristic for "doubling" strategy, the resulting TD-Gammon 1 played well enough to rival a strong human opponent. By adding a two-ply look ahead to the supervising program, TD-Gammon 2 improved still further and, by increasing the number of hidden units to 80, TD-Gammon 2.1 is reckoned to play at strong master level and is extremely close to equalling the world's best human players. The performance of the various versions of TD-Gammon is listed in Table 1.

Table 1: Performance of TD-Gammon

	Training Games	Hidden Units	Results
TD 1.0	300,000	80	Lost 13 points in 51 games
TD 2.0	800,000	40	Lost 7 points in 38 games
TD 2.1	1,500,000	80	Lost 1 point in 40 games

Although the performance of reinforcement learning has been good when applied to other games, nothing has matched the outstanding success of TD-Gammon. Games that have been tried include Go (Schraudolph, Dayan & Sejnowski, 1994) and Chess (Thrun, 1995 and Baxter, Tridgell & Weaver, 1998). See Chapter Three for more details.

The problem that TD-Gammon presents is that it is a complex learning system and it is difficult to see which of its features, or which features of the game, were important to its success – neural network design, TD(λ), the particular game feature set, the random play training regime, the stochastic nature of the game and so on.

Learning by co-evolution

The ability to learn by playing a reinforcement learning system against itself, termed "co-evolution", is an opportunity that is unique to reinforcement learning. The lack of need for a supervisor or teacher means that a reinforcement learning system can be pitted against itself at various stages of its history. This sounds like a very good idea but in practice it has proved disappointing. To quote Pollack & Blair, 1997,

"... self-playing learners usually develop eccentric and brittle strategies which allow them to draw each other, yet play poorly against humans and other programs."

It is as if the two systems agree not to test each other by playing more creative strategies.

The best counter-example of this folklore is TD-Gammon but why the method works in this case is difficult to determine. There is evidence that the co-evolution learning of TD-Gammon is more likely to be due to the structure of the game rather than any of the details of the learning algorithm (Pollack & Blair, 1997). Clearly more work is needed to understand what makes co-evolution a successful learning method.

The role of TD(λ) in reinforcement learning

Tesauro's (1994) results with TD(λ) applied to backgammon and, to a lesser extent TD(λ) applied to chess (Baxter et al, 1998) and Go (Schraudolph, Dayan & Sejnowski, 1994), indicate that it is an appropriate learning method to use when taking a reinforcement learning approach to games. However the question of why

Tesauro's backgammon program performs so well is still open. The question remains, "is TD(λ) so good an estimation procedure that it makes a decisive difference to reinforcement learning as suggested by Tesauro's backgammon program", (Tesauro, 1994). To quote again from Sutton and Barto (1998):

> *"If both TD and Monte Carlo methods converge asymptotically to the correct predictions, then a natural next question is "Which gets there first?" In other words which method learns faster? Which makes the more efficient use of limited data? At the current time this is an open question."*

The TD philosophy

The philosophy associated with the TD(λ) estimator essentially says that learning does not have to be based just on the rewards obtained but on any current estimates of the value of the states so far encountered (Sutton, 1988 and Sutton and Barto,1998). The basis for this idea is simply that the current estimate of a state's value represents the expected reward given that the agent finds itself in this state. In this sense the current estimate of the value can act as a surrogate for the eventual "real" reward. Actions that result in the agent moving to states with a high estimated value are as good as actions that eventually result in a final "real" reward – as long as the estimates are good. The advantage of using the current value estimates as surrogates for the reward is that it allows learning to occur as actions are selected by the agent, rather than having to wait until the final reward is received.

To quote from Sutton and Barto (1998):

> *"The next most obvious advantage of TD methods over Monte Carlo method is that they are naturally implemented in an on–line, fully incremental fashion. With Monte Carlo methods one must wait until the end of an episode, because only then is the return known, whereas with TD methods one need wait only one time step."*

Thus although TD(λ) is a specific estimator of the value function it can also be regarded as just one possible application of the more general principle of temporal difference learning. It is derived from a consideration of the idea that existing estimates of the value of subsequent states should be used to update the estimate of the value of the current state.

The philosophy of temporal difference learning is more subtle than this simple explanation might suggest. The principle of temporal difference learning is based on the observation that the dynamics of a wide class of reinforcement models possess a special property. In any realisation of the model the expected value of the value function at the next step is constant as the agent moves from state to state and it is equal to the expected final reward. In other words, a large class of reinforcement learning models have the Martingale property, where the expected value of the quantity of interest at this step is the same as the expected value at the next step (Chung, 1974). A Martingale is the probabilistic process which best characterises a range of gambling situations that are judged "fair". For example, consider a situation where a gambler starts with £M and the game being played is fair with a 0.5 probability of winning or losing £x. In this case the gambler's expected wealth at the next turn is $[= 0.5(M + x) + 0.5(M - x)]$, that is, it remains at £M. This argument, by extension, also shows that the expected wealth after any number of games is also £M.

The same reasoning applies to the value function during reinforcement learning. In this case the expected value at the next step is equal to the current value by its definition rather than by any imposition of fairness. The value function at a given state can be expressed as the expected value at the next state by the Bellman equations (see Chapter Four).

In general terms the Bellman equations say:

Value of state s_i= Expected value of the state that occurs immediately after.

That is, the value function has the Martingale property because by definition it satisfies the Bellman equation.

The principle of temporal difference learning takes this simple observation and finds ways of constructing estimates using it. For example, Sutton and Barto (1998) in their "Driving Home" example suggest,

> *"Each day as you drive home from work, you try to predict how long it will take to get home..."*

There then follows a description of estimating the total time of the journey given the times to various points on the journey. In this case the intermediate points along the

journey are the states and the times are the intermediate rewards. The expected total time to home from any point on the way is its value function. The example continues:

> *"Is it necessary to wait until the final outcome is known before learning can begin? Suppose on another day you again estimate when leaving your office that it will take 30 minutes to drive home, but then you become stuck in a massive traffic jam. Twenty-five minutes after leaving the office you are still bumper-to-bumper on the highway. You now estimate that it will take another 25 minutes to get home, for a total of 50 minutes. As you wait in traffic, you already know that you initial estimate of 30 minutes was too optimistic. Must you wait until you get home before increasing your estimate for the initial state? According to the Monte Carlo approach you must because you don't yet know the true return.*
>
> *According to a TD approach, on the other hand, you would learn immediately, shifting your initial estimate from 30 minutes toward 50. In fact, each estimate would be shifted toward the estimate that immediately follows it... Each error is proportional to the change over time of the prediction, that is to the* ***temporal difference*** *in prediction."*

It is important, in this example, not to confuse the gain in the prediction accuracy for this realisation against any long-term gain in accuracy that the adjustment may produce. It is quite obvious that after sitting in a traffic jam for 25 minutes you should increase your estimate for this realisation to 50 minutes and this is a correction that is not only in the direction of the temporal difference but is most sensibly taken to be the whole of the temporal difference. This makes it seem obvious that temporal differences are a sensible method of correcting estimates but it says nothing about the accuracy of prediction on the next journey! Clearly if your new estimate to get home from the office is 50 minutes and there is no traffic jam next time you make the journey your estimate is worse, not better. It all hinges on the probability of the traffic jam. If this was a common event your improved estimate will be better in the long run, but if a traffic jam is a low probability your estimate is made worse by the correction.

The real key to temporal difference estimation is not an appeal to the improvement in the estimate in the current realisation, but that the corrections are made often enough and with a frequency proportional to the probability that the event occurs to reflect

the long term behaviour of the system. That is, in the case of the time to get home estimator, the estimate is raised every time a traffic jam occurs and lowered every time a traffic jam does not occur in just the right proportion to allow the estimate to converge (in some sense) to the average time to drive home. So the example is correct and appropriate but only when the role of the probability of a traffic jam, or some other similar event, is clearly understood in making the estimate more accurate in the long run and not just in the single realisation.

Early learning?

There is another issue that reflects on the philosophy of TD learning. Namely that TD(λ) is repeatedly shown to be a better estimator early in the stages of data acquisition, (Sutton and Barto, 1998). In the case of the driving home estimator it is stated that the TD approach should be preferred because it allows learning to start sooner than a simple "wait until finished" approach.

The Bellman equation can again be used to relate the state's value function to the next state's expected value function. This implies that estimates already obtained can be used to improve the current estimate, so speeding up the learning process by "bootstrapping". To quote Sutton and Barto (1998),

> *"TD methods learn their estimates in part on the basis of other estimates. They learn a guess from a guess – they bootstrap"*

and

> *"TD methods have usually been found to converge faster than constant α MC methods on stochastic tasks..."*

Indeed it is possible to show that TD(λ) estimators do work better than alternative estimators for small sample sizes. The key result here being the Markov chain random walk experiment used by Sutton (1988), Sutton and Barto (1998), Singh and Dayan (1998) and many others to show that TD(λ) does indeed outperform alternative estimators. It is indisputable that TD(λ) is a better estimator than simple Monte Carlo estimators for some models for small sample sizes, but this doesn't mean that the TD philosophy is the reason for this superiority.

It is clear from the Bellman equations that TD estimators could make use of existing information more efficiently than simple estimators. For example, if a state with an unknown value is "connected" to a set of states with value function estimates that are accurate then TD learning can be applied to estimate the unknown value without any need to use the final reward statistics and without needed to obtain complete realisations or the process. All that is needed is to start the system off from the state and observe the average reward at the first transition to another state and use this as the estimate of the state's value. However if the "connected" states values are different from the real values then this procedure fails. Of course a complete TD estimator would allow the value estimates at the "connected" states to be updated in the same way and hence everything converges to the true value function.

It seems more reasonable that the TD philosophy is likely to create estimators that are at their worst for small samples, when the early estimates are likely to be further from their true values, than later in the procedure. What is more, on examination of various models and TD estimators, see later chapters, much of the small sample advantage occurs on the very first step of estimation when the initial estimates have no information about the expected rewards. Whatever it is that gives these estimators their first step advantage it is difficult to see how it can be Temporal Difference learning.

Clearly the superiority of TD(λ) in small samples deserves further investigation.

Outline of thesis

The purpose of this study is to evaluate the TD(λ) estimator against an appropriate range of alternatives and find out what aspects of it are important to its functioning. The following brief outline is given to help the reader see how the argument develops.

The detailed model used in studying reinforcement learning is presented in Chapter Two - the Markov Decision Process (MDP) along with the Bellman equations that govern its value function. The three standard approaches to value function estimation are then presented – Monte Carlo, Dynamic Programming and Temporal Difference method. The detailed form of the TD(λ) estimator is introduced along with a discussion of some of the motivation for its construction. This is then specialised to

the case of no discounting and only terminal rewards. The eligibility trace formulation of the TD(λ) estimator is also introduced.

Following on from the brief histories of TD(λ) and its connection to the wider reinforcement learning context, Chapter Three focuses specifically on the work done to explore and evaluate the TD(λ) estimator and on how it has been used.

By selecting a policy the problem of estimating the value function reduces to that of estimating functions of state of a Markov chain. In Chapter Four the classical analysis of absorbing Markov chains is presented along with an analysis of functions of state which obey generalised Bellman conditions. The analysis provides tools that make computing some expected values much easier.

Chapter Five presents an initial examination of the behaviour of the TD estimator as compared to the simple Monte Carlo Alpha (MCA) update rule. The reason for the divergence of the Accumulate or Every Visit Off-line form of TD(λ) for some values of α is explored and a modification that removes the divergence is suggested and explored. The various forms of the eligibility trace TD(λ) estimator are related to first, every and last visit estimators and the reasons for the divergence of Replacing Traces and Accumulate traces is investigated.

Following the preliminary investigation of the way in which recurrent states are used in estimation, Chapter Six looks in detail at the first and every visit forms of the MCA estimator and their performance.

In Chapter Seven the focus shifts from the effects of different ways of using recurrent states to the central question of why TD(λ) delivers a better small sample performance than the MCA estimator. An approximation to the TD(λ) estimator, WR(λ) is derived and explored in the context of the standard 19-state Simple Random Walk (SRW) Model and a First Visit estimator. As WR(λ) contains no temporal difference components its behaviour illuminates why TD(λ) is better in small samples.

Chapter Eight extends the exploration of the TD(λ) and WR(λ) first visit estimators to additional models designed to reveal any differences between the two that might be due to temporal difference principles.

Chapter Nine then extends the exploration of the TD(λ) and WR(λ) estimators to Accumulating and Replacing traces forms using the models described in the two preceding chapters. and the potential for using WR(λ) as an estimator in its own right, rather than just as an approximation to TD(λ) is considered in Chapter Ten

Chapter Eleven considers whether or not the moderate parameter-sensitive gains in RMS error TD(λ) offers really make any difference to reinforcement learning tasks. The "concordance" coefficient is introduced and used to determine if TD(λ) or WR(λ) offer any substantial advantages over alternative estimators when the estimated value functions are used to derive new policies. The performance of value functions estimated using TD(λ) and MCA are then compared under conditions of different exploration/exploitation.

The final chapter gathers together the results and conclusions of the entire thesis and puts them in context. It discusses the relevance of temporal difference learning as a philosophy and the importance of estimators like TD(λ) or WR(λ).

Chapter Two

Markov Decision Processes and Estimation

After the general and non-mathematical consideration of reinforcement learning and temporal difference methods in Chapter One, this chapter sets out to make precise the ideas by introducing the most common mathematical model of reinforcement learning, a Markov Decision Process (MDP). This leads on to the Bellman equation of dynamic programming which the value function satisfies. The chapter then moves on to consider the three methods in common use for estimating the value function – Monte Carlo, Dynamic Programming and Temporal Difference. The detailed form of the TD(λ) estimator is introduced, along with a discussion of some of the motivation for its construction. The most general form of TD(λ) is very complicated and to see how it works a simpler form, which results when there is no discounting and only terminal rewards, is introduced. Finally an alternative formulation based on eligibility traces, which is much more suitable for implementation, is described.

MDPs - Markov Decision Processes

The most common probability model applied to reinforcement learning is the Markov Decision Process – MDP (Sutton & Barto, 1998; Sutton, 1997).

An MDP consists of:

- a set of states, S
- a set of actions, A
- a reward function $R:S \times A \times S' \rightarrow R$ where S' represents the set of states subsequent to s.
- a state transition function $T: S \times A \rightarrow \Pi(S)$ where $\Pi(S)$ is a probability distribution over S

For the most general model the reward function is assumed to depend on the current state s, the action a selected by the policy and the subsequent state, s'. That is, the reward is a function R(s,a,s'). By including a in the reward function, we can also

model the fact that actions might have different costs and so the reward might depend on how the state s' was reached.

The state transition function, T(s,a) gives a probability distribution which in turn give the probability of each state s' being the result of s,a, i.e. Π(s') is the probability of s' being the next state after action a in state s. For simplicity it is usual to write the function T as T(s,a,s') which gives the probability of state s' following action a in state s. The model is Markovian, and hence an MDP, if T(s,a,s') is independent of any previous states or actions other than the current state and action, (Bellman, 1957; Bertsekas, 1987; Howard, 1960; Puterman, 1994).

We can associate a policy π with the model which deterministically selects actions at each stage of the development. If π is a function only of s, i.e. if it is a deterministic Markovian policy, then π induces a Markov process on the MDP. That is, if the policy is followed the MDP makes state transitions given by T(s,π(s),s') which is clearly just a function of s and s' and can be written T^{π}(s,s') and

$$T^{\pi}(s,s') = T(s,\pi(s),s')$$

Similarly if π(s,a) is a Markovian stochastic policy, i.e. the probability of choosing action a depends only on state s, then the state transition probabilities are given by

$$T^{\pi}(s,s') = \sum_{a} \pi(s,a)T(s,a,s')$$

For a discussion of the generality of Markovian models, see Sutton & Barto, 1998.

The value of a state and an action

A key idea in reinforcement learning is the optimal value of a state. If a policy π is followed subsequent to state s being reached then the value of the state V^{π}(s) can be defined as the expected reward under the policy. In most cases the expected discounted reward is used to define the value of a state:

$$V^{\pi}(s) = \underset{\pi}{E}[\sum_{t=0}^{\infty} \gamma^{t} r_{t}]$$

The expected value operator clearly depends on the policy being followed.

The justification for using a discounted reward is that it values a reward received now more than one of equal magnitude received some time in the future. But it also neatly solves the problem of computing the total reward in models that terminate and in models that do not terminate in the same way. That is, we don't care if the task terminates or continues indefinitely - the discounted reward can be computed in the same way. In the case of a terminal state s_t encountered at time t it is clear that V(s) is zero as no further rewards are received and r_{t+k}, k>0, is zero. Setting γ to 1 produces an undiscounted total reward so there is no restriction entailed in working with the discounted reward as a theoretical device to define the value function.

If the policy is known to be an optimal policy π^* then the optimal value, $V^*(s)$ is the value of the state following the optimal policy. That is:

$$V^*(s) = \max_{\pi} E[\sum_{t=0}^{\infty} \gamma^t r_t]$$

and V^* can be used to define π^* as that policy which maximises $V^{\pi}(s)$.

If the optimal value function is known then the optimal policy can be derived from it by simply selecting the action which gives the largest expected value of V^* at the next state. That is, the optimal policy is "greedy" with respect to the optimal value function.

In the same way as a state can be assigned a value, so can an action in a given state. The value of an action $Q^{\pi}(s,a)$ is the expected discounted reward obtained by taking action a in state s and then following policy π thereafter. Notice that the action a may not be the one specified by the policy when in state s. This can be thought of as forcing the action onto the policy and then discovering the expected reward.

$$Q^{\pi}(s,a) = \underset{\pi}{E}[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a]$$

In the same way $Q^*(s,a)$ can be defined to be the optimal value of the action under the optimal policy π^*.

As before, action a may not be the action selected by the optimal policy and thus the overall policy followed may not be optimal. It is only optimal after the selection of a in s. By definition:

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a)$$

That is, the optimal value of an action is its largest value under any policy selected after taking action a.

If the optimum action value function is known then it is clear that the optimum state value function can be obtained from it using:

$$V^*(s) = \max_{a} Q^*(s,a)$$

A range of methods is available that enable an initial policy to be improved using an estimate of its value function to compute the optimal policy and optimal value function. These include Q learning (Watkins, 1989; Watkins & Dayan,1992), Policy Iteration (Bellman, 1957), and Value Iteration (Puterman & Shin, 1978).

Value functions for MDPs – the Bellman equations

The state and action value functions can be expressed in terms of the parameters of an MDP and in these forms alternative methods of estimation and iterative solution become possible. Essentially the probability structure of the MDP allows us to evaluate the expectation operators in terms of the transition and reward functions.

That is, for an MDP the value of a state using the recursive form is:

$$\begin{aligned} V^{\pi}(s) &= \underset{\pi}{E}[R_t | s_t = s] \\ &= \underset{\pi}{E}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s] \\ &= \sum_{k=0}^{\infty} \gamma^k \underset{\pi}{E}[r_{t+k+1} | s_t = s] \\ &= \underset{\pi}{E}[r_{t+1} | s_t = s] + \sum_{k=1}^{\infty} \gamma^k \underset{\pi}{E}[r_{t+k+1} | s_t = s] \end{aligned}$$

The first term is just the expected reward at the next step and this can be computed using the one-step transition probabilities:

$$\underset{\pi}{E}[r_{t+1}|s_t = s] = \sum_a \pi(s,a)\sum_{s'} T(s,a,s')R(s,a,s')$$

where $\pi(s,a)$ is the probability of selecting action a while in state s, T(s,a,s') is the state transition probability function and R(s,a,s') is the reward function.

The second term is a little more difficult to evaluate but by simple change of the lower summation limit we obtain:

$$\sum_{k=1}^{\infty}\gamma^k E_\pi[r_{t+k+1}|s_t = s] = \gamma\sum_{k=0}^{\infty}\gamma^k E_\pi[r_{t+k+2}|s_t = s]$$
$$= \gamma E_\pi[R_{t+1}|s_t = s]$$

The expectation of the reward at time t+1 given the state at time t can also be evaluated using the one-step transition probabilities:

$$\underset{\pi}{E}[r_{t+1}|s_t = s] = \sum_a \pi(s,a)\sum_{s'} T(s,a,s')E[R_{t+1}\,|\,s_{t+1} = s']$$

but by definition:

$E\{R_{t+1}|s_{t+1} = s'\} = V^\pi(s')$ Putting both terms back together gives:

$$V^\pi(s) = \sum_a \pi(s,a)\sum_{s'} T(s,a,s')R(s,a,s') + \gamma\sum_a \pi(s,a)\sum_{s'} T(s,a,s')V^\pi(s')]$$
$$= \sum_a \pi(s,a)\sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^\pi(s')]$$

The sums are taken over all actions permissible in state s at time t and over all possible successor states s′ at time t+1. This equation is also intuitively reasonable for a "value" function. It says that the value of a state is the expected reward received in moving to the next state plus the expected value of the successor state discounted by γ.

This system of equations relating $V^\pi(s)$ to the values of $V^\pi(s')$ on the successor states is called the Bellman equation and it is the fundamental equation of Dynamic Programming, (Bellman, 1957).

Estimating the value function

Many methods of finding the optimum policy involve the use of an estimate of the value function corresponding to a current policy. Clearly the Bellman equations can be solved either recursively or exactly to give V^{π}. In a sense this is a complete solution to the problem of finding the value function but only if the parameters of the MDP are known exactly. In practice we have to use statistical techniques to estimate V^{π}.

Sutton and Barto (Sutton & Barto, 1998) propose three distinct methods of obtaining approximations to V^{π}:

- Monte Carlo methods
- Dynamic Programming
- Temporal Difference methods

There is a sense in which TD methods are also Dynamic Programming methods but it is customary to distinguish them.

Monte Carlo methods

Monte Carlo methods correspond to classical statistical estimation of $V^{\pi}(s)$ or $Q^{\pi}(s,a)$ as data is accumulated by observing examples of the behaviour. In this case $V^{\pi}(s)$ is estimated as the average return obtained over many samples or realisations of the behaviour. Similarly $Q^{\pi}(s,a)$ is estimated as the average return obtained as the result of taking action a in state s.

The estimation procedure is to allow the system to run or to simulate the system and record the states visited and the rewards received. After each trial or realisation we can form a new estimate of the value of each state visited in the realisation. The estimate of V(s) is updated using the discounted reward actually obtained when the sequence terminates at time $\tau(t)$, after t realisations i.e.:

$$R_t = r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots + \gamma^{\tau(t)-1} r_{\tau(t)}$$

and then forming the new sample average:

$$V_t(s_i) = \sum_{j=1}^{t} \frac{R_j}{\kappa_j(t)}$$

where $\kappa_i(t)$ is the number of times the state s_i has been visited in the t realisations.

The calculation of the sample mean can be re-written as in iterative update involving the old estimate of the mean and the new data:

$$V_t(s_i) = \frac{V_{t-1}(s_i)(\kappa_i(t)-1) + R_t}{\kappa_i(t)}$$

or

$$V_t(s_i) = V_{t-1}(s_i)(1 - \frac{1}{\kappa_i(t)}) + R_t \frac{1}{\kappa_i(t)}$$

Writing $\alpha_t = \frac{1}{\kappa_i(t)}$ gives the update rule in the form

$$V_t(s_i) = V_{t-1}(s_i)(1-\alpha_t) + R_t\alpha_t$$

This is also the form of the Robbins Munro or Stochastic Iterative algorithm for estimating the mean (Bertsekas & Tsitsiklis, 1996). In this case the iteration is:

$$V_t(s_i) = V_{t-1}(s_i)(1-\alpha_t) + R_t\alpha_t$$

where α_t is not now no longer taken to be $\frac{1}{\kappa_i(t)}$ but is a general function of t.

The iteration can be shown (Bertsekas & Tsitsiklis, 1996) to converge to the expected value of R if:

$$\lim_{t\to\infty} \alpha_t = 0 \quad \sum_{t=1}^{\infty} \alpha_t = \infty \text{ and } \sum_{t=1}^{\infty} \alpha_t^2 < \infty .$$

These are all reasonable conditions. The first states that α gets smaller as the amount of data increases, so reducing the variance of the estimator; the second makes sure that α has enough variation for the estimate to allow whatever initial bias there may be in V_0 to be removed in a finite number of samples; and the final condition states that this variation isn't too great to allow the estimate to settle down. Clearly all three conditions are satisfied by the harmonic series 1/N and hence the sample mean converges to the population mean.

In many applications the update rule is written:

$$V_t(s_i) = V_{t-1}(s_i)(1-\alpha) + R_t\alpha$$

where the dependence of α on the estimation step is taken for granted. This is sometimes referred to as the "alpha update rule" or the Monte Carlo Alpha (MCA) estimator and α is referred to as the "step size". In many cases the step size is set heuristically to allow the estimator to track a changing population parameter.

The alpha update rule can also be written:

$$\begin{aligned} V_t(s_i) &= V_{t-1}(s_i)(1-\alpha) + R_t\alpha \\ &= V_{t-1}(s_i) - \alpha(V_{t-1}(s_i) - R_t) \\ &= V_{t-1}(s_i) - \alpha\Delta_t \end{aligned}$$

where Δ_t is the "error" between the current estimate and the observed value.

This form of the estimator makes it less clear that the estimator is derived from a generalisation of the standard sample average. It makes it look more like a gradient descent error minimisation process where the current estimate is adjusted in the direction that minimises the error on the current sample.

So now all we have to do is observe the process, record the rewards, compute the discounted reward and use the alpha update rule to obtain the new estimate of the value function. There is also a small complication caused by the possibility of more than one visit to state s before the process terminates. One approach is to ignore additional visits and compute an estimate based only on a first visit. Another approach is to treat each occurrence of the state as if it was the start of a new realisation and compute an every visit update. The issue of how to deal with First and Every Visit estimation is considered in detail in Chapters Six and Seven.

The alpha update rule has been extensively analysed (Wetherill, 1975), (Bertsekas & Tsitsiklis, 1996). As well as general convergence theorems it is possible to derive analytic expressions for the variance and bias of the alpha update rule, see Chapter Seven and Singh and Dayan (1998).

Dynamic Programming - Certainty Equivalence methods

An alternative Monte Carlo/Dynamic Programming approach in the case of an MDP is to estimate the parameters of the MDP model and then solve the Bellman equations for $V^{\pi}(s)$ or $Q^{\pi}(s,a)$. If a fixed policy π is being followed on an MDP then the transitions between states become a Markov process with fixed transition probabilities – the induced Markov process. The basic approach of the certainty equivalent estimator (Sutton & Barto, 1998) can be summarised as, "estimate the parameters of the model from the data and then solve the model for the quantities of interest as if the estimates where the population values". In the case of an MDP this reduces to "estimate the transition probabilities in the induced Markov chain that results from following policy π and then use the Bellman equations to find the value function". Of course if the transition probabilities are accurate the value function derived from them will also be exact. As the sample size increases the estimated parameters converge to the population parameters and the solution to the estimated model becomes the solution to the true model.

If we are simply interested in estimating the value function then we have to solve the Bellman equations using estimated quantities:

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^{\pi}(s')]$$

These can be simplified by combining the effect of $\pi(s,a)$ and $T(s,a,s')$ into a single function $P(s,s')$ which gives the probability of transition from state s to state s', i.e. $P(s,s')$ is the Markov transition matrix induced on the MDP by following the policy π. By definition $P(s,s') = \sum_{a} \pi(s,a) \sum_{s'} T(s,a,s')$.

Changing the order of summation in the Bellman equations gives:

$$\begin{aligned} V^{\pi}(s) &= \sum_{a} \pi(s,a) \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^{\pi}(s')] \\ &= \sum_{s'} \sum_{a} \pi(s,a) T(s,a,s')[R(s,a,s') + \gamma V^{\pi}(s')] \\ &= \sum_{s'} \sum_{a} \pi(s,a) T(s,a,s') R(s,a,s') + \sum_{s'} \sum_{a} \pi(s,a) T(s,a,s') \gamma V^{\pi}(s') \\ &= \sum_{s'} \sum_{a} \pi(s,a) T(s,a,s') R(s,a,s') + \gamma \sum_{s'} P(s,s') V^{\pi}(s') \end{aligned}$$

The first term:

$$\sum_{s'}\sum_{a}\pi(s,a)T(s,a,s')R(s,a,s') = R(s)$$

is clearly the expected value of the reward obtained in state s while following policy π. Putting all this together gives a much simpler form for the Bellman equations:

$$V^{\pi}(s) = R(s) + \gamma\sum_{s'}P(s,s')V^{\pi}(s')$$

It is clear that R(s) can be estimated simply by recording the average immediate reward received by each state s and P(s,s') can be estimated by recording the number of times the transition s to s' occurs.

When these estimates are available the simplified Bellman equation can be solved by any of a number of methods but most generally by the iteration:

$$V_{k+1}(s) = \overline{R(s)} + \gamma\frac{1}{N}\sum_{s'}f_{ss'}V_k(s')$$

where $\overline{R(s)}$ is the sample estimate of the immediate reward for state s; $f_{ss'}$ is the number of times the transition s to s' occurred; and N is the total number of transitions in the realisation. The iteration is performed until the estimates converge to the required degree of accuracy.

It is easy to see that CE estimator is the Maximum Likelihood (ML) estimator. It is the ML estimator because if $L(\theta)$, the likelihood function is maximised at θ_m and ϕ is a function of θ, i.e. $\phi=f(\theta)$, then $L(\phi)$ is maximised by $\phi_m=f(\theta_m)$. Thus the Maximum Likelihood estimator of a function of a model parameter is the same function of the maximum likelihood estimator of the model parameter, i.e. the CE estimator.

Of course being the ML estimator doesn't guarantee optimal properties in any small sample, only the asymptotic unbiasedness and efficiency are guaranteed (Cox & Hinkley, 1974). In addition there is the problem of computing the CE estimator. If there are n(S) states the CE estimator needs at least $2^{n(S)}$ estimates and this rules it out as a practical proposition.

Temporal Difference methods

Temporal Difference (TD) methods can be understood as a modification to Monte Carlo methods or to Dynamic Programming methods. They make use of predictions at time t+1, t+2 and so on to update the prediction at time t. Monte Carlo methods wait until time T, the termination of the sequence, before updating the estimate made at time t and Dynamic Programming makes use of data from all terminal states and complete knowledge of the transition probabilities before updating.

TD(λ) in its most general form is a complicated estimator and there are many different ways of looking at it, explaining it, and justifying it. It can even be written in different ways so as to emphasise these different aspects. The following account starts from the most general form of the TD estimator and works towards a simpler, more specialised form, that in fact proves to be sufficient for the situations considered in later chapters.

The standard form of TD(λ)

As described earlier in the Monte Carlo method the estimate of V(s) is updated using the rewards actually obtained when the sequence terminates at time $T = \tau(t)$, i.e., supposing that state s_i occurs at position m in realisation t:

$$R_m = r_{m+1} + \gamma r_{m+2} + \gamma^2 r_{m+3} + \ldots + \gamma^{\tau(t)-1} r_T$$

(suppressing for the moment the obvious dependencies on t) and the updated estimate is:

$$V_{t+1}(s_i) = V_t(s_i)(1-\alpha) + \alpha R_m$$

where $0<\alpha<1$.

Rewriting the Monte Carlo estimate as:

$$\begin{aligned} V_{t+1}(s_i) &= V_t(s_i) + \alpha[R_m - V_t(s_i)] \\ &= V_t(s_i) + \alpha\Delta_{\tau(t)} \end{aligned}$$

reveals the fact that the update to V(s) can be viewed as proportional to the difference between the predicted value of the state $V_k(s_t)$ and the discounted reward actually obtained, i.e. the value of R_m.

That is, it is the difference between prediction at time m when the state s_i occurs in the realisation and the reality at time $T = \tau(t)$ when all the rewards are known, i.e. a temporal difference. This suggests extending the use of differences between prediction and "reality" to times other than $\tau(t)$. Of course the only "reality" available to us at time m, with $m<\tau(t)$, is the set of rewards obtained so far and the estimate of the value of the state that we have reached, i.e. $V_t(s_m)$ where s_m is the state that occurs at position m in the realisation.

The general TD method (Sutton & Barto, 1998) uses, in place of the final reward at time $\tau(t)$:

$$R_m^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}[\sum_{j=1}^{n-1}\gamma^{j-1}r_{m+j} + \gamma^n V_t(s_{m+n})]$$

where s_{m+n} is the state that occurs at m+n in the realisation and r_{m+j} is the reward obtained at m+j.

The inner bracket is called the "corrected n-step truncated return" or just the "n-step return":

$$R_m^{(n)} = \sum_{j=1}^{n-1}\gamma^{j-1}r_{m+j} + \gamma^n V_t(s_{m+n})$$

This is just the discounted return calculated using $V_t(s)$ as an estimate of the returns when real rewards have not yet occurred, i.e. at more than or equal to n time periods in the future.

That is, the rewards for state s at time m+1, m+2, .. to m+n-1 are assumed known but $V_t(s_{m+n})$ is used as an estimate of the total reward thereafter. This is very reasonable as $V_t(s_{m+n})$ is supposed to be an estimate of exactly this – the expected total reward starting from state s_{m+n}. Clearly how useful it is to substitute an existing estimate in place of the actual returns depends on how accurate and stable the estimate of $V_t(s_{m+n})$ is.

For example, for n=1 we have:

$$R_m^{(1)} = r_{m+1} + \gamma V_t(s_{m+1})$$

i.e. the actual reward at time m+1 and the estimate of the expected reward thereafter.

For n=2

$$R_m^{(2)} = r_{m+1} + \gamma r_{m+2} + \gamma^2 V_t(s_{m+2}) \quad \text{and so on.}$$

Any of the $R^{(n)}$ can in principle be used in place of R_t in the Monte Carlo estimator. As the expected value of $\gamma^n V_t(s_{m+n})$ is simply the rest of the discounted reward series it is clear that the expected value of $R_m^{(n)}$ is also the expected reward in states s, i.e. $V(s_i)$, which in turn is just the expected value of the discounted reward.

To be precise we have:

$$E[R^{(n)}] = V(s_{m+n}) = E[R]$$

The question is which of the $R_m^{(n)}$ should be used in an update rule for the current estimate of $V(s_i)$? The quality of any such modified estimator would depend on the value of n chosen and the quality of the $V_t(s_{m+n})$ as estimators of the expected reward.

Rather than select a particular value of n, the value R^{λ}_m used in the TD(λ) method, is a weighted average of all of the n-step returns (Sutton & Barto, 1998):

$$R_m^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} R_m^{(n)}$$

Clearly this quantity, being a weighted average of the $R_m^{(n)}$, has the same expectation value as they do, i.e. the discounted reward, and so is suitable as a choice for the update rule for $V(s_i)$.

The TD(λ) estimator is thus:

$$V_{t+1}(s_i) = V_t(s_i)(1-\alpha) + \alpha R^{\lambda}_m$$

As λ varies from 0 to 1 the weight put on $R^{(n)}$ in the computation of R_m^{λ} varies. When λ is small values of n close to 1 are weighted more strongly and as λ increases the weight is more evenly distributed across values of n.

Temporal difference form of TD(λ)

The TD(λ) estimate can be written as a difference between the prediction and the "true" value:

$$\begin{aligned} V_{t+1}(s_i) &= V_t(s_i)(1-\alpha)+\alpha R^{\lambda}_{m} \\ &= V_t(s_i)+\alpha[R^{\lambda}_{m}-V_t(s_i)] \\ &= V_t(s_i)+\alpha\Delta^{\lambda} \end{aligned}$$

In this form there is no clue as to why the method is called "temporal difference" because it appears to be just an average of estimates of reward at times m+1, m+2 and so on. The name arises from an alternative view of the same estimate. The general TD(λ) method can be re-written, using simple algebra, to display that it is a temporal difference method; see Appendix 1, The Difference Form of TD(λ).

When expressed in this form the learning increment can be divided into two distinct parts:

$$\Delta^{\lambda} = \Delta^{\lambda}_{r} + \Delta^{\lambda}_{V}$$

The first part of this expression:

$$\Delta^{\lambda}_{r} = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}[\sum_{j=1}^{n-1}\gamma^{j-1}r_{m+j}]$$

is simply a weighted average of the discounted actual rewards at each step and is not of any great interest in terms of temporal difference learning.

The second term is:

$$\begin{aligned} \Delta^{\lambda}_{V} &= \sum_{n=1}^{\infty}\beta^{n-1}[\gamma V_t(s_{m+n})-V_t(s_{m+n-1})] \\ &= \sum_{n=0}^{\infty}\beta^{n}\nabla^{\gamma}V_t(s_{m+n}) \end{aligned}$$

where:

$$\nabla^{\gamma}V_t(s_{m+n}) = \gamma V_t(s_{m+n+1})-V_t(s_{m+n})$$

is the discounted first difference operator and $\beta=\lambda\gamma$

It is now clear that the new estimate of the value of a state can be formed using just the difference in estimates at each step and is indeed a "temporal difference" estimate.

A simple form of TD(λ)

A simpler and very common situation – no discounting and only terminal rewards - provides a clearer expression of the interaction between existing estimates of V(s) and actual data, i.e. the terminal reward in the TD(λ) estimate.

If no discounting is used the discounted difference operator reverts to the standard difference operator and if the only reward is the terminal reward R_T received at time T-1 (i.e. on transition to the terminal state) then the first part of the TD increment, i.e. that depending on the received rewards becomes:

$$\Delta_r^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}[\sum_{j=1}^{n-1}\gamma^{j-1}r_{m+j}]$$
$$= \lambda^{T-m-1}R_T$$

Because $\gamma=1$, and $V(s_{m+n})=0$ for $m+n\geq T$, the second term becomes:

$$\Delta_V^\lambda = \sum_{n=1}^{\infty}\lambda^{n-1}[V_t(s_{m+n})-V_t(s_{m+n-1})]$$
$$= \sum_{n=1}^{T-t}\lambda^{n-1}[V_t(s_{m+n})-V_t(s_{m+n-1})]$$

If the two terms are now put together again:

$$\Delta^\lambda = \Delta_r^\lambda + \Delta_V^\lambda$$
$$= \sum_{n=1}^{T-m}\lambda^{n-1}[V_t(s_{m+n})-V_t(s_{m+n-1})]+\lambda^{T-m-1}R_T$$
$$= \sum_{n=1}^{T-m-1}\lambda^{n-1}[V_t(s_{m+n})-V_t(s_{m+n-1})]+\lambda^{T-m-1}R_T-\lambda^{T-m-1}V_t(s_{m+n-1})$$
$$= \sum_{n=1}^{T-m-1}\lambda^{n-1}[V_t(s_{m+n})-V_t(s_{m+n-1})]+\lambda^{T-m-1}[R_T-V_t(s_{m+n-1})]$$

the last term in this sum looks just like rest of the series, i.e. it is a weighted difference but in this case a difference between R_T, the actual reward received, and its estimate. All of the other terms in the series are differences between estimates of the future reward.

By writing $R_T=V_t(s_T)$ we can make the series look even more uniform:

$$\Delta^{\lambda} = \sum_{n=1}^{T-m-1}\lambda^{n-1}[V_t(s_{m+n}) - V_t(s_{m+n-1})] + \lambda^{T-m-1}[R_T - V_t(s_{t+m-1})]$$
$$= \sum_{n=1}^{T-m}\lambda^{n-1}[V_t(s_{m+n}) - V_t(s_{m+n-1})]$$

This clearly demonstrates that the TD philosophy is to always correct the current estimate by an increment proportional to the weighted sum of the difference between the current estimates separated by one time interval. This is the original form of TD(λ) introduced by Sutton (Sutton, 1988).

It also serves to demonstrate more clearly what happens in the limit when λ=0 or λ=1. In the case that λ=0 we have for TD(0):

$$\Delta^{\lambda} = \sum_{n=1}^{T-m}\lambda^{n-1}[V_t(s_{m+n}) - V_t(s_{m+n-1})]$$
$$= V_t(s_{m+1}) - V_t(s_m)$$

and the increment is just the difference between the value estimates of the states at m and m+1 in the realisation.

If λ=1 then we have for TD(1):

$$\Delta^{\lambda} = \sum_{n=1}^{T-m}\lambda^{n-1}[V_t(s_{m+n}) - V_t(s_{m+n-1})]$$
$$= \sum_{n=1}^{T-m}[V_t(s_{m+n}) - V_t(s_{m+n-1})]$$
$$= V_t(s_T) - V_t(s_m)$$

and the increment is just the difference between the actual reward and the current estimate of the state's value, i.e. the standard alpha update Monte Carlo estimate, or Monte Carlo Alpha (MCA) .

Eligibility traces form of TD(λ)

An alternative way of computing the TD estimator also leads on to alternative forms of the estimator. Eligibility traces (Sutton & Barto, 1998) allow TD to be computed using local storage associated with each of the states.

The eligibility trace for a (non-discounted process) for state s step m in a realisation is given by:

$$e_i(m) = \begin{cases} \lambda e_i(m-1) + 1 \text{ if } i = s_m \\ \lambda e_i(m-1) \text{ if } i \neq s_m \end{cases}$$

Initially eligibility traces are set to zero and at each step the trace decays by a factor λ. In addition each time a state is visited the eligibility trace is increased by 1. This type of eligibility trace is called an Accumulating trace because it accumulates on each visit to the state, see Figure 1.

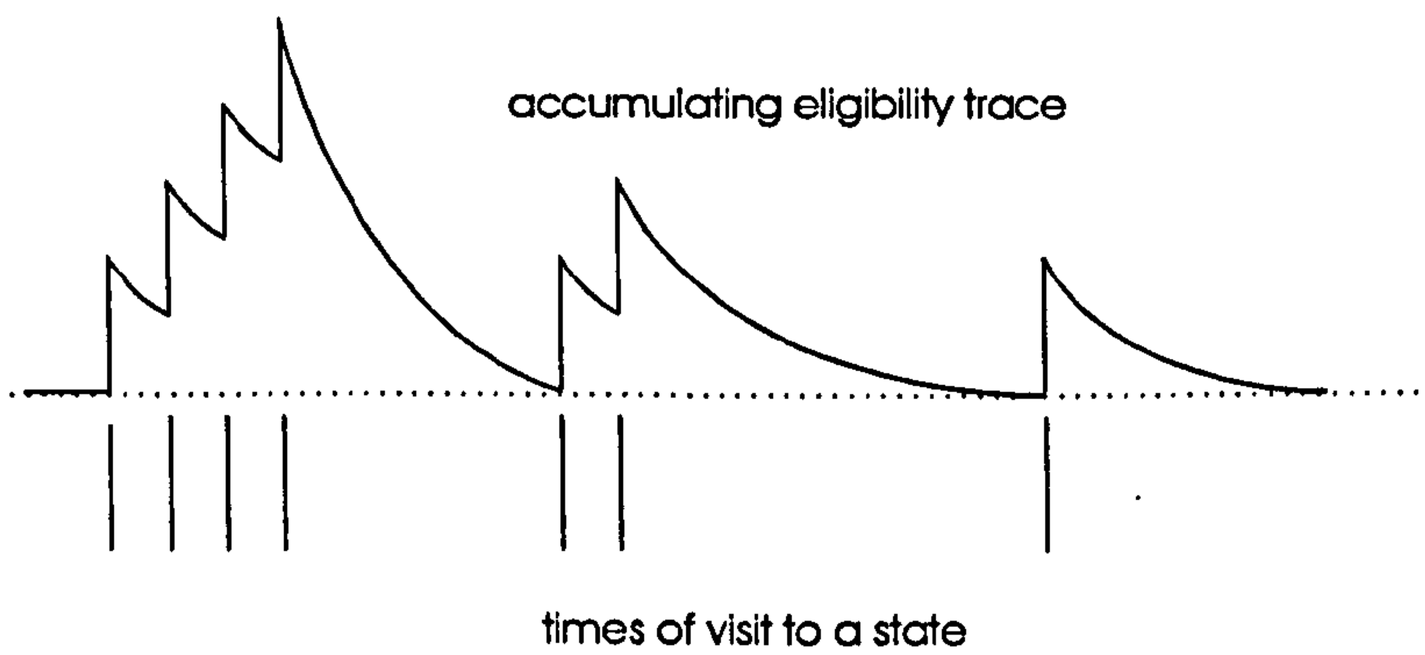


Figure 1: Accumulating eligibility traces

The terminology derives from the idea that the eligibility trace measures the state's eligibility for undergoing a learning change should a reinforcing event occur. In the case of the simplified TD(λ) the eligibility traces are used to form Δ_i^λ used to update state i:

$$\Delta_i^\lambda = \sum_{n=1}^{T-1} [V_t(s_{n+1}) - V_t(s_n)]e_i(n) + [R_T - V_t(s_T)]e_i(T)$$

This is exactly equivalent to TD(λ) with every visit to state i in a realisation being treated as if it came from an independent realisation, i.e. it is an every visit form of TD(λ). That is, Accumulating traces TD(λ) is a way of implementing Every Visit TD(λ).

The eligibility trace formulation naturally leads on to alternative forms of TD(λ) simply by modifying the way traces are computed. For example, replacing trace TD(λ), Singh & Sutton (1996) use the following update rule:

$$e_i(m) = \begin{cases} 1 \text{ if } i = s_m \\ \lambda e_i(m-1) \text{ if } i \neq s_m \end{cases}$$

This resets the trace to 1 when the state has been visited.

Replacing trace TD(λ) is a form of Last visit TD(λ) and is discussed further in Chapter Six.

Finally there is the First TD(λ), introduced by Singh and Dayan (1998), where the eligibility trace is updated using:

$$e_i(m) = \begin{cases} 1 \text{ if } i = s_m \text{ and this is a first visit} \\ \lambda e_i(m-1) \text{ otherwise} \end{cases}$$

First TD(λ) is a form of First visit TD(λ) and is discussed further in Chapter Six.

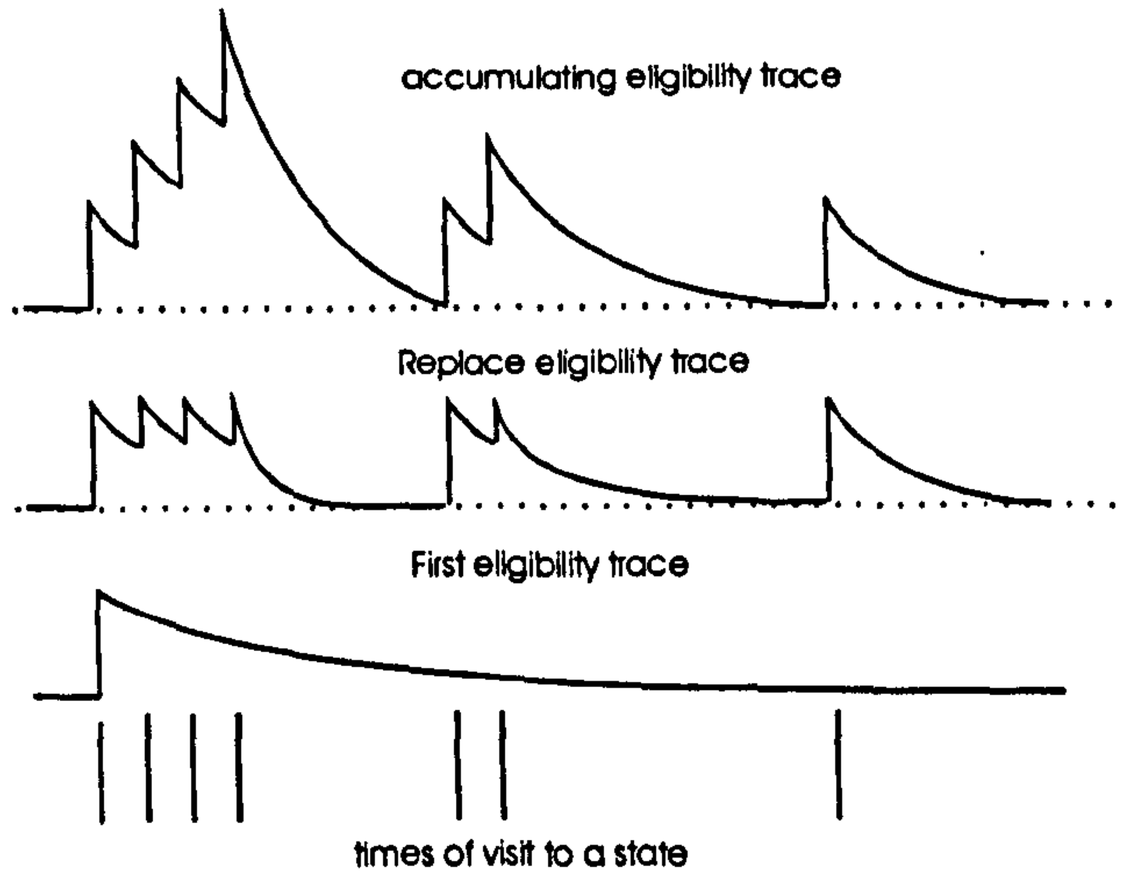


Figure 2: The three types of eligibility trace

The key point about the eligibility trace approach to TD(λ) is that it provides a simple algorithm for computing the estimate. The three methods differ only in the way that they treat states that recur within a realisation. Indeed, in the case of a non-recurrent MDP the three methods produce the same result.

Function approximation

Any of the estimation methods can be used with function approximation (Sutton & Barto, 1998) to simplify the value function. In this case a particular form for the value function is assumed and standard statistical methods are used to fit the function to the current estimate. In practice the estimation of the value function is combined with the functional approximation. For example, the data obtained from sampling the model can be used to train a neural network or be used in direct gradient minimisation of the mean-square error of the approximating function. A particularly important case is the linear approximation. In this case each state is "coded" by a vector of features ϕ_s and the value function is computed using: $V(s) = \theta^T \phi_s$ where θ is a vector of parameters adjusted to make V(s) close to the estimated value function. (Sutton & Barto, 1998)

Clearly functional approximation restricts the form that an estimate can take and for this study a table-based estimator, i.e. a separate estimate of value for each state, is used so that the effect of using a particular estimator isn't confounded by the structure of the assumed function. This is equivalent to using a neural network with sufficient nodes to store the values in the table or any functional approximation with sufficient "degrees of freedom" to reproduce the estimated value function exactly. For example, if a linear approximation is used with the components of ϕ_s all zero apart from the s^{th} component, which is set to one, then the vector θ is simply a table of V(s) estimates.

Conclusion

The Markov Decision Process – MDP is a useful model for many reinforcement-learning situations. In this case the value function satisfies the Bellman equations. When a policy is adopted the MDP can be treated as a Markov chain and realisations can be generated.

There are three broad approaches to constructing suitable estimators of the value function:

- The Monte Carlo estimator is essentially the sample mean cast into an iterative form – the alpha update rule. This can be viewed either as a generalisation of the sample mean, a stochastic iterative estimate related to the Robbins Munro estimator or a gradient descent error minimisation procedure.

- The Certainty Equivalence (CE) estimator is based on solving the model equations using sample estimates of the model parameters – the transition probabilities and rewards. This is also the maximum likelihood estimator of the value function assuming that the underlying model is an MDP. The CE estimator is generally considered to be so computationally expensive as not be a practical proposition.

- The Temporal Difference (TD) estimator is in its most general form a complex looking estimator based on the idea of using the current estimates of the value function as "surrogates" for the eventual reward. The form of the estimator is simpler in the absence of discounting and intermediate rewards and this is the form that is most commonly used and analysed. The eligibility trace form of TD is particularly easy to calculate and its three variations correspond to different approaches to the use of recurrent states.

Chapter Three
A Brief History of TD Estimation

A brief overview of the history of Temporal Difference estimators and their evaluation, exploration and use is presented. Some key results relating to the hypothesis of this thesis are described.

Surveys

There are a number of useful surveys of the area of reinforcement learning and although none deals exclusively with TD they include much about it. The surveys by Keerthi and Ravindran (1995), Kaelbling, Littman and Moore (1996), Dayan (2001), Dayan and Watkins (2001) and Wyatt(2002) provide a good starting point for the exploration of the wider reinforcement learning literature.

The book *Reinforcement Learning: An Introduction* by Sutton and Barto (1998) not only covers general reinforcement learning but does so with an emphasis on TD methods and a number of new results and demonstrations are presented. It provides an introductory account of reinforcement learning topics that had previously been scattered in the literature. The review by Rao (2000) gives some idea of how important an event its publication was:

> *"Given its broad and in-depth coverage of all the important issues in reinforcement learning, this book appears destined to become the standard text in the field in the years to come."*

Neuro-Dynamic Programming by Bertsekas and Tsitsiklis (1996) also contains a full discussion of TD methods but from the perspective of engineering and operations research.

The development of temporal difference and TD(λ)

This history of temporal difference learning is a long one and it took some time for a fully formed theory to emerge. Ideas relating to the need for consistency among value predictions can be traced back to the work of Holland (1975, 1976), studies of animal learning Klopf (1972) and game playing (Samuel, 1959). The TD(λ) estimator in the form described in Chapter Two was introduced by Sutton (1988)

along with a proof of convergence for the TD(0) case. Sutton's 1988 paper, "Learning to predict by the methods of temporal differences", also introduced the term "Temporal Difference learning" and a much used Simple Random Walk (SRW) example which demonstrated that TD(λ) could outperform a simple Monte Carlo alpha (MCA) rule. The idea of n-step returns and the λ return was introduced by Watkins (1989), who also discussed their error reduction properties.

Eligibility traces have their origin in the work of Klopf (1972) who hypothesised that under certain conditions a neuron's synapses would become "eligible" for subsequent modification should reinforcement arrive at the neuron. This biological principle was extended by Sutton (1978a, 1978b, 1978c) and by Barto and Sutton (1981a, 1981b), Sutton and Barto (1981), Barto, Sutton and Anderson (1983). Eligibility traces as a way of computing TD(λ) were introduced by Sutton (1984) in his PhD thesis, "Learning to predict by the methods of temporal differences", in the form of Accumulating traces TD, which is exactly equivalent to an Every visit TD(λ). Replacing traces TD was introduced by Singh and Sutton (1996) "Reinforcement learning with eligibility traces", along with a discussion of the relationship of Accumulate and Replace TD to each other and to Every and First visit Monte Carlo estimators. First TD was introduced in the paper "Analytical Mean Squared Error Curves for Temporal Difference Learning" by Singh and Dayan (1998) as a generalisation of Replace TD.

There have also been many efforts to improve the use or simplify the form of the TD(λ) estimator. For example, Boyan (1999) extends the work of Bradtke and Barto (1996) to product a least squares form of TD which has no adjustable parameters. Dayan (1993) showed how the successor representation could be learned using TD(λ) to improve its generalisation. Cichosz and Mulawka (1995) introduced a truncation procedure, based on an idea by Watkins (1989) to speed up the computation of TD(λ).

Precup, Sutton and Dasgupta (2001) have proposed a new algorithm based on TD(λ) which makes it possible to implement off-policy TD learning, so overcoming one of the biggest problems in utilising the TD estimator in procedures such as Q learning Watkins (1989).

Convergence and bounds

Sutton (1988) proved that TD(0) converged in the mean to the true value function. Barnard (1993) illuminated the way that TD(0) worked by deriving it as a combination of one step of an incremental method for learning a model of the Markov chain and one step of a method for computing predictions from the model. A general proof that TD(λ) converged in the mean was provided by Dayan (1992). Peng (1993), Dayan and Sejnowski (1994) and Tsitsiklis (1994) proved that it converged with probability 1. Jaakkola, Jordan and Singh (1994) extended the proof to on-line updating where estimates are changed before the realisation ends. Gurvits et al. (1994) proved convergence for a more general class of eligibility trace methods. Schapire and Warmuth (1994) provided a worse case analysis for TD(λ) with linear functional approximation.

Bertsekas and Tsitsiklis (1996) provided perhaps the most general proof of convergence with probability 1 for a range of TD like estimator. The proof concerns an estimator of the form:

$$V_{t+1}(i) = V_{t+1}(i) + \alpha(i)\sum_{m=0}^{\infty} z_m(i)d_m$$

where the d_m are the temporal differences defined by:

$$d_m = r_m + (V(s_{m+1}) - V(s_m))$$

i.e. the immediate reward received plus the difference between the current value estimates at the state that occurred at m and at m+1 in the realisation. The z_m are eligibility coefficients which can depend on the realisation in a range of different ways. For example, if $z_m(i) = \lambda^{m-m_1}$ with m_1 being the position in the realisation of the first visit to state *i* then we have the first visit TD(λ) estimator. Similar definitions give estimators that are equivalent to the other well known forms of TD(λ) and many others. The only conditions that the z_m and $\alpha(i)$ have to satisfy are:

1. $z_m(i) \geq 0$, i.e. no negative eligibilities

2. $z_{-1}(i) = 0$, which is needed to guarantee that eligibility traces are initially zero.

3. $z_m(i) \leq z_{m-1}(i)$ if the state at m isn't state i. This together with conditions (1) and (2) forces the eligibility coefficient to be zero until the first visit to the state.

4. $z_m(i) \leq z_{m-1}(i)+1$ if the state at m is state i. This allows the eligibility coefficients to increase by at most 1 with each visit to state i.

5. $z_m(i)$ is completely determined by the realisation, together with information collected before the realisation starts.

6. $z_m(i) = 1$ the first time that it becomes positive.

7. $\alpha(i) \geq 0$ when state i occurs and $\alpha(i) = 0$ if it doesn't.

8. $\sum \alpha(i) = \infty$ for all i where the sum is over all occasions when the state occurs.

9. $\sum \alpha^2(i) < \infty$ for all i where the sum is over all occasions when the state occurs.

Conditions 1 to 6 are very undemanding on the form of the eligibility traces and are met by all version of TD(λ) encountered in practice including every visit, online/offline and replacing traces forms. Conditions 7 to 8 are the usual conditions placed on α in an alpha update estimator to ensure convergence, but with the possibility that the state might not occur in every realisation. Notice that a fixed value of α does not satisfy condition 9.

If conditions 1 to 9 are met then Bertsekas and Tsitsiklis (1996) proved that the estimator converges for all *i* to the true value function with probability 1.

The most recent analysis of general TD(λ), "Bias-Variance Error Bounds for Temporal Difference Updates" by Kearns and Singh (2000) provides upper bounds on its error in a form that doesn't involve the parameters of the MDP. The results apply to an estimator where $\alpha = \frac{1}{t}$ and not a fixed α where t is the number of realisations. Under these conditions convergence is proved to be exponentially fast, i.e. proportional to c^t where c is a constant for fixed λ. Of course, this also applies to

the case of TD(1), i.e. the MCA estimator, and indeed c is proved to be a decreasing function of λ, i.e. convergence gets faster as $\lambda \to 1$

The upper bound also shows the bias variance trade off observed in other experiment-based studies, e.g. Sutton and Barto (1998) and Singh and Dayan (1998). That is, the asymptotic error upper bound increases as $\lambda \to 1$. In addition the upper bound is smaller for intermediate values of λ. The advantage of these results is that they are independent of the MDP's parameters, i.e. they are universal.

Lagoudakis and Parr (2001) extend the error bounds of Bertsekas and Tsitsiklis (1996) to a zero-sum two-person game for both TD(λ) and the least squares variant of the estimator.

RMS error studies

The earliest study of the behaviour of TD(λ) was Sutton's (1988) use of the SRW to illustrate its advantage over the MCA estimator for small samples. Many simulation studies of the behaviour of TD(λ) are included in papers mainly dealing with other topics. One of particular note is by Sutton and Singh (1994), "On bias and step size in temporal-difference learning", where the problem of choosing values of α and λ is considered in detail. Various schemes for varying α and λ with sample size are compared using an a-cyclic model. The CE (maximum likelihood) estimator was also computed as part of the simulation and this proved to provide the best RMS error.

Another simulation study White (1995), looked the comparative performance of Accumulate and Replace TD using the SRW model and a maze task. Optimum values of α and λ were found for the average RMS error over ten steps.

The largest and most complete study of the behaviour of TD(λ) is that by Singh and Dayan (1998) where analytic expressions for the RMS error are given for Accumulate and Replace TD and for First TD, which is introduced for the first time. Analytic expressions are also presented for First and Every Visit MCA.

Singh and Dayan (1998) also demonstrated for the first time that simulations can be very slow to converge to true values using a case in which the empirical method badly failed to match the analytical learning curve after more than 12 million

simulation runs on a small 5-state SRW problem for parameters $\alpha = 0.432$ and $\lambda = 0.5$. The analytic formulae and the corresponding C programs presented in the paper provide a way of exploring the RMS error behaviour of the three common forms of TD without the danger of encountering estimation problems using simulation.

As a result of investigating the behaviour of the RMS curves for the 19-state SRW MDP, Singh and Dayan present four hypotheses:

H1: *For a fixed Markov reward process and a constant λ increasing α has two general effects on the learning curve: there is a largest value of α below which the bias converges to zero and above which the bias diverges (Sutton, 1988; Dayan, 1992), and there is a largest value of α below which the variance converges to a non-zero value and above which it diverges. These largest feasible values of α need not be the same for bias and variance. Based on our limited investigation of learning curves, we conjecture that the largest feasible value of α for bias is greater than or equal to the corresponding value for variance.*

H2: *For each algorithm, increasing α while holding λ fixed increases the asymptotic value of MSE. Similarly, increasing λ in the feasible range while holding α fixed increases the asymptotic value of MSE. Therefore, the smaller the constant α and λ the smaller the asymptotic MSE.*

H3: *For each algorithm, larger values of α or λ lead to faster convergence to the asymptotic value of MSE if there exists one. This may break down for λ very near to 1.*

H4: *In general, for each algorithm as one decreases λ, the feasible range of α shrinks, i.e., larger α can be used with larger λ without causing excessive MSE.*

Although it isn't possible to prove these hypotheses at the present time, they are amply borne out by the RMS curves presented in the paper and elsewhere.

The availability of analytic formulae for the RMS error of each type of TD(λ) estimator applied to a specific MDP allows the calculation of a one-step optimal value of α and λ. In the case of the 19-state SRW the optimal values λ were initially close to one for each form of TD. Although not formally stated as a hypothesis,

Singh and Dayan's comment:

> *"...in fact the drop in MSE may be very insensitive to the value of λ except in the very first few trials, given the ability to schedule α appropriately."*

seems a very reasonable conjecture. This is demonstrated by plotting the RMS ratio for the optimum λ against other values of λ for a range of sample sizes. The charts presented demonstrate that the sensitivity of RMS reduction to λ is clearly confined the first few (5-10) steps of estimation.

After investigation of RMS curves for a cyclic MDP with recurrence controlled by a parameter, an additional hypothesis is added to the first four:

> ***H5:*** *If the initial value function has a high bias, one should begin with a large λ, while if the initial value function has a low bias, one should begin with a small λ. Over time the effect of the initial bias weakens and the asymptotic λ should depend mainly on other problem parameters.*

A comparison of the various TD, Monte Carlo alpha (MCA) and the CE (Maximum Likelihood) estimates is presented using the 5-state SRW model. The optimum values for α and λ were used for the three TD estimators and the optimum values for α in the case of the First and Every visit form of the MCA estimators. The results demonstrate that the CE estimator forms a lower bound for the RMS error in this model and that the other estimators are close to each other with the exception of the Every visit MCA estimator, which has a worse RMS error for all sample sizes. On closer inspection it is clear that Replace TD has a lower RMS than First TD, which in turn is lower than Accumulate TD and First visit MCA.

Finally, Singh and Dayan provide an eigenvalue analysis of the matrices used in the analytical expressions for the 19-state SRW to obtain information on convergence rates and regions. This confirms, for this model, the behaviour of the TD estimators observed in the RMS curves.

One of the important findings is stated clearly in their conclusion:

> *"We had expected that there would be large differences between the three different TD algorithms: accumulate TD, replace TD and first TD. Singh & Sutton (1996) analyzed slightly different versions of accumulate TD and replace TD for $\lambda = 1$, showing that the MSE of accumulate TD is lower at the start of learning, but becomes higher than that of replace TD after some number of trials. However, our results show that given suitable choices of α and λ, the algorithms are essentially indistinguishable - we have cases in which accumulate TD does better, worse, or the same as replace TD."*

This suggests that the differences between the three forms of TD are essentially to do with rescaling of α and λ.

First and Every visit estimators

The issue of First and Every visit estimators and the variations that are possible on this idea are not specific to the TD(λ) estimator but are generally applicable to any estimator where a state can recur within a realisation. One of the earliest discussions of the problem is to be found in Billingsley (1961) although it is clear that it was well known before this. Singh and Sutton (1996) provide an analysis of the MCA estimator in the First and Every visit case with $\alpha=1/N$, i.e. the sample mean. A number of useful theorems are proved but from the point of view of estimation the key result is that after one trial the Every visit estimator is at least as good as the First visit estimator but this possible relative advantage always reverses itself as more samples are obtained. This result applies not only to situations with terminal rewards but with immediate rewards and an example is given where the Every visit estimator is better, in terms of RMS error, than the First visit estimator for the first five steps in a simple model using immediate rewards. Bertsekas and Tsitsiklis (1996) expand on the results of Singh and Sutton (1996) and come to similar conclusions.

Singh and Dayan (1998) give exact analytic expressions for the RMS error of the First and Every visit MCA estimator for arbitrary α.

Uses of TD methods

After the success of Tesauro's Backgammon playing program (Tesauro, 1994) many researchers turned to TD(λ) as ways of learning to play games. Indeed there are at least seven Backgammon programs which now use TD-learning and neural networks, including the open source GNU Backgammon, and the commercially available Snowie and Jellyfish.

There have been several attempts to learn chess using TD methods. One of the earliest, Thrun (1995), specialised Tesauro's design to chess. The KnightCap program (Baxter, Trigdell & Weaver, 1997, 1998, 2000) developed methods of integrating TD(λ) estimation with search procedures. Beal and Smith applied TD(λ) to the existence of pieces as inputs for Chess in their 1997 paper, Learning piece values using temporal differences, and to Shogi in 2001. They used heuristic search and game playing in general (2000). Trinh, Bashi and Deshpande (1998) used Chinese chess as a test bed for TD methods. Mannen and Wiering (2004) also applied TD methods to chess.

The game of Go has also been approached by a variety of TD methods as explored in Schraudolph, Dayan and Sejnowski (1994), Chan, King and Lui (1996) and Enzenberger (2003), and Ekker, van der Werf and Schomaker (2004). For a recent survey of AI including TD approaches applied to Go see Bouzy & Cazenave (2001).

A project is currently underway to add TD learning to an already highly successful checkers playing program, Chinook (Schaeffer, Hlynka & Jussila, 2001). With TD(λ) the designer hopes to produce a checkers program that can finally beat the best human players. Patist and Wiering (2004) have also applied TD learning to Draughts. For a recent overview of TD(λ) and board games see Ghory (2004).

TD(λ) has also been used in applications other than game playing. For example Zhang and Dietterich (1995) apply TD(λ) to train a neural network to schedule space shuttle payloads with good results. Guo and Kuh (1997) use TD learning in sequential decision making task as an alternative to the classical SPRT test and Hansen and Cohen (1992) show how a TD estimator of task duration used in scheduling gives as good a result as dynamic programming methods. Gosavi, Bandla, and Das (2002) describe an application of TD to airline seat pricing.

Reinforcement learning and TD methods in particular are often used in robotics. Gu and Hu (2002), for example, describe how TD(λ), using eligibility traces, can be used in combination with a fuzzy controller to allow a robot to learn to walk. Nie et al (2001) use TD(λ) to learn low level skills in a robot football team which competed in the 2000 world championships. TD learning is also supported in the latest version of IBM's Agent Building and Learning Environment (ABLE) (IBM, 2004). Grigoriadis and Paliouras (2004) apply the TD(λ) estimator to the problem of directing a web crawler searching for information on a specified topic.

To be added to these examples of TD learning in action are the case studies in Bertsekas and Tsitsiklis (1996) – Parking, Football, Tetris, Combinatorial Optimisation, Dynamic Channel Allocation and Backgammon - and in Sutton and Barto (1998) – TD-Gammon, Samuel's Checkers Player, The Acrobat, Elevator Dispatching, Dynamic Channel Allocation and Job-Shop Scheduling.

Chapter Four

MDPs as Markov Chains

By selecting a policy, the problem of estimating the value function reduces to that of estimating functions of the state of a Markov chain. A further restriction to an MDP that involves no discounting and only terminal rewards allows us to consider only the much simpler problem of estimating linear functions of the terminal probabilities of a finite absorbing Markov chain. In this chapter the classical analysis of absorbing Markov chains is presented along with an analysis of functions of state which obey generalised Bellman conditions. This leads to a simple way of computing the expected value of any function of the terminal rewards and path length of an absorbing Markov chain. This allows us to analyse the value function and its variance as a function of state of an absorbing process. These results are used in following chapters to derive theoretical results concerning the performance of estimators related to TD(λ).

MDP + Policy=Markov chain

Given an MDP and a policy π the result of following the policy is to induce a Markov chain on the MDP. In this case the value function is defined via the simplified form of the Bellman equations as shown in Chapter Two:

$$V^{\pi}(s) = R(s) + \sum_{s'} P(s,s')V^{\pi}(s')$$

where R(s) is the expected immediate reward in state s and P(s.s') is the Markov transition matrix. If we restrict our attention to models with no immediate rewards and no discounting then the only possibility is that rewards are received on reaching a state in a terminal set T.

In this case the value function for any given state s is simply:

$$V^{\pi}(s) = R(s) + \sum_{s'} P(s,s')V^{\pi}(s')$$
$$= \sum_{s' \in T} P^{*}(s,s')R(s')$$

where P^{*} is simply the probability of making a transition from s to s' in any number of steps i.e. it is the probability of the process starting in s and terminating in s'.

In other words, for a large class of MDP models the problem of estimating the value function reduces to the problem of estimating the terminal probabilities of the induced Markov finite chain. This reduces our problem to comparing different estimators of linear functions on the terminal probabilities on a Markov chain.

Absorbing processes

Given that an MDP with only terminal rewards can be reduced to a Markov chain by following a fixed policy, it is worth examining the standard theory of absorbing processes. This provides us with expressions for many of the quantities needed in the examination of TD(λ) and related estimators operating on a such absorbing processes. Although much of this theory is standard (Kemeny & Snell, 1983) the needs of statistical analysis of the TD estimator place a different emphasis on the quantities of interest.

A Markov chain that has a set of states that have zero probability of exiting, i.e. a closed set, is called an absorbing process. The transition matrix for an absorbing process can always be put into the form:

$$P = \begin{bmatrix} S & T \\ 0 & I \end{bmatrix}$$

where S is the matrix of transition probabilities between the set of non-absorbing states, T is a transition matrix from the non-absorbing states to the absorbing states and I is an identity matrix which represents the probability of transition within the absorbing states. It is possible consider more general processes where the transition matrix between terminal states is something more complex than I but, as once absorbed the process ceases to be interest, there is nothing to be gained. We can also complicate matters unnecessarily by considering chains composed of non-communicating subsets but again nothing is gained in generality. (Essentially we are only considering irreducible chains.)

The probability of terminating in state j starting from state i after k steps where the agent was not absorbed is simply:

$$[S^{k-1}T]_{ij}$$

where $[]_{ij}$ is the i,j^{th} element of the matrix within the brackets. This can be thought of

as the probability of making k-1 transitions within the non-absorbing set and then a single transition to the absorbing set.

That is:

$$P_k^* = S^{k-1}T$$

is the matrix of absorption probabilities corresponding to not being absorbed until the k^{th} step.

If the reward on reaching the terminal state j is $r_j=[\mathbf{r}]_j$ and if the expected reward starting in state i after k-1 non absorbing steps is $[\mathbf{v}_k]_i$ we have:

$$\mathbf{v}_k = S^{k-1}T\mathbf{r}$$

The expected reward vector irrespective of the number of steps to absorption is simply:

$$\mathbf{v} = \sum_{k=1}^{\infty} S^{k-1}T\mathbf{r}$$

Using the standard result from matrix analysis (Varga, 2000):

$$\sum_{k=1}^{\infty} A^{k-1} = (I-A)^{-1}$$

which is true if A is a stochastic or sub-stochastic matrix (i.e. row sums are 1 or less than one). As S is the probability matrix for an absorbing Markov chain the infinite sum converges and we can write:

$$\mathbf{v} = (I-S)^{-1}T\mathbf{r}$$

This gives us the expected reward for each of the states of the Markov chain induced by the policy π and hence an explicit solution for the value function in terms of the parameters.

The fundamental matrix of the process:

$$Q = \sum_{k=1}^{\infty} S^k = (I-S)^{-1}$$

has a simple interpretation. The element $q_{ij} =[Q]_{ij}$ is the sum of the probability of starting at i and making the transition to j at the k^{th} step. This is the expected number

of times the process enters j starting from i before absorption. The expected number of steps to absorption when starting from state i, i.e. $E[n_i]$, is simply:

$$E[n_i] = \sum_j q_{ij}$$

or in matrix terms

$$\mathbf{n} = Q\mathbf{l} = (I - S)^{-1}\mathbf{l}$$

where $\mathbf{n}$ is the vector of the expected number of steps to absorption and $\mathbf{l}$ is a vector of ones of the correct dimension to form the row sums.

Finally $P^* = QT$ gives the probability or ending in each of the absorbing states. That is $[P^*]_{ij}$ is the probability of starting in state i and terminating in the absorbing state j. Once again we have the connection between estimating the terminal probabilities of a Markov chain and the value function in that:

$$\mathbf{v} = (I - S)^{-1}T\mathbf{r} = P^*\mathbf{r}$$

A summary of these results is given in Table 2.

Functions on a Markov chain

There is another equally valid approach to analysing the behaviour of a Markov chain in terms of the recursive relationships between functions of state and this particularly illuminating when it comes to the behaviour of TD estimators. It provides an analysis in terms of recursively defined state function similar to the Bellman equations. These results make computing means and variances of estimators similar to TD(λ) easier and more transparent.

A function of state of a Markov chain is defined to be a function f(s) where $s \in S$ the set of states of the Markov chain. For any state function simple probability theory and the definition of the conditional expectation gives:

$$E[f(s_{t+1}) | s_t] = \sum_{s' \in S} p_{s_t s'} f(s')$$

where $p_{s_t s'}$ is the probability of making a transition from s_t to the state s', i.e. it is the one-step transition probability. Notice that the expression $E[f(s_{t+1}) | s_t]$ means the expected value of f at time t+1 given the current state is s_t.

Table 2: Summary of finite absorbing Markov chains

$P = \begin{bmatrix} S & T \\ 0 & I \end{bmatrix}$	S is the sub-transition matrix between the non-absorbing states and T is the sub-transition matrix between the absorbing states.
$P_k^* = S^{k-1}T$	Probability of absorption in each of the terminal states at step k and not before.
$P^* = \sum_{k=1}^{\infty} S^{k-1}T = (I-S)^{-1}T = QT$	Total probability of absorption in each of the terminal states.
$\mathbf{v}_k = S^{k-1}T\mathbf{r} = P_k^*\mathbf{r}$	Vector of expected value of each state if absorption occurs at step k and not before.
$\mathbf{v} = \sum_{k=1}^{\infty} S^{k-1}T\mathbf{r} = (I-S)^{-1}T\mathbf{r} = QTr = P^*\mathbf{r}$	Expected value function of each state.
$Q = (I-S)^{-1}$	Number of times process enters state j starting from I
$\mathbf{n} = Ql = (I-S)^{-1}\mathbf{l}$	Expected number of steps to absorption

If a state function is such that it satisfies the following recursive relationship:

$$f(s) = \sum_{s' \in S} p_{ss'} f(s')$$

then:

$$E[f(s_{t+1}) | s_t] = f(s_t)$$

That is, the expected value of f at time t+1 is its current value at time t.

Thus if a function of state satisfies:

$$f(s) = \sum_{s' \in S} p_{ss'} f(s')$$

it is a Martingale[1] because its expectation at time t+1 is the same as its current value.

[1] For a fuller definition of a Martingale than is given in Chapter One see Chung 1974, Appendix 3.

This something of a circular argument because:

$$f(s) = \sum_{s' \in S} p_{ss'} f(s')$$

defines the function f(s) as the expected value of f at the next step but this clear connection quickly gets lost in any real situation. In fact it is easy to see that any such state function can be interpreted as an expected value of some function of the terminal rewards in a finite absorbing chain.

Equally by definition and using the previous result:

$$\begin{aligned} E[f(s_{t+2}) | s_t] &= \sum_{s' \in S} p_{s_t s'} \sum_{s'' \in S} p_{s's''} f(s'') \\ &= \sum_{s' \in S} p_{s_t s'} f(s') \\ &= f(s_t) \end{aligned}$$

a result that can be extended by induction to:

$$E[f(s_{t+n}) | s_t] = f(s_t)$$

for all n.

The condition:

$$f(s) = \sum_{s' \in S} p_{ss'} f(s')$$

is just a simplified form of the Bellman equation and so any quantity which satisfies the Bellman equation gives rise to a Martingale induced on the Markov chain. As the value function satisfies the Bellman equation it is a particular example of a Martingale and so its expectation is constant with time as the process moves from state to state.

For another useful example, the probability of reaching a particular nominated state s^* from state s satisfies:

$$p_{s^*}(s) = \sum_{s' \in S} p_{ss'} p_{s^*}(s')$$

and so it satisfies a Bellman-like equation and has constant conditional probability.

That is:

$$p_{s^*}(s_t)=p_{s^*}(s_{t+n}|s_t)$$

and this means that the expectation value of the termination probability is constant with time.

Extended Bellman state functions

The full Bellman equation includes intermediate rewards and if we are to use the methods described above to deal with this more general condition we have to extend the definition of a state function. The state function equivalent of the more general Bellman equation is:

$$f(s)=\sum_{s'\in S} p_{ss'}[R(s,s')+f(s')]$$

where R(s,s') is some function of the transition from s to s'.

In the more general Markov context this corresponds to a state function which is obtained by adding an amount that depends on the path taken to get to state s' and value of the function at s'. If a state function satisfies this more general Bellman condition then:

$$\begin{aligned} f(s) &= \sum_{s'\in S} p_{ss'}[R(s,s')+f(s')] \\ &= \sum_{s'\in S} p_{ss'}R(s,s')+E[f(s_{t+1})|s_t] \end{aligned}$$

or

$$\begin{aligned} E[f(s_{t+1})|s_t] &= f(s_t)-\sum_{s'\in S} p_{ss'}R(s,s') \\ &= f(s_t)-E[R|s_t] \end{aligned}$$

While this process is not a Martingale is still satisfies an interesting recursive relationship between its current value and the expected value of the subsequent state.

Extending this to t+n gives:

$$\begin{aligned} E[f(s_{t+n})|s_t] &= f(s_t)-E[R|s_t]-E[R|s_{t+1}]\ \ldots -E[R|s_{t+n-1}] \\ &= f(s_t)-\sum_{j=0}^{n-1} E[R|s_{t+j}] \end{aligned}$$

This seems an even less useful relationship but in the case where R is independent of s and s' things look more promising.

In particular if $R(s,s') = r$ then:

$$E[f(s_{t+1})|s_t] = f(s_t) - \sum_{s' \in S} p_{ss'} r$$
$$= f(s_t) - r$$

which generalises by induction to:

$$E[f(s_{t+n})|s_t] = f(s_t) - nr$$

A practical example of this sort of state function is quite easy to find. Consider $L_{s^*}(s)$, the expected path length to particular nominated state s^* from state s, i.e. the count of transitions starting at s until the process satisfies:

$$L_{s^*}(s) = 1 + \sum_{s' \in S} p_{ss'} L_{s^*}(s')]$$

and so:

$$E[L_{s^*}(s_{t+n})|s_t] = L_{s^*}(s_t) - n$$

The expected path length at time t+n clearly going to be n steps less than that at t. Functions which satisfy this condition can be thought of as the expected values of any function of the terminal rewards combined with a function of path length.

Matrix form of recursion

There is a very direct connection between the state function recursions described in the previous sections and the fundamental matrix equations discussed in the section on absorbing Markov processes. The value function satisfies the basic recursion:

$$V(s) = \sum_{s'} p_{ss'} V(s')$$

This can be written in matrix form by using the decomposition into non-terminal and terminal states discussed earlier.

The recursion can be split into a sum over terminal T and non-terminal NT states:

$$V(s) = \sum_{s'} p_{ss'} V(s') = \sum_{s' \in NT} p_{ss'} V(s') + \sum_{s' \in T} p_{ss'} r(s')$$

Translating this to matrix form gives:

$$\mathbf{v} = S\mathbf{v} + T\mathbf{r}$$

where $\mathbf{v}$ is the vector of state values i.e. $E[\mathbf{r}]$, $\mathbf{r}$ is the vector of rewards and S and T are the transition matrices for transitions to the non-terminal and terminal states respectively. In other words, the value vector is the solution to this equation and so:

$$\begin{aligned}\mathbf{v} &= S\mathbf{v} + T\mathbf{r} \\ &= (I - S)^{-1}T\mathbf{r} \\ E[\mathbf{r}] &= QT\mathbf{r}\end{aligned}$$

which agrees with the earlier derivation.

Using the fact that the same equations apply to any function of the terminal rewards we have:

$$E[f(\mathbf{r})] = QTf[\mathbf{r}]$$

Hence:

$$\begin{aligned}E[\mathbf{r}^2] &= (I - S)^{-1}T\mathbf{r}^2 \\ &= QT\mathbf{r}^2\end{aligned}$$

where $\mathbf{r}^2$ means the element-wise square often written as $\mathbf{r} \otimes \mathbf{r}$.

Putting these two together gives:

$$Var(\mathbf{r}) = QT\mathbf{r}^2 - (QT\mathbf{r})^2$$

To summarise, for any absorbing finite Markov chain with fundamental matrix Q:

$E[\mathbf{r}] = QT\mathbf{r}$
$E[\mathbf{r}^2] = QT\mathbf{r}^2$
$Var(\mathbf{r}) = QT\mathbf{r}^2 - (QT\mathbf{r})^2$

These formulae can be used to give the mean and variance of the reward for any absorbing finite Markov chain.

As another example of the use of the matrix recursion we have for path length to absorption:

$$L_s\bullet(s) = 1 + \sum_{s'\in S} p_{ss'} L_s\bullet(s')]$$
$$= 1 + \sum_{s'\in S} p_{ss'} L_s\bullet(s') + \sum_{s'\in T} p_{ss'} 0$$

which on conversion to matrix form gives the standard result for path length:

$$\mathbf{n} = \mathbf{l} + S\mathbf{n}$$
$$\mathbf{n} = (I - S)^{-1}\mathbf{l} = Q\mathbf{l}$$

An important final example that proves useful later is the recursion for $\lambda^{T-m-1}\mathbf{r}$ i.e. the reward weighted by λ raised to the path length from the occurrence of the state. It is clear that the expectation of $\lambda^{T-m-1}\mathbf{r}$ at state s is related to its expectation and the subsequent state by:

$$E_{\lambda^d}[s] = \sum_{s'\in S} \lambda p_{ss'} E_{\lambda^d}[s']$$
$$= \lambda \sum_{s'\in S} p_{ss'} E_{\lambda^d}[s'] + \sum_{s'\in T} p_{ss'} r(s')$$

which on conversion to matrix form gives :

$$\mathbf{w} = \lambda S\mathbf{w} + T\mathbf{r}$$
$$\mathbf{w} = (I - \lambda S)^{-1} T\mathbf{r}$$

where **w** is the vector of expected values of $\lambda^{T-t-1}\mathbf{r} = \lambda^d\mathbf{r}$ for each state.

By analogy with the corresponding equation for the expected reward, i.e. the value function, we have:

$$\mathbf{w} = (I - \lambda S)^{-1} T\mathbf{r} = Q(\lambda)T\mathbf{r}$$

where $Q(\lambda) = (I - \lambda S)^{-1}$ and obviously Q(1)=Q and Q(0)=I.

Following the previous argument the variance of **w** is:

$$\mathrm{Var}(\mathbf{w}) = Q(\lambda^2)T\mathbf{r}^2 - (Q(\lambda)T\mathbf{r})^2$$

To summarise:

$E[\lambda^d \mathbf{r}] = (I - \lambda S)^{-1} T\mathbf{r} = Q(\lambda) T\mathbf{r}$
$Var(\lambda^d \mathbf{r}) = Q(\lambda) T\mathbf{r}^2 - (Q(\lambda) T\mathbf{r})^2$

Conclusion

An MDP becomes a simple Markov process, the induced Markov process, under a given policy π. In the case of only terminal rewards and no discounting the induced Markov chain is a finite absorbing process. In this case the value function is a simple function of the probabilities of terminating in each of the absorbing states. As a result the standard results of finite Markov chain theory is important and the results needed to develop expressions relevant to the linear chain used as a testing ground for TD(λ) are presented.

The value function is just one of many possible functions of state of a Markov chain and the more general theory provides a number of simplified approaches to calculating quantities of interest. When expressed in matrix form the connection between the recursive formulae and the standard theory of finite Markov chains becomes clear.

Chapter Five

The Alpha Update Rule and Divergence

A preliminary examination of the 19-state SRW Markov chain example confirms the small sample superiority of the TD(λ) estimator but raises the question of why some forms diverge. To help answer this question the RMS error for the simple alpha update estimator is derived which reveals that it should converge for all α. The TD(λ) estimator is shown to be a modified alpha update estimator and conditions for this to converge in the mean are derived. Finally after considering the behaviour of the three most commonly used forms of TD(λ) a convergent alternative to the Accumulate trace estimator is suggested.

TD as alpha update

Although the "temporal difference" form and the replacing traces form of the TD estimator given in previous chapters are useful, there is a third way to write TD estimators which makes clear their relationship with the alpha update estimator and is easier to use in theoretical calculations.

For example the first visit TD estimator for a Markov chain with terminal rewards can be written:

$$V_{N(s)+1}(s_t) = V_{N(s)}(s_t)(1-\alpha) + \alpha R_t^{\lambda}$$

$$= V_{N(s)}(s_t)(1-\alpha) + \alpha(1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}V_{N(s)}(s_{t+n}) + \alpha\lambda^{T-t-1}r_T$$

where N(s) is the number of times the state has occurred in the realisations or samples.

This can be compared to the standard incremental MC estimator:

$$V_{N(s)+1}(s_t) = V_{N(s)}(s_t)(1-\alpha) + \alpha r_T$$

From this it is clear that the TD(λ) estimator replaces r_T with:

$$R_t^{\lambda} = (1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}V_{N(s)}(s_{t+n}) + \lambda^{T-t-1}r_T$$

As:

$$(1-\lambda)\sum_{n=0}^{T-t}\lambda^n+\lambda^{T-t-1}=1$$

R_t^λ can be viewed as a weighted average of the $V_{N(s)}$ and r_T.

The expected value of the reward at each step is constant and equal to the expected value of the reward at the current state. Thus TD(λ) forms a weighted average of a set of quantities which all have the same expected value, which is equal to the expected reward at the current state. If these quantities have a smaller variance than the terminal reward then the estimate formed using them should be better than the estimate using just the terminal reward.

Divergence of Accumulating traces TD

As an empirical demonstration of the efficiency of the Accumulate traces TD method, Sutton and Barto (1998) presented the RMS error of estimation for a 19-state linear SRW example with equal probability of a left or right transition. The reward was set to be –1 when the process terminated on the left and +1 when it terminated on the right. This choice of reward causes the estimates to converge to $p_i^* - 0.5$, i.e. the probability of terminating on the right minus 0.5. The initial value of the estimate was set to zero. The estimate was computed using 10 trials and repeated 10 times to give an average RMS per state error of estimation. The measure of error used is designed to demonstrate how good the estimation method is in the early stages of collecting data.

Sutton and Barto (1998) make the point that best performance is achieved for intermediate values of λ and hence TD(λ) is demonstrated to be a better estimator than TD(1), i.e. the Monte Carlo Alpha (MCA) estimator. The subject of the performance of TD(λ) is taken up in detail in a later chapter. The feature examined in this chapter is the divergence of TD(λ). From the chart in Figure 3 this covers only the range α=0 to 0.3. This behaviour is related to the way recurrent states are treated in the Accumulate form of the TD estimator. As will be explained in Chapter Six there are a number of possible ways of handling recurrent states.

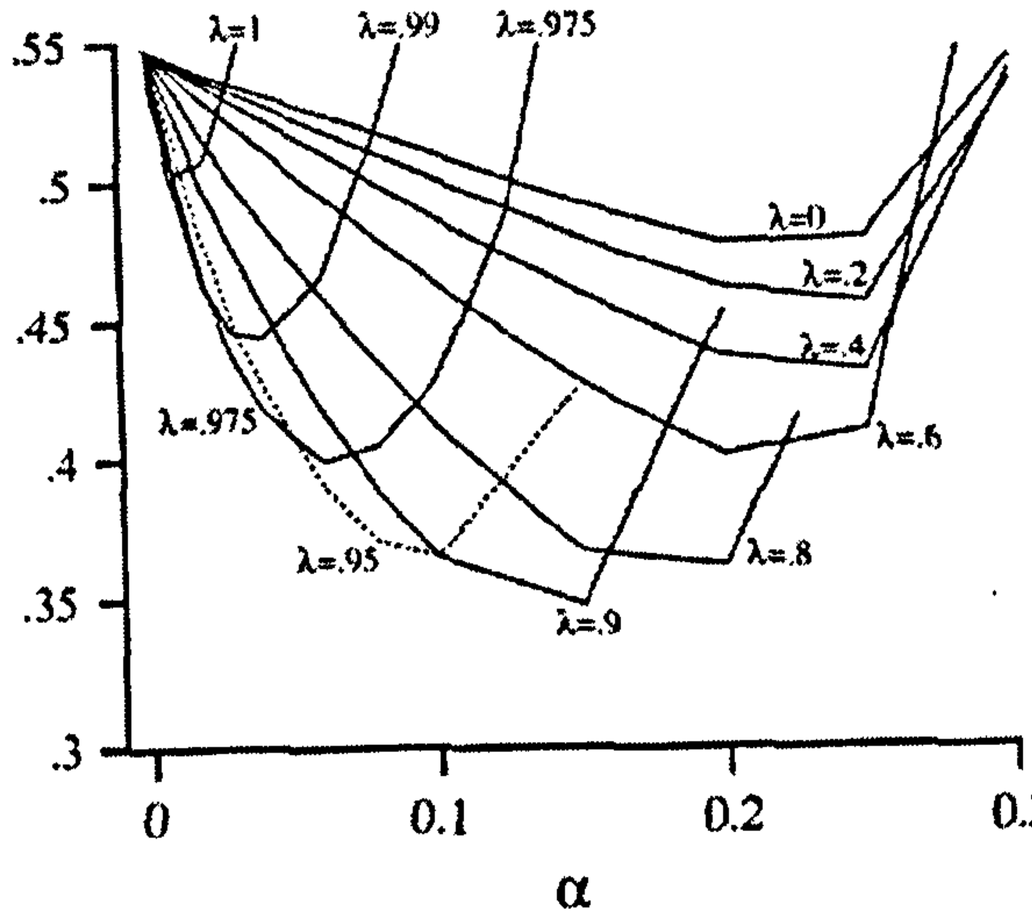


Figure 3: RMS error of estimation (from Sutton and Barto, 1998)

The RMS error of the MCA estimator

In order to understand the behaviour of the general TD(λ) estimator it is worth deriving expressions for the RMS error of the simple alpha update estimator – the Monte Carlo Alpha (MCA) estimator.

To make things simpler we will consider a single state and assume that it occurs in each realisation, which allows the update to be written.

$$V_{k+1} = V_k(1-\alpha)+\alpha r_T$$

Notice that this applies to a general estimation procedure where V_k is the current estimate and r_T is the latest sample.

As a first step to deriving the RMS error of the MC estimator we first need its expectation. Assuming that α is a constant we can solve the first order recurrence relation with constant coefficients, by the standard method (Goldberg, 1958), for V_k:

$$V_k = \sum_{t=1}^{k} \alpha(1-\alpha)^{(k-t)} r_t + (1-\alpha)^k V_o$$

where V_0 is the initial value assigned to the estimator and r_t is the reward at realisation t.

The expectation of V_k can now be easily evaluated:

$$E[V_k] = \sum_{t=1}^{k} \alpha(1-\alpha)^{(k-t)} E[r_t] + (1-\alpha)^k V_o$$

And if r_t is a independent sample from a distribution we have $E[r_t]=E[r]$ and summing the series gives:

$$E[V_k] = E[r][1-(1-\alpha)^k] + (1-\alpha)^k V_o$$
$$= E[r] + (1-\alpha)^k (V_0 - E[r])$$

The bias is:

$$\text{Bias} = (1-\alpha)^k (V_o(s) - E[r])$$

Of course if $0<\alpha<1$ then when $k \to \infty$ the bias goes to zero and

$$E[V_\infty] = E[r]$$

That is, the estimate is asymptotically unbiased irrespective of the starting value V_0 and α.

The variance can be found in the same way:

$$\text{Var}[V_k] = \text{VAR}[\sum_{t=1}^{k} \alpha(1-\alpha)^{(k-t)} r_t + (1-\alpha)^k V_o]$$
$$= \text{VAR}[\sum_{t=1}^{k} \alpha(1-\alpha)^{(k-t)} r_t]$$
$$= \sum_{t=1}^{k} \alpha^2 (1-\alpha)^{2(k-t)} \text{VAR}[r_t]$$
$$= \text{VAR}[r]\alpha \frac{1-(1-\alpha)^{2k}}{(2-\alpha)}$$

using the standard properties of VAR[x] and the fact that $\text{VAR}[V_0]=0$ and the r_t are independent.

Putting the bias and variance together gives the RMS error of the estimate:

$$\text{RMS}[V_k]^2 = \text{VAR}[r]\alpha \frac{1-(1-\alpha)^{2k}}{(2-\alpha)} + (1-\alpha)^{2k} (V_0 - E[r])^2$$

From this it is clear that for α=0 the RMS^2 error is $[V_0 - E[r]]^2$, i.e. the initial bias, and for α=1 it is VAR[r], i.e. the variance in the data - which agrees with common sense. It is also clear that the RMS error never diverges and it is always bounded by $Max[(V_0 - E[r]), VAR[r]]^2$ for all α.

The RMS behaviour of the MCA estimator

Another useful way of expressing the relationship between the RMS error, the initial bias and the variance is to divide through by VAR[R] and write VR as the ratio of the initial bias to data variance (variance ratio).

$$\frac{RMS[V_k]^2}{VAR[r]} = \alpha\frac{1-(1-\alpha)^{2k}}{(2-\alpha)} + (1-\alpha)^{2k} VR$$

This allows us to draw a graph of how the variance in the estimator compares to the variance in the raw data.

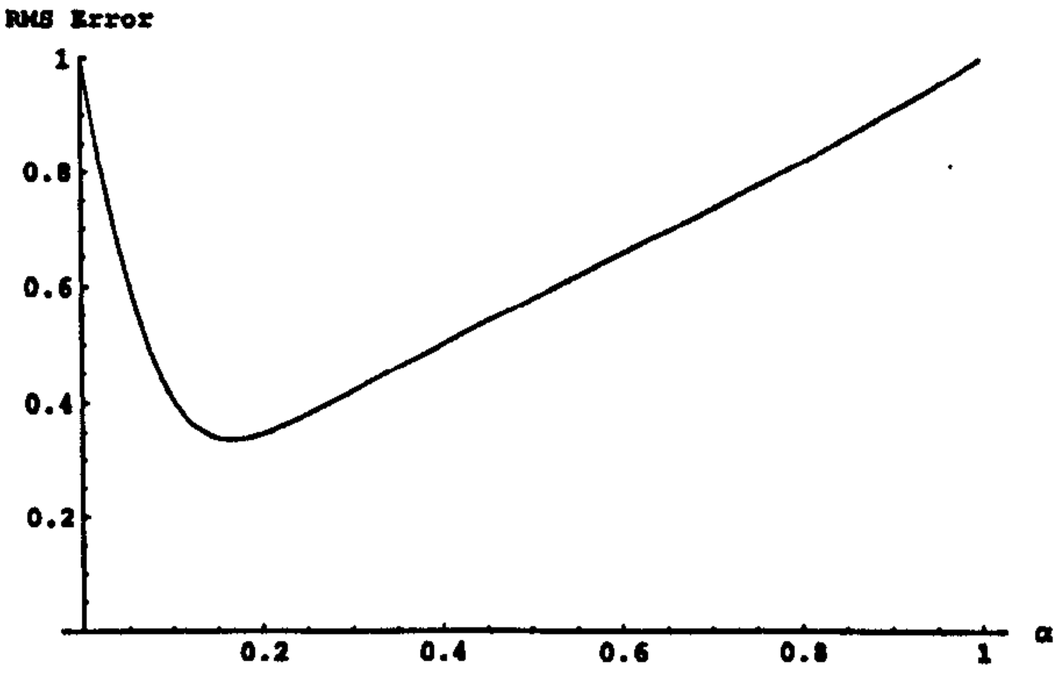


Figure 4: RMS error against α for k=10 and VR=1.

Changing VR and Var[r] simply alters the value of the curve at α=1, i.e. RMS^2= Var[r], and α=0, i.e. RMS^2=Var[r]VR, and doesn't change the location of the minimum. As k increases the minimum moves towards lower and lower α.

There are two other interesting measures of the behaviour of the alpha update estimator. The first is the asymptotic RMS error with increasing sample size:

$$\lim_{k\to\infty} \frac{RMS[V_k]^2}{VAR[r]} = \frac{\alpha}{(2-\alpha)} VR$$

This indicates the error reducing properties of any given selection of α. It is the fraction by which the variance in the sample data is reduced the estimate in the long run. The following chart shows the obvious fact that the smaller α the smaller the residual variance in the estimator.

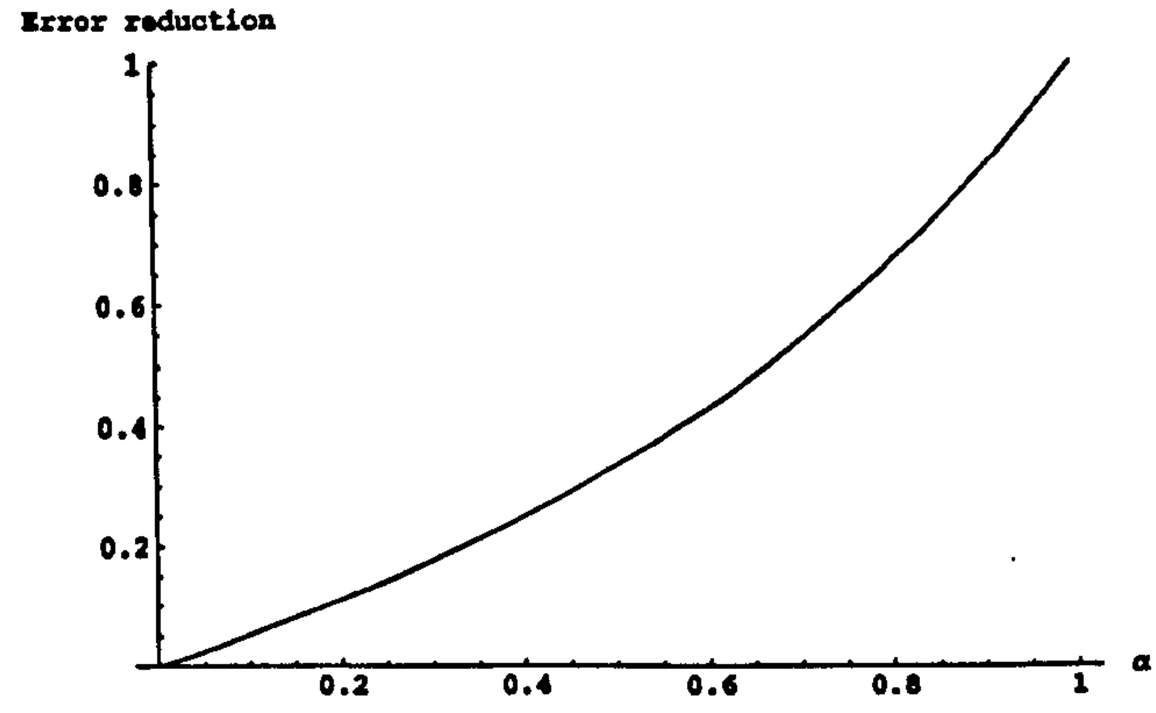


Figure 5: Asymptotic error reduction against α

An interesting measure of how a good a particular choice of α is at both is the number of steps needed to produce equal weighting for bias and variance in the RMS error. That is, given that at step k the RMS error is:

$$RMS[V_k]^2 = VAR[r]\alpha\frac{1-(1-\alpha)^{2k}}{(2-\alpha)} + (1-\alpha)^{2k}(V_0 - E[r])^2$$

For equal weighting we have:

$$\alpha[\frac{1-(1-\alpha)^{2k_{50}}}{(2-\alpha)}] = (1-\alpha)^{2k_{50}}$$

with solution

$$k_{50} = \frac{-\log\left[\frac{2}{\alpha}\right]}{2\log[1-\alpha]}$$

Figure 6 shows the way k_{50} varies with α and it is clear that the tracking ability of the alpha update rule decreases very rapidly for $\alpha>0.2$.

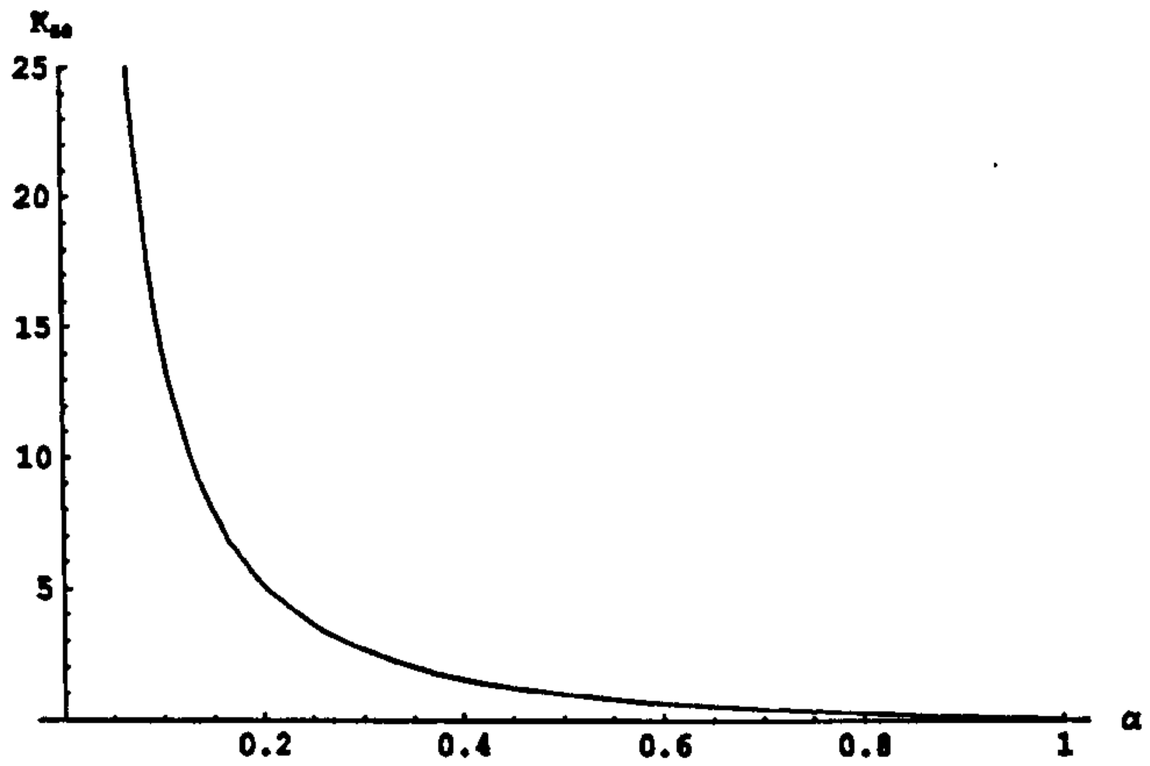


Figure 6: Tracking ability of the alpha update rule

Multiple increment updates

What is clear is that the behaviour of the Accumulate TD(λ) estimator for larger values of α does not conform to the pattern described in the earlier sections and something else must be responsible for the divergence.

The Accumulate TD(λ) is an every visit estimator. What this means is that each occurrence of a state within a realisation is treated as if it was the start of an independent realisation. That is, if a realisation produces the state sequence:

$$s_1 \rightarrow s_3 \rightarrow s_5 \rightarrow s_1 \rightarrow s_4 \rightarrow \cdots$$

then a first visit estimator for state s_1 would treat this as the single realisation:

$$s_1 \rightarrow s_3 \rightarrow s_5 \rightarrow s_1 \rightarrow s_4 \rightarrow \cdots$$

but an every visit estimator would treat the realisation as two realisations starting from s_1 i.e.

$$\underbrace{s_1 \rightarrow s_3 \rightarrow s_5 \rightarrow \overbrace{s_1 \rightarrow s_4 \rightarrow \cdots}^{\text{Second realisation}}}_{\text{First realisation}}$$

In an every visit estimator the alpha update rule would be applied each time the state occurred. There are in fact two ways of doing this – on-line and off-line (Sutton and Barto, 1998). The distinctions between these two methods are discussed in Chapter Six, but essentially the on-line method performs the first update as soon as possible

so changing the estimate before the second update is performed. The off-line form attempts to make a single update when the realisation is complete.

That is, the on-line every visit form makes the update:

$$\begin{aligned} X_{i+1} &= X_i(1-\alpha)+\alpha x_i \\ &= X_i + \alpha\Delta_i \\ &\text{where} \\ \Delta_i &= (x_i - X_i) \end{aligned}$$

each time the state occurs in the realisation. The off-line form accumulates the increments, i.e. the "errors" for each observation, and then batch updates the current estimate in one step. That is, if the state occurs n_i times there are n_i increments Δ_k (or errors) to apply to convert the old estimate into the new estimate. This generalises the previous rule to give the i^{th} update as:

$$X_{i+1} = X_i + \alpha\sum_{k=1}^{n_i} \Delta_k^i$$

where

$$\Delta_k^i = (x_k^i - X_k)$$

When this is converted back into a non-incremental form it is clear that this is not equivalent to the weighted average:

$$\begin{aligned} X_{i+1} &= X_i + \alpha\sum_{k=1}^{n_i}(x_k^i - X_k) \\ &= X_i(1-n_i\alpha)+\alpha\sum_{k=1}^{n_i} x_k \end{aligned}$$

This is not a weighted average because $(1\text{-}n_i\alpha)+\alpha \neq 1$. Assuming that the x_k have the same expected value then it is equivalent to using $n\alpha$ as the step size and even if $\alpha<1$, $n\alpha$ might not be and then the iteration may not converge. Taking expectations gives:

$$\begin{aligned} E(X_{i+1}) &= E(X_i(1-n_i\alpha)+\alpha\sum_{k=1}^{n_i} x_k) \\ &= E(X_i)(1-E(n_i)\alpha)+\alpha E(n_i)E(x) \end{aligned}$$

assuming that n_i is independent of x_i.

Solving this recursion gives:

$$E(X_i) = E(x)(1-(1-E(n)\alpha)^i] + (1-E(n)\alpha)^i X_0$$

In this case if $E(n)\alpha>1$ or $E(n)>1/\alpha$ then the $E(X_i)$ diverges.

In the case of the Markov chain TD(λ) example the expected path length is 81 states, giving a value just less than 5 for E(n), i.e. each state is visited roughly 5 times. Using this value the estimate could be expected to diverge for values of $\alpha> 1/5$, i.e. 0.2 which corresponds reasonably well to the observed behaviour.

Convergence of TD(λ)

For any TD(λ) estimator the update that results at time t+1 can be written (see Appendix 2) in the form:

$$\mathbf{v}_{t+1} = (I-\alpha Diag[\chi])\mathbf{v}_t + \alpha(M\mathbf{v}_t + \mathbf{n}r_t)$$

where M is the matrix of weights for the non-reward based terms, **n** is the vector of weights for the reward based term, χ is a vector of indicator functions for the event that state i occurred in the realisation, Diag[**x**] is the diagonal matrix with the elements of **x** along the diagonal and r_t is the (scalar) reward actually obtained.

For example for a First Visit estimator M and **n** are:

$$[M]_{ij} = (1-\lambda)\sum_{d=1}^{\kappa_j(t)} \lambda^{n_j(t,d)-n_i(t;1)-1}$$

$$[\mathbf{n}]_i = \lambda^{\tau(t)-n_i(t;1)}$$

if the state occurred and zero otherwise.

Similar expressions exist for other variations on TD(λ) and the general idea is clear: the i, j^{th} element of M consists of the λ weights applied to state j and the i^{th} element of **n** is just the λ weight applied to the reward in the calculation of state i's update.

If M and **n** are non-negative and satisfy a relationship of the form:

$$\mathbf{n} + M\mathbf{l} = \chi$$

where $\|\chi\| \leq 1$

The norm (Householder, 1964); Bellman, 1960) is any suitable matrix norm with a compatible vector norm, e.g. the absolute row max norm is defined as: $\|X\| = \max(\sum_j |x_{ij}|)$, then it can be shown (Appendix 2) that:

$$\|\mathbf{v}_{t+1}\| \le \|\mathbf{v}_t\|(1-\alpha) + \alpha|\mathbf{r}_t|$$

or

$$\|\mathbf{v}_{t+1}\| \le \mathrm{Max}(\|\mathbf{v}_t\|, |\mathbf{r}_t|)$$

Thus as long as $\mathbf{v}_0$ and $\mathbf{r}_t$ are bounded so is the iteration and hence it does not diverge for any $0 \le \alpha \le 1$.

Similarly if $\mathbf{n} + M\mathbf{l} = \chi$, where χ is a vector of indicator functions for the event "state i occurred" in the realisation, then it can be also shown (Appendix 2) that the iteration converges to E[**r**], the vector of expected rewards, for $0 < \alpha < 1$. In other words, as long as the lambda weights for any state that occurs in the realisation sum to one the iteration converges in the mean to the value function.

Eligibility trace TD

The eligibility traces form of TD(λ) make use of recurrent states in different ways. In the light of the results given in the previous section it is now worth examining them in more detail to make clear how they use recurrent states and to investigate if and why they diverge.

Accumulating traces TD

The Accumulating traces form of TD(λ) can be written as given in Singh and Dayan (1998):

$$v_i(t) = v_i(t-1) + \alpha\left(\sum_{n=1}^{\tau(t)} K_i(t;n)\left(\sum_{m=n+1}^{\tau(t)} (1-\lambda)\lambda^{m-n-1} v_{s_m}(t-1) + \lambda^{\tau(t)-n_{\tau(t)}} r(t)\right) - \kappa_i(t) v_i(t-1)\right)$$

where $K_i(t;n)$ is an indicator function for state i occurring at position n in realisation t, $\kappa_i(t)$ is the number of times state i occurs in the realisation, and $\tau(t)$ is the length of the realisation.

This computes a TD(λ) update for every position in the realisation but the $K_i(t;n)$ indicator function is a zero unless state i occurs at position n in the realisation. It is easy to show that the λ weights in:

$$\sum_{m=n+1}^{\tau(t)} (1-\lambda)\lambda^{m-n-1} v_{s_m}(t-1) + \lambda^{\tau(t)-n_{\tau(t)}} r(t)$$

sum to one and hence can be regarded as forming a weighted average of the non-reward and reward terms. There are $\kappa_i(t)$ non-zero such terms so the total weighting adds up to $\kappa_i(t)$. Thus the row sum of the matrices corresponding to M, and n in this case, does not sum to one but to $\kappa_i(t)$ and hence the iteration does not converge for all α.

If we write:

$$R = \frac{1}{\kappa_i(t)} \sum_{n=1}^{\tau(t)} K_i(t;n) \left(\sum_{m=n+1}^{\tau(t)} (1-\lambda)\lambda^{m-n-1} v_{s_m}(t-1) + \lambda^{\tau(t)-n_{\tau(t)}} r(t) \right)$$

R is a true weighted average of the non-reward and reward terms.

The update can now be written as:

$$v_i(t) = v_i(t-1)(1-\alpha\kappa_i(t)) + \alpha\kappa_i(t)R$$

Thus Accumulating traces TD diverges when $\alpha\kappa_i(t)$ is larger than 1. When λ=1 R reduces to r(t) we recover the well known result that Replacing traces TD(1)= Every Visit MCA.

From the update rule given above it appears that λ has no effect on the convergence of the Accumulating Traces form of TD(λ) but this ignores the fact that R contains terms that involve $v_i(t-1)$ and these do effect the convergence for fixed α.

First TD(λ)

The First TD(λ) estimator can be written as (Singh and Dayan (1998)):

$$v_i(t) = v_i(t-1) + \alpha\left(\sum_{m=n_i(t;1)+1}^{\tau(t)} (1-\lambda)\lambda^{m-n_i(t;1)-1} v_{s_m}(t-1) + \lambda^{\tau(t)-n_i(t,1)} r(t)\right) - K_i(t) v_i(t-1))$$

where the function $n_i(t;d)$ gives the position in the realisation at which the d^{th} occurrence of state i occurs and $K_i(t)$ is one if the state occurred and zero otherwise Again it is clear that the λ weights sum to one and hence can be regarded as forming a weighted average of the non-reward and reward terms. That is the row sums of the matrices corresponding to M and **n** in this case do sum to one and the iteration does not diverge for any α in the usual range.

This can also be seen by defining:

$$R = \sum_{m=n_i(t;1)+1}^{\tau(t)} (1-\lambda)\lambda^{m-n_i(t;1)-1} v_{s_m}(t-1) + \lambda^{\tau(t)-n_i(t;1)} r(t)$$

The update can now be written as:

$$v_i(t) = v_i(t-1)(1-\alpha K_i(t)) + \alpha K_i(t) R$$

using the fact that R is zero if K_i is. This can be seen to be a true First Visit form of TD(λ) i.e. the TD weighted average is computed just once starting from the first occurrence of the state $n_i(t;1)$. It is also clear that when λ=1 R reduces to r(t) and we recover the well known result that First TD(1)= First Visit MCA.

Replacing traces TD

The Replacing traces form of TD(λ) can be written as (Singh and Dayan,1998):

$$v_i(t) = v_i(t-1) + \alpha\Bigg(\sum_{d=1}^{\kappa_i(t)-1} \sum_{m=n_i(t;d)+1}^{n_i(t;d+1)} (1-\lambda)\lambda^{m-n_i(t;d)-1} v_{s_m}(t-1) + \sum_{m=n_i(t;\kappa_i(t))+1}^{\tau(t)} (1-\lambda)\lambda^{m-n_i(t;\kappa_i(t))-1} v_{s_m}(t-1) + \lambda^{\tau(t)-n_i(t;\kappa_i(t))} r(t) - K_i(t) v_i(t-1)\Bigg)$$

The first term is a sum from the d^{th} occurrence to the $(d+1)^{th}$ with weights given by the distance from the d^{th} occurrence. The second term is the same but from the final occurrence of the state to the final state.

The reward is weighted by $\lambda^{\tau(t)-n_i(t;\kappa_i(t))}$ which uses the number of steps from the last occurrence of the state to the reward. In this sense the Replacing traces TD estimator is a "last visit" estimator however the non-reward terms are not "last visit" terms in quite the same sense. The λ weights can be summed in the usual way:

$$\sum_{d=1}^{\kappa_i(t)-1}\sum_{m=n_i(t;d)+1}^{n_i(t;d+1)}(1-\lambda)\lambda^{m-n_i(t;d)-1}+\sum_{m=n_i(t;\kappa_i(t))+1}^{\tau(t)}(1-\lambda)\lambda^{m-n_i(t;\kappa_i(t))-1}+\lambda^{\tau(t)-n_i(t;\kappa_i(t))}$$

$$=\sum_{d=1}^{\kappa_i(t)-1}1-\lambda^{n_i(t;d+1)-n_i(t;d)}+1-\lambda^{\tau(t)-n_i(t;\kappa_i(t))}+\lambda^{\tau(t)-n_i(t;\kappa_i(t))}$$

$$=\kappa_i(t)-\sum_{d=1}^{\kappa_i(t)-1}\lambda^{n_i(t;d+1)-n_i(t;d)}$$

and it is clear that they no longer sum to one unless $\lambda=1$. Unless this is the case the matrices corresponding to M and **n** do not have rows that sum to one and the iteration does not converge for all α in the usual range.

Defining R as:

$$R=\sum_{d=1}^{\kappa_i(t)-1}\sum_{m=n_i(t;d)+1}^{n_i(t;d+1)}\frac{(1-\lambda)\lambda^{m-n_i(t;d)-1}}{\kappa_i(t)-\sum_{d=1}^{\kappa_i(t)-1}\lambda^{n_i(t;d+1)-n_i(t;d)}}v_{s_m}(t-1)$$

$$+\sum_{m=n_i(t;\kappa_i(t))+1}^{\tau(t)}\frac{(1-\lambda)\lambda^{m-n_i(t;\kappa_i(t))-1}}{\kappa_i(t)-\sum_{d=1}^{\kappa_i(t)-1}\lambda^{n_i(t,d+1)-n_i(t;d)}}v_{s_m}(t-1)+\frac{\lambda^{\tau(t)-n_i(t;\kappa_i(t))}}{\kappa_i(t)-\sum_{d=1}^{\kappa_i(t)-1}\lambda^{n_i(t;d+1)-n_i(t;d)}}r(t)$$

it is clear that it is a true weighted average of the non-reward and reward terms.

This allows the alpha update to be written as:

$$v_i(t)=v_i(t-1)\left(1-\alpha K_i(t)\right)+\left(\alpha\kappa_i(t)-\alpha\sum_{d=1}^{\kappa_i(t)-1}\lambda^{n_i(t;d+1)-n_i(t;d)}\right)R$$

As in the case of Accumulating traces we have a factor of $\alpha\kappa_i(t)$ which can cause the estimate to diverge if it becomes too large however the additional term acts to reduce $\alpha\kappa_i(t)$ as $\lambda \to 1$. Indeed when λ=1 we have $\sum_{d=1}^{\kappa_i(t)-1} \lambda^{n_i(t;d+1)-n_i(t;d)} = \kappa_i(t)-1$ and the update reduces to:

$$v_i(t) = v_i(t-1)\big(1-\alpha K_i(t)\big)+\alpha K_i(t)R$$

which is equivalent to a Last Visit MCA which is in turn the same as a First Visit MCA estimator.

That the divergence gets worse when λ=0 can be seen in Figure 7. A Replacing traces TD(0) that just diverges with a value of α=0.6 in the 19-state linear SRW model shows that increasing the value of λ allows the estimator to converge. (In this case the estimator is first observed to converge when λ>0.7.)

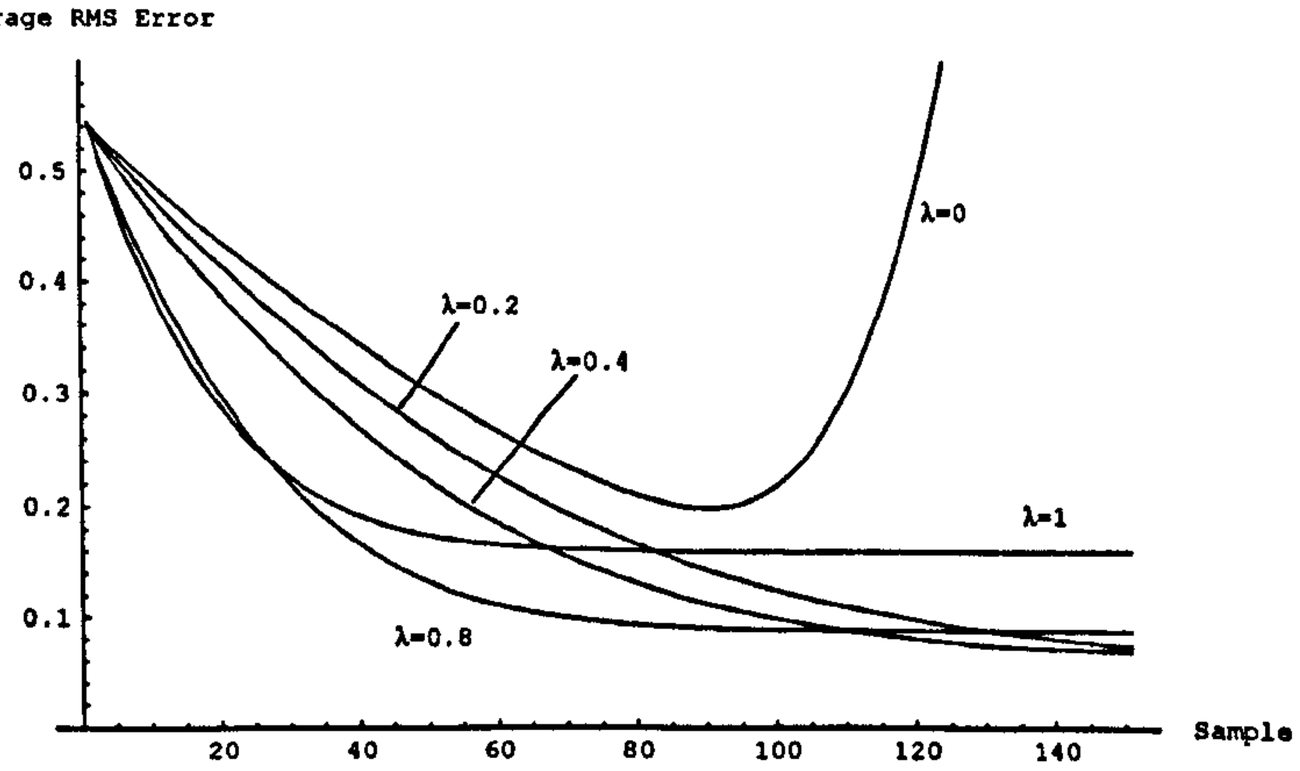


Figure 7: Replacing traces TD with α=0.6 converges for values of λ>0.

n^{th} visit and last visit estimators

The Replacing traces TD estimator raises the idea of using a last visit form of estimator and this in turn raises the idea of generalising the first visit form to the nth occurrence form. In the case of the simple MCA estimator an nth occurrence estimator gives essentially the same result as a First visit estimator – the same reward is received and the update is the identical. The only thing that changes is the probability of occurring for the nth time in a realisation. The same is true even for rewards that are functions of the path taken to reach the terminal state, such as TD

estimators. The reason in this case is the Markov property, which implies that the statistical properties of an nth occurrence estimator do not depend upon n beyond a basic probability of occurrence. This all suggests that the only result of using an nth visit estimator rather than a first visit estimator is to decrease the number of realisations that contribute to the estimate of a state's value.

A last visit estimator isn't subject to the Markov property argument. This is because a last visit estimator isn't an nth occurrence estimator for any fixed n and the realisation after the last occurrence is special in that the state doesn't recur. What this means is that rewards or estimators that vary according to the path taken to the terminal state will give different results in a last visit form.

Correcting the off-line Accumlate TD(λ) estimator

Given that the off-line Accumulating traces form of TD(λ) can be written as:

$$v_i(t) = v_i(t-1) + \alpha\left(\sum_{n=1}^{\tau(t)} K_i(t;n)\left(\sum_{m=n+1}^{\tau(t)} (1-\lambda)\lambda^{m-n-1} v_{s_m}(t-1) + \lambda^{\tau(t)-n_{\tau(t)}} r(t)\right) - \kappa_i(t) v_i(t-1)\right)$$

it is clear that changing this to:

$$v_i(t) = v_i(t-1) + \alpha\frac{1}{\kappa_i(t)}\left(\sum_{n=1}^{\tau(t)} K_i(t;n)\left(\sum_{m=n+1}^{\tau(t)} (1-\lambda)\lambda^{m-n-1} v_{s_m}(t-1) + \lambda^{\tau(t)-n_{\tau(t)}} r(t)\right) - \kappa_i(t) v_i(t-1)\right)$$

gives the update as:

$$v_i(t) = v_i(t-1)(1-\alpha) + \alpha R$$

where

$$R = \frac{1}{\kappa_i(t)}\sum_{n=1}^{\tau(t)} K_i(t;n)\left(\sum_{m=n+1}^{\tau(t)} (1-\lambda)\lambda^{m-n-1} v_{s_m}(t-1) + \lambda^{\tau(t)-n_{\tau(t)}} r(t)\right)$$

then R is a true weighted average of the non-reward and reward terms and the row sums of the corresponding M and **n** do sum to one and the iteration converges for all α in the usual range. Similarly the iteration converges in the mean to the value function for all α in the usual range (Appendix 2). It is also clear that an eligibility trace form of this estimator satisfies the conditions listed by Bertsekas & Tsitsiklis

(1996), see Chapter Three, and hence the estimator converges to the value function with probability one.

This corrected version of the estimator could be computed using a single extra element of storage at each state and hence it is not computationally much more expensive. All that is needed is to keep a count of the number of times a state is visited and divide the "error increment" by this before using it in an update.

Repeating the 19-state linear SRW Markov chain experiment with the form of off-line update used in Sutton & Barto (1998) and elsewhere reproduces the results given in the literature, see Figure 8.

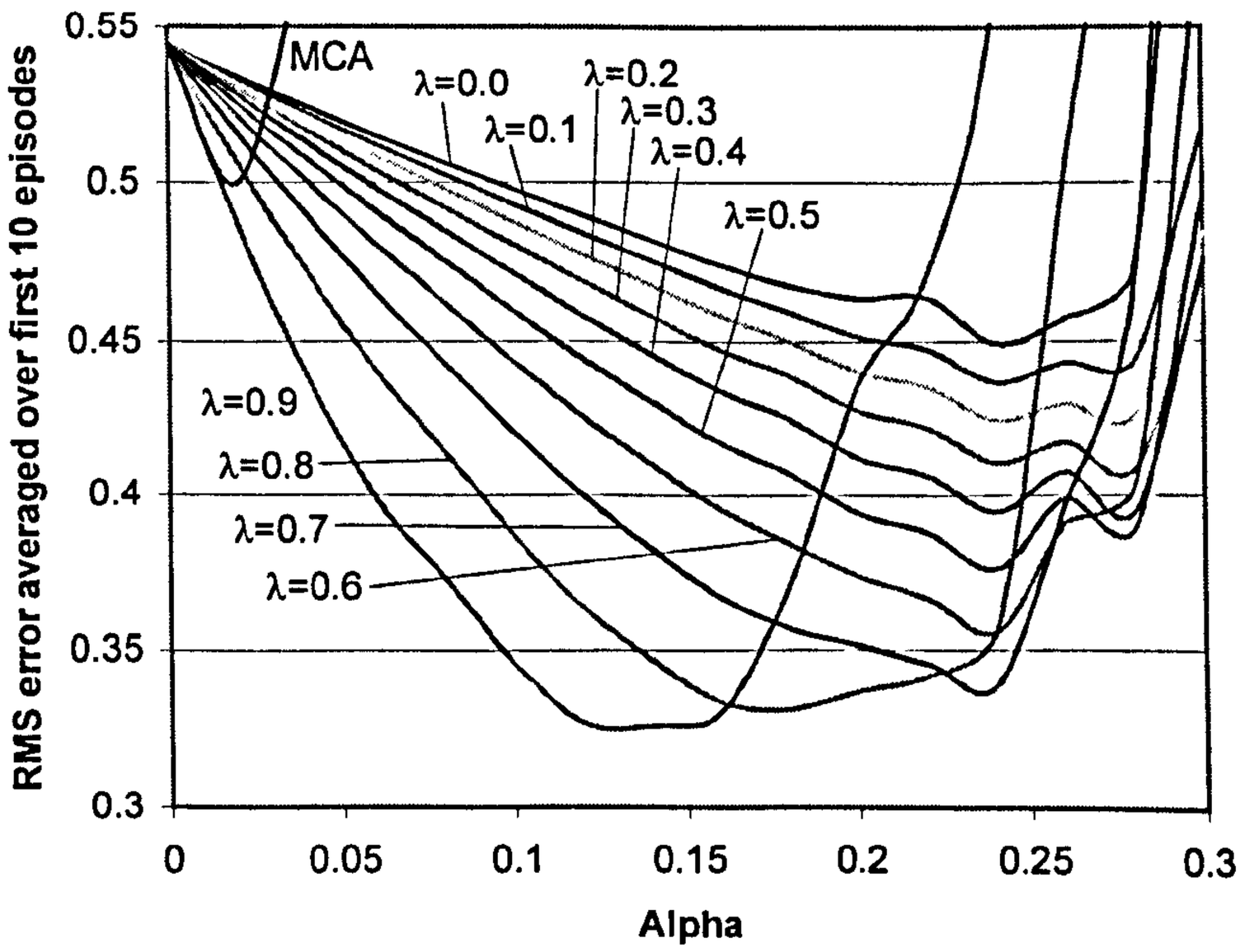


Figure 8: Off-line Accumulate TD

The corrected off-line rule produces RMS error curves that do not diverge for any value of α in the usual range – see Figure 9 (note the change of scale).

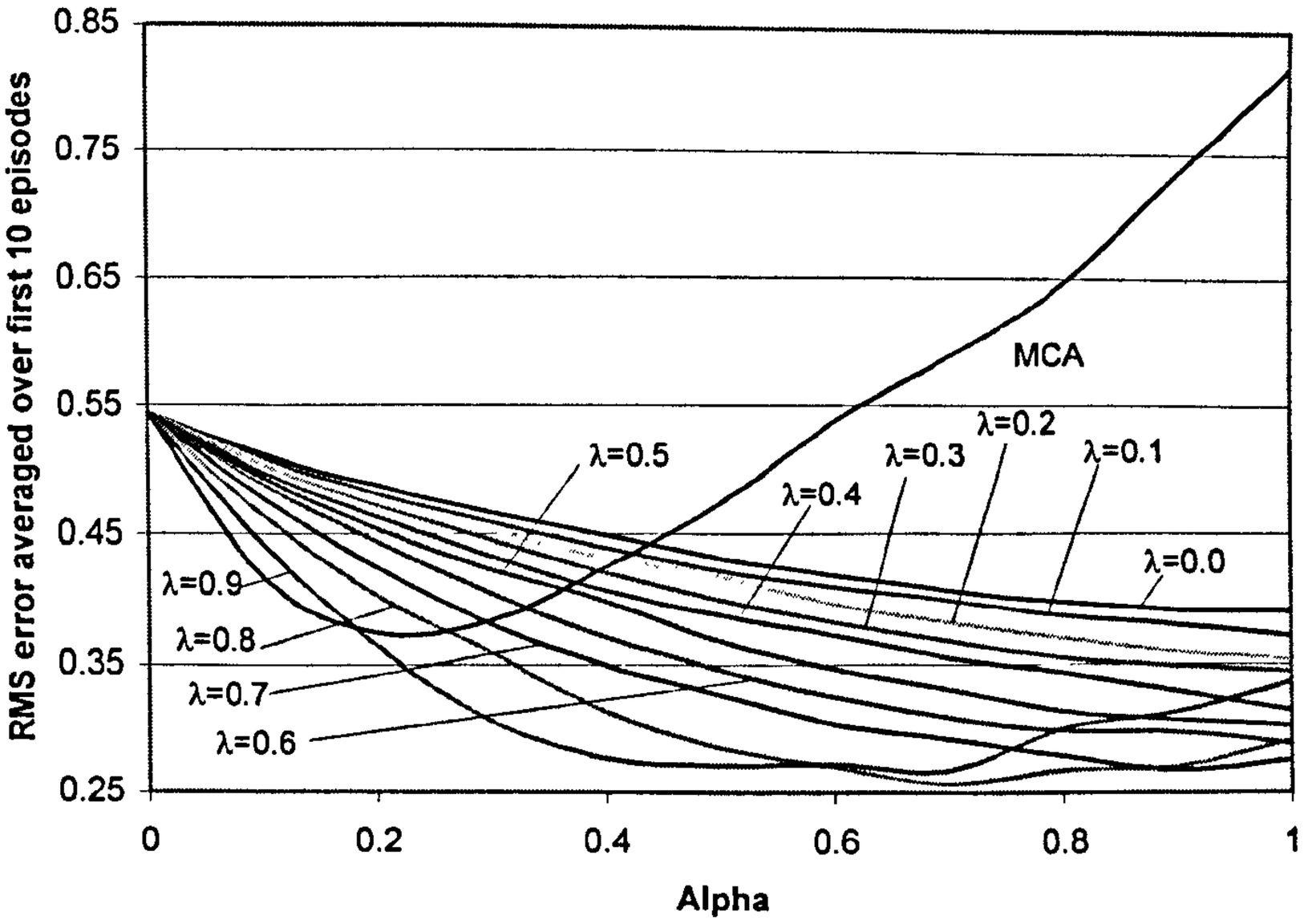


Figure 9: Off-line corrected Accumulate TD

As well as not diverging, the lowest RMS value for a given value of λ, i.e. using the optimum α in each case, is lower for the asymptotically unbiased estimator

Given that there is no way to know what value of α will cause divergence for any situation in which the parameters of the model are unknown and for any α, there is clearly an MDP that produces recurrent states often enough to cause the standard Accumulating traces TD to diverge. It therefore seems reasonable to prefer the corrected form. However while it has been demonstrated that it provides good performance on the 19-state SRW it cannot be assumed that it always produces a better RMS error than the standard Accumulating traces TD. To make progress on this question, an analytic expression for its RMS error is desirable such as that provided by as Singh and Dayan (1998) for the other eligibility trace forms of TD(λ).

Conclusion

The TD(λ) estimator does out-perform simpler estimators, specifically the MCA estimator which corresponds to TD(1), on the linear Markov chain task for some ranges of λ. An analysis of the MC estimator in the alpha update form reveals that it does not diverge for any α in the usual range of 0 to 1 irrespective of the parameters of the model. Clearly the form of the iteration used for all alpha update estimators, MCA or TD indicates that divergence will occur in all cases for some $\alpha > a$ independent of the form of the model e.g. the MCA estimator diverges if $\alpha > 2$. Conditions for a general TD style iteration not to diverge and to converge in the mean to the value function are presented. The relationship of the eligibility trace forms of TD(λ) to first and every visit estimators has been made clear and the reasons for the divergence of the Accumulating and Replacing traces estimators have been outlined. A non-divergent form of the off-line Accumulating traces TD has been proposed and initial investigations indicate that it works as well, if not better, than the divergent form in the case of the 19-state SRW.

Chapter Six

Sampling and MDPs

The issues raised in Chapter Five, concerning the way that recurrent states are used makes it worth focusing on the question of First visit versus Every visit estimation. Here we look at the way the attempt to use all the information contained in a realisation leads to a number of different types of estimator. The properties of the MCA form of these estimators is then explored in detail.

Taking samples from a recurrent Markov chain presents an interesting question of how to deal with multiple occurrences of states within realisations. It has long been known, Billingsley, (1961), that there are two distinct approaches to using the data derived from a realisation usually summed up by the descriptions "Every visit" and "First visit" estimators. In fact, Every and First visit are just two possibilities, as becomes apparent when generalisations applied to the alpha update rule.

Sampling and estimators

The simplest estimator of the expected reward in a MDP is the Monte Carlo (MC) estimator, which corresponds to the sample mean reward. There are two possible sampling schemes (Bertsekas, and Tsitsiklis, 1996) used to compute the MC estimator, which correspond to a fixed and variable sample size.

The simpler sampling scheme is to generate a fixed number of samples for each state:

- Take a sample from the MDP starting it in state i in the form of N realisations x_j j=1,2..N and the total reward received r_{ij}

- The estimate of the expected reward for state i is then simply $\bar{r}_i = \sum_{j=1}^{N} \frac{r_{ij}}{N}$

Of course this is just the usual sample mean of the quantity of interest computed from a random sample.

Given the typical cost of generating a sample, this isn't an efficient use of the data contained in the sample and most practitioners prefer to use each realisation to estimate the average reward for each state that occurs within the realisation.

That is:

- Take a sample from the MDP in the form of N realisations x_{jj}=1,2..N and the reward received r_{ij}

- The estimate of the expected reward for state i is now $\bar{r}_i = \sum_{j=1}^{N} \frac{r_{ij}}{N_i}$

  where N_i is the number of realisations in which state i occurs

The reason why this method of estimation can be used is the Markov property inherent in the MDP. As the process has no "memory" of what happened before the current state, each realisation can be taken as multiple samples starting the process off from each state that occurs in the realisation. The big change between the two estimators is that now N_i is a stochastic variable and not a fixed quantity set before the sample is taken.

However as:

$$E[\bar{r}_i] = E[E\left[\sum_{j=1}^{N} \frac{r_{ij}}{N_i} \middle| N_i\right]]$$

then, as long as $E\left[\sum_{j=1}^{N} \frac{r_{ij}}{N_i} \middle| N_i\right]$ is the sample mean, the estimator is unbiased and, by a similar argument, the variance of the estimator is the same as in the case of fixed N (Bertsekas, and Tsitsiklis, 1996).

As the conditional expectation of reward is clearly not affected by the number of times a state occurs in a realisation this small problem can be ignored and the estimator can be treated as if it were being computed with a fixed N_i.

It is worth noticing that this problem doesn't arise if an alpha update estimator is used. In the case of an alpha update estimate we have:

$$\bar{r}_{iN_i+1} = \bar{r}_{iN_i+1}(1-\alpha) + \alpha r_{ij}$$
$$E[\bar{r}_{iN_i+1}] = E[\bar{r}_{iN_i}](1-\alpha) + \alpha E[r_{ij}]$$

which is of course of exactly the same form as the alpha update estimator for fixed N_i and so its mean and variance are the same. As the sample size doesn't figure in the expression for the alpha update estimator it doesn't have the potential to introduce bias in the same way as in the case of the sample mean. Of course the alpha update is still biased but no more so than in the case of fixed sample size.

First and Every visit estimators for an MDP with terminal rewards

If an MDP has only terminal rewards then any realisation produces a single reward, r say, and this is used to calculate the estimate of the average reward for each state. This makes the form of the First and Every visit estimators particularly simple because each sub-sequence of the realisation results in the same reward.

To be more precise, suppose that state s_i occurs in N_i of the realisations at least once, and these are numbered 1,2,3 to N_i. If each realisation in which it occurred produces a terminal reward of r_j then the First visit estimator is:

$$\bar{r}_i^F = \sum_{j=1}^{N_i} \frac{r_j}{N_i} = \frac{1}{N_i} \sum_{j=1}^{N_i} r_j$$

and it is clearly unbiased.

If in addition we know that state s_i occurred n_{ij} times in the j^{th} realisation then, because the reward is the same for each of the recurrences, the Every visit estimator is simply:

$$\bar{r}_i^E = \sum_{j=1}^{N_i} n_{ij} \frac{r_j}{\sum_{k=1}^{N_i} n_{ik}} = \frac{1}{N_i^E} \sum_{j=1}^{N_i} n_{ij} r_j$$

writing N_i^E for the total number of times s_i occurred in all realisations.

In this case the Every visit estimator can be seen to be a weighted average of the rewards. The weights are proportional to the number of times the state occurred in the realisation. It is clear that the expectation of $\bar{r}_i^E$ isn't the expected reward as n_{ij} and r_j need not be independent. We might hope that any dependency between n_{ij} and r_j might reduce the variance of the estimator rather than increase the bias.

First and Every visit on-line and off-line estimators

As well as the First and Every visit estimator there is the distinction between the way estimators are evaluated using the "on-line" and "off-line" approaches, which have already been introduced briefly in Chapter 5. The idea is that an off-line estimator

performs one up-date per realisation to the estimator whereas an on-line estimator makes multiple updates per realisation, changing the current estimate each time. Clearly the on-line and off-line form of the First visit estimator is the same as only one update is ever made per realisation.

The rationale for on-line updating is usually to make better estimates available as soon as possible but it can be seen as a consequence of the same idea that motivates the use of the Every visit estimator. Indeed it can be argued that the natural form of the Every visit estimator is an on-line estimator.

Every visit on-line MCA

The First visit MCA estimator for state i is simply:

$$\bar{r}^{F\alpha}_{N_i+1} = \bar{r}^{F\alpha}_{N_i}(1-\alpha)+\alpha r$$

where r is the reward obtained at the current realisation (for simplicity of notation its dependence on N_i is not shown). If we want to create an Every visit version of this update rule then the most obvious way of doing it is to apply the rule each time a state occurs in the realisation, i.e. treating each occurrence and its sub-realisation as if it was really an independent realisation. This "independent" update method is, of course an on-line update because it changes the value of the estimator once for each occurrence of the state. For example, for the first occurrence of the state the update is performed giving a new estimate:

$$\bar{r}^{Eon-line}_{2} = (\bar{r}^{Eon-line}_{1}(1-\alpha)+\alpha r)$$

and for the second occurrence the update is performed on the new estimate :

$$\bar{r}^{Eon-line}_{3} = (\bar{r}^{Eon-line}_{2}(1-\alpha)+\alpha r) = ([\bar{r}^{Eon-line}_{1}(1-\alpha)+\alpha r](1-\alpha)+\alpha r)$$

and so on.

In general if the state occurs n_{ij} times that the on-line Every visit update is given by:

$$\begin{aligned}\bar{r}^{Eon-line}_{N_i+1} &= \bar{r}^{Eon-line}_{N_i}(1-\alpha)^{n_{ij}} + \alpha r\sum_{k=0}^{n_{ij}}(1-\alpha)^{k-1} \\ &= \bar{r}^{Eon-line}_{N_i}(1-\alpha)^{n_{ij}} + r[1-(1-\alpha)^{n_{ij}}] \\ &= \bar{r}^{Eon-line}_{N_i}(1-\beta)+\beta r\end{aligned}$$

It can now be seen that the Every visit on-line update is equivalent to a First visit update using a value of α of $\beta = 1-(1-\alpha)^{n_{ij}}$. As this depends on the data, i.e. the number of times the state occurs, this is no longer a simple alpha update rule.

Divergent Every visit off-line MCA

The off-line form of the Every visit MCA estimator has to make a single update per realisation. The most obvious way is to re-write the First visit update as an error increment rule:

$$\begin{aligned}\bar{r}_{N_i+1}^{F\alpha} &= \bar{r}_{N_i}^{F\alpha} + \alpha(r - \bar{r}_{N_i}^{F\alpha}) \\ &= \bar{r}_{N_i}^{F\alpha} + \alpha\Delta_{N_i}^{F\alpha}\end{aligned}$$

Now we can simply sum the increments associated with each occurrence of the state to convert the First visit rule into an off-line Every visit rule.

Each time the state occurs an error increment is added to the total increment and the resulting update is:

$$\bar{r}_{N_i+1}^{Eoff-line} = \bar{r}_{N_i}^{Eoff-line} + \alpha(r - \bar{r}_{N_i}^{Eoff-line}) + \alpha(r - \bar{r}_{iN_i}^{Eoff-line}) + \ldots$$

and in general we have:

$$\bar{r}_{N_i+1}^{Eoff-line} = (\bar{r}_{N_i}^{Eoff-line}(1-n_{ij}\alpha) + n_{ij}\alpha r)$$

In this form it is clear that off-line Every visit update also weights the reward by the number of times the state occurs. Unfortunately this weighting takes the form of a simple multiplier making it possible for $n_{ij}\alpha$ to be greater than 1 and this makes it diverge. The usual solution to this problem is to reduce α to a value that is sufficiently small to make sure that $n_{ij}\alpha$ is smaller than 1. However, as in the case of the Accumulating traces TD estimator, it is difficult to know what value of α will cause divergence and for any selected value there are clearly MDPs that are recurrent enough to make the estimate diverge.

Non-divergent Every visit off-line MCA

An alternative way of using the multiple increments to produce an off-line update is to "normalise" the error increments so that they have the same expected value as a

single error increment, i.e. by dividing the increment by the number of times the state occurred. This update is given by:

$$\begin{aligned}\bar{r}_{N_i+1}^{ENoff-line} &= \frac{1}{n_{ij}}\left(\left[\bar{r}_{N_i}^{ENoff-line}(1-\alpha)+\alpha r\right]+\left[\bar{r}_{N_i}^{ENoff-line}(1-\alpha)+\alpha r\right]+\ldots\right)\\ &= \left(\bar{r}_{N_i}^{ENoff-line}(1-\alpha)+\alpha r\right)\end{aligned}$$

which is the same as the First visit update. That is, the non-divergent form of the Every visit off-line estimator is arithmetically the same as the First visit estimator.

It is important to notice that this only happens because the quantity associated with each occurrence of a state within a realisation is the same, i.e. it is the same reward. In general this is not the case and in particular if there are intermediate rewards then the total reward for each occurrence will be different.

So to summarise:

	First visit	Every visit
Off-line	First visit MCA	Divergent Every visit MCA or Non-divergent Every visit MCA*
On-line	First visit MCA	Every visit on-line MCA

* If there are only terminal rewards then Non-divergent Every visit MCA = First visit MCA

Thus in the case of only terminal rewards there are three distinct estimators: First visit, on-line Every visit and off-line divergent Every visit. It is possible to derive analytical expression for the RMS of each of these estimators.

RMS error for First visit MCA

It is possible to derive an analytic expression for the RMS error for the first visit MCA estimator as given in Singh and Dayan (1998) but in a slightly different form. The First visit MCA update is:

$$v_i(n) = v_i(n-1) + \alpha K_i(n)(r(n) - v_i(n-1))$$

where $K_i(n)$ is an indicator function which is 1 if state i occurred in the nth realisation and 0 otherwise.

The expected value is:

$$E[v_i(n)] = E[r_i](1-(1-\alpha f_i)^n) + (1-\alpha f_i)^n E[v_i(0)]$$

where f_i is the probability that state i occurs in a realisation. For a Markov chain we have $f_i = \left[\mu^T\{I + S_{-i}(I - S_{-i})^{-1}\}\right]_i$ where S is the transition matrix for the transient states and S_{-i} is the matrix with row i set to zero.

The RMS error can be found in a similar way to be:

$$RMS_{E[r_i]}[v_i(n)] = (1+f_i((1-\alpha)^2-1))^n RMS[v_i(0)] + \frac{1-(1+f_i((1-\alpha)^2-1))^n}{1-(1-\alpha)^2}\alpha^2 VAR[r_i]$$

It is easy to show that the estimator converges for all α and that it is asymptotically unbiased.

Notice that if $f_i = 1$ these expressions reduce to those for the simple MCA estimator given in Chapter 5 for the MCA estimator which assumed that the state occurs in every realisation.

RMS error for the divergent off-line Every visit MCA

The RMS error for the divergent every visit off-line MCA estimator is as given in Singh and Dayan (1998) but put into a slightly different form.

In this case the up-date takes the form

$$v_i(n) = v_i(n-1) + \alpha\kappa_i(n)(r(n) - v_i(n-1))$$

where $\kappa_i(n)$ is the number of times that state i occurred in the nth realisation.

The mean is given by:

$$E[v_i(n)] = E[r_i](1-(1-\alpha E[\kappa_i])^n) + (1-\alpha E[\kappa_i])^n E[v_i(0)]$$

$E[\kappa_i] = \mu^T(I-S)^{-1}$ where S is the transition matrix of the transient states of the Markov chain and μ is the initial distribution of states when the process is started, i.e the vector of probabilities that the system is started in state i.

The RMS error is given by:

$$RMS_{E[r_i]}[v_i(n)] = E[(1-\alpha\kappa_i)^2]^n RMS[v_i(0)] + \frac{1-E[(1-\alpha\kappa_i)^2]^n}{1-E[(1-\alpha\kappa_i)^2]} VAR[r_i]\alpha^2 E[\kappa_i{}^2]$$

The expectation values of the various expressions involving κ are:

$$E[\kappa_i] = \mu^T (I-S)^{-1}$$

$$\begin{aligned} E[\kappa_i^2] &= [\mu^T (I-S)^{-1}]_i + 2[\mu^T (I-S)^{-1}]_i [S(I-S)^{-1}]_{ii} \\ &= [\mu^T (I-S)^{-1}]_i [I + 2S(I-S)^{-1}]_{ii} \\ &= E[k_i][I + 2S(I-S)^{-1}]_{ii} \end{aligned}$$

or in terms of First visit probabilities:

$$E[\kappa_i] = \frac{f_i}{(1-f_{ii})}$$

$$E[\kappa_i^2] = \frac{f_i(1+f_{ii})}{(1-f_{ii})^2} = \frac{(1+f_{ii})}{(1-f_{ii})} E[\kappa_i]$$

$$\begin{aligned} E[(1-\alpha\kappa_i)^2] &= 1 - 2\alpha E[\kappa_i] + \alpha^2 E[\kappa_i{}^2] \\ &= 1 - 2\alpha E[\kappa_i] + \alpha^2 \frac{(1+f_{ii})}{(1-f_{ii})} E[\kappa_i] \\ &= 1 - 2\alpha \frac{f_i}{(1-f_{ii})} + \alpha^2 \frac{f_i(1+f_{ii})}{(1-f_{ii})^2} \end{aligned}$$

where f_i is the probability of state i occurring and f_{ii} is the probability that state i occurs given that state i has occurred.

It is also clear that when

$$\alpha \geq \frac{2(1-f_{ii})}{(1+f_{ii})}$$

the RMS error diverges and for a fixed α a process with

$$f_{ii} \geq \frac{2-\alpha}{2+\alpha}$$

causes the RMS error to diverge.

RMS error for the Every visit on-line MCA

The expression for the on-line version of the Every visit estimator is more complicated due its non-linear dependence on the number of times the state occurs. In this case the update is given by:

$$v_i(n) = v_i(n-1) + (1-(1-\alpha)^{\kappa_i(n)})(r(n) - v_i(n-1)$$

the mean is given by:

$$E[v_i(n)] = E[r_i](1 - E[(1-\alpha)^{\kappa_i}]^n) + E[(1-\alpha)^{\kappa_i}]^n E[v_i(0)]$$

and the RMS error is

$$RMS_{E[r_i]}[v_i(n)] = RMS_{E[r_i]}[v_i(0)]E[(1-\alpha)^{2\kappa_i}]^n + \frac{1 - E[(1-\alpha)^{2\kappa_i}]^n}{1 - E[(1-\alpha)^{2\kappa_i}]} E[(1-(1-\alpha)^{\kappa_i})^2] VAR[r_i]$$

We also have for the expected values used in the formulae:

$$E[(1-\alpha)^{k_i}] = E[\beta^{k_i}] = (1-f_i) + \frac{\beta f_i (1-f_{ii})}{1-\beta f_{ii}}$$

$$E[(1-\alpha)^{2k_i}] = E[\beta^{2k_i}] = E[(\beta^2)^{k_i}] = (1-f_i) + \frac{\beta^2 f_i (1-f_{ii})}{1-\beta^2 f_{ii}}$$

$$E[(1-(1-\alpha)^{k_i})^2] = 1 - 2E[\beta^{k_i}] + E[\beta^{2k_i}]$$

Comparing First and off-line Every visit MCA

The RMS error for both the First and off-line Every visit MCA estimator has the form

$$RMS_t = A^t RMS_0 + \frac{1-A^t}{1-A} B Var[r_i]$$

with

$$A = A(\alpha, f_i, f_{ii}) = 1 - 2\alpha \frac{f_i}{(1-f_{ii})} + \alpha^2 \frac{f_i(1+f_{ii})}{(1-f_{ii})^2}$$

and

$$B = B(\alpha, f_i, f_{ii}) = \alpha^2 \frac{f_i(1+f_{ii})}{(1-f_{ii})^2}$$

The expression for the First visit estimator is recovered by setting $f_{ii} = 0$ in both A and B.

This equation has been checked against simulations for a range of models and against the First and Accumulate TD programs from Singh and Dayan (1998) with λ=1. In all cases the differences were within the tolerances of the simulation or computation. For example, see Figure 10 which shows the differences between predictions for the First and off-line Every visit estimator for the 19-state linear SRW model.

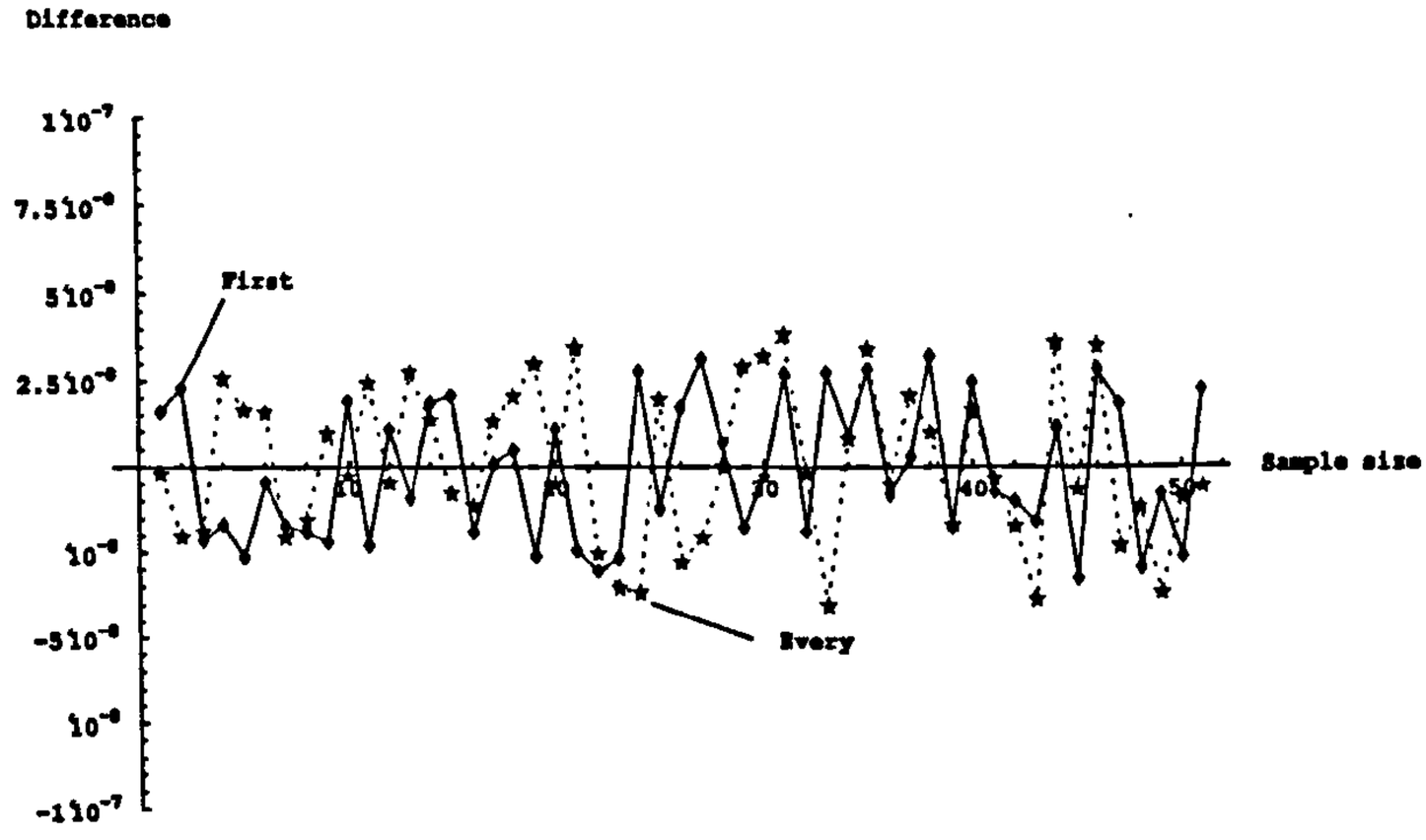


Figure 10: Difference in prediction for the First and off-line Every visit estimators

The formula can be used to compare the two forms of the MCA estimator. To do this we first make the asymptotic error of each form of the estimator the same and then compare the rate of convergence, i.e. the value of A

The First visit estimator has asymptotic RMS:

$$RMS^{F}{}_{E[r_i]}[v_i(\infty)] = \frac{\alpha_F}{2-\alpha_F} VAR[r_i]$$

The Every visit estimator has asymptotic RMS

$$RMS^{E}{}_{E[r_i]}[v_i(\infty)] = \frac{\left(\frac{1+f_{ii}}{1-f_{ii}}\right)\alpha_E}{2-\left(\frac{1+f_{ii}}{1-f_{ii}}\right)\alpha_E} VAR[r_i]$$

Thus the same asymptotic bias is achieved if $\alpha_E = \left(\frac{1-f_{ii}}{1+f_{ii}}\right)\alpha_F$ and $\alpha_E < 2\left(\frac{1-f_{ii}}{1+f_{ii}}\right)$

for convergence of the Every visit estimator.

In this case i.e. with equal asymptotic bias we have:

$$A_F = A(\alpha_F, f_i, 0)$$
$$= 1 - \alpha_F f_i (2 - \alpha_F)_i$$

and for $\alpha_E = \left(\frac{1-f_{ii}}{1+f_{ii}}\right)\alpha_F$ the equivalent A_E is:

$$A_E = A(\alpha_E, f_i, f_{ii})$$
$$= 1 - \alpha_F f (2 - \alpha_F)_i \frac{1}{(1+f_{ii})}$$

This implies that:

$$\frac{1-A_F}{1-A_E} = \frac{\alpha_F f_i (2-\alpha_F)}{\alpha_F f_i \left(\frac{1}{1+f_{ii}}\right)(2-\alpha_F)} = (1+f_{ii})$$

and hence

$$\frac{1-A_F}{1-A_E} = (1+f_{ii}) \geq 1$$
$$\Rightarrow A_F \leq A_E$$

We therefore have the result that if both estimators are adjusted to have the same asymptotic RMS error then the First visit constant α estimator converges faster than the Every visit on-line constant α estimator and this holds for all Markov models.

For example, in the case of the 19-state linear SRW with α_E=0.01 it is clear that the First visit estimator converges faster and has the same asymptotic error.

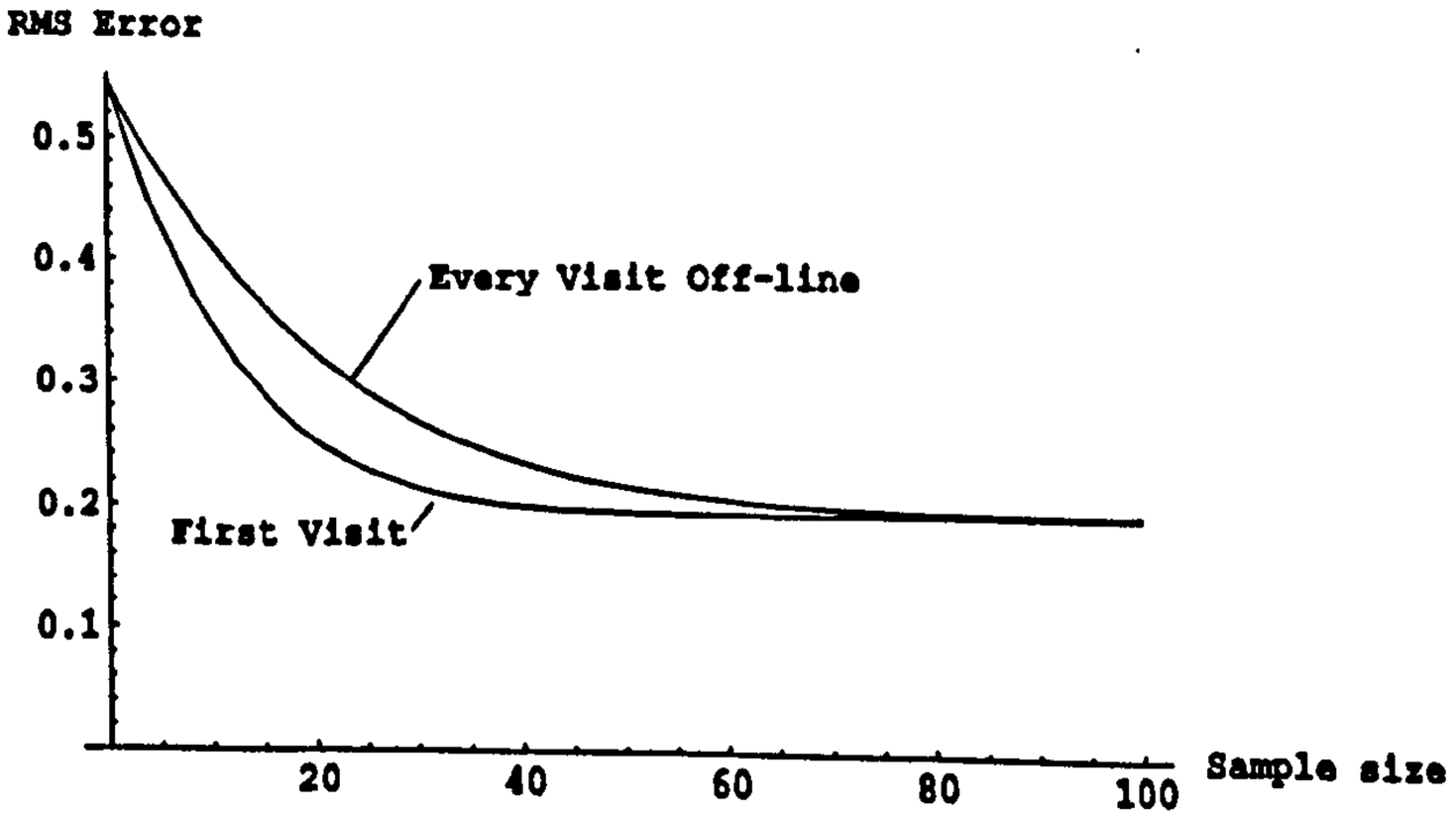


Figure 11: First and Every visit off-line estimators compared

Comparing First and Every visit on-line MCA

The RMS error for the on-line every visit estimator is:

$$RMS_{E[r_i]}[v_i(n)] = RMS_{E[r_i]}[v_i(0)]E[(1-\alpha)^{2k_i}]^n + \frac{1-E[(1-\alpha)^{2k_i}]^n}{1-E[(1-\alpha)^{2k_i}]}E[(1-(1-\alpha)^{k_i})^2]VAR[r_i]$$

As there is no alternative formula for the on-line Every visit MCA estimator this was tested by simulation for a range of parameter values. For example, see Figure 12 which shows the difference between predicated and empirical values for a single state with f_i=0.4, f_{ii}=0.6 and α=0.5. The formula and empirical testing always agreed to three or better decimal places with the statistical variation being of the same magnitude.

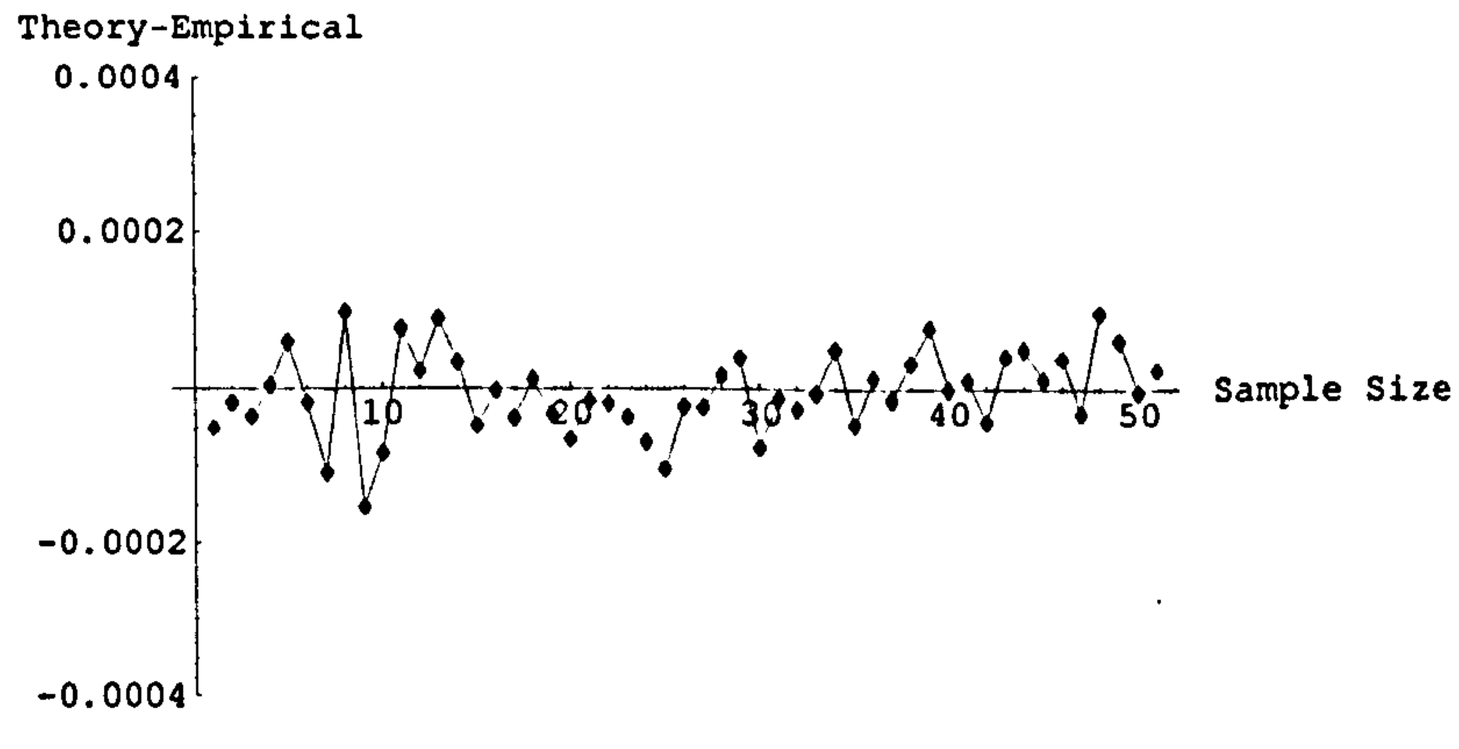


Figure 12: A typical Theory-Empirical curve for Every visit on-line MCA

The RMS error is again of the form:

$$RMS_t = A^t RMS_0 + \frac{1-A^t}{1-A}BVar[r]$$

with

$$A = E[(1-\alpha)^{2k_i}] = (1-f_i) + \frac{\beta^2 f_i(1-f_{ii})}{1-\beta^2 f_{ii}}$$

$$B = E[(1-(1-\alpha)^{k_i})^2]$$

$$= 1-(1-f_i)-2\frac{\beta f_i(1-f_{ii})}{1-\beta f_{ii}} + \frac{\beta^2 f_i(1-f_{ii})}{1-\beta^2 f_{ii}}$$

Clearly when the series converges the asymptotic RMS error is:

$$\begin{aligned} RMS^{Online}{}_{E[r_i]}[V_i(\infty)] &= \frac{B}{1-A} Var[r] \\ &= \frac{(1-2E[\beta^{k_i}]+E[\beta^{2k_i}])}{1-E[\beta^{2k_i}]} Var[r] \\ &= \left(\frac{\alpha}{2-\alpha}\right)\frac{1+f_{ii}(1-\alpha)}{1-f_{ii}(1-\alpha)} Var[r] \end{aligned}$$

Thus the on-line asymptotic error is:

$$RMS^{Online}{}_{E[r_i]}[V_i(\infty)] = \frac{\alpha_o}{2-\alpha_o}\frac{(1+f_{ii}(1-\alpha_o))}{(1-f_{ii}(1-\alpha_o))} VAR[r]$$

The on-line Every visit estimator has the same asymptotic error as the First visit estimator if:

$$\left(\frac{\alpha_o}{2-\alpha_o}\right)\frac{(1+f_{ii}(1-\alpha_o))}{(1-f_{ii}(1-\alpha_o))} = \left(\frac{\alpha_F}{2-\alpha_F}\right)$$

or

$$\alpha_F = \alpha_o\left(\frac{1+(1-\alpha_o)f_{ii}}{1-(1-\alpha_o)^2 f_{ii}}\right)$$

For equal asymptotic RMS we have for on-line Every visit and First visit:

$$A_o = 1-\alpha_o f_i \frac{2-\alpha_o}{1-(1-\alpha_o)^2 f_{ii}}$$

$$A_F = 1-f_i\alpha_o \frac{1+(1-\alpha_o)f_{ii}}{1-(1-\alpha_o)^2 f_{ii}}\left(2+\alpha_o \frac{1+(1-\alpha_o)f_{ii}}{1-(1-\alpha_o)^2 f_{ii}}\right)$$

Hence:

$$\frac{1-A_F}{1-A_o}=\frac{1+(1-\alpha_o)f_{ii}}{2-\alpha_o}\left(2+\alpha_o\frac{1+(1-\alpha_o)f_{ii}}{1-(1-\alpha_o)^2 f_{ii}}\right)$$

$$(1+(1-\alpha_o)f_{ii})\geq(1-(1-\alpha_o)^2 f_{ii})\Rightarrow$$

$$\frac{1+(1-\alpha_o)f_{ii}}{1-(1-\alpha_o)^2 f_{ii}}\geq 1\Rightarrow$$

$$\frac{(1+(1-\alpha_o)f_{ii})}{(2-\alpha_o)}\left(2+\alpha_o\frac{1+(1-\alpha_o)f_{ii}}{1-(1-\alpha_o)^2 f_{ii}}\right)\geq\frac{(1+(1-\alpha_o)f_{ii})}{(2-\alpha_o)}(2+\alpha_o)$$

$$\frac{2+\alpha_o}{2-\alpha_o}\geq 1\Rightarrow(1+(1-\alpha_o)f_{ii})\frac{2+\alpha_o}{2-\alpha_o}\geq(1+(1-\alpha_o)f_{ii})$$

$$\Rightarrow\frac{(1+(1-\alpha_o)f_{ii})}{(2-\alpha_o)}(2+\alpha_o)\geq(1+(1-\alpha_o)f_{ii})\geq 1$$

$$\frac{1-A_F}{1-A_o}\geq 1\Rightarrow A_F\leq A_o$$

We therefore have the result that if both estimators are adjusted to have the same asymptotic RMS error then the First visit constant α estimator converges faster than the Every visit off-line constant α estimator and this holds for all Markov models.

For example in the case of the 19-state linear SRW with α_o=0.01 it is clear that the First visit estimator converges faster and has the same asymptotic error, see Figure 13.

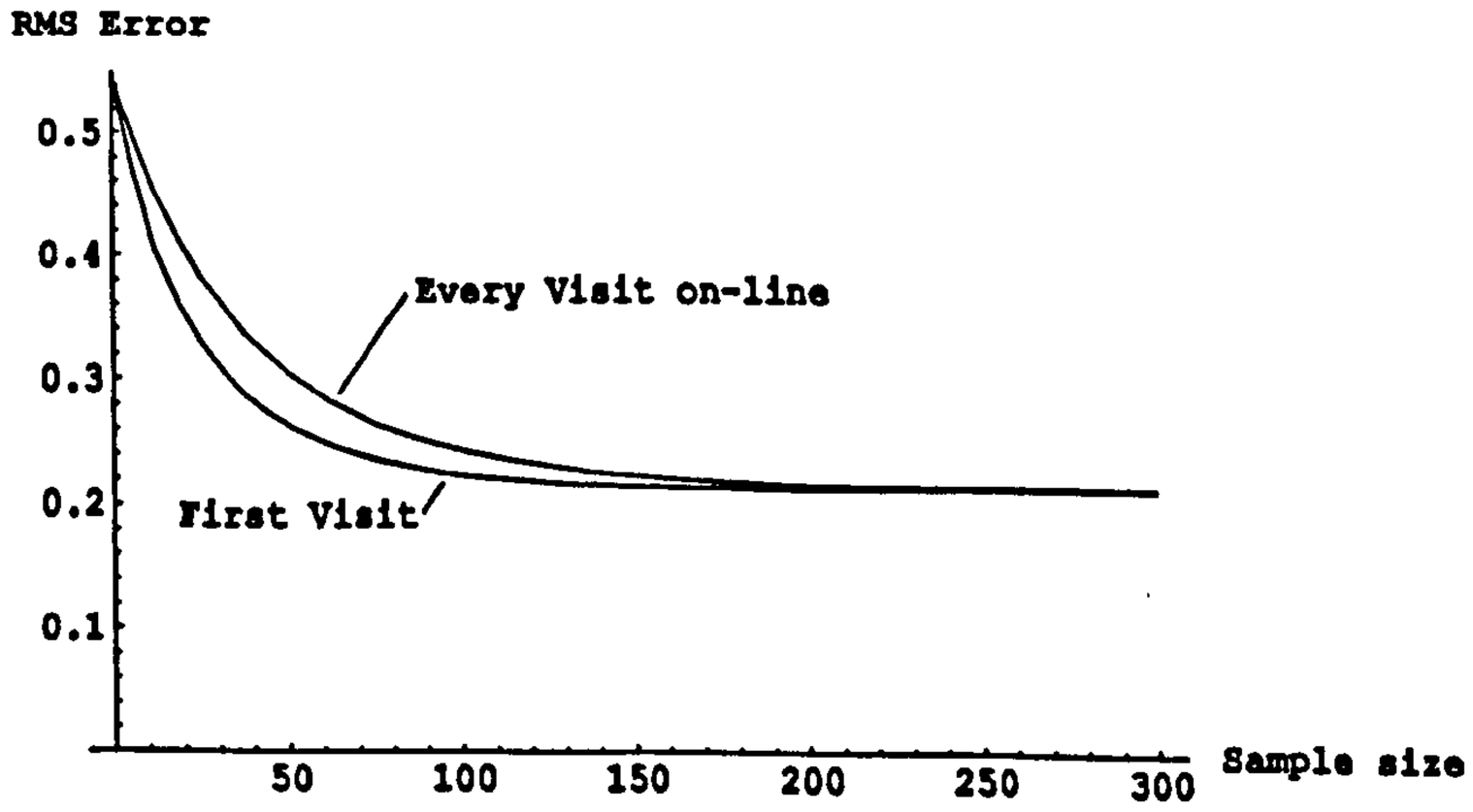


Figure 13: First and Every visit on-line estimators compared

Conclusion

How to make use of all of the available data in a set of realisations from a Markov process is an intricate topic. Sampling schemes result in variations on estimators that can be classified as First/Every visit and as on-line/off-line, although this is more usually thought of as being about how soon to use the available data to update the current estimate.

The question then arises of which form of the estimator is best? In the case of the simple MCA estimator exact analytic expressions for the RMS error can be used to reach firm conclusions that do not depend on the parameters of the Markov model.

If all estimators are adjusted to have the same asymptotic RMS error then the First visit constant α estimator converges faster than either the Every visit on-line or Every visit off-line constant α estimator and this holds for all Markov models. Given this simple fact there seems little point in using anything but a First visit MCA estimator. Unfortunately while it is tempting to conjecture that the same holds for more complex estimators currently this is not easy to prove.

Chapter Seven

An Exploration of TD and Time Weighted Reward

Even though there are problems with selecting values of α and λ to optimise performance, there is no doubt that TD(λ) performs well for small samples. The standard explanation of this behaviour is that it is due to "temporal difference learning". As commented earlier, this seems to be paradoxical as temporal difference learning can only work when there are already good estimates of the value function to be used. This raises the question of what the mechanism is that gives TD(λ) its advantage in the very early stages of learning.

One of the standard "testbeds" for the estimation of value function is the n-state linear Simple Random Walk (SRW). This has been used many times to demonstrate properties of TD estimation - Sutton (1988), White (1995), Sutton and Barto (1998), Singh and Dayan (1998), and many more. All such studies conclude that all of the various forms of the TD(λ) estimator have an advantage over the simpler MCA estimators. In this chapter this model is used in an initial exploration of the small sample behaviour of TD(λ). The First visit TD estimator is used but the results do not change in character if an alternative form of TD is used apart from the need to avoid regions where the estimator diverges and the need to rescale α and λ - see Chapter Nine. All of the RMS curves in this chapter were computed using the analytic RMS programs given by Singh and Dayan (1998) or by the expression for the First visit MCA given in the previous chapter.

The SRW model

The 19-state linear SRW model is very simple and consists of 17 transient states and two terminal states as shown in Figure 14.

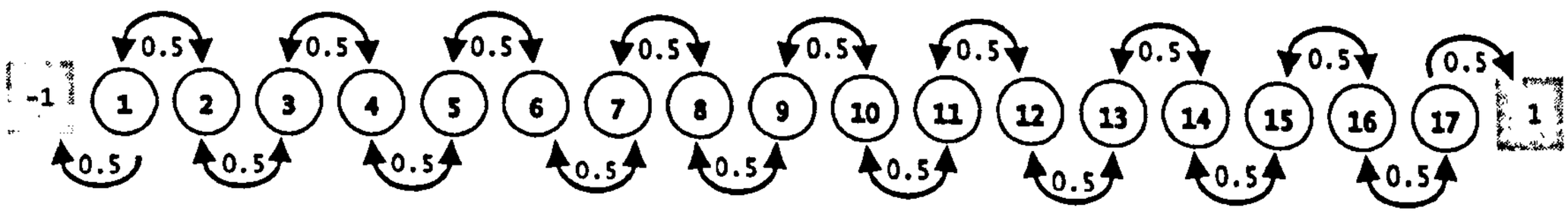


Figure 14: The 19-state linear SRW model

The generalisation to an n-state linear SRW is obvious. The main characteristics of this model is that, with 19 states, a typical realisation is relatively long, averaging 81 states for realisations starting from the middle state. Equal left and right transition probabilities create the longest realisations and the highest recurrence rate.
The theoretical expected reward and variance at each state can be seen in Figure 15. It is clear that the SRW model is highly structured.

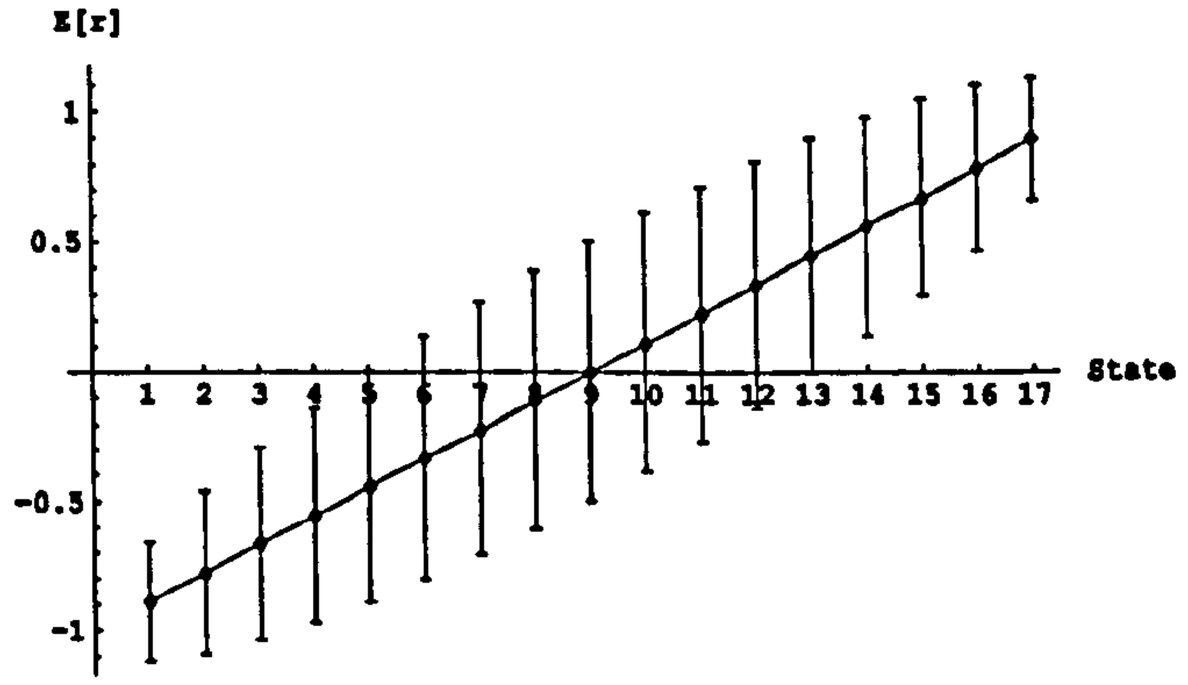


Figure 15: The expected reward at each state of the SRW with RMS error bars

RMS learning curves

The behaviour of the TD estimator is well described in Singh and Dayan (1998) and there is no doubt that it can outperform a simple MCA estimator in some models and for appropriate choices of λ and α. For example Figure 16 shows the TD estimator for TD λ=0.95 α=0.4 and MCA with α=0.146. While these two estimators have the same asymptotic RMS error, the TD estimator has a faster rate of convergence.

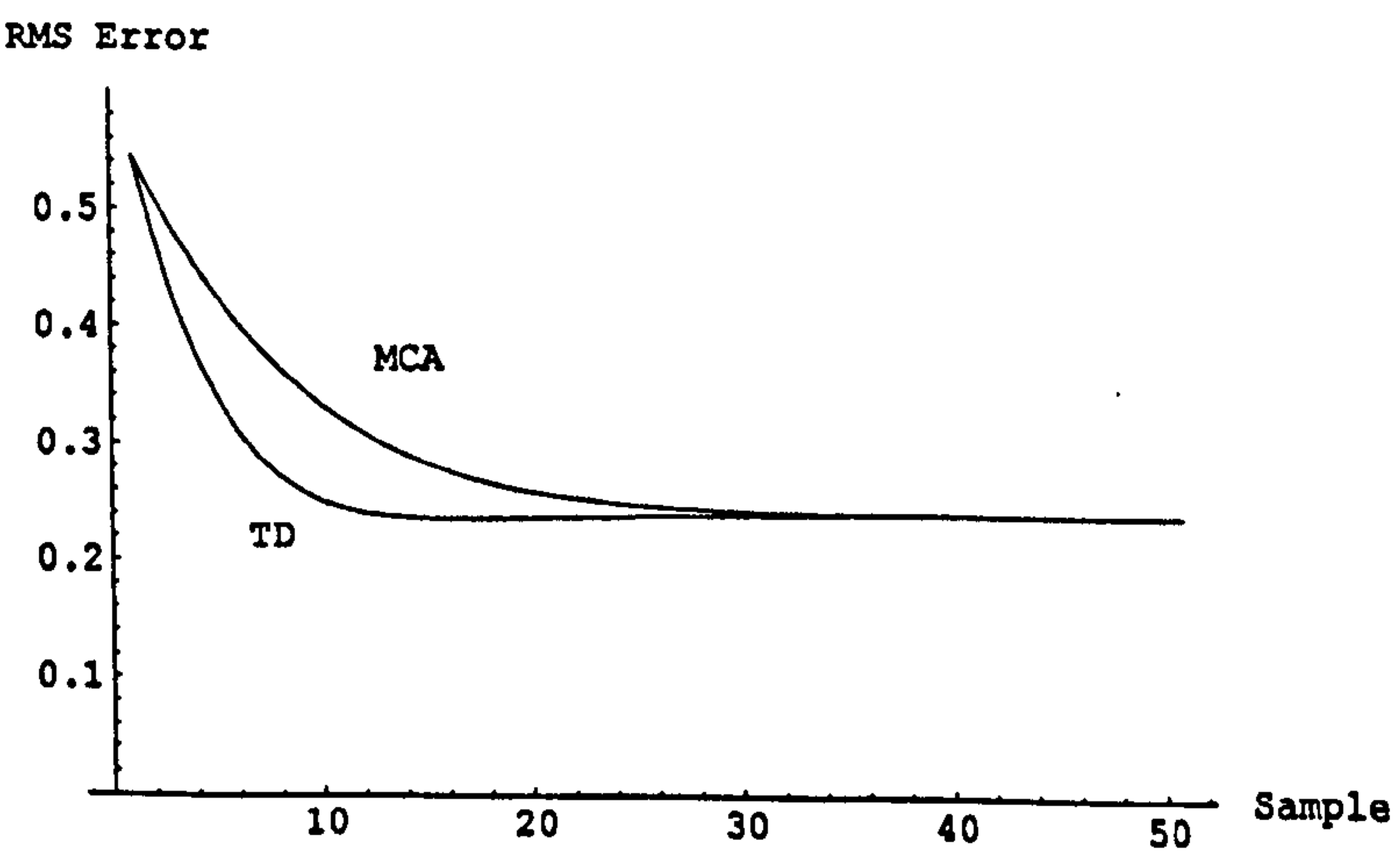


Figure 16, The TD estimator for λ=0.95 α=0.4 and MCA α=0.146

The only main problem with summarising the behaviour of the TD estimator is that it has two parameters affecting its behaviour at each sample size. One solution is to present performance measures that reduce one of the variables. The average RMS over the first ten steps of estimation is one standard method Sutton (1988), Sutton and Barto, (1998) of presenting the performance of TD varying λ and α. In the case of the SRW the results can be seen in Figure 17 and it is clear that values of λ in the range 0.9 to 0.7 have an early advantage over the MCA estimator (λ=1) for a reasonably large range of α.

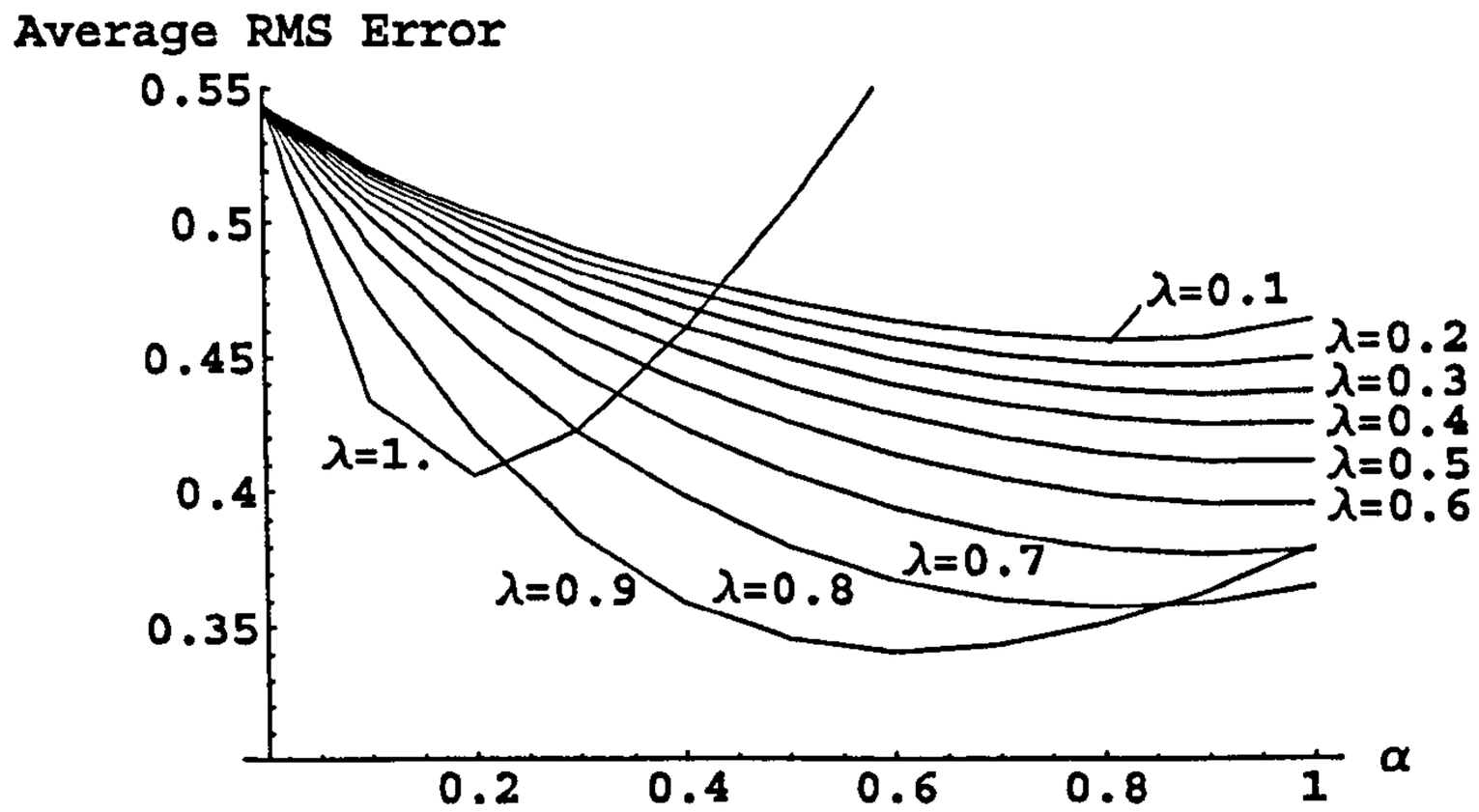


Figure 17: The average RMS error over 10 steps for the SRW for First TD

The 10-step average RMS error is a useful performance measure that indicates ranges of λ and α for which TD might have an advantage over simpler estimators but it doesn't explore what happens in large samples. One method of showing the advantage that TD might have over the simpler MCA estimator is to plot the minimum RMS error achievable at a given sample size by adjusting α. This optimum α plot shows where one estimator is better than other for a fixed sample size. The optimum α plot can be constructed using the analytic RMS programs of Singh and Dayan (1998) and the numerical optimisation commands provided by Mathematica. The behaviour of First TD on the SRW can be seen in Figure 18. Again it is clear that λ in the range .8 to .9 has an early advantage over the optimum α MCA estimator (λ=1) and that intermediate values of λ have a longer-term advantage.

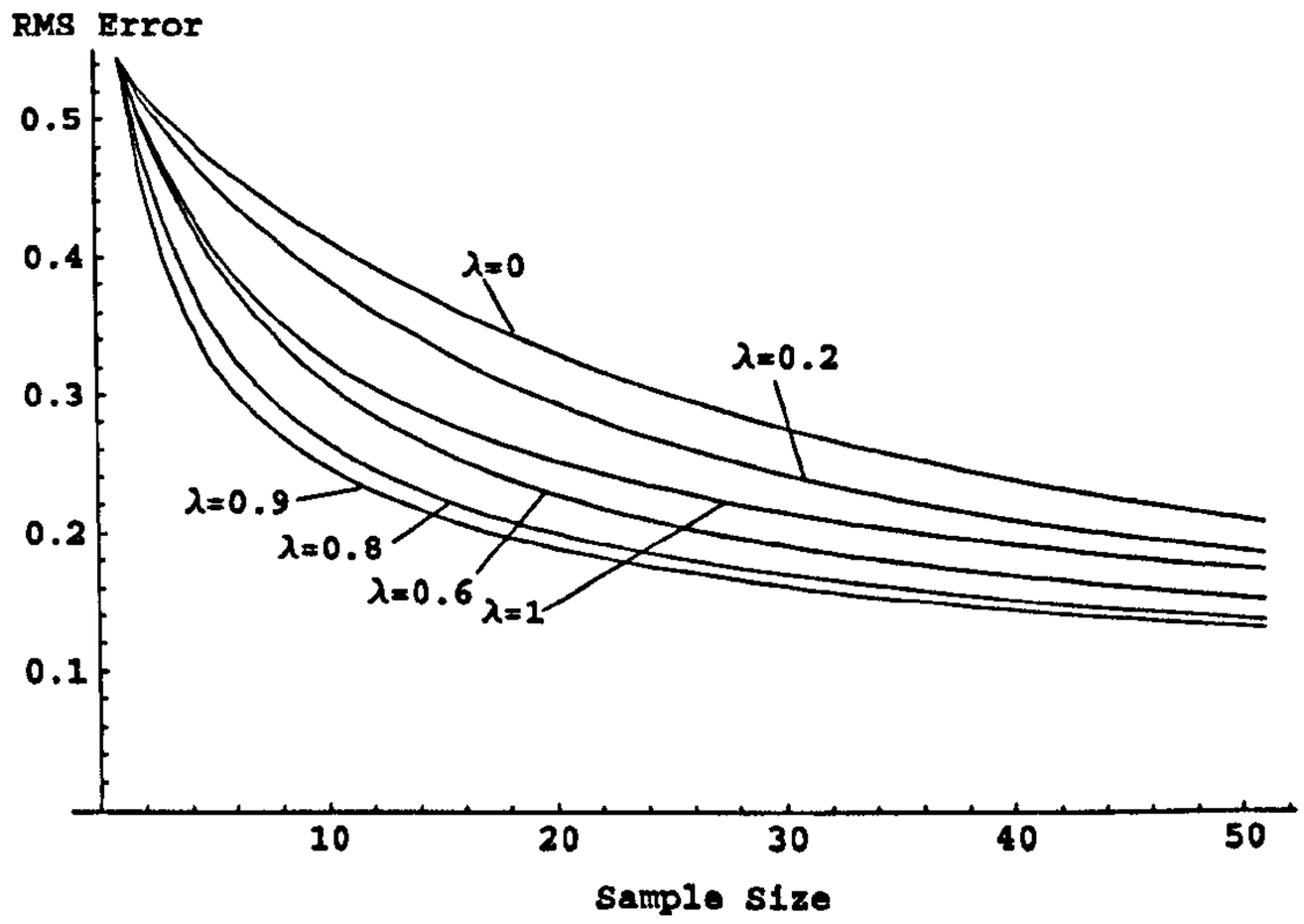


Figure 18: Optimum α RMS error curves for First TD

Both average RMS over ten steps and the optimum α RMS curves are used to compare TD and MCA estimators on a range of models in the subsequent chapters.

Time-weighted reward

Examination of the behaviour of estimators early in their application suggests that part of the problem with the MCA and CE estimators is their inability to assign non-extreme estimates in the early stages of learning. Both make use of the full reward, i.e. +1 or –1, to update the current estimates. TD(λ), on the other hand, weights the final reward by λ^{T-t-1} when updating the estimate at s_t:

$$V_{N(s)+1}(s_t) = V_{N(s)}(s_t)(1-\alpha)+\alpha(1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}V_{N(s)}(s_{t+n})+\alpha\lambda^{T-t-1}r_T$$

That is, each state's estimate is updated using a weight that depends on the distance of the state from the terminal state in the realisation. In other words, if a state occurs just before the reward is received it is updated by λr_T, if it is two steps away from the end of the realisation it is updated by $\lambda^2 r_T$ and so on. In terms of a general philosophy of reinforcement learning this isn't unreasonable in that it assigns more credit for the reward to states that are close to the state that resulted in the reward.

This use of a time-weighed reward makes TD(λ) quite different from the other estimators and it raises the question if this mechanism alone could account for much of its advantage in the early stages of learning.

Small sample approximation

If the initial estimate $V_0(s_t)$ is assumed to be zero, a common choice, then the TD(λ) estimator at the first step, using a first visit form, is:

$$V_1(s_t) = V_0(s_t)(1-\alpha) + \alpha(1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}V_0(s_{t+n}) + \alpha\lambda^{T-t-1}r_T$$

$$= 0(1-\alpha) + \alpha(1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}0 + \alpha\lambda^{T-t-1}r_T$$

$$= \alpha\lambda^{T-t-1}r_T$$

So after the first update the estimate of the value of each state s_i is simply $\alpha\lambda^{d_i}r_T$ where d_i indicates how far in terms of number of states away from the end of the realisation state s_i was.

At the next step the estimator is of the form:

$$V_2(s_t) = V_1(s_t)(1-\alpha) + \alpha(1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}V_1(s_{t+n}) + \alpha\lambda^{T-t-1}r_T$$

$$= \alpha\lambda^{d_i}r_T(1-\alpha) + \alpha(1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}\alpha\lambda^{d_{(t+n)}}r_T + \alpha\lambda^{T-t-1}r_T$$

The sum over states that occurred in the realisation contains terms like $\alpha^2(1-\lambda)\lambda^{(n-1)d_{(t+n)}}r_T$ which are very small compared to the first and last terms, $\alpha\lambda^{d_i}r_T(1-\alpha)$, and $\alpha\lambda^{T-t-1}r_T$ respectively.

A similar argument applies to subsequent steps in the computation of the estimator and so we can conclude that initially TD(λ) behaves like:

$$V_{N(s)+1}(s_t) = V_{N(s)}(s_t)(1-\alpha) + \alpha\lambda^{T-t-1}r$$

This is a much simpler expression and it is clearly an estimator based on a time-weighted reward. That is, the reward is multiplied by a factor that is a function of how "close" the state was to the reward in that particular realisation.

That is the update for each state can be written:

$$V_{N(s)+1}(s_i) = V_{N(s)}(s_i)(1-\alpha) + \alpha\lambda^{d_i}r_T$$

where d_i is the number of states from the end of the realisation at which s_i occurred.

Of course the same reasoning can be extended to the every visit case, with the lambda wieght being generalised to include multiple occurrences of the state.

This new non-temporal difference form of the estimator has both λ and α as variable parameters and can be thought of as a time-weighted reward estimator WR(λ) defined by:

$$V_{N(s)+1}(s_t) = V_{N(s)}(s_t)(1-\alpha) + \alpha\lambda^{T-t-1}r$$

Notice that even though this estimator is a function of only the terminal reward, the weighting involves the position the state occupies in the realisation and this means that the first and non-divergent off-line every visit estimators are distinct.

From the simple-minded derivation of the small sample estimator it is reasonable to conclude that it is most effective for small values of λ but this doesn't take into account the way that the weights are distributed between the reward and non-reward based components of the update. Consider the expression for the TD estimator in terms of this division:

$$\begin{aligned} V_1(s_t) &= V_0(s_t)(1-\alpha) + \alpha(1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}V_0(s_{t+n}) + \alpha\lambda^{T-t-1}r_T \\ &= V_0(s_t)(1-\alpha) + \alpha(1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}\text{non}-\text{reward} + \alpha\lambda^{T-t-1}\text{reward} \end{aligned}$$

It is clear that the non-reward based terms are weighted by $(1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1} = (1-\lambda^d)$

and the reward is weighted by λ^d. That is:

$$V_1(s_t) = V_0(s_t)(1-\alpha) + \alpha(1-\lambda^d)\text{non}-\text{reward} + \alpha\lambda^d\text{reward}$$

The weighting on the reward is equal to that on the rest non-reward based terms when $d = \dfrac{Log[1/2]}{Log[\lambda]}$.

A graph of the equal weight threshold d can be seen in Figure 19.

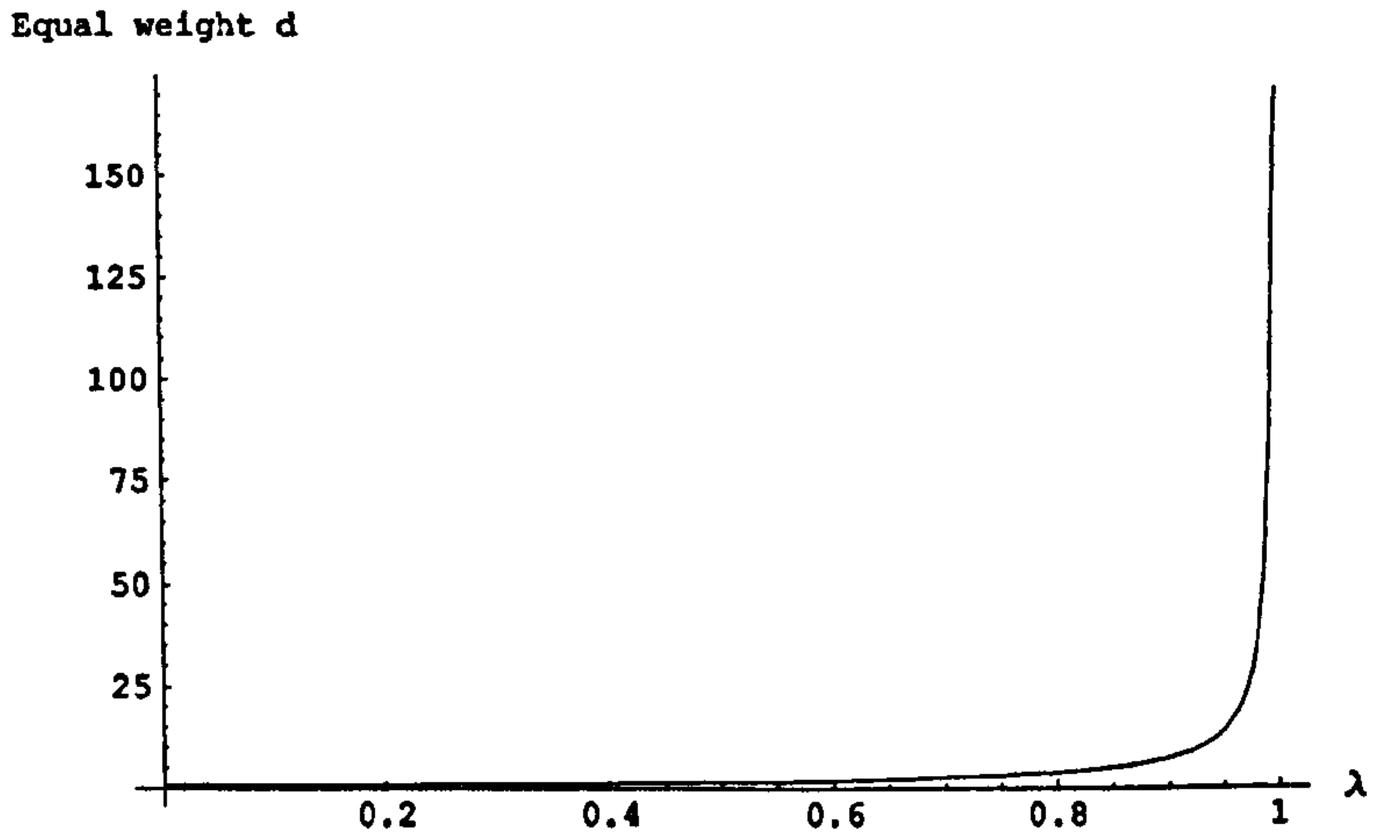


Figure 19: Equal weight threshold for a range of λ

The conclusion is that as λ increases the TD estimator puts increasing weight on the final reward and becomes more like the WR estimator.

Thus there are two reasons why WR can be regarded as a small sample approximation to TD. The first applies when the non-reward terms are close to zero and the second when λ is close to 1. Of course when the non-reward terms are small and λ is close to 1 then both effects tend to make TD close to WR.

To summarise:

- The TD estimator should tend to the WR estimator as $\lambda \to 1$ irrespective of the initial estimate. (TD=WR=MCA for λ=1.)
- The TD estimate is close to WR if the current estimates of reward are small.
- In particular if the initial estimate of reward is zero then the TD and WR estimators are equal at the first update for all λ and remain close for λ near 1.

It therefore seems reasonable to test the hypothesis that the small sample advantages shown by TD are in fact due to the inclusion of weighted reward component. That is, where TD shows a small sample advantage this is due to it being like the WR estimator. Given the complexity of the TD estimator the best that can be done is to demonstrate for a reasonable range of models that when TD does show a small sample advantage that WR also shows the same behaviour and advantage.

Large sample behaviour

It is clear that while TD may be approximated by WR in small samples this cannot remain so as the sample size increases. The WR estimator converges to $E[\lambda^d r]$ and TD converges to E[r] and they are likely to have different asymptotic errors. However, the effect of the inclusion of a weighted average of non-reward terms can be seen as a way of changing the effective λ on the weighted reward as the sampling proceeds so as $\underset{\lambda \to 1}{E[\lambda^d r]} \to E[r]$.

An intuitive explanation of how the inclusion of the non-reward elements changes the effective λ in the weighted reward mechanism is easy to construct. As the estimates of other states move towards their expected values, the values of the non-reward terms move towards E[reward], as described in Chapter Four. In this case the TD estimator becomes more like:

$$V_1(s_t) = V_0(s_t)(1-\alpha) + \alpha(1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}E[\text{reward}] + \alpha\lambda^{T-t-1}\text{reward}$$

As the lambda weights sum to 1 this tends to:

$$V_1(s_t) = V_0(s_t)(1-\alpha) + \alpha((1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1} + \alpha\lambda^{T-t-1})\text{reward} \to$$

$$V_1(s_t) = V_0(s_t)(1-\alpha) + \alpha\,\text{reward}$$

To summarise:

- Initially TD(λ) behaves like WR(λ) but as the sample size increases its behaviour tends to become more like WR(λ`) where $\lambda` \to 1$ as the sample size increases.

  This can be seen as the effect of the "temporal difference" component of TD(λ) which slowly changes the effective value of λ applied to the weighted reward as better estimates become available.

Empirical results

A simulation of the Simple Random Walk (SRW) with 19 states provides evidence that the small sample approximation holds good in at least this case. For example, the RMS error for the first 10 steps of estimation using λ=0.9 and α=0.7 for the first visit TD and WR estimator (10,000 realisations) can be seen in Figure 20.

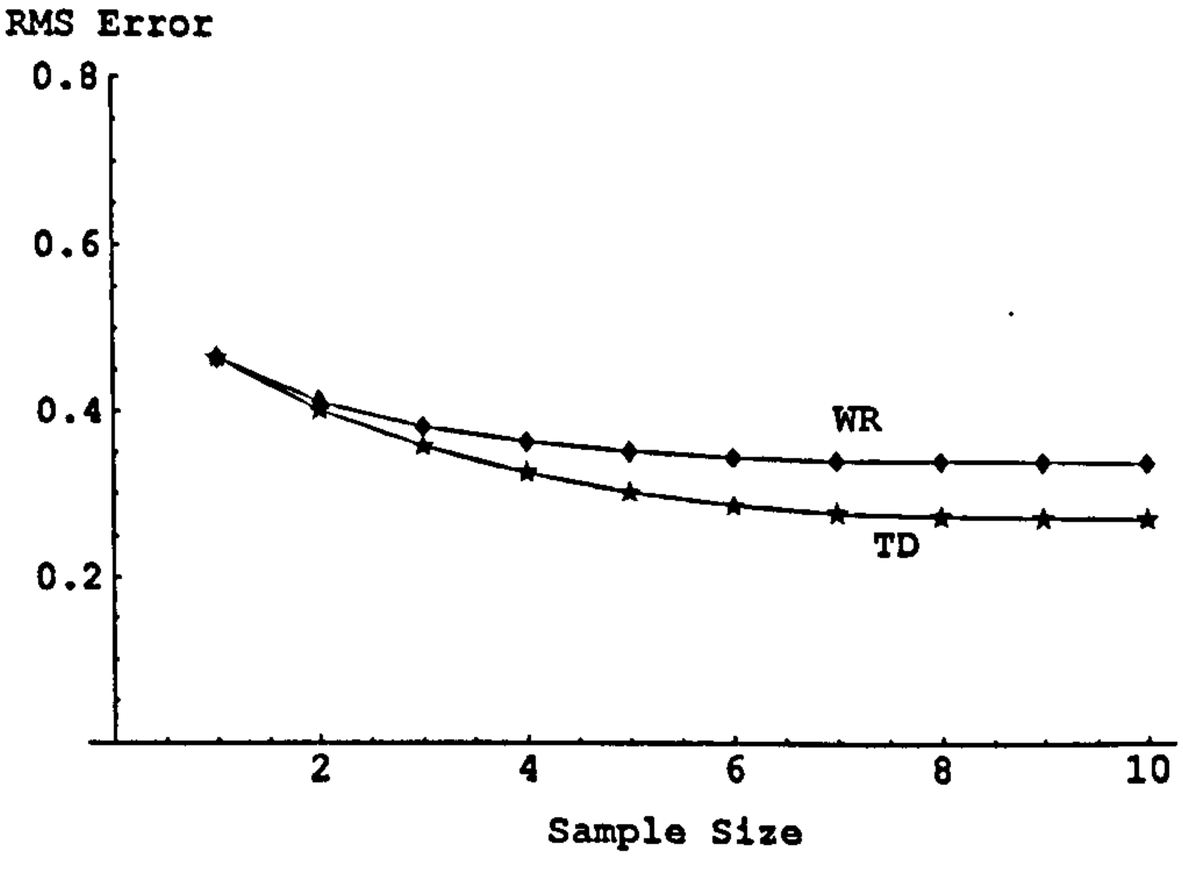


Figure 20: RMS error for TD and WR for λ=0.9 and α=0.7

The RMS errors of TD and WR are typically closer for smaller values of λ and α but the values illustrated correspond to a region of operation where TD performs well and hence is of practical significance. The empirical studies using the SRW strongly suggest that further investigation of the behaviour of the WR estimator is warranted.

Statistics of the weighted reward

The WR(λ) estimator is a biased estimator of the reward because it is a more natural estimator of $\lambda^{d_t} r_T$ where d is the number of steps before the reward is received. The expected value and variance of $\lambda^{d_t} r_T$ have already been derived in Chapter Four using the recursive methods developed there. That is:

$$E[\lambda^d \mathbf{r}] = (I - \lambda S)^{-1} T\mathbf{r} = Q(\lambda)T\mathbf{r}$$

and

$$Var(\lambda^d \mathbf{r}) = Q(\lambda)T\mathbf{r}^2 - (Q(\lambda)T\mathbf{r})^2$$

where $Q(\lambda) = (I - \lambda S)^{-1}$.

Plotting these expressions reveals why the WR(λ) estimator may well have a small sample advantage when $\lambda < 1$.

For λ=1 it has the same expectation as the value function and as λ reduces the expectations move away from the value function, less so for those states near the terminal states.

This can be seen in Figure 21, where λ=1 gives the true (population) value function.

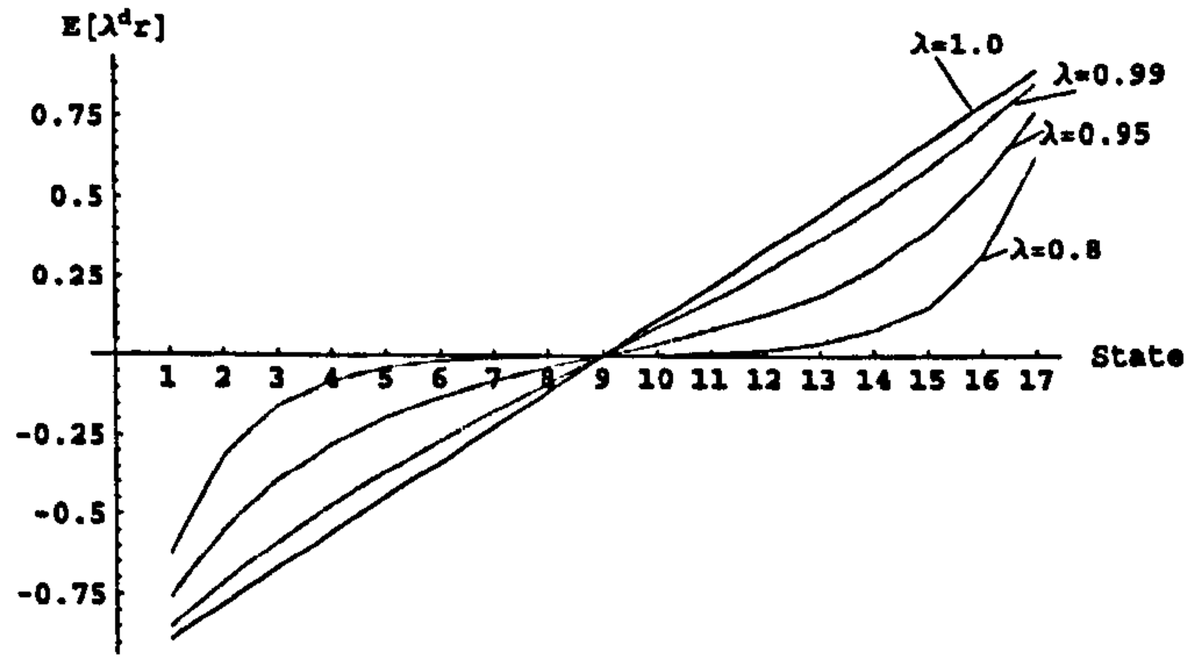


Figure 21: Expected value of λ^d r for each state for four values of λ

A corresponding chart for the RMS error of $\lambda^{d_i} r_T$ about its mean, $RMS_{E[\lambda^{d_i} r]}[\lambda^{d_i} r]$, (i.e. the standard deviation), see Figure 22, indicates clearly why the estimator has a lower RMS error than the unbiased estimator WR(1).

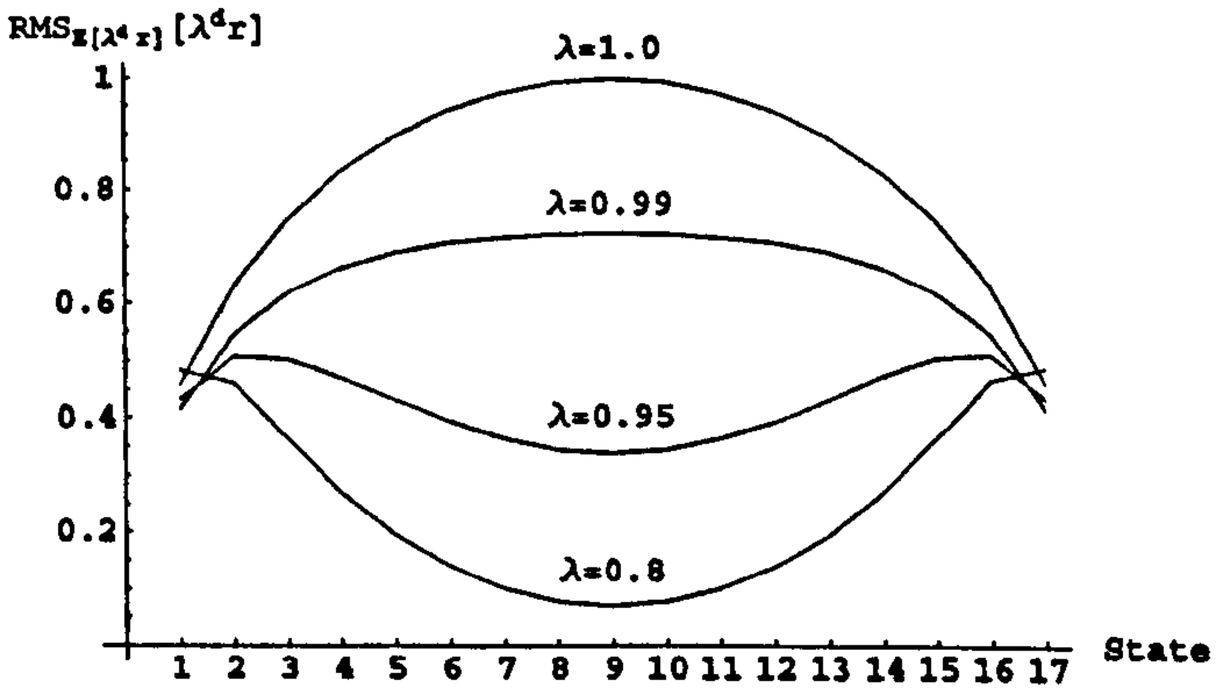


Figure 22: RMS error of λ^d r for each state for four values of λ

As λ decreases the variance of $\lambda^{T-t-1} r$ also decreases. Another way of viewing the advantage that a weighted reward gives is to plot the RMS deviation from E[r] i.e. the RMS deviation from the expected reward $RMS_{E[r]}[\lambda^{d_i} r]$, as shown in Figure 23.

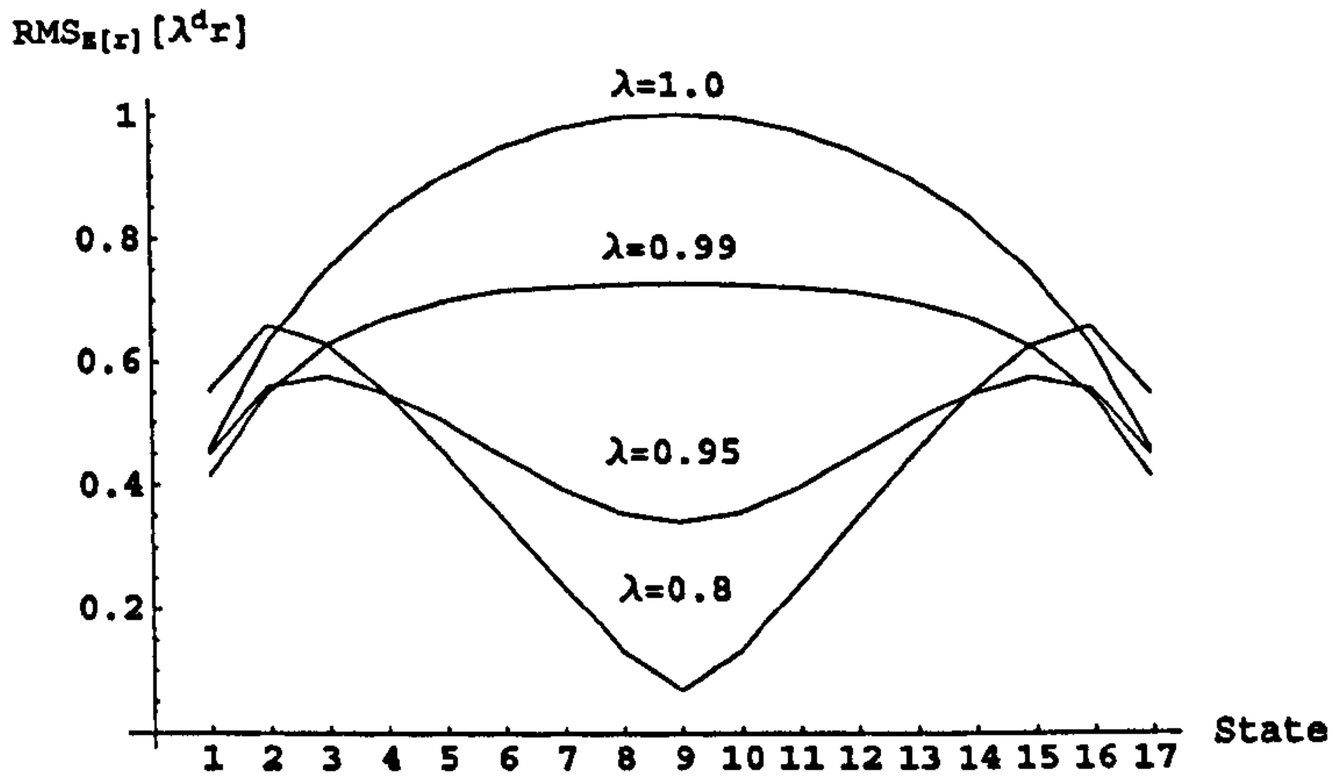


Figure 23: RMS error of λ^d for each state for four values of λ

The lower RMS deviation for "middle" states can clearly be seen in this chart. However, it should also be recalled that estimators such as the alpha update rule can reduce the variance component of the quantity being estimated but not the bias.

Figure 24, shows the per state average of $RMS_{E[\lambda^{d_i} r]}[\lambda^{d_i} r]$. In the case of the RMS error of WR considered as an estimator of the reward it makes sense to normalise by the average per state RMS of the reward. This yields a figure that shows the decrease in RMS value for values of λ less than 1. This is what is used in Figure 25 and it can be seen that the RMS value falls to 60% of its original value. Interestingly there is also an optimum value of λ at approximately 0.95. This type of normalised diagram is a useful tool in comparing different types of model because it provides an indication of the way a weighted reward estimator should behave.

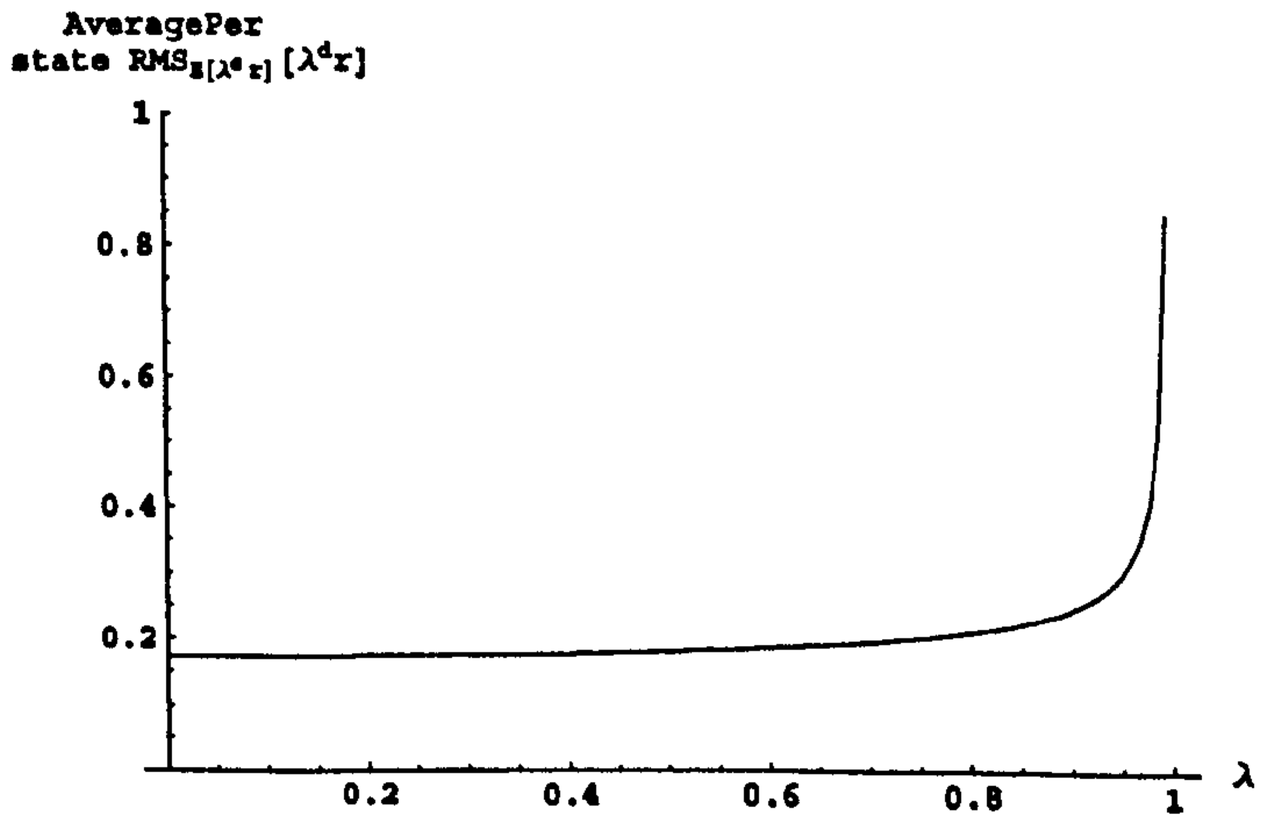


Figure 24: Average per state $RMS_{E[\lambda^{d_i} r]}[\lambda^{d_i} r]$ error of λ^d r for values of λ

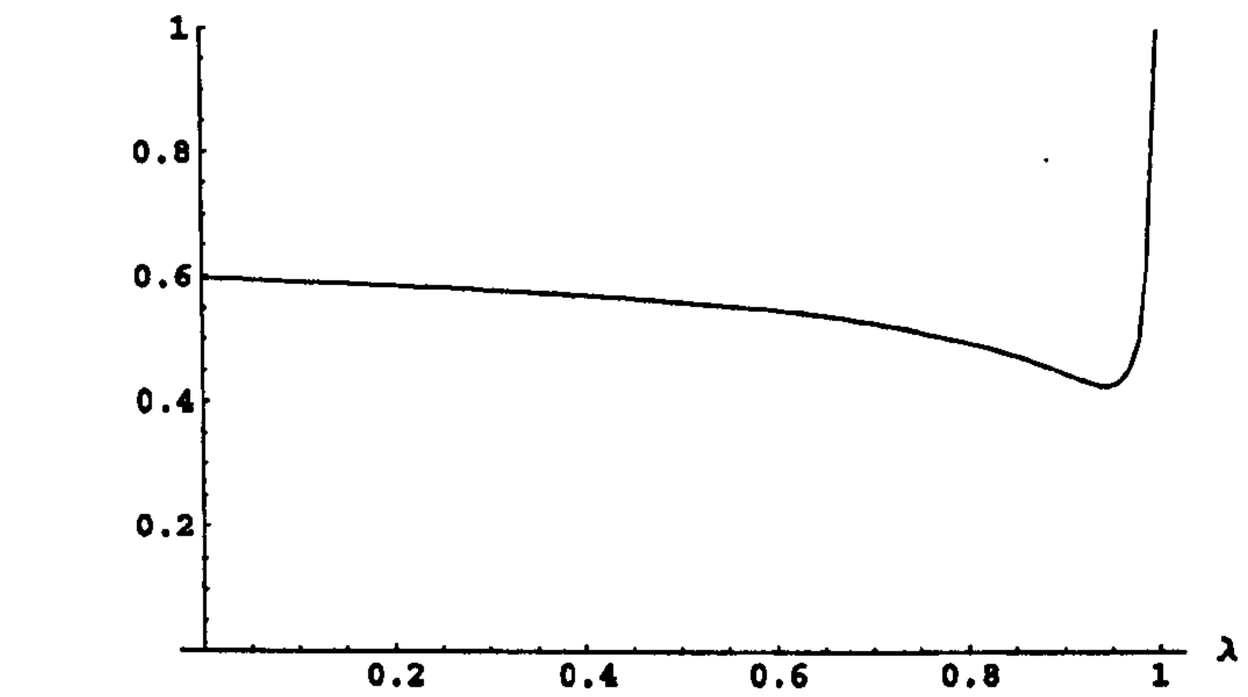


Figure 25: Normalised average per state $RMS_{E[r]}[\lambda^{d_i} r]$

Analytic RMS curves

As the WR estimator takes the form of a simple alpha update rule its expected value and variance can be easily computed following the methods developed in Chapters Four and Six but allowing for the difference in the expectation of $\lambda^{T+1}r$ and r. The corrected RMS error for a sample size n for a given α and λ is:

$$RMS^2_{E[w_i]}[WR_n(s_i)] = \frac{1-\{(1-\alpha f_i(2-\alpha)\}^n}{(2-\alpha)}\alpha RMS^2_{E[w_i]}[w_i] + \{(1-\alpha f_i(2-\alpha)\}^n RMS^2_{E[w_i]}[WR_0(s_i)]$$

However the RMS operators are all evaluated with respect to $E[w_i]$ and as W is a biased estimator we need to correct this to allow for the fact that $E[w_i] \neq E[v_i]$.
To correct the RMS error for the bias we need to subtract $(E[WR_n]-E[w])^2$ and add $(E[WR_n]-E[r])^2$ using:

$$E[WR_n(s_i)] = (1-\alpha f_i)^n\{WR_0(s_i) - E[w_i]\} + E[w_i]$$

$$RMS^2_{E[r_i]}[WR_n(s_i)] = \frac{1-\{(1-\alpha f_i(2-\alpha)\}^n}{(2-\alpha)}\alpha VAR[w_i] + \{(1-\alpha f_i(2-\alpha)\}^n (WR_0(s_i) - E[w_i])^2 - (E[WR_n(s_i)] - E[w_i])^2 + (E[WR_n(s_i)] - E[r_i])^2$$

Notice that as both $VAR[w_i]$ and $E[w_i]$ depend on λ all of the above expressions are functions of λ as well as α.

As $WR_n(s_i)$ is asymptotically unbiased as an estimator of $E[w_i]$, i.e. $WR_n(s_i) \underset{n\to\infty}{\to} E[w_i]$, we have:

$$RMS^2[WR_\infty(s_i)] = \frac{\alpha}{(2-\alpha)} VAR[w_i] + (E[w_i] - E[r_i])^2$$

which represents the asymptotic RMS error for $W_n(s_i)$ estimating $E[r_i]$.

From this it can be seen that $WR(\lambda)$ is asymptotically biased by $(E[w_i]-E[r_i])^2$ even if α is reduced to zero. Of course as $E[w_i] \underset{\lambda\to 1}{\to} E[r_i]$, $WR(\lambda)$ is asymptotically unbiased in a scheme where $\alpha \to 0$ and $\lambda \to 1$ with sample size.

Empirical v Theoretical RMS

The formulae developed in the previous section can be used to predict the RMS error for any state in any MDP. For example for state 1 in the 17-state linear MDP the theoretical estimate compares well with empirical values (20,000 trials) for α=0.2 and λ=0.5 which is typical, see Figure 26.

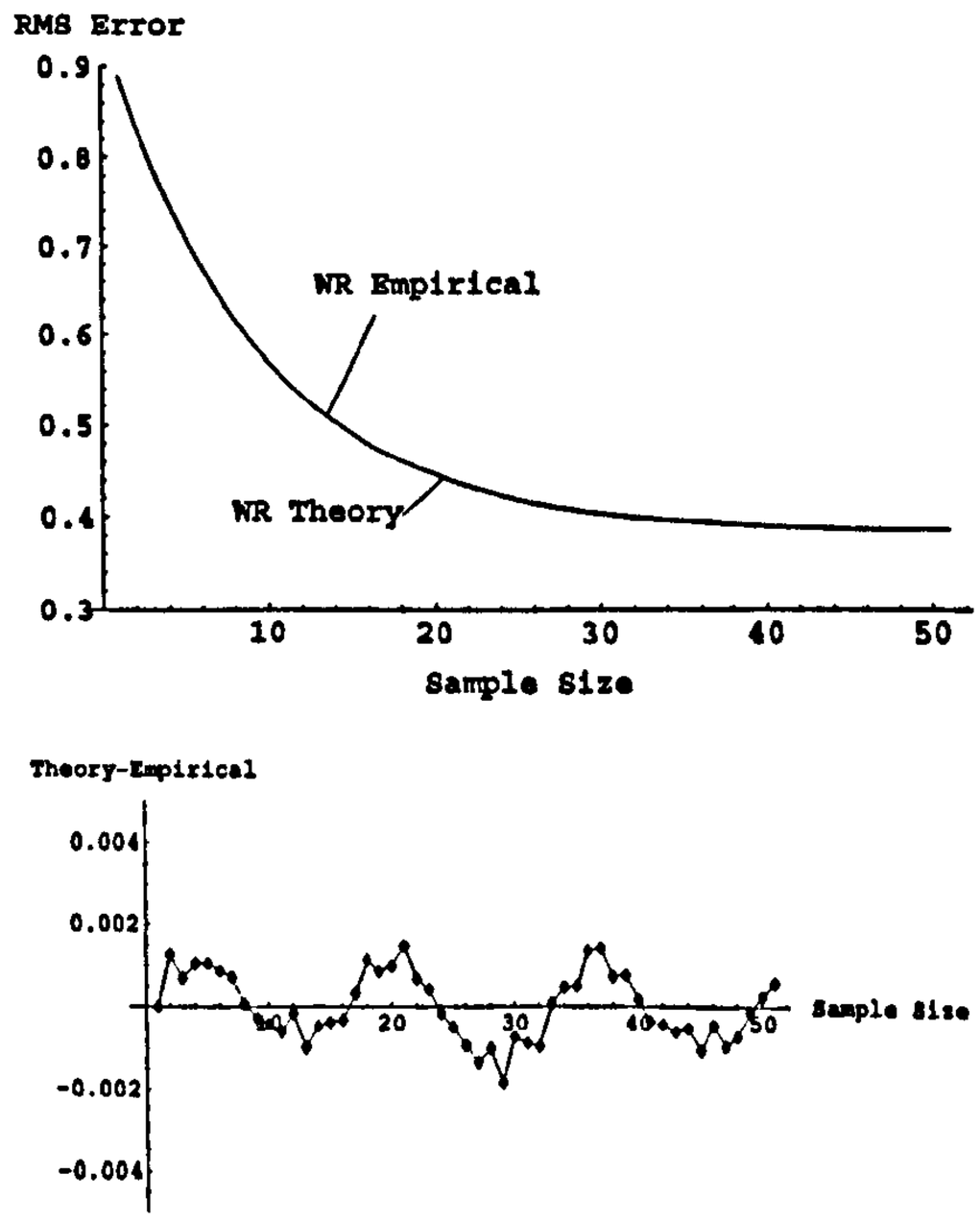


Figure 26: Empirical v theoretical RMS for State 1, α=0.2 and λ=0.5

The agreement between the theoretical predictions for average RMS per state is also excellent. For example for λ=0.9 and α=0.7 the empirical (20,000 trials) and predicted RMS curves are very close, as shown in Figure 27.

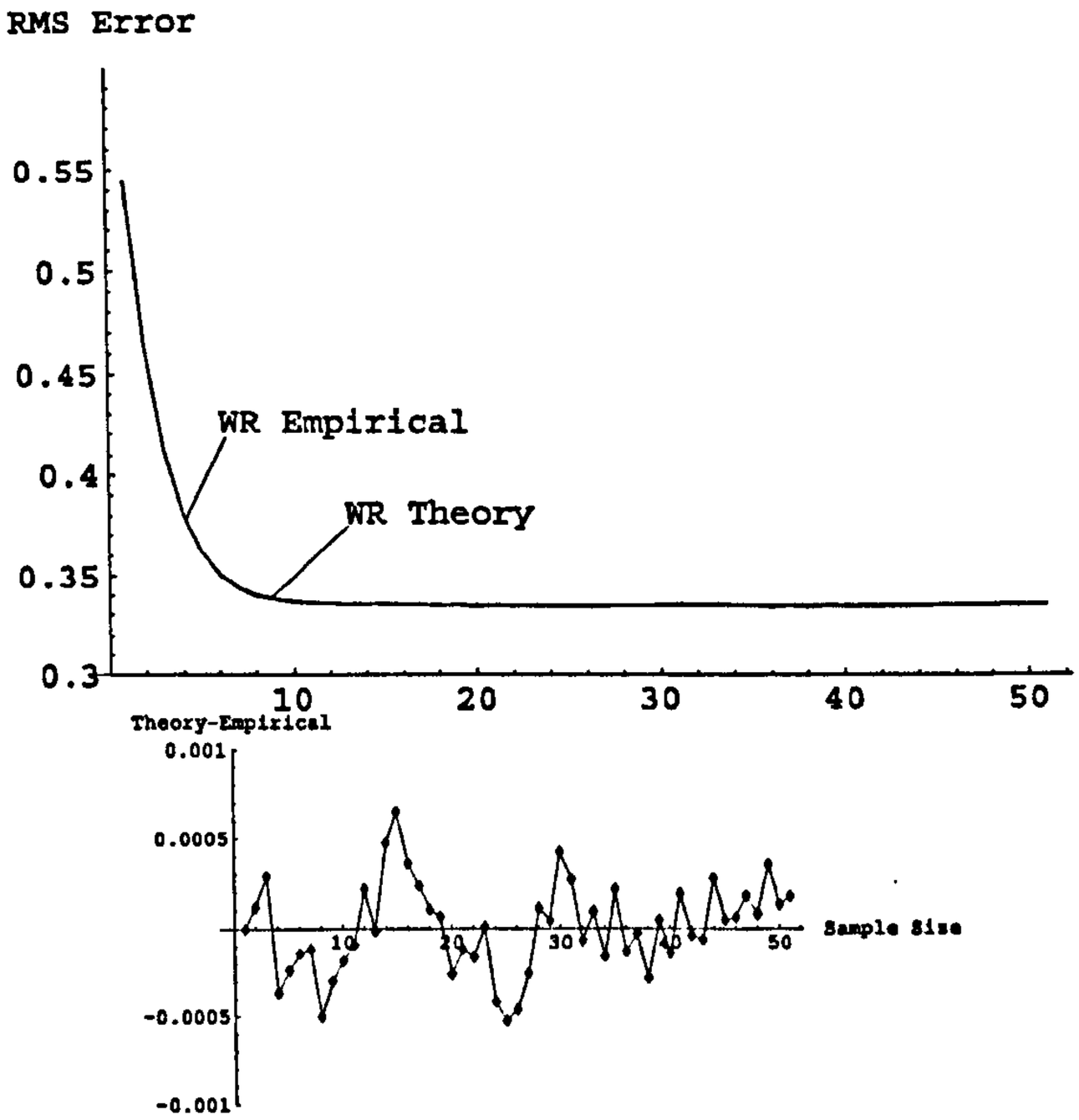


Figure 27: Empirical v theoretical Average RMS per state for α=0.7, λ=0.5

Comparing WR(λ) with TD(λ) using the SRW

The WR(λ) estimator can be compared to the equivalent TD(λ) estimator in an effort to see how much of the early learning performance can be attributed to its action. This was achieved using analytic formula for First TD derived and implemented as C programs by Singh and Dayan (1998). The program was modified to compute an un-weighted RMS error and was integrated into a Mathematica notebook which set up the model matrices and computed the RMS error of the WR estimator using the formulae given earlier. First TD is used throughout this chapter; the behaviour of the other forms of the TD estimator is considered in Chapter Nine.

The first comparison to be made is the average RMS error in the first ten steps of the SRW since this is often presented as an illustration of the small sample superiority of TD over other estimators (Sutton, 1984; Singh. and Dayan, 1998.).

The average RMS error for a range of λ and α for the first visit WR estimator produced the result shown in Figure 28 which should be compared to the equivalent chart for TD(λ) in Figure 29. As can be clearly seen the behaviour of the WR(λ) estimator is very similar to TD(λ) and is, of course, identical for λ=1 where the W(1) and TD(1) are both equivalent to the Every visit MCA update rule.

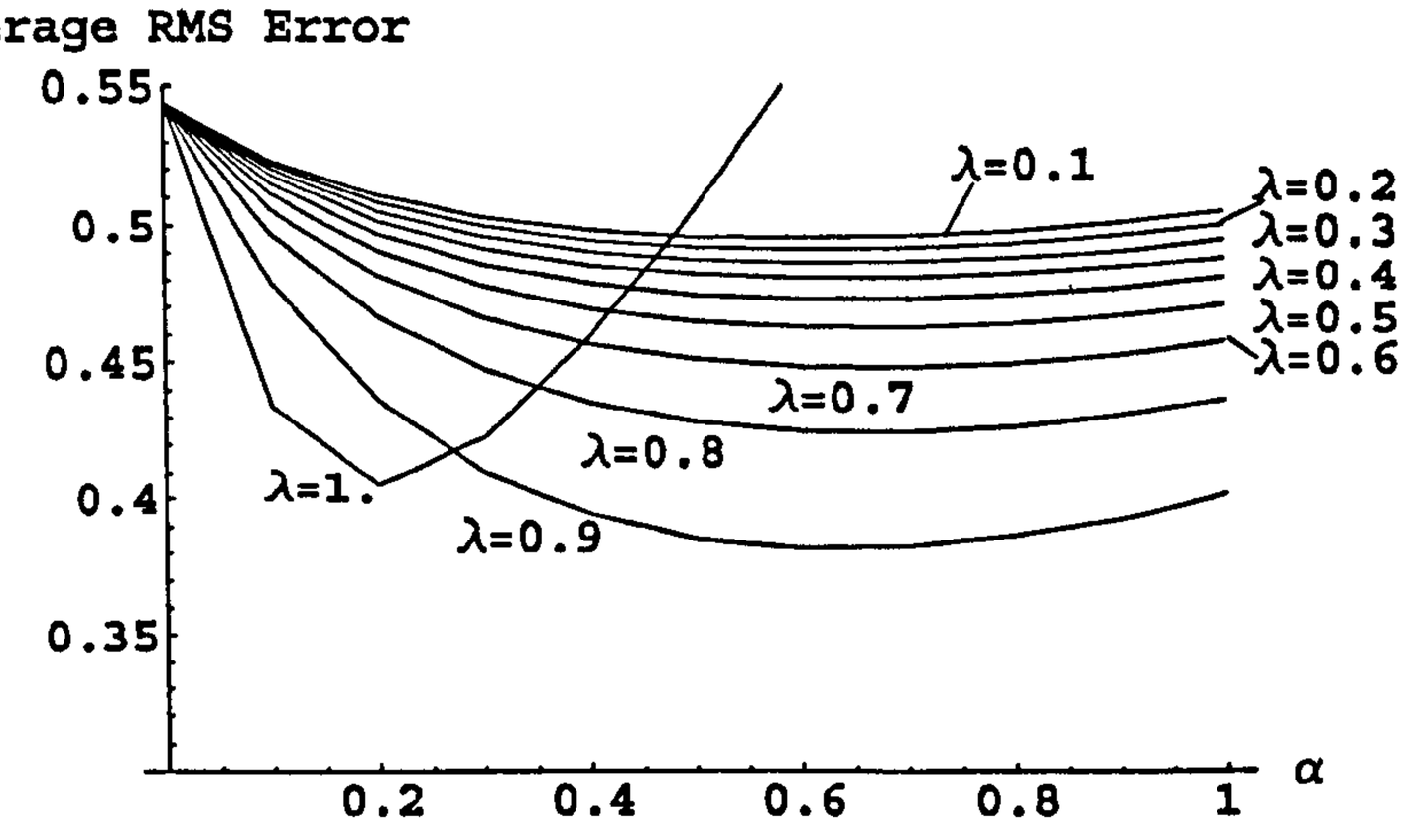


Figure 28: RMS error for a range of λ and α - WR(λ)

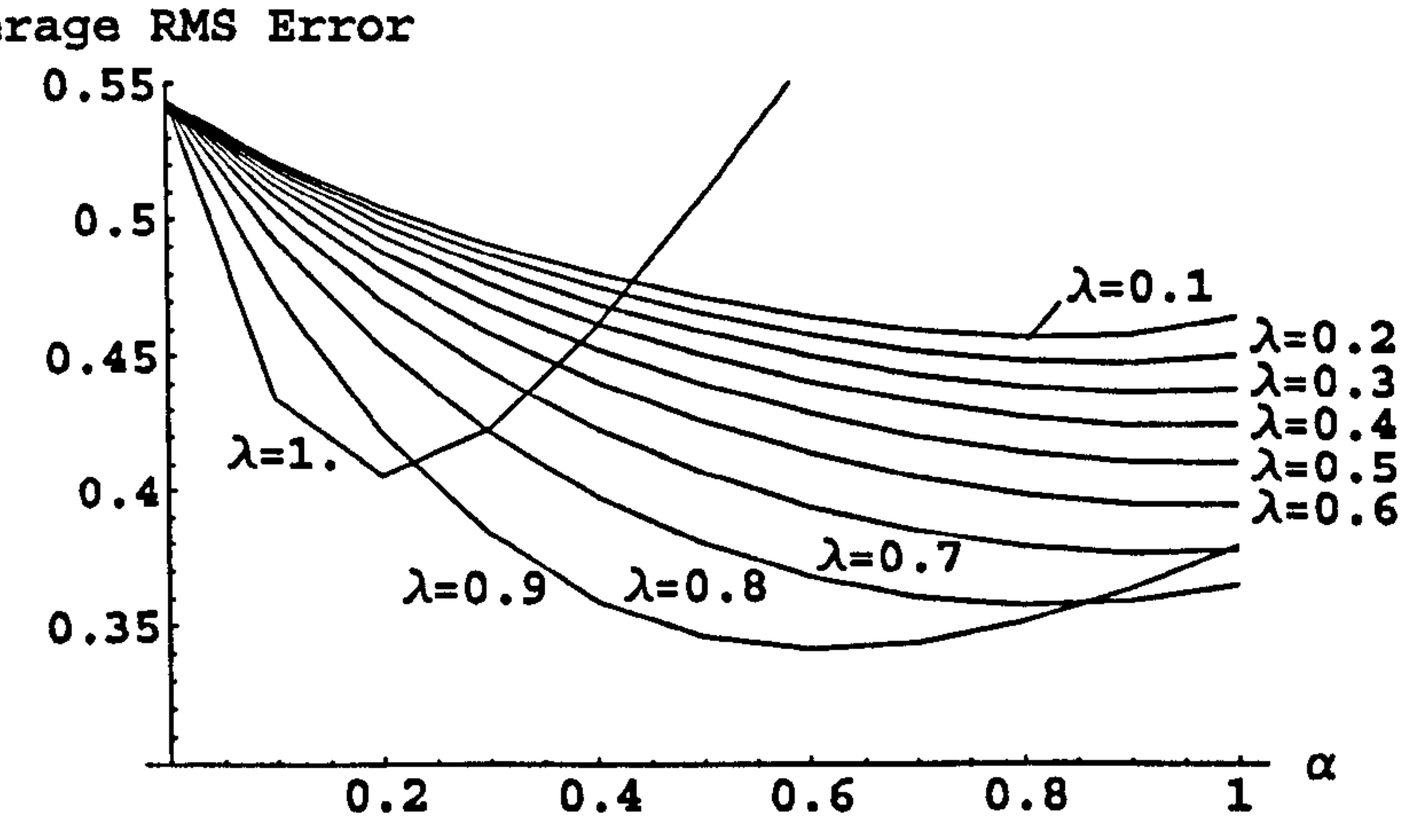
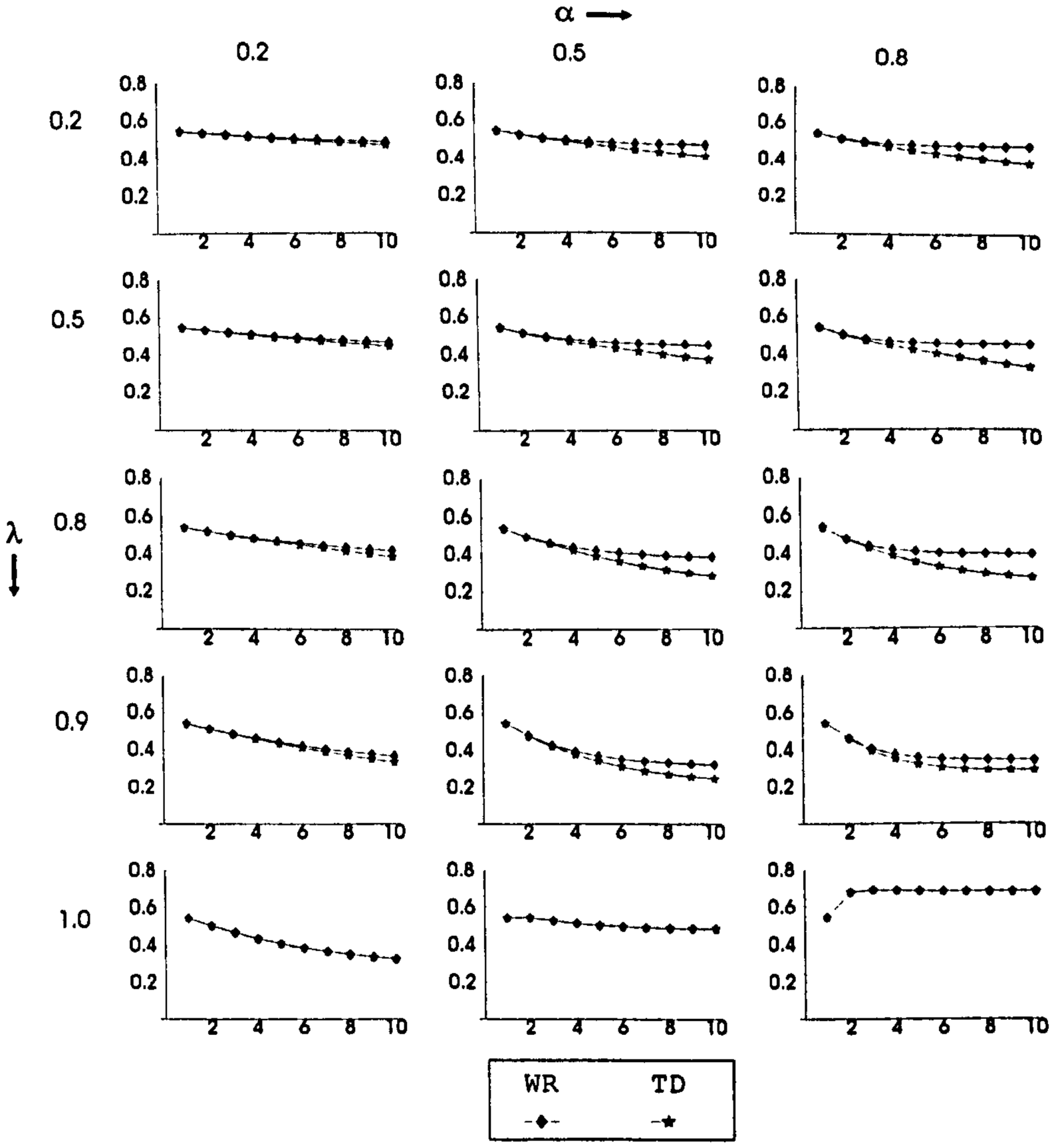


Figure 29: RMS error for a range of λ and α -TD(λ)

Comparing the actual RMS error curves over the first 10 steps of the estimator, see Figure 30, for a range of λ and α reveals the general behaviour of the two estimators.



NB: vertical axis is the average per-state RMS error horizontal axis is sample size

Figure 30: Comparison of WR(λ) with TD(λ) over first 10 steps

The graphs confirm that the difference between the WR and TD estimators tends to increase with α and the sample size for small λ. This is probably due to the fact that WR is asymptotically biased and for small λ this bias, $(E[w_i] - E[r_i])^2$, is large and reached in only a few samples. After this WR cannot reduce its RMS error but TD, being asymptotically unbiased can, and its asymptotic RMS decreases as α decreases.

The behaviour with λ is more complex but can be best summarised as being an increase in difference with sample size as λ increases and then a decrease for λ sufficiently large to shift the weights to the reward. For values of λ that do better than MCA for small samples, the WR estimator is close to the TD estimator and shows the same small sample advantage.

Large sample behaviour

Examining the behaviour of the estimators over more steps clearly reveals the importance of the bias in WR. Figure 31 shows RMS curves for TD, WR, the best MCA estimates and the difference TD-WR for a typical λ (λ=0.9 and α=0.8) that produces a good small sample performance.

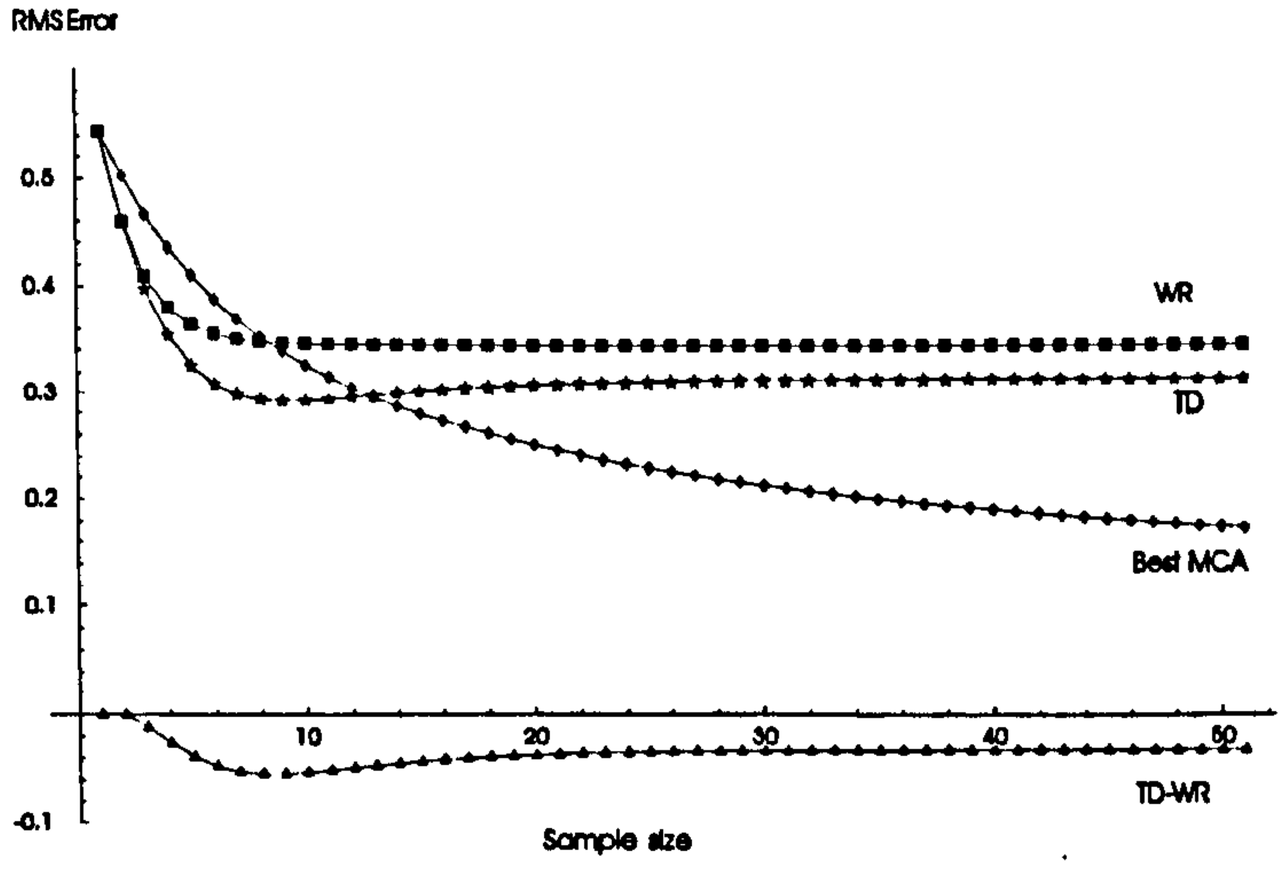


Figure 31: TD and WR for λ=0.9 and α=0.8

It can be seen that the WR estimator is initially close to the TD estimator until they converge to their respective asymptotic errors. This effect becomes more pronounced with smaller λ. This is shown in Figure 32, for a value of λ=0.7 and α=0.8, where the TD estimator first starts to show a small advantage over the best MCA estimator.

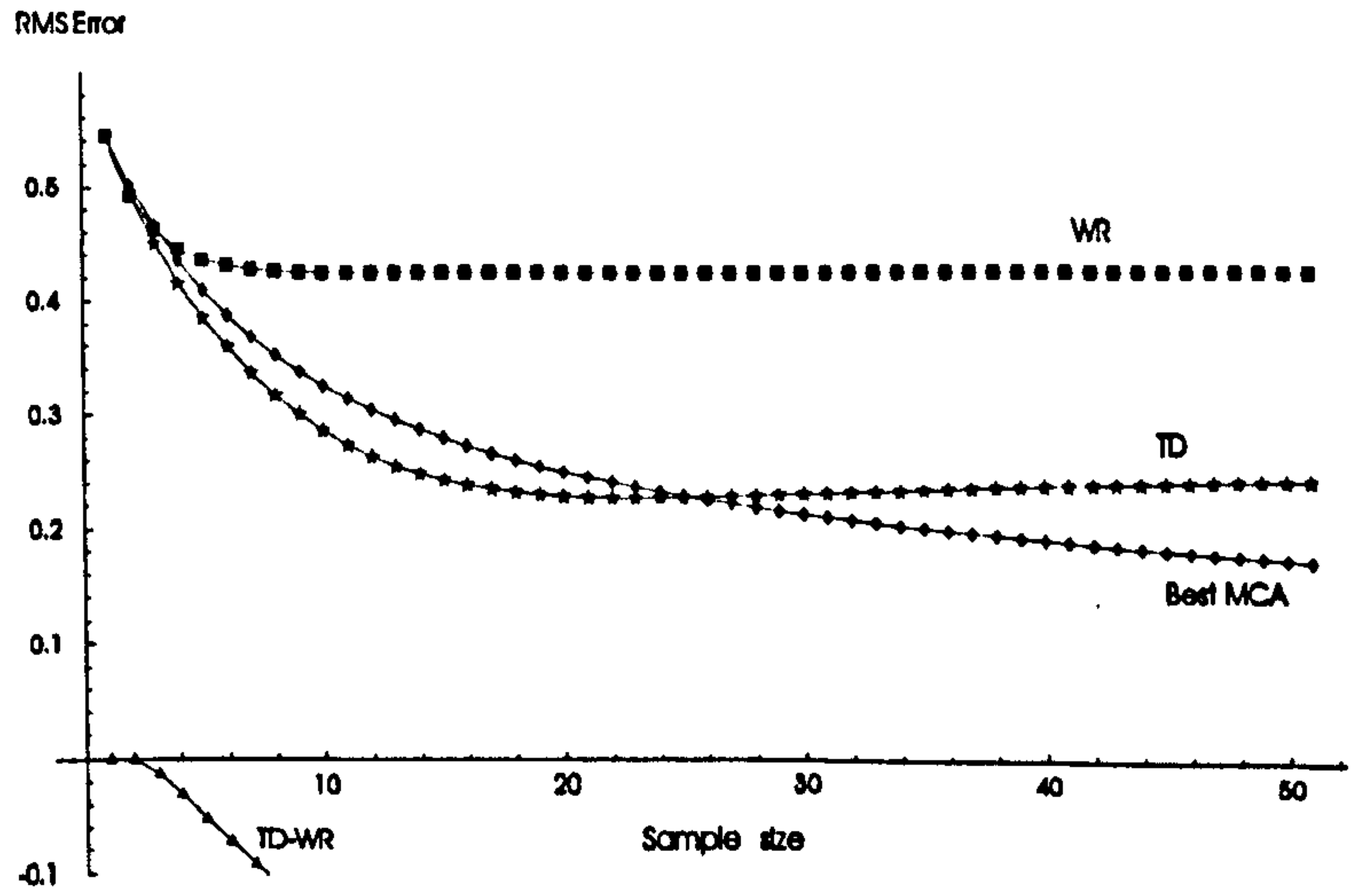


Figure 32: TD and WR for λ=0.7 and α=0.8

It is clear that the contributions from the non-reward elements in the TD estimator act to reduce the bias as described earlier. That is, as the sample size increases the non-reward terms in the TD estimator start to make the it behave like WR with λ tending to one.

This raises the idea of comparing the TD estimator to the WR estimator with the same asymptotic error achieved by increasing the value of λ selected. This can be achieved by solving:

$$RMS^2[TD_\infty(\lambda)] = RMS^2[WR_\infty(\lambda')]$$

for λ' using the value of asymptotic error computed for the TD estimator. However, for small values of λ and large (fixed) α there is no WR estimator with the same α and asymptotic error. That is, it is impossible to find a λ' that gives the same asymptotic error for the same α but it is possible to find a set of λ' α' which give the same asymptotic error. The further raises the question of which λ' α' values to choose to compare with the TD estimator? A reasonable approach is to find the λ' α' that gives the best fit to the TD RMS error curve. For example in the case of TD using λ=0.9 and α=0.8 the best fitting WR estimator uses λ'=0.96 and α'=0.79. As can be seen in Figure 33, the fit is good.

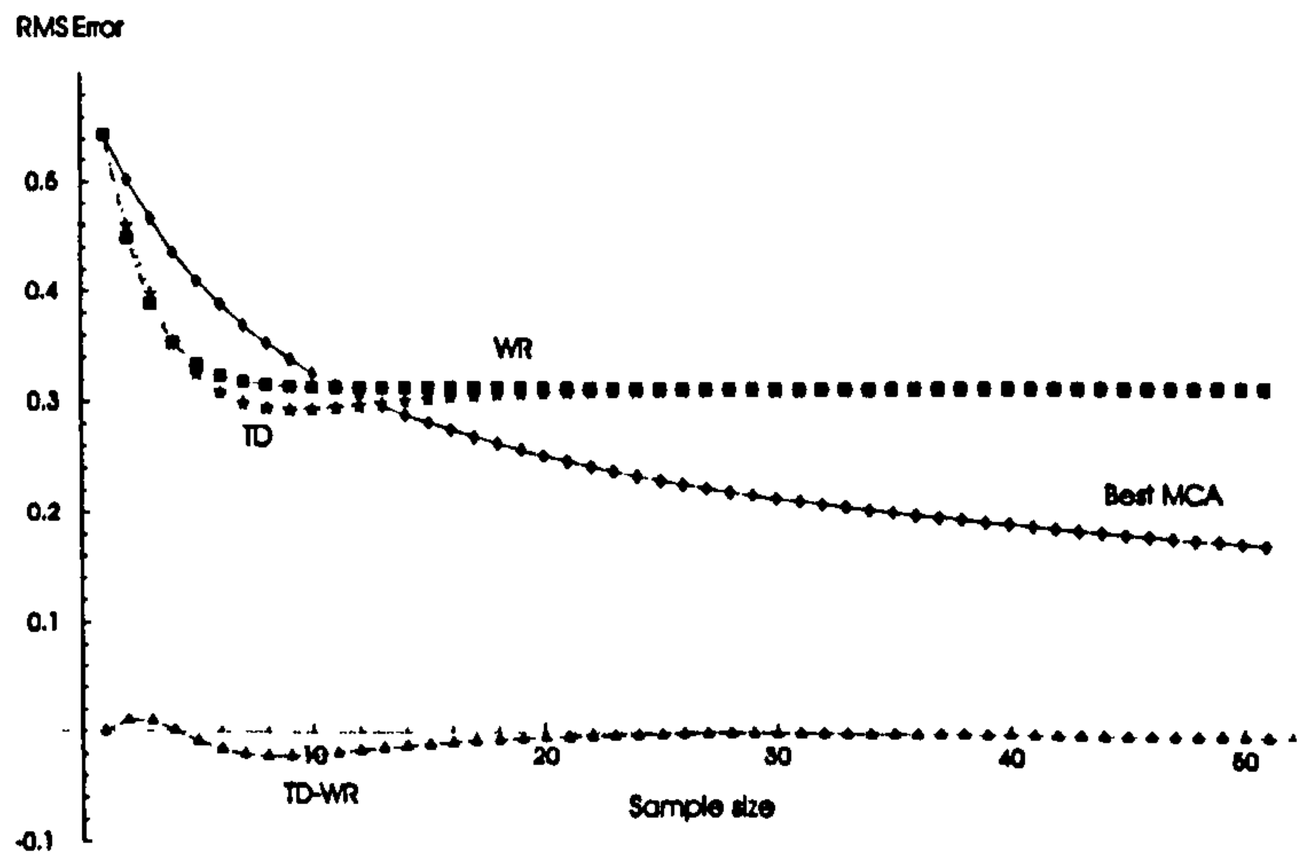


Figure 33: TD for λ=0.9 and α=0.8 and WR for λ'=0.96 and α'=0.79

For smaller values of λ the best fitting values of λ' α' move further away from the TD values but the fit of the RMS curves remains good. For example, for TD using

λ=0.7 and α=0.8 the best fitting WR estimator uses λ'=0.94 and α'=0.31.In this case the larger value of λ is needed to reduce the asymptotic bias but again the fit is good, as can be seen in Figure 34.

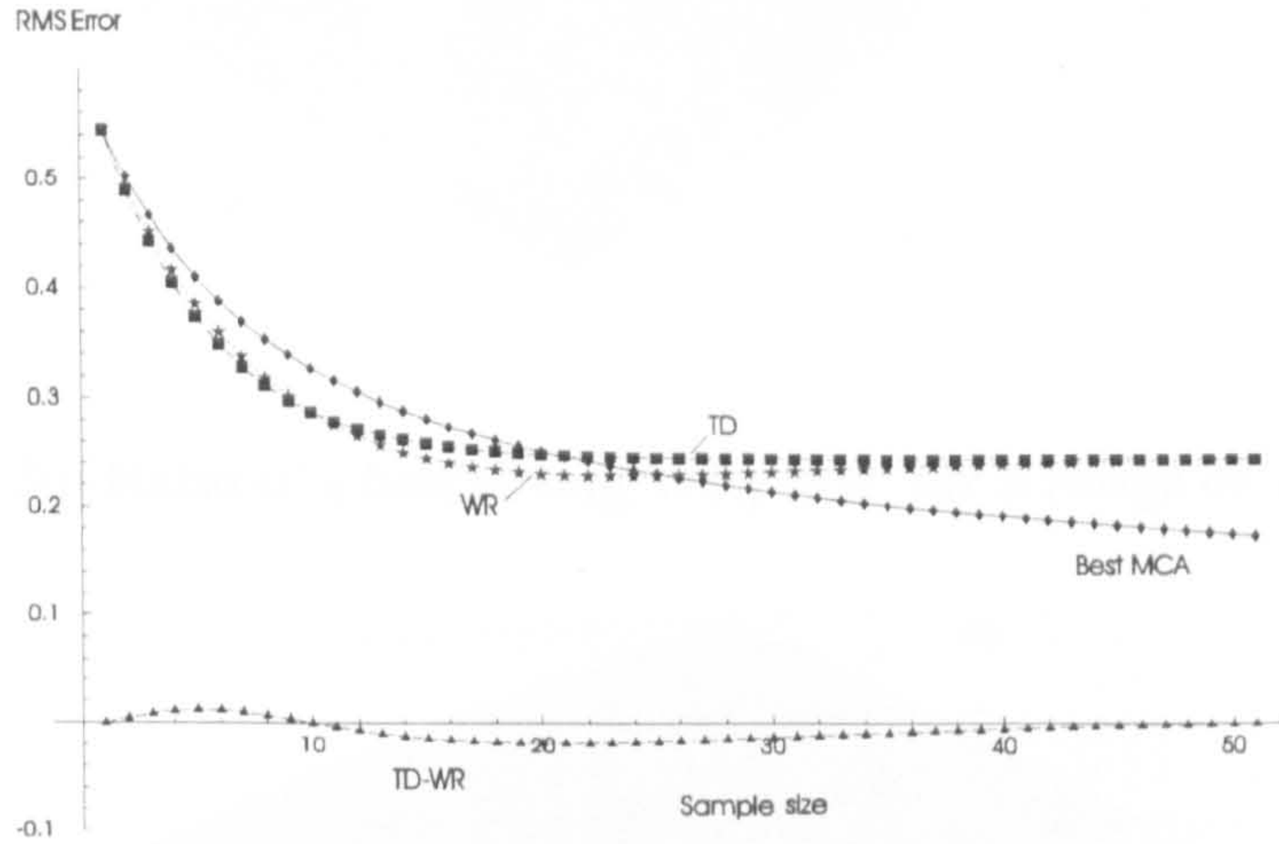


Figure 34: TD for λ=0.7 and α=0.8and WR for λ'=0.94 and α'=0.31

The RMS error between the two estimators over a sample size of 50 can be seen to be small for the full range of λ and α in Figure 35.

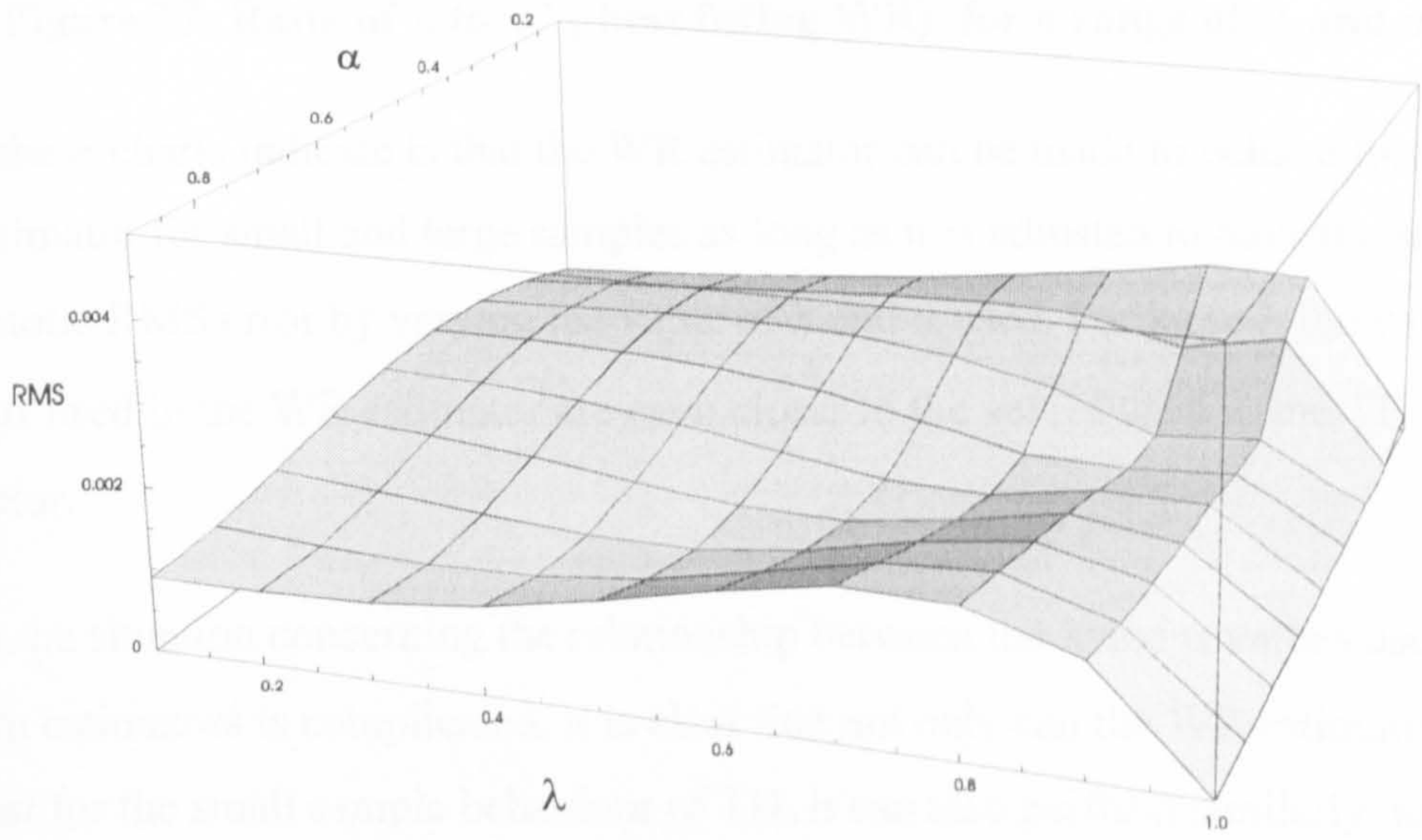


Figure 35: RMS error between TD and the best fitting WR

The ratio of the best α' to the α used in the TD estimator can be seen in Figure 36 and the ratio of the λ used in the TD estimator to the best λ' can be seen in Figure 37.

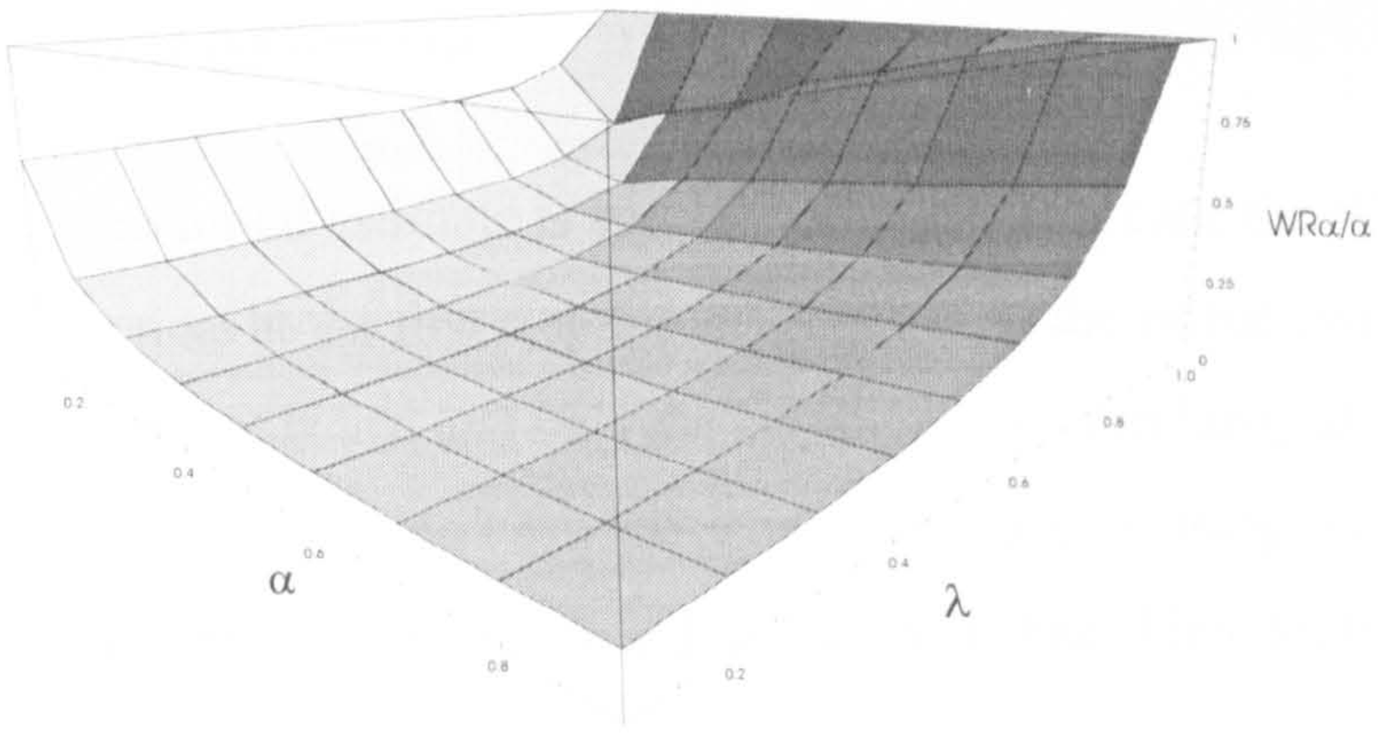


Figure 36: Ratio α' (best fitting WR) to α for a range of λ and α

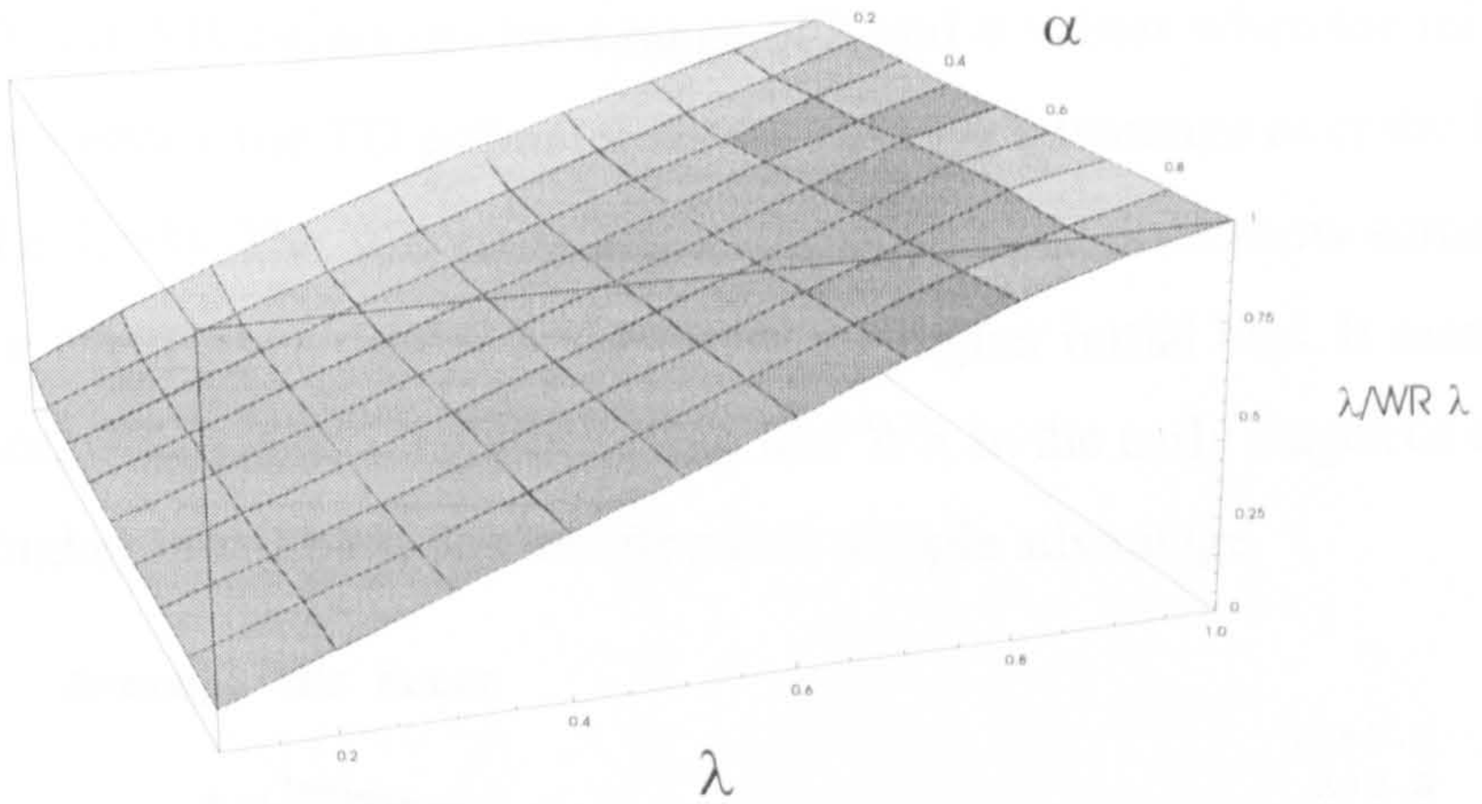


Figure 37: Ratio of λ to λ' (best fitting WR) for a range of λ and α

What these charts indicate is that the WR estimator can be made to behave like the TD estimator for small and large samples as long as it is adjusted to have the same asymptotic RMS error by varying the value of λ and α used. For large λ the values of λ and α used in the WR estimator are even closer to the values used in the TD estimator.

While the situation concerning the relationship between the λ and α values used in the two estimators is complicated, it is clear that not only can the WR estimator account for the small sample behaviour of TD, it can also perform similarly, to the point where both estimators have settled down to their asymptotic error rate. This suggests that there is no need to postulate any other mechanism than the use of weighted rewards in the action of the TD estimator. The WR estimator, at least in this case, seems to be capable of extracting information as efficiently as the TD estimator.

Non-zero initial estimate

The case of the zero initial estimate is special because in this case the WR and TD estimators are identical at the first step for any λ and α. If the initial estimate is non-zero then this agreement is lost and only the arguments for similarity at large values of λ apply. It is worth examining what happens in this case by using the SRW model with an initial estimate of expected reward of 1 at each state. This doubles the initial RMS bias.

As can be seen in Figure 38, which shows the average RMS error over ten steps for both the TD and WR estimators for a range of λ and α values when the initial estimate is non-zero, the TD estimator shows no early advantage over the MCA estimator (i.e. $\lambda = 1$). The WR estimator on the other hand does show some improved performance when given time to make up for the higher initial bias. It seems that the TD estimator, which is no longer behaving like WR in the early stages of estimation due to the higher initial bias, has lost its small sample advantage.

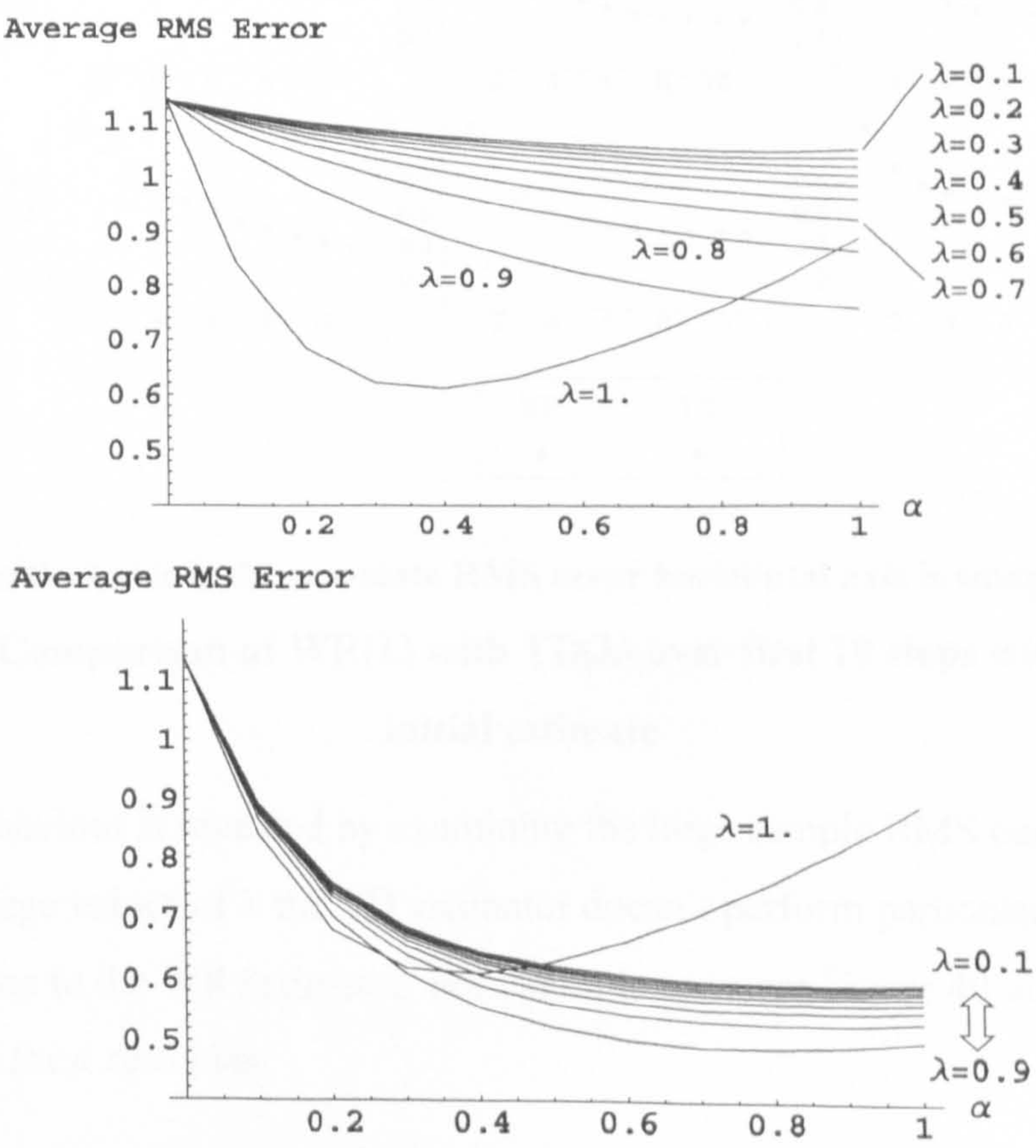
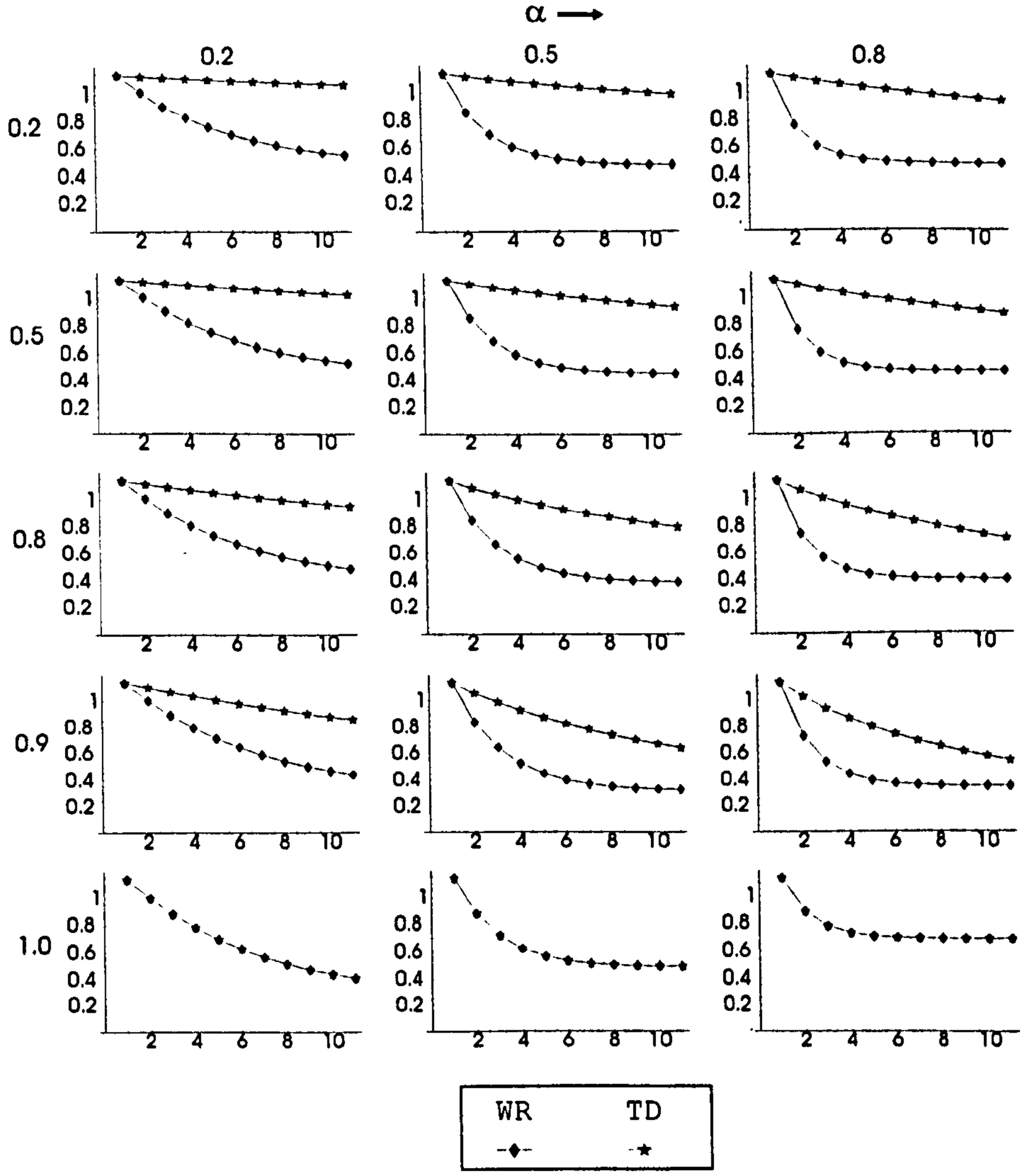


Figure 38: Average RMS error for TD (upper graph) and WR (lower graph) using a non-zero initial estimate

The same situation is revealed by the individual RMS error curves for the first ten steps in Figure 39. As can be clearly seen TD gets closer to WR as λ tends to 1 but it is a worse approximation than in the case of a zero initial estimate.



NB: vertical axis is the per-state RMS error horizontal axis is sample size

Figure 39: Comparison of WR(λ) with TD(λ) over first 10 steps with non-zero initial estimate

The same behaviour is revealed by examining the large sample RMS curve. Even for reasonably large values of λ the TD estimator doesn't perform particularly well and isn't very close to the WR estimator. For example compare Figure 40 with the previous one for a zero bias.

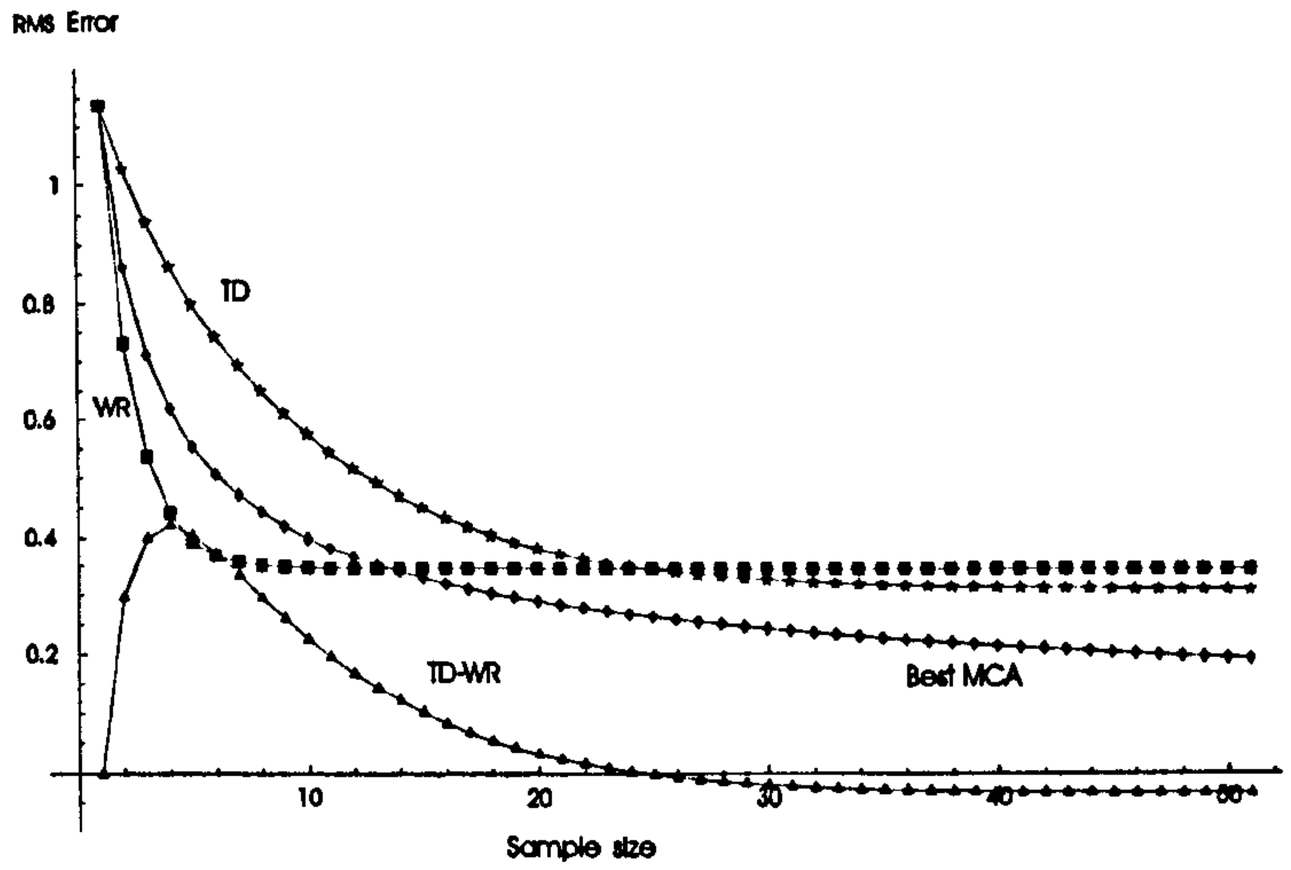


Figure 40: Non-zero initial estimate TD and WR for λ=0.9 and α=0.8

As λ increases, however, the behaviour does slowly converge to that displayed by WR. See Figure 41 for λ=0.95.

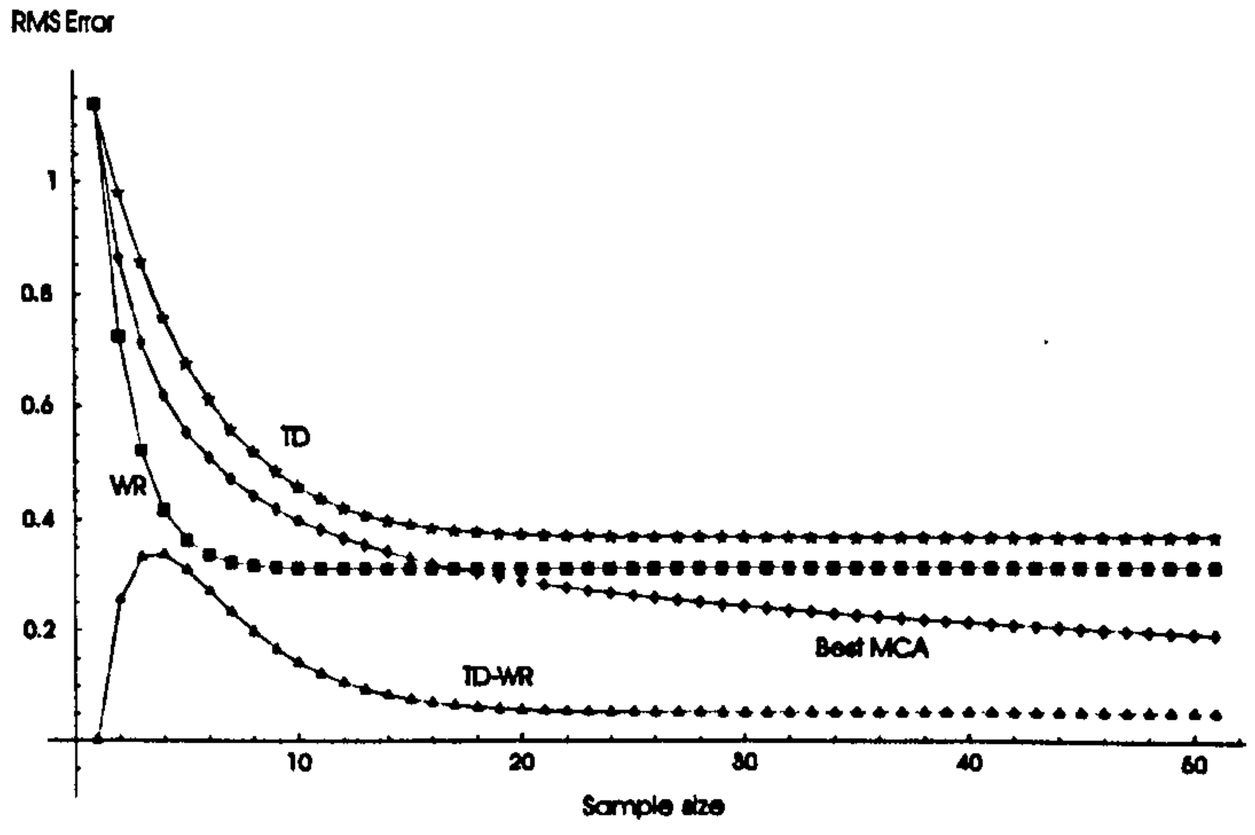


Figure 41: Non-zero initial estimate TD and WR for λ=0.95 and α=0.8

The conclusion is that the non-reward elements in the TD estimator damage rather than enhance its small sample performance when the initial estimate isn't zero. They do this by taking it away from the WR estimator which continues to perform well in its own right.

Weighted assignment versus temporal difference

The performance of the WR(λ) estimator compared to TD(λ) indicates that the superiority of the small sample performance of TD(λ) doesn't rely on temporal difference philosophy but can be accounted for by a simple and classical variance

reduction technique. By trading variance for bias, the WR(λ) estimator with λ<1 performs better earlier than the simple MCA estimator. Of course, given that the WR(λ) estimator is asymptotically biased, it isn't equivalent to TD(λ) as the sample size increases because TD(λ) is asymptotically unbiased. The way that the inclusion of the non-reward terms in TD(λ) removes the bias can be seen as the action of temporal difference but this does not seem to contribute to its small sample behaviour. In every case of the SRW where TD has shown a small sample advantage then it has been close to the WR estimator. Even in the large sample case it has been possible to match the behaviour of TD by a suitably adjusted WR estimator.

If TD(λ) has a "philosophy" is it possible to ascribe a "philosophy" to WR(λ)? The basic idea behind WR(λ) is that the reward is weighted by its "distance" from the state that produced it. The nature of the weighting λ^d , where d is the number of states encountered before the terminal reward, may seem an arbitrary choice but it is proportional to the probability of the realisation from the state to the terminal state. That is, if the probability of the transition from the state at time t to the state at time t+1 is simply $p_{x_t x_{t+1}}$, the probability of the sequence $X=\{x_1,x_2,\ldots x_{n+1}\}$ occurring is, by definition:

$$P(X) = p_{x_1} p_{x_1 x_2} \cdots p_{x_n x_{n+1}}$$

If we assume that the probabilities are constant and equal to λ then:

$$P(X) = p_{x_1} p_{x_1 x_2} \cdots p_{x_n x_{n+1}} = \lambda^n$$

In other words, the λ^n weighting factor can be thought of as an approximation to the probability of the rest of the realisation occurring. Now consider the way that this affects the estimation of reward for a state that is "close" to a terminal state with reward R and a "long way" from a terminal state with reward –r. Clearly the value function for the state will be closer to r than –r and this is reflected in the likely position of the state in a realisation that ends with a reward r and one that ends with a reward –r. By using the position in the realisation to weight the reward it can be seen that we are taking account of the likelihood that the state is "associated" with the reward. By doing this we are not forming an unbiased estimate of the reward but one that over- or under-estimates the value according to the "distance" that the state is from each of the terminal states.

Conclusion

The small sample performance of TD(λ) seems to be mainly due to the effect of it incorporating a weighted reward. In the case of a zero initial estimate TD and WR are equal at the first step. When the initial estimate is non-zero then TD and WR are not identical at the first step but in either case TD becomes increasingly like WR as λ tends to one.

In any situation where TD performs well in a small sample it is close to the WR estimator with the same parameters. This makes good sense and explains how TD can provide improved estimates so early, even when the information in the current estimates is so low. The small sample performance of TD can be accounted for by classical variance reduction principles.

Chapter Eight
WR and TD in Other Models

The behaviour of TD in the SRW model agrees with the hypothesis that its small sample advantage is due to the use of weighted rewards. However the SRW model has a number of special features. This raises the question of the performance and behaviour of both WR and TD in a wider context. Two additional models are analysed to discover if temporal difference learning can provide a small sample advantage over and above the WR estimator.

Factors affecting TD and WR

The SRW is a very special model as far as TD and WR is concerned. First it is a recurrent model in the sense that any non-terminal state can occur any number of times in a realisation. As TD makes use of the current estimate of reward each time a state occurs and WR doesn't, increasing recurrence increases the difference between these two estimators. The SRW with equal left and right probabilities has the longest expected path length and the maximal recurrence of any given state. On average each state occurs 4.7 times in each realisation. In this sense it is already a test of the effect of recurrence on TD and WR. The second important feature is the strong association between expected path length and expected reward – as can be seen in Figure 42.

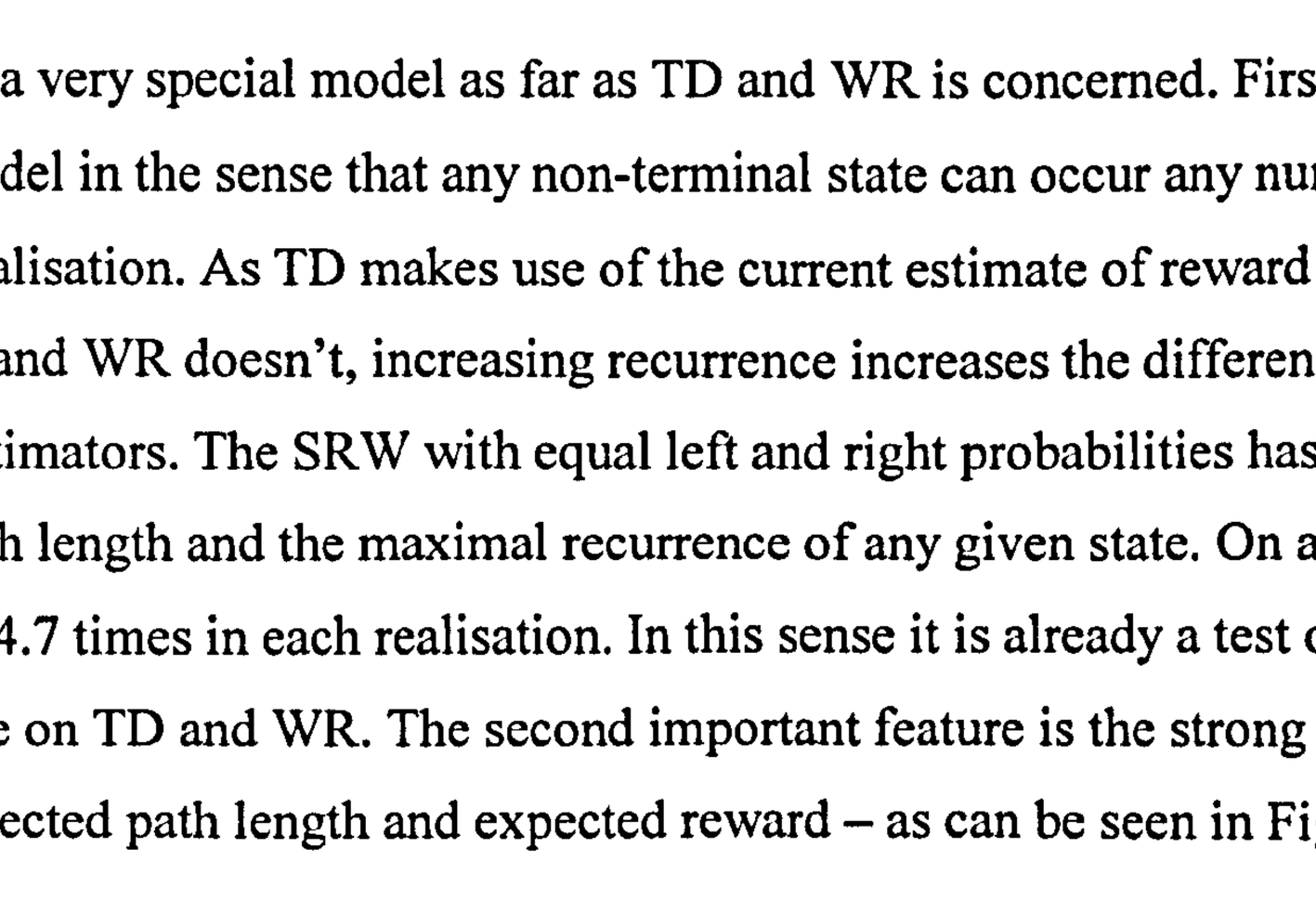


Figure 42: The variation in expected reward with expected path length.

This relationship between expected path length and reward makes the SRW an ideal candidate for a weighted reward estimator to outperform the MCA estimator. This raises the question as to whether or not such relationships are common or very specific to the SRW? As the expected path length is given by:

$$E[path] = Ql$$

where l is a vector of ones.

The expected reward is given by:

$$E[r] = QTr = Q\mathbf{R}$$

where **R** is a vector of the expected reward at each state by direct transition to a terminal state. In both cases Q is the same fundamental matrix and so in general there is likely to be relationship between expected path length and reward. However, even if there isn't a relationship between the means this doesn't rule out a relationship between the actual path length and actual reward received or the variability of the reward received.

It is still worth investigating the behaviour of TD and WR in additional models that differ from SRW in determinism, recurrence and in expected path length/reward relationships. In this chapter two models are explored in detail – the cyclic model and the bottleneck model. The cyclic model is interesting because it has a fixed expected path length and the bottleneck model should provide an opportunity for TD to outperform WR if temporal difference learning provides it with an early advantage. The bottleneck model also offers the opportunity to investigate the effect of changing the level of determinism in the model i.e. the degree of variability of the reward.

A cyclic model

An explicitly cyclic model is that used by Singh and Dayan (1998) to examine the behaviour of the TD estimator. This consists of a ring of states each directly connected to its own terminal state and hence direct reward – see Figure 43.

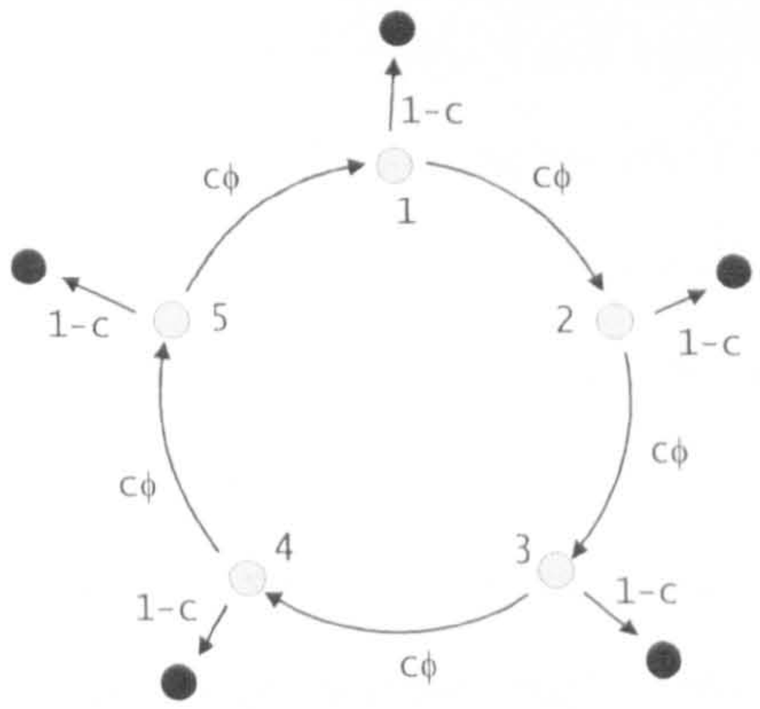


Figure 43: The cyclic model.

Two parameters control the model, c and ϕ. The probability of transition to a terminal state is given by 1-c and so c alone controls the expected path length. As c increases to 1 the expected path length increases as shown in Figure 44 and the expected number of times a state occurs in a realisation is as shown in Figure 45. It is worth pointing out that for n=5 the expected path length only reaches that of the SWR at c>0.99.

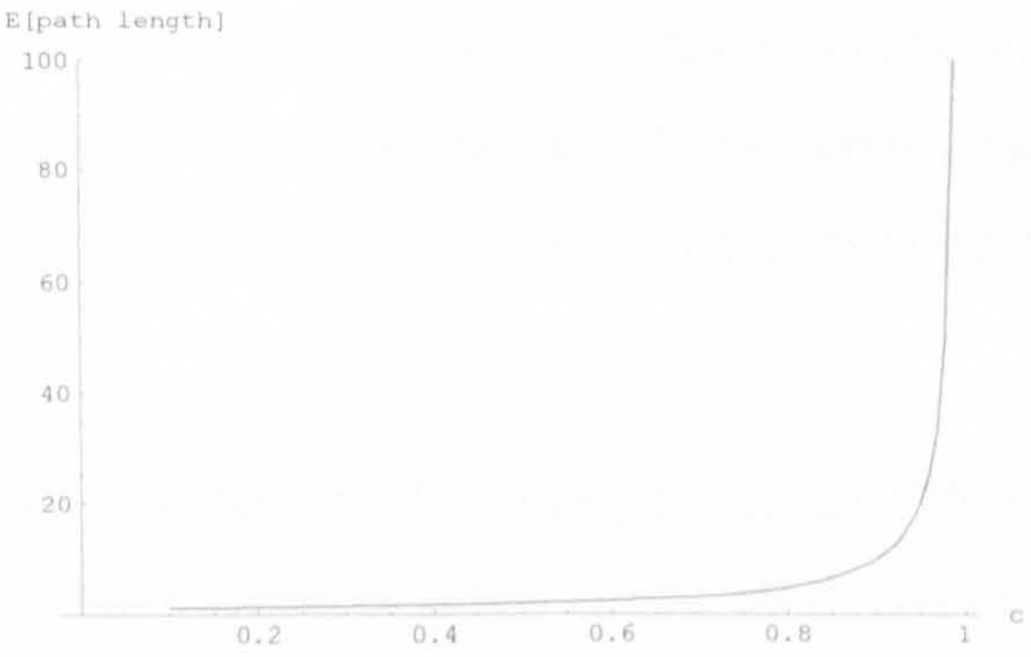


Figure 44: The effect of c on expected path length

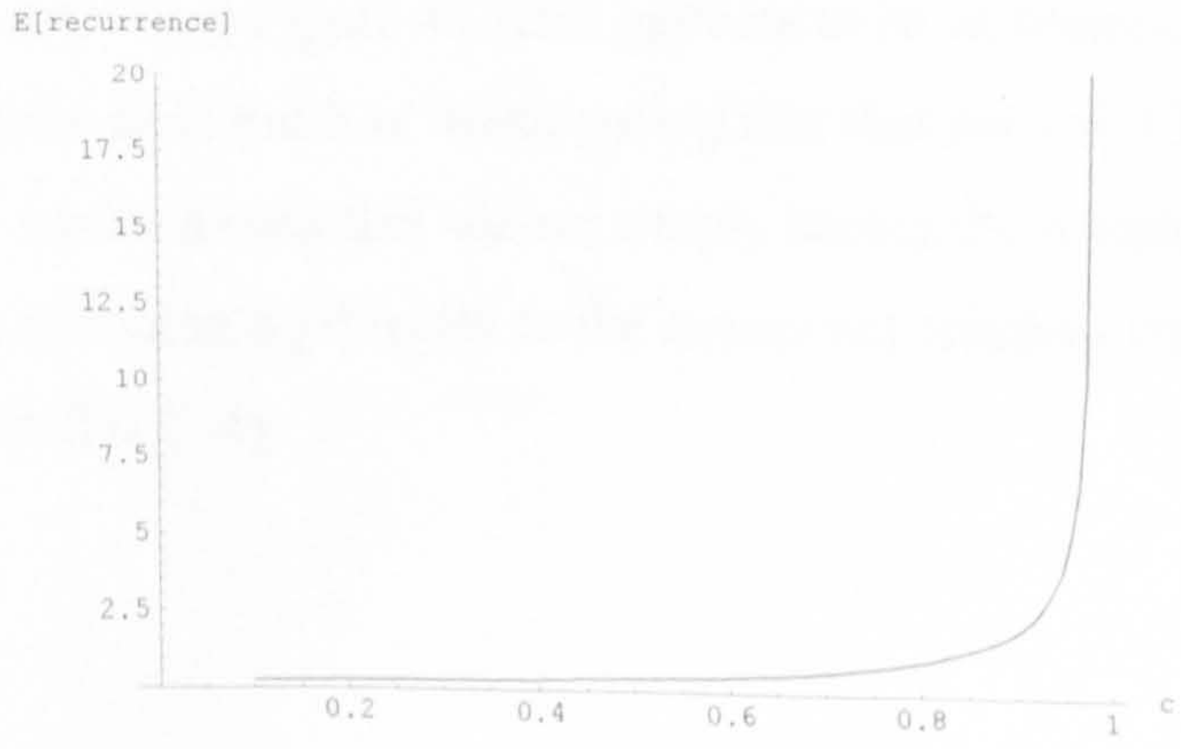


Figure 45: The effect of c on the expected "recurrence" of a state

The ϕ parameter doesn't affect path length or recurrence, instead it controls the "rotation" of the model in the sense that as it increases to 1 the probability that the next move will be to the adjacent state increases. For ϕ less than 1 the remaining probability, i.e. $c(1-\phi)$, is distributed equally among transitions to the other non-terminal states in the ring (transition arrows not drawn in the diagram). Thus for ϕ close to 1 the current state tends to move round cyclically until it terminates. As ϕ reduces towards 0 the cyclic behaviour merges with a disordered jumping about between states.

What makes this model interesting from the point of view of testing TD and WR is that for all c and ϕ the expected path length starting from state i is a constant. In this situation we would expect WR to have little to no advantage over the MCA estimator and if TD has no additional learning abilities then it too would do no better. It is also worth pointing out that this is a very difficult model to learn. The reason is that it is much like a chaotic roulette wheel where the initial position of the ball has little predictive value for the eventual reward. As c is increased the "spin" of the "wheel" becomes more vigorous and the connection between reward and initial state becomes more tenuous. To make the model relevant to practical situations it seems reasonable to set c to a value that keeps a connection between initial state and reward.

The rewards used in the original paper (Singh and Dayan 1998) are 2i-n-1 where i is the state number and n the number of states. (n=5). With this choice the expected reward at each state reduces towards zero as c tends to one. For example, with c=0.1 and ϕ=1 (although ϕ has little effect on the expected reward) the expected reward is different at each state – see Figure 46. This appears to be an interesting model and a good test of the estimators but it is worth noting that that for c=0.1 the expected path length is only 0.2 which means that we are simply seeing the rewards obtained by starting in a state and jumping directly to the connected terminal state (for five states the rewards are –4,-2,0,2,-4).

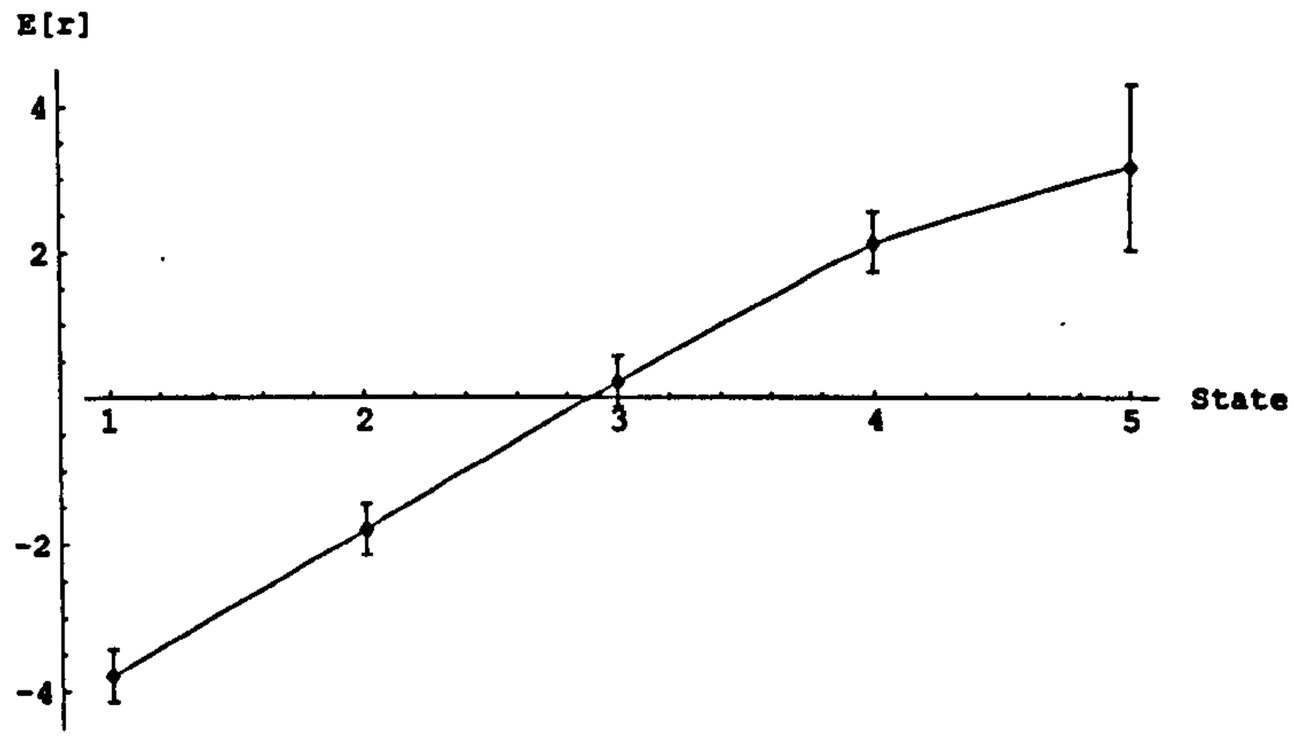


Figure 46: The expected reward at each state for c=0.1 with RMS error bars

If c is increased to 0.5, as in Figure 47, the effect is to reduce the expected reward towards zero and to increase the variability of reward received (as indicated by the larger RMS error bars). In this case the expected path length is just 2 states and so the model is still much simpler than the 19 state SRW.

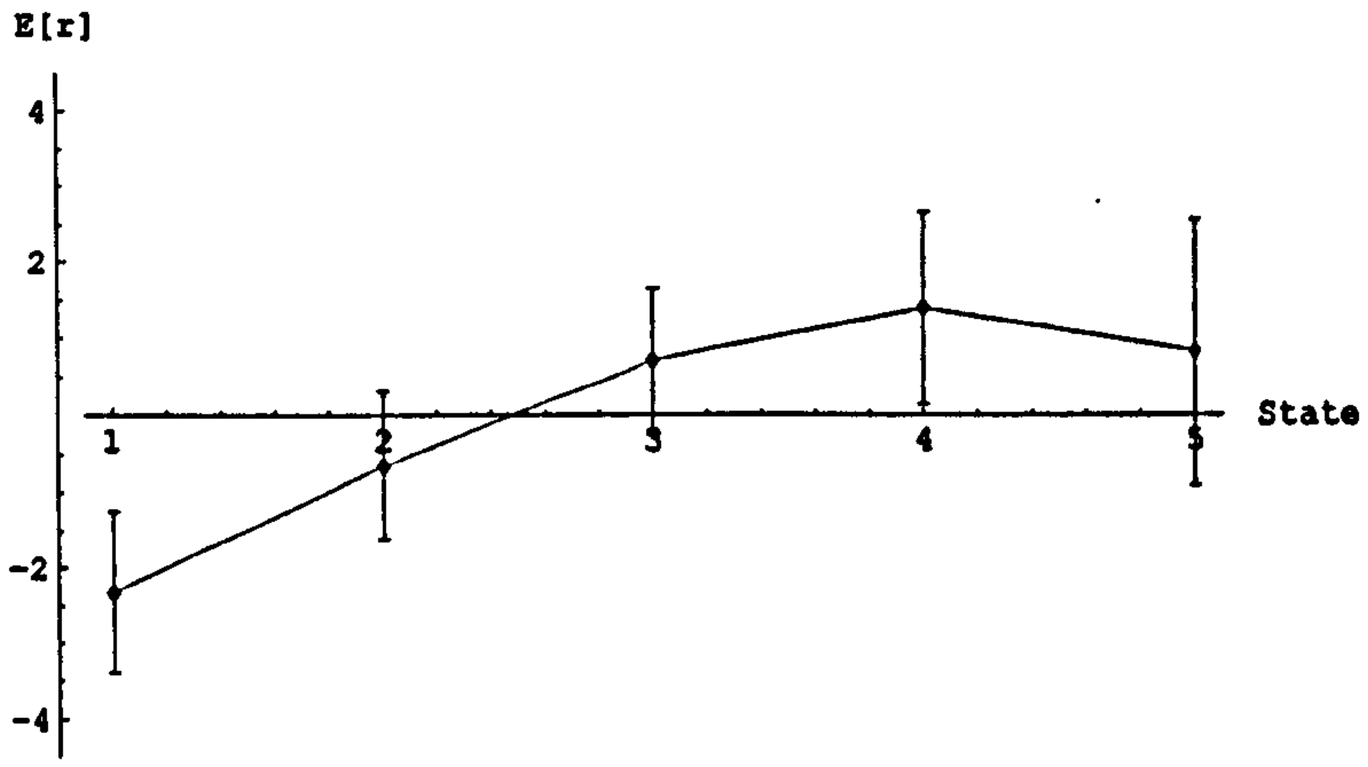


Figure 47: The expected reward at each state for c=0.5 with RMS error bars.

With c set to 0.9 the expected path length is still only 10 and now the differences between the expected rewards are clearly swamped by the RMS error at each state, see Figure 48. Thus the cyclic model provides a difficult estimation problem but it isn't of practical importance because the expected reward at each state becomes equal as the model complexity increases.

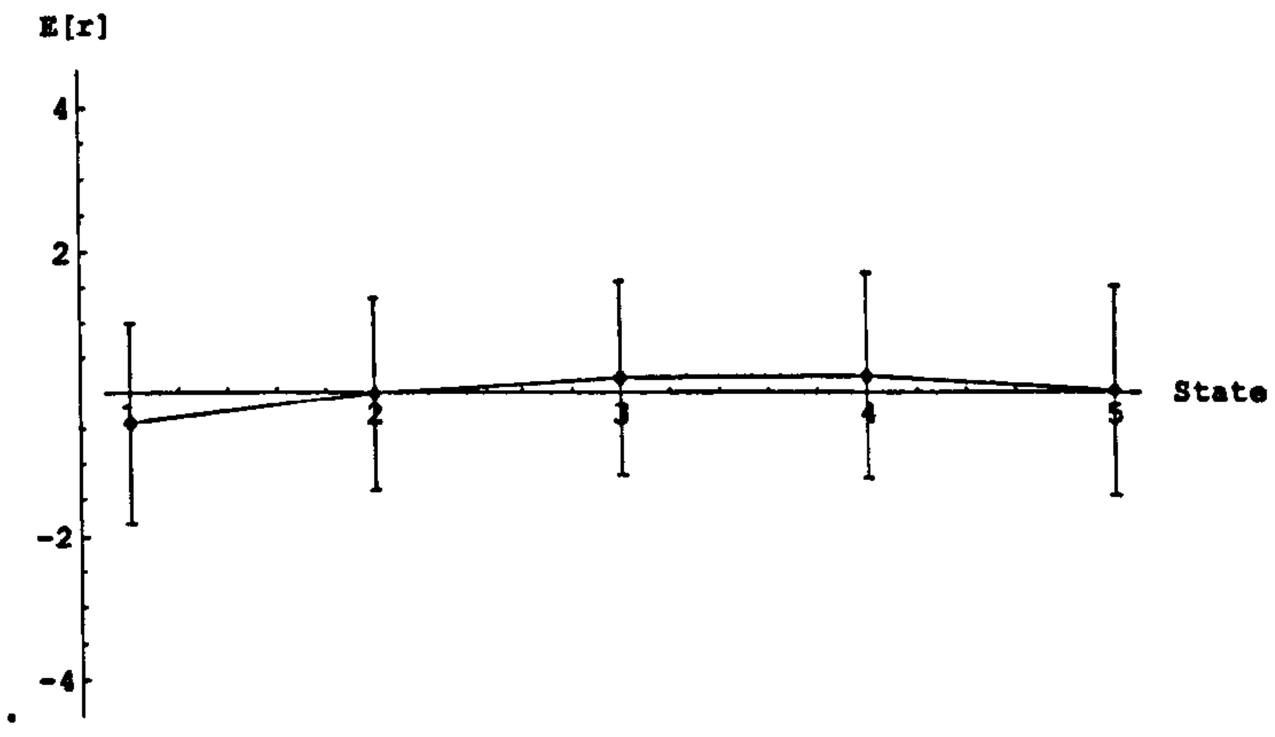


Figure 48: The expected reward at each state for c=0.9 with RMS error bars

Now that the fundamental behaviour of the cyclic model has been explored the next question is what the effect of a weighted reward would be? A simple-minded approach would suggest that as the expected path length at each state is constant then weighting by λ^d would have a negligible effect and this is indeed the case. The curves for the expected weighted reward and its RMS variation can be seen in Figure 49 for c=0.9 and ϕ=1 but there is no qualitative change for other values of c and ϕ - although there is a small reduction in the expected values and the RMS variation. Notice that these graphs are for a wider range of λ values than in the case of the SRW.

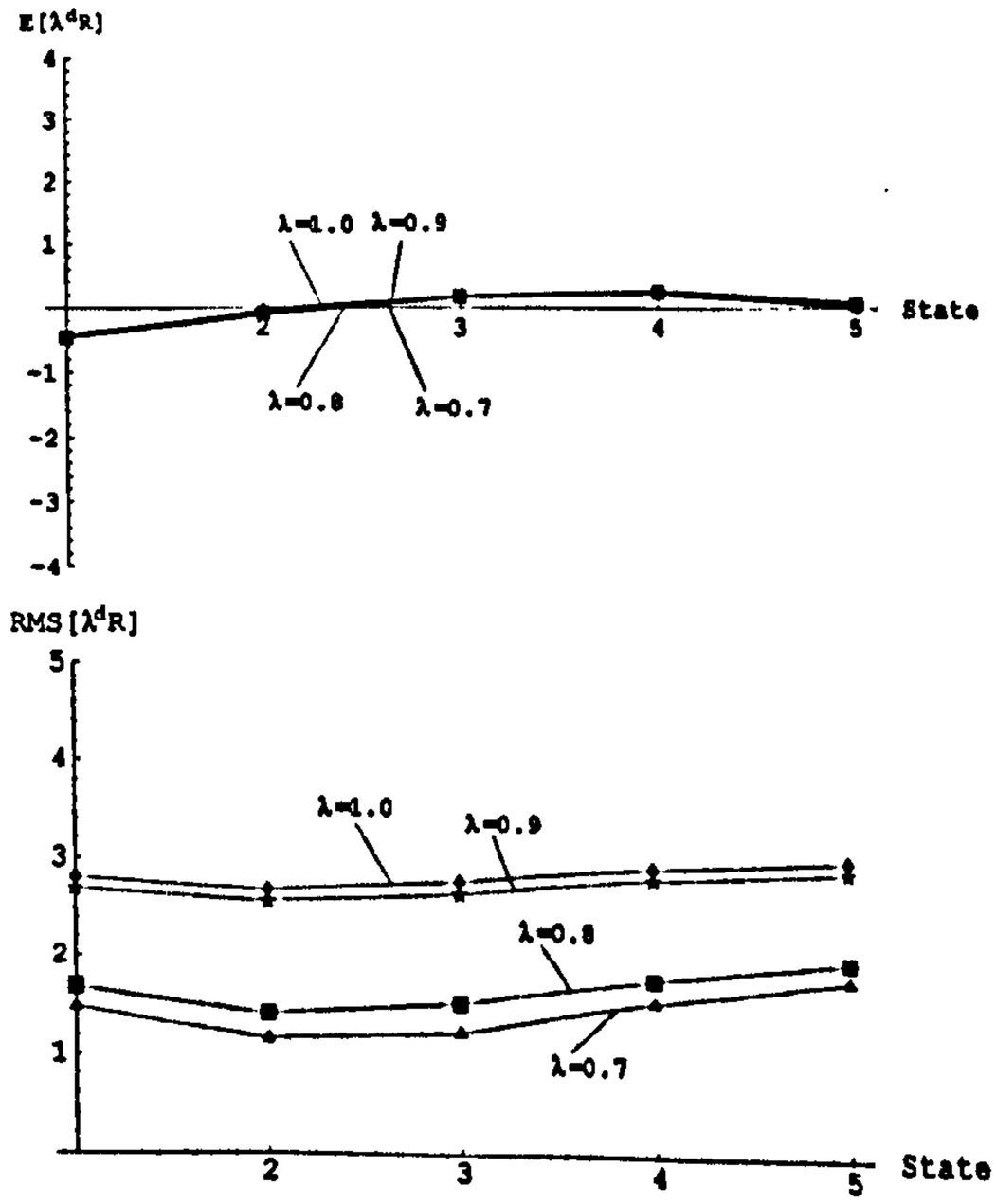


Figure 49: Expected value and RMS error of λ^d r for each state for 4 values of λ

Finally, it is worth examining the per-state error reduction for the range of values of λ. Figure 50 shows that the RMS error reduces as λ gets smaller, which is quite different behaviour from the SRW model where there was a distinct dip for λ values around 0.95. This is reasonable given the way that the cyclic model's means hardly vary with λ but its variances reduce with λ.

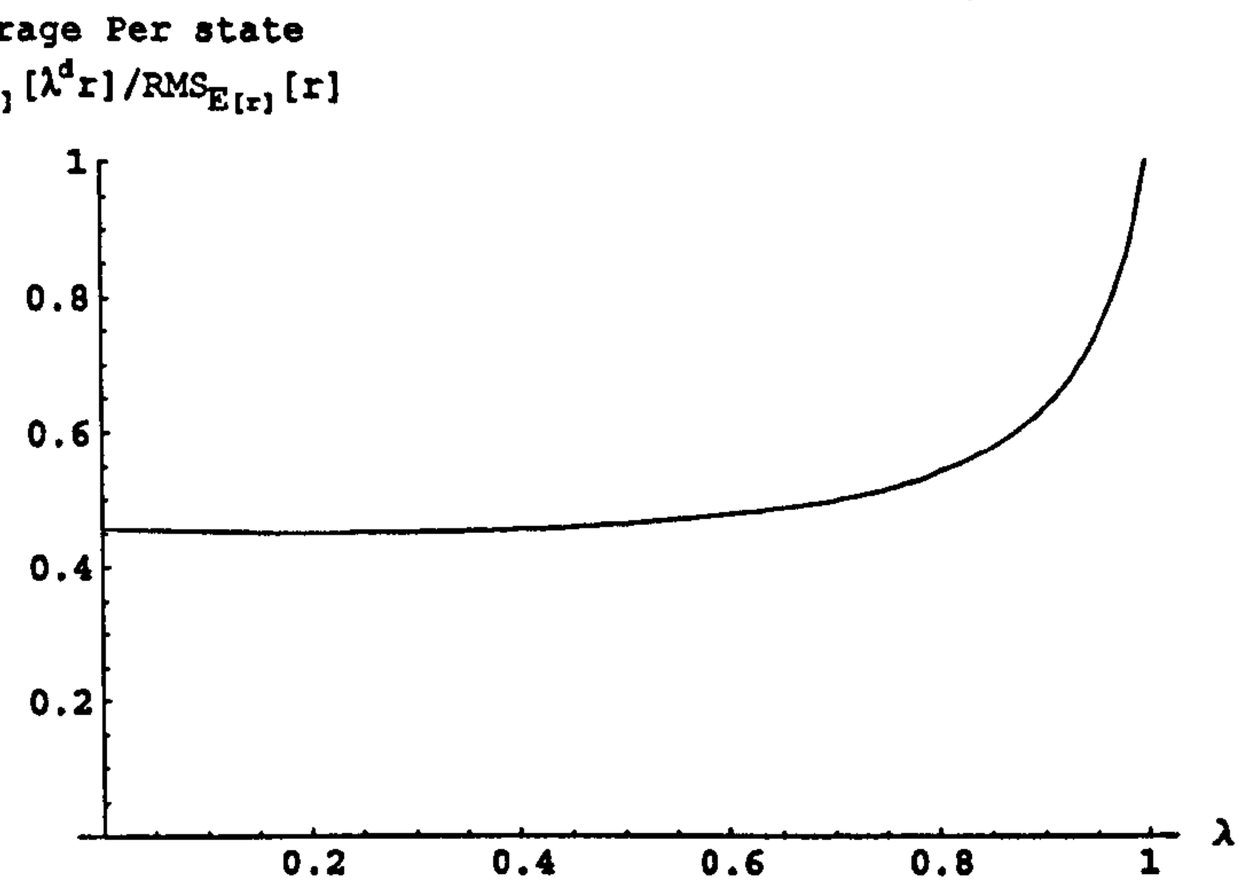


Figure 50: RMS reduction for a range of λ values (c=0.9 ϕ=1)

TD and WR with zero initial estimate

Before starting to examine the behaviour of TD and WR on this model one small change needs to be made. The expected reward for each state is very close to zero and starting the estimator off with an initial estimate of zero results in a smaller initial RMS error than the asymptotic error for any α in the MCA estimator. This means that the optimal α is zero or very nearly zero because adding information from the data simply makes the estimate worse than the initial estimate. This is a well-known phenomenon and the usual solution is simply to test the estimators using a non-zero initial estimate. However, we need to examine TD and WR when the initial estimate is zero and non-zero so in this case adding a constant, one say, to the rewards to move their expected value away from zero is a better solution. This change to the reward structure makes it possible to compute an optimal value of α for every samples size and use this to evaluate the performance of the TD estimator. This reveals that there is no value of α or λ that makes TD better than MCA for an optimal choice of α. As λ tends to 1 the best-performing TD tends to the optimal

MCA but it fails to outperform it. A typical optimal α performance chart can be seen in Figure 51 for c=0.9 ϕ=1. The value of ϕ has little effect and this is a reasonable behaviour if the additional power of TD comes from the association between reward and path length observed in the SRW.

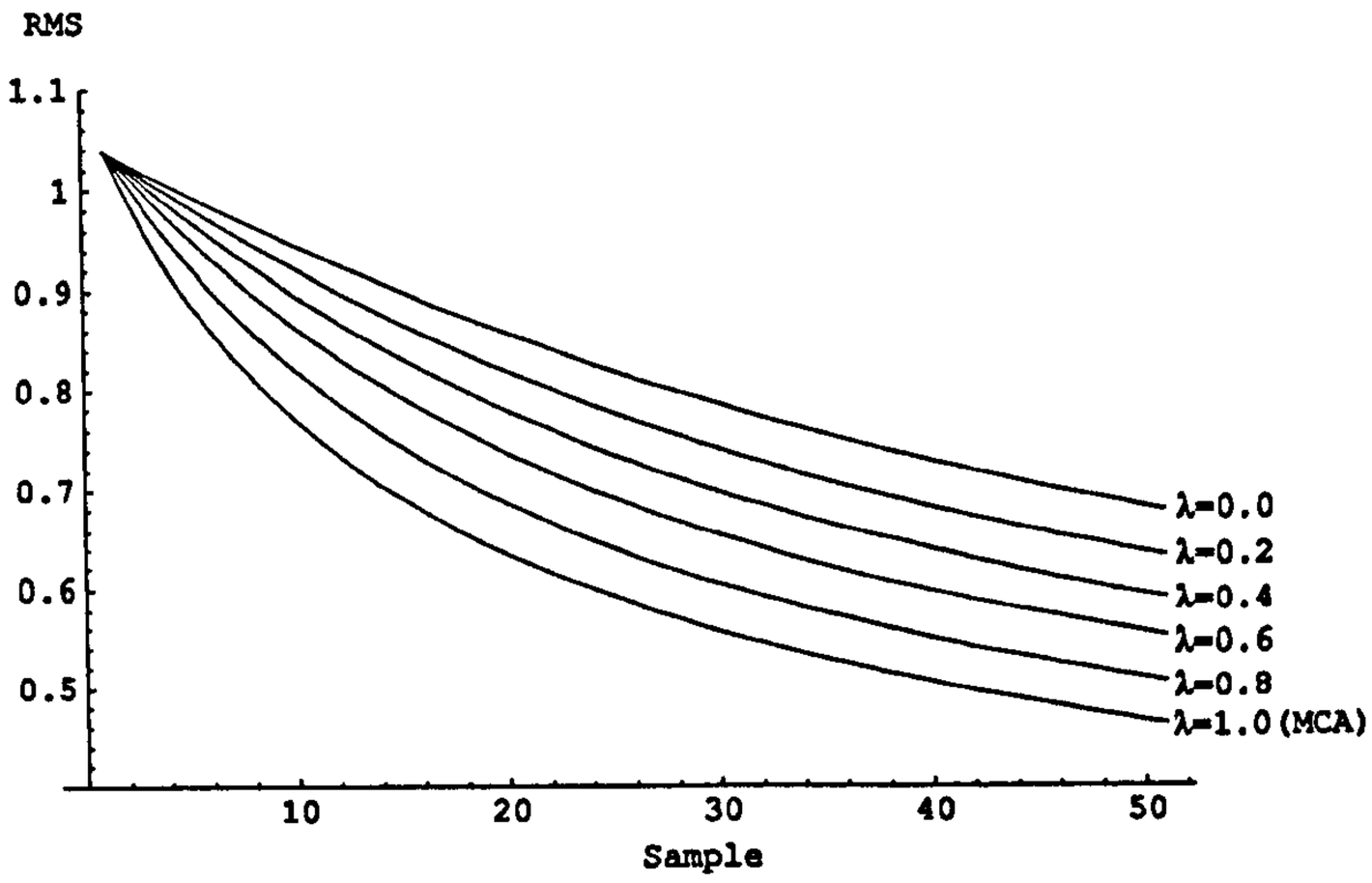


Figure 51: Optimal α TD estimators for c=0.9 ϕ=1

Comparing the average RMS error over the first ten steps for TD and WR reveals that once again they are similar but neither does better than MCA for the best choice of α - see Figure 52. Even though neither estimator is better than a simple MCA estimator they show a clear similarity as in the case of the SRW model. It seems that even if TD isn't better, it is still close to the WR estimator.

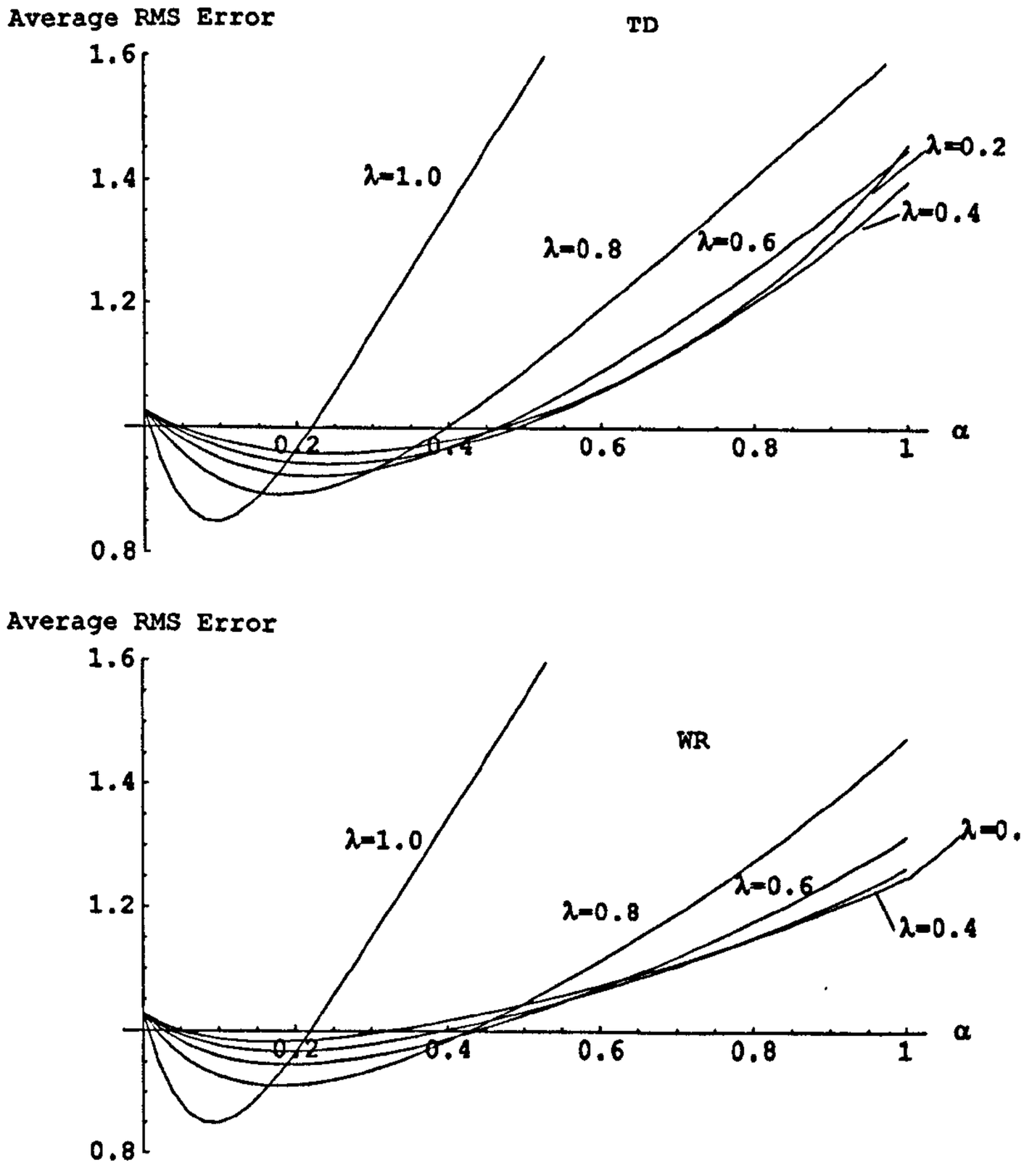
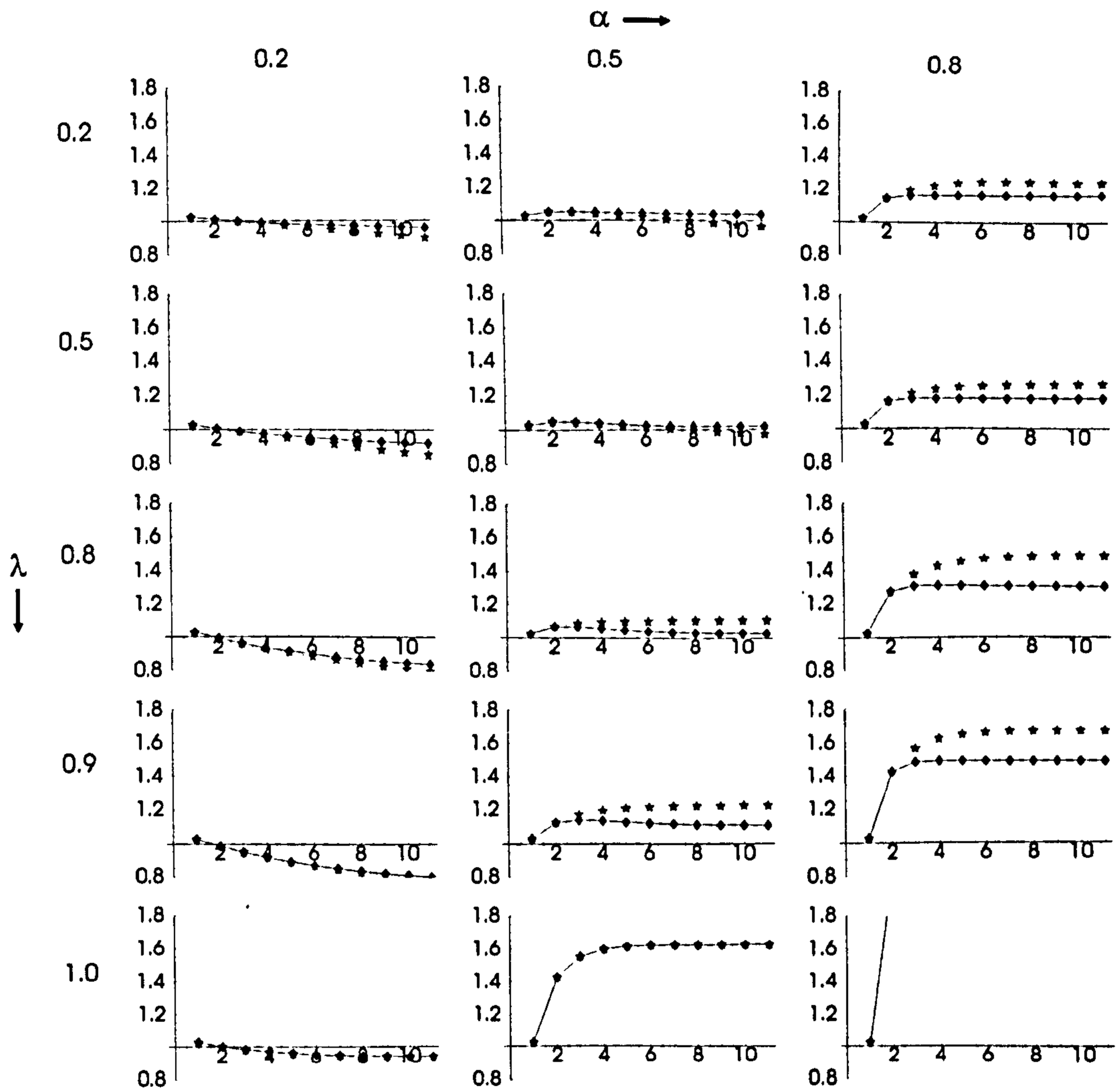


Figure 52: Average RMS error for the first ten steps for TD and WR c=0.9 ϕ=1

Comparing the detailed RMS behaviour for the first ten steps again reveals how close TD and WR are – see Figure 53. For large λ TD and WR are close, even when the α value is such that the initial estimate is better that the asymptotic error.



NB: vertical axis is the per-state RMS error, horizontal axis is sample size

Figure 53: Comparison of WR(λ) with TD(λ) over first 10 steps

Large sample performance is also similar to the SRW case with TD and WR close for large values of λ and moving further away due to the difference in asymptotic error as λ decreases. One difference is that α has to be small to ensure that the asymptotic error is smaller than the initial error. The results are qualitatively unaffected by the value for c and ϕ. An example for c=0.9, ϕ=1, λ=0.95 and α=0.1 can be seen in Figure 54.

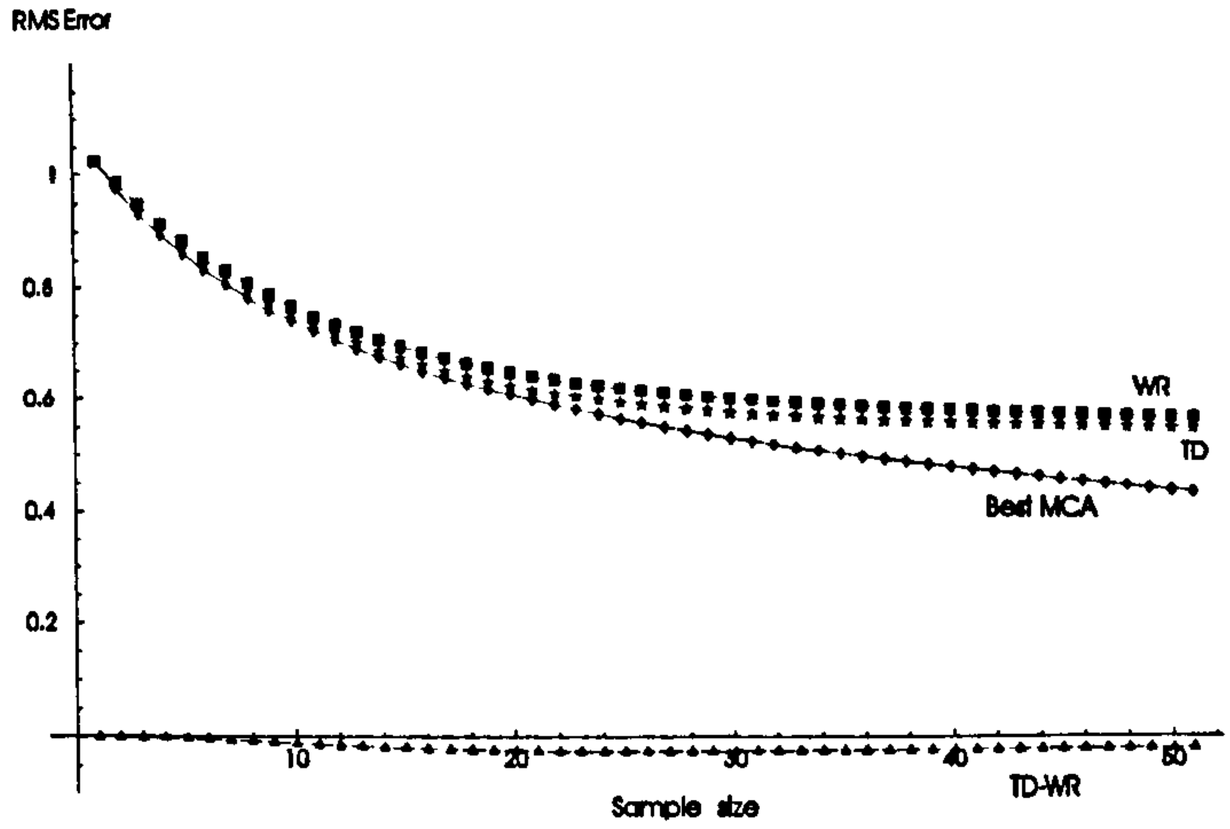


Figure 54: TD and WR for λ=0.95 and α=0.1 (c=0.9, ϕ=1)

As in the SRW case, using a WR estimator with the same asymptotic error produces a good match over the full range of λ and α. For example, Figure 55 shows the WR RMS curve for the same λ and α and for optimal λ and α for λ =0.6 and α=0.2. The optimal λ and α.were 0.88 and 0.08 respectively.

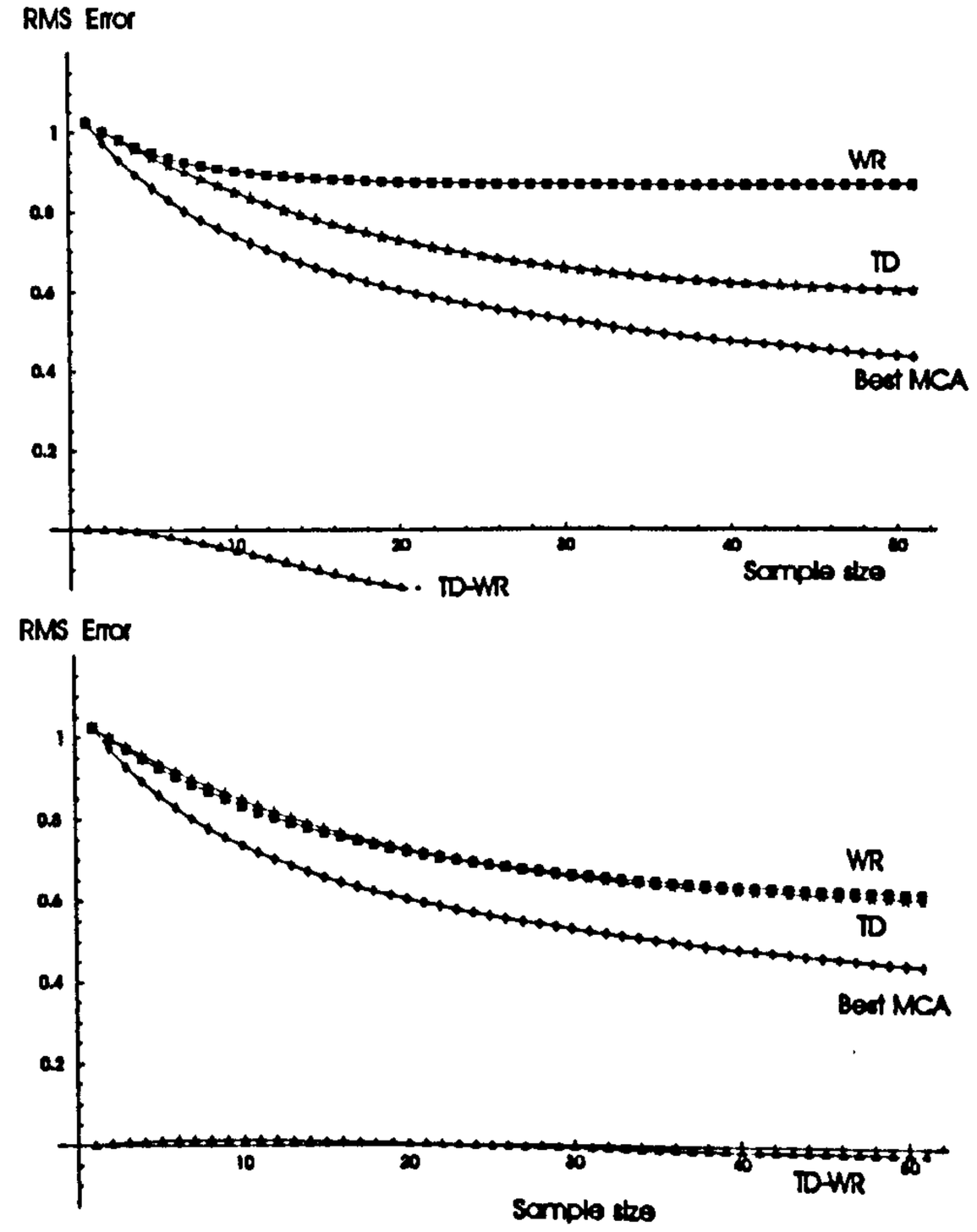


Figure 55: RMS curves for TD and WR for λ =0.6 and α=0.2 (above) and for WR optimal λ and α.(0.88 and 0.08).

TD and WR with non-zero initial estimate

When the estimators are started off with non-zero initial estimates then the pattern observed in the case of the SRW model is encountered again. Setting the initial estimate to one (and restoring the reward structure so that it has an mean of zero) reveals that TD performs worse than the WR estimator, presumably because of its use of poor initial estimates. For example, the average RMS error for the first ten estimation steps can be seen in Figure 56. Notice that the TD estimate is worse and it is still beaten by the MCA estimate. However the WR estimate is not only better then the TD estimate, it outperforms the MCA estimator for a range of values of λ and it gets better as λ decreases which is in agreement with the theoretical RMS reduction curve (Figure 50).

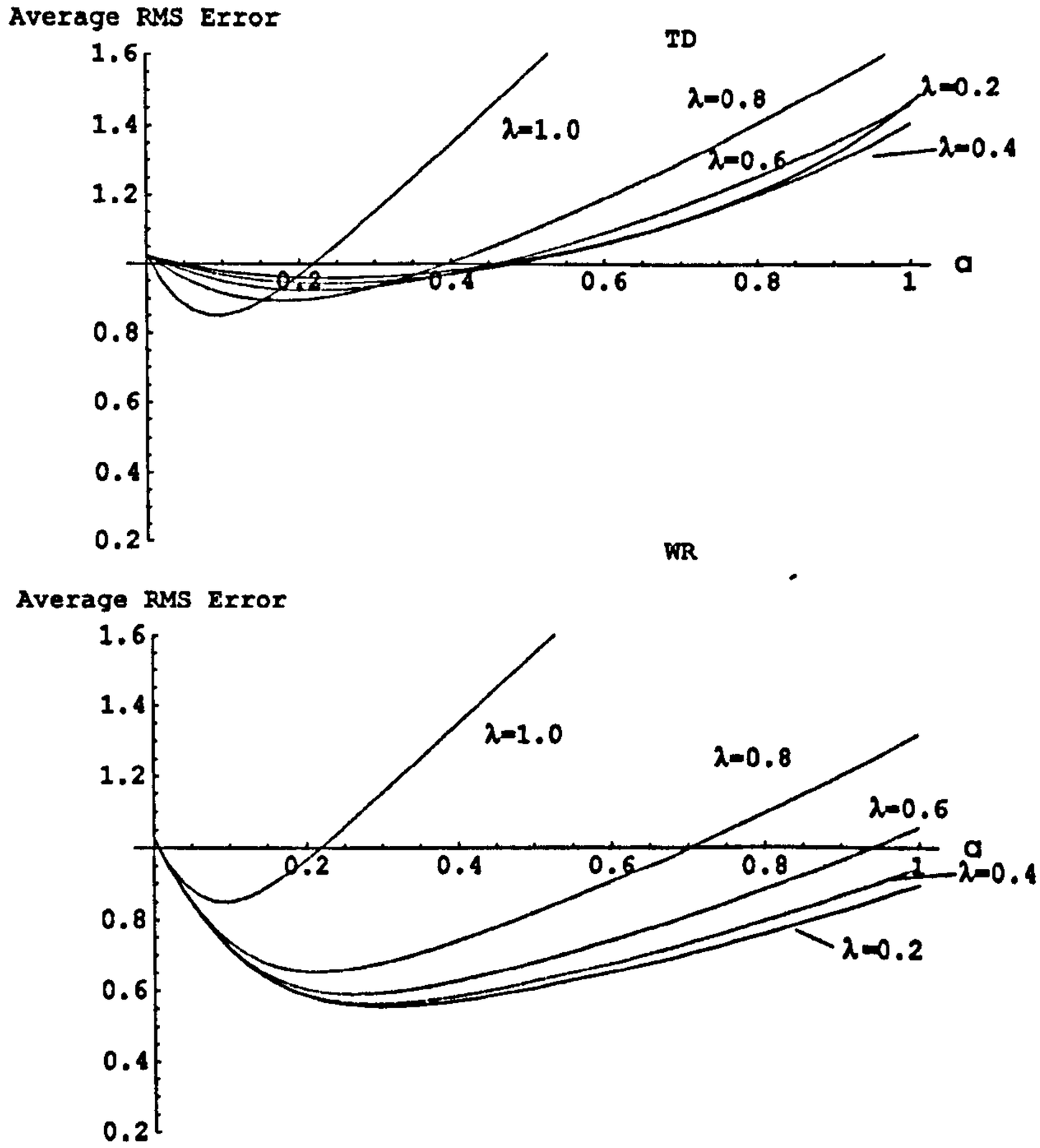


Figure 56: Average RMS error for the first ten steps for TD and WR c=0.9 ϕ=1 with a non-zero initial estimate.

The fact that WR appears to be less affected by the bias can also be seen in the large sample behaviour of the RMS error. Figure 57 shows the RMS curve for λ =0.6 and

α=0.2 which can be compared to the result in Figure 55 using a zero initial estimate. This behaviour is typical and again qualitatively unaffected by the model parameters.

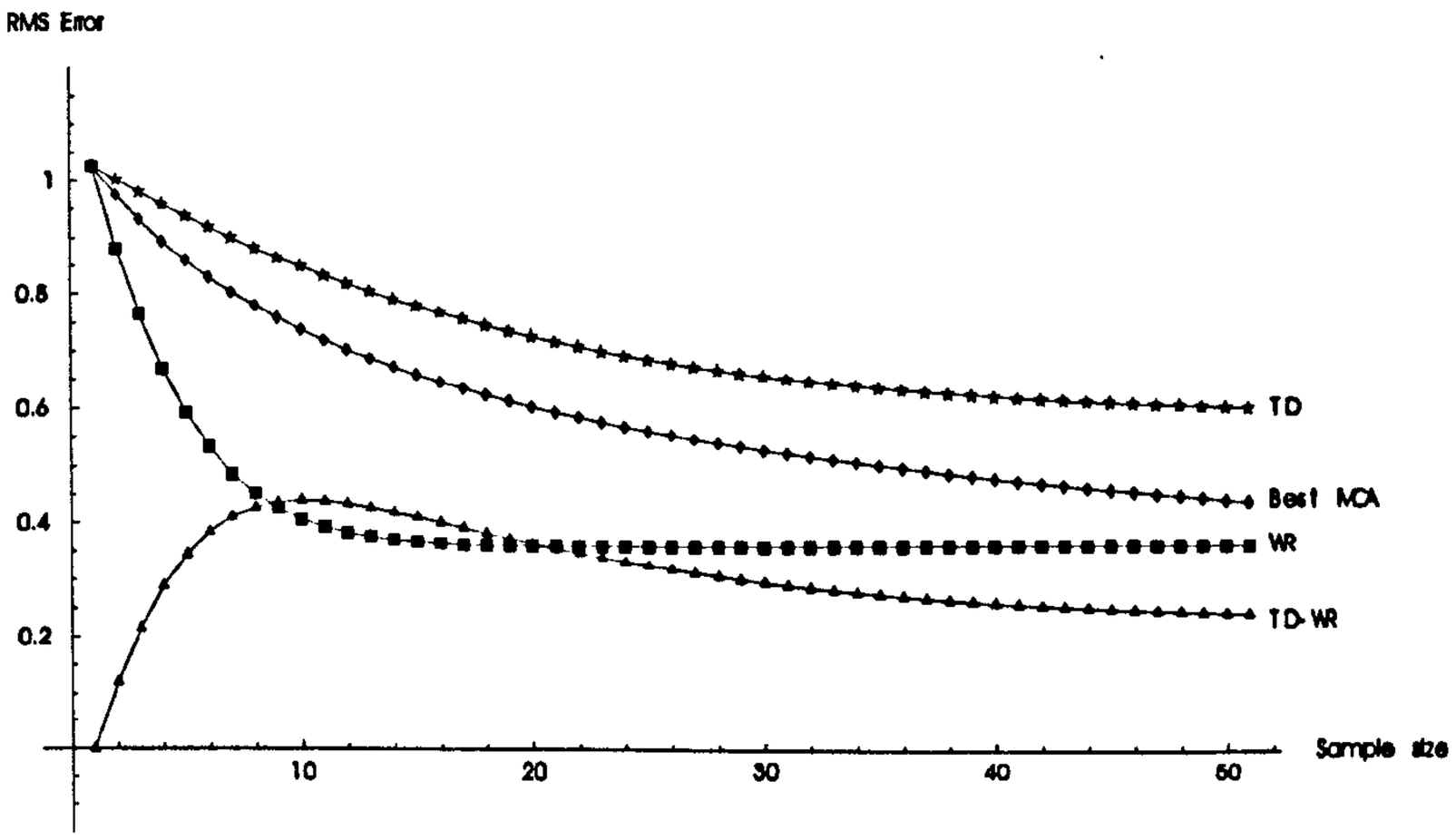


Figure 57 RMS curve for TD and WR for λ =0.6 and α=0.2 with non-zero initial estimate

Summary of cyclic model

Investigating the behaviour of TD and WR with a parameterised model clearly results in a huge amount of data as the parameter space provided by the model and the estimators, i.e n,c,ϕ λ and α, has to be explored. However the variation in behaviour caused by values of c and ϕ that produce realisations with more than a few states in a realisation is minor.

The main conclusions to be drawn are:

1) TD doesn't out perform a simple MCA estimator for any values of λ and α irrespective of the initial estimate.

2) TD is close to WR for a range of λ in the first few steps of estimation, especially if the difference in the asymptotic error is corrected for. However, they both perform poorly compared to the simple MCA estimator when the initial estimate is zero.

3) When the initial estimate is non-zero the performance of TD is similar to WR for λ close to 1 but worse than the WR estimator when λ is small. The WR

estimator outperforms the simple MCA estimator (using the same initial estimate) in this case.

Point one is to be expected if TD is making use of weighted rewards to improve its estimation as this particular model shows no association between expected path length and expected reward. Point two is further confirmation that the behaviour of TD can be accounted for by weighted reward principles. Point three, however, provides something new to be considered. At first sight it appears that the WR estimator is out-performing the MCA and the TD estimator due to a lower sensitivity to a higher initial bias. However, while it might well be less sensitive to the initial bias, WR is also being helped by the fact that in this case the expected rewards are close to zero. As λ is reduced the expected value of $\lambda^d r$ tends to zero, which just happens to be closer to the expected value of the reward in this case. As a result WR improves as λ gets smaller, which is not in line with its general behaviour. The fact that WR is better than TD or MCA for larger values of λ is still evidence that it isn't as influenced by initial bias.

The bottleneck model

One of the ideas of temporal difference learning is that good estimates of the expected reward at some states can be used to improve the estimates at other states. For example, if there are a few states that have to be entered on the way to a final reward then good estimates of reward at these "bottleneck" states quickly propagate to other states further separated from the reward. It could be argued that the SRW model is already an example of a bottleneck because the two states at either end of the linear chain "learn" their expected reward quickly (because they have a low reward variance) and because they have to be passed through to reach the reward. A more explicit bottleneck model is, however, worth exploring to see if there are any gains in estimation performance due to temporal difference learning in TD as opposed to weighted reward learning.

A bottleneck model can be constructed from a Simple Random Walk with reflecting end states by adding a probability of direct transition to one of a number of states leading to the reward states. For simplicity two bottleneck states are used, one

connected to three positive terminal rewards and one connected to three negative terminal rewards. Clearly the bottleneck states quickly learn their expected rewards.

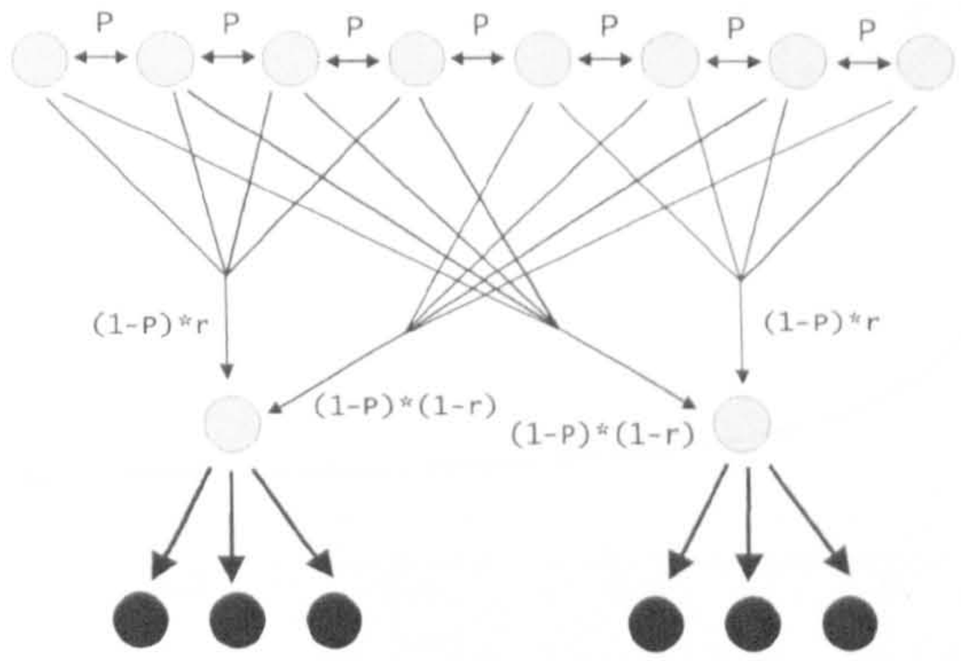


Figure 58: The bottleneck model

Two parameters, p and r, control the model. The probability of making a transition to a neighbouring state in the SRW is p/2 and the probability of making a transition to one of the bottleneck states is 1-p. This probability is divided in the ratio r to 1-r between the two states. The states in the left hand portion of the SWR divide the probability up in the ratio r to r-1 and in the right hand half in the ratio r-1 to r. Thus p controls how long the model remains in the SRW part of the model and r controls the relative strength of connection of the left and right hand parts to the two bottleneck states. Clearly as p increases both the expected path length and the cyclicity increases and this tends to blur the differences between the expected rewards at each state – much as in the case of the cyclic model. Equally as r tends to .5 the expected rewards in the two halves of the SWR tend to the same value as they have equal chance of jumping to either of the bottleneck states. Clearly r can be thought of as controlling the level of determinism in the reward. That is, for r=1 each half of the SRW always jumps to the same bottleneck states

Only p affects the expected path length and recurrence, as shown in Figures 59 and 60.

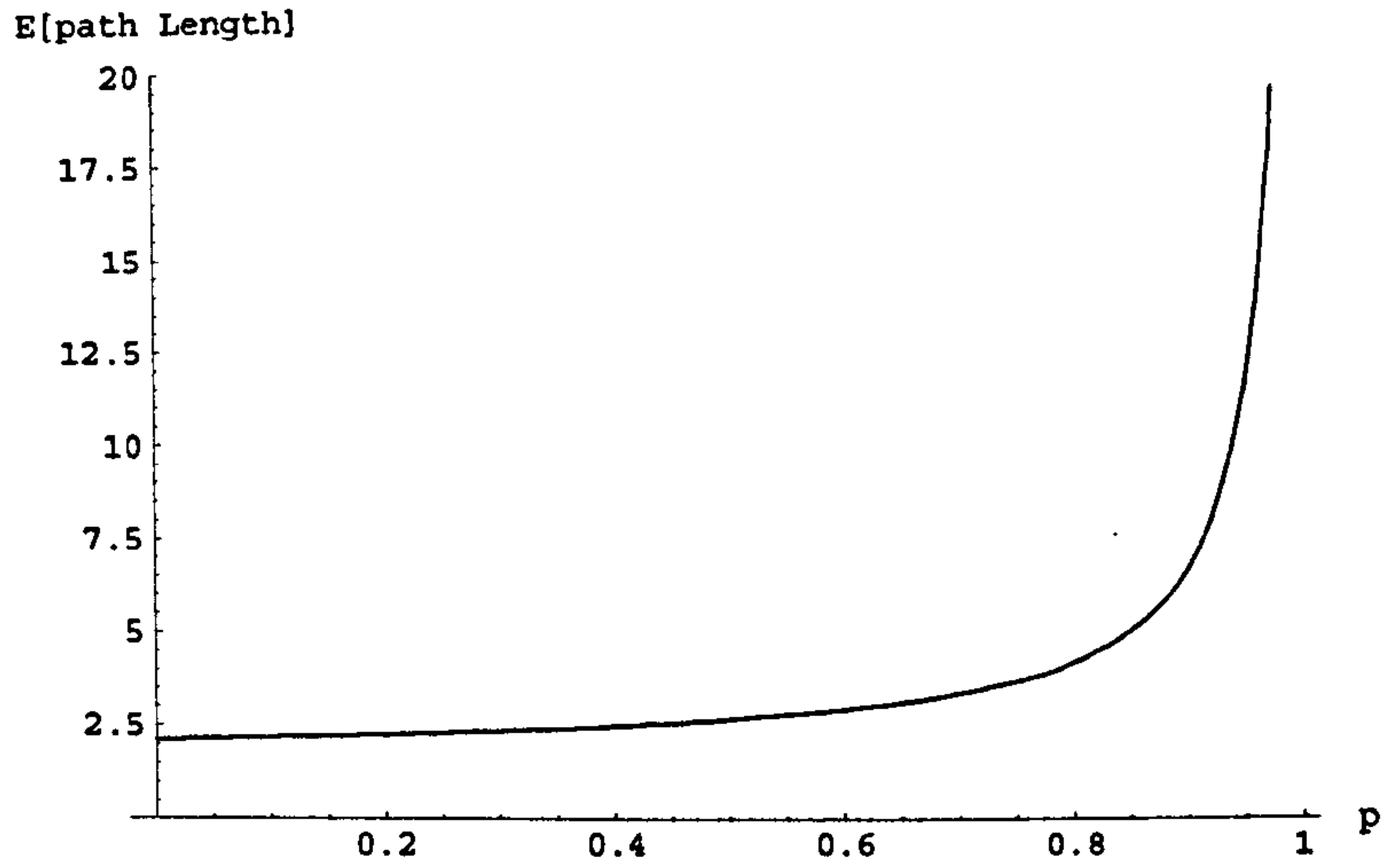


Figure 59: The effect of p on expected path length

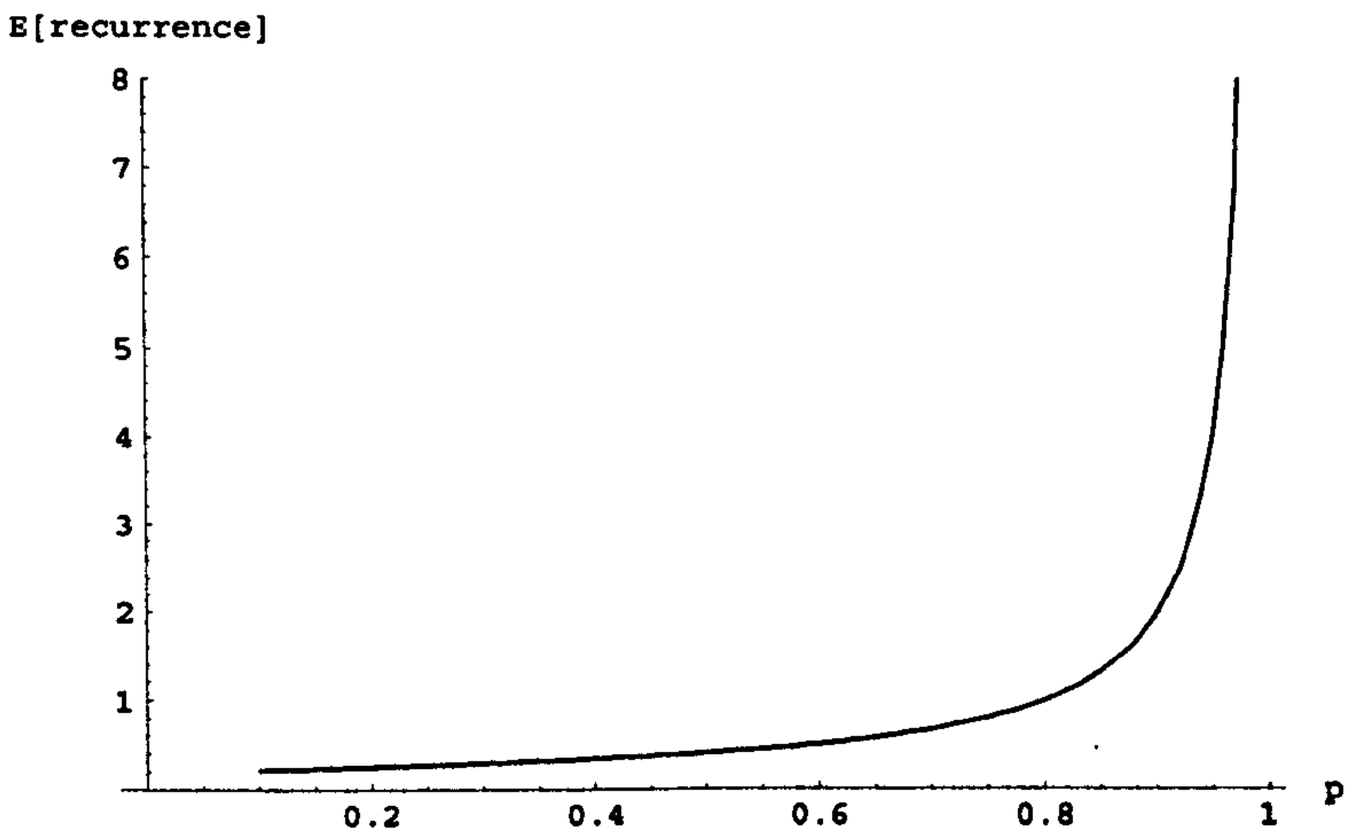


Figure 60: The effect of p on the expected recurrence of a state

The rewards used are i-*term*-0.5 where i is the state number and *term* the number of terminal states. For values of r close to 0.5 the expected reward at each state tends to zero and produces a model that is uninteresting from the point of view of policy improvement. As r increases towards 1 (or decreases towards 0) the difference between the expected reward in states in the left and right half of the SRW increases. The effect of increasing p for a fixed r is to increase the expected path length and to decrease the difference in the expected rewards between the right and left half of the SRW. This behaviour can be seen in Figure 61 where expected rewards are plotted with error bars for the 8 states in the SRW (the expected rewards at the bottleneck states don't vary with p or r).

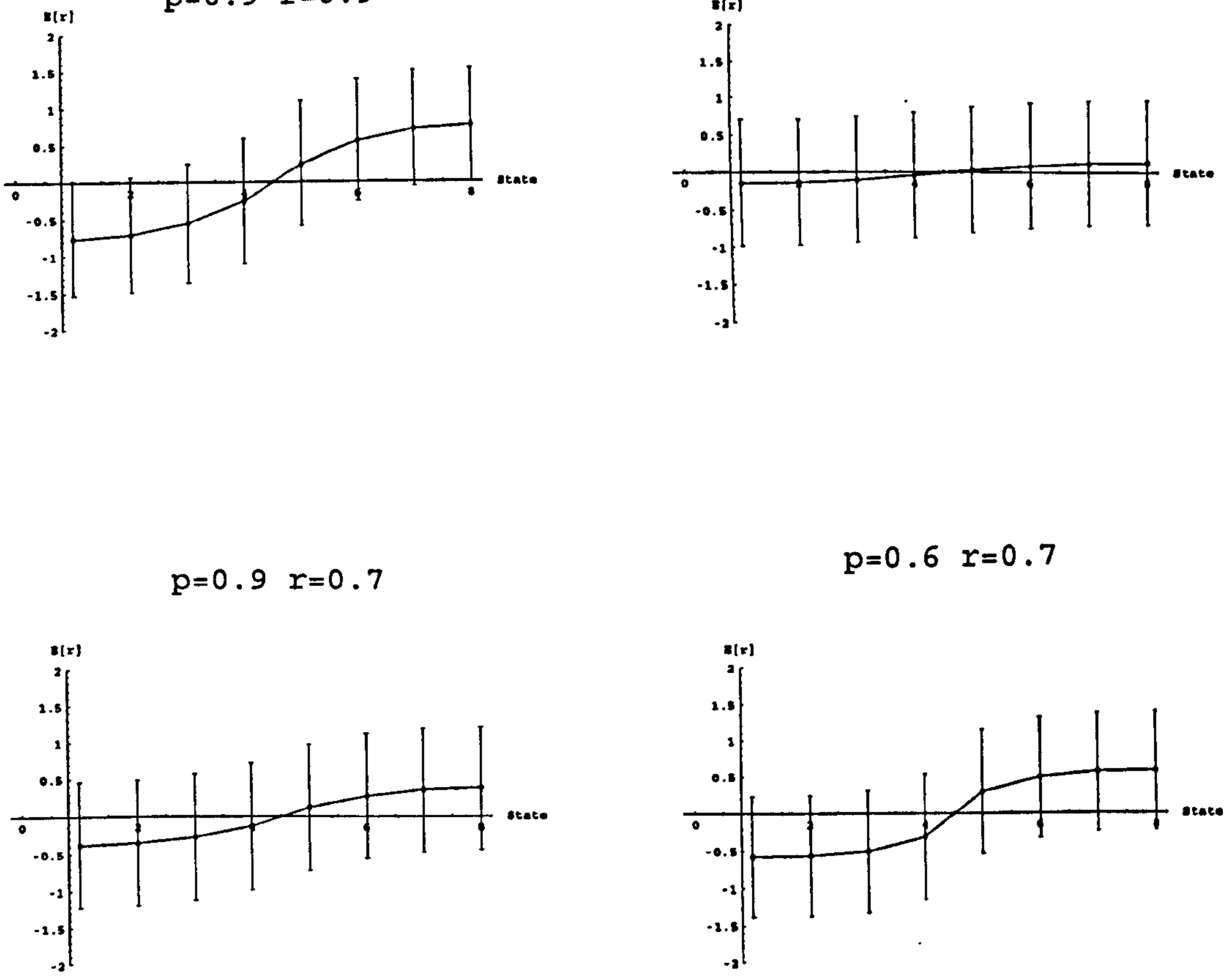
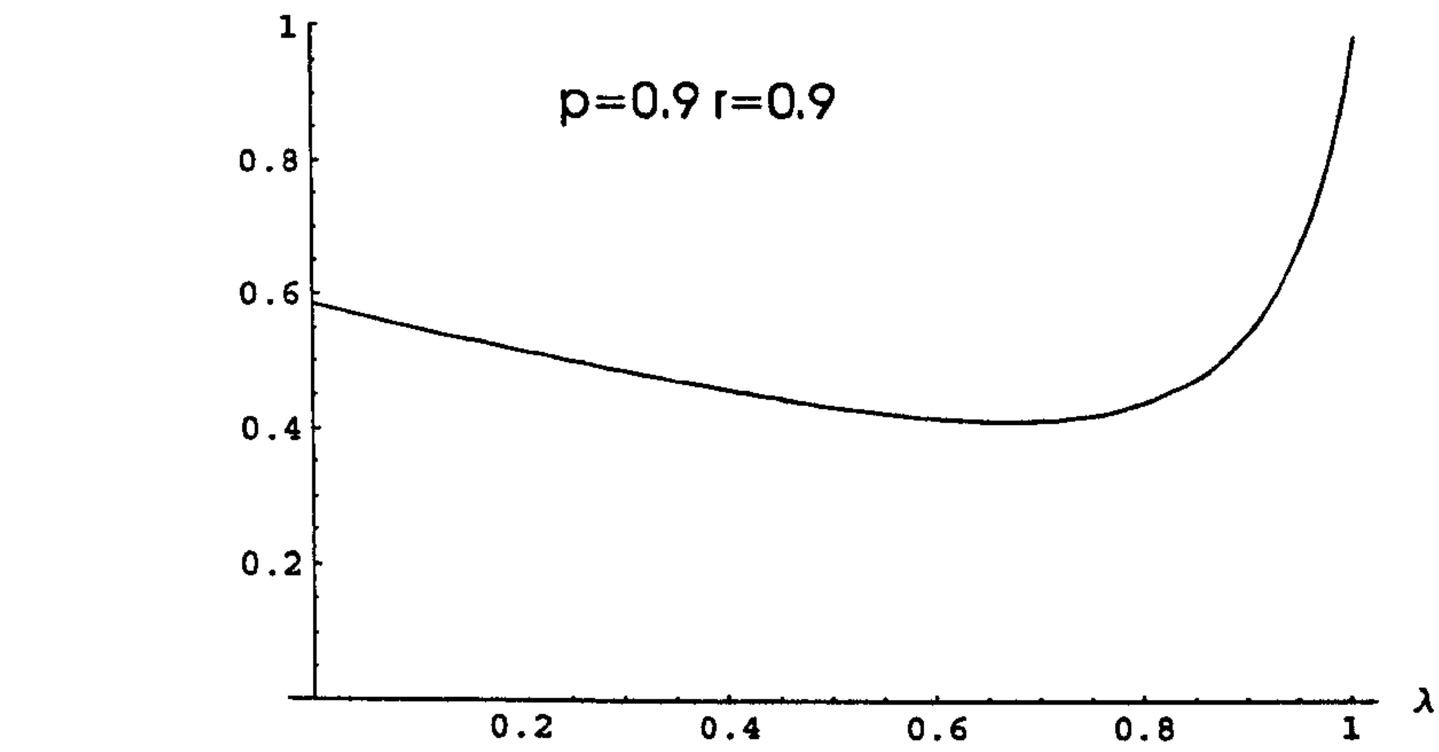


Figure 61: The expected reward RMS error bars for a range of p and r values

As with the cyclic model, expected rewards that are close to zero tend to remove asymptotic bias from the WR estimator and hence give it an unfair advantage. For this reason the behaviour of the estimators with two models using p=0.9 r=0.9 (high path length and determinism) and p=0.6 r=0.7 (lower path length and determinism).

Now that the fundamental behaviour of the bottleneck model has been explored the next question is what the effect of a weighted reward would be? The simplest way to illustrate the behaviour of WR is to examine the per-state error reduction for the range of values of λ. Figure 62 shows that the RMS error reduces for values of λ smaller than one and there is likely to be an optimal value of λ each case. This behaviour is similar to that of the SRW model.



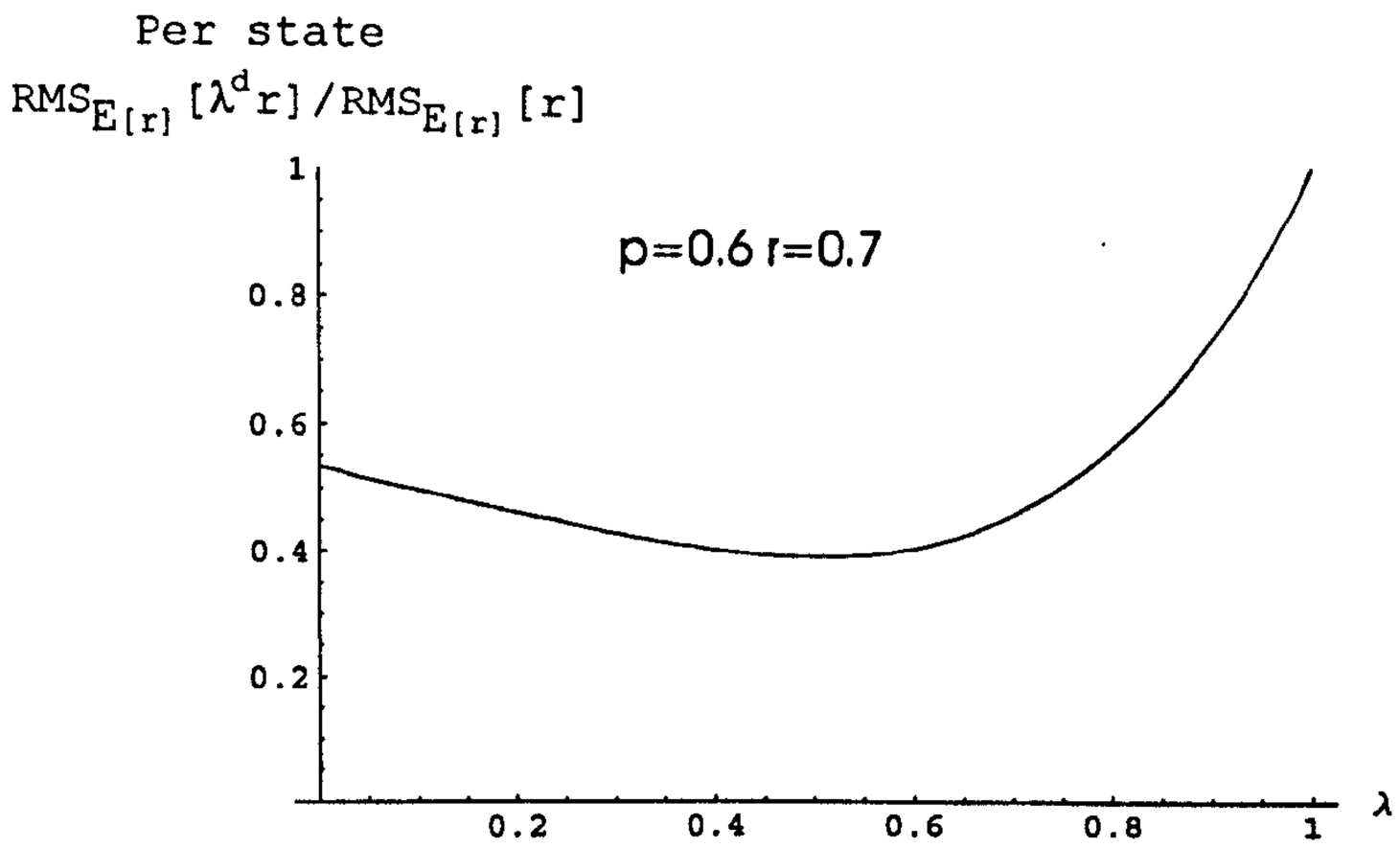


Figure 62: RMS reduction for a range of λ values (using p=0.9 r=0.9 and p=0.6 r=0.7.)

TD and WR with zero initial estimate

Typical optimal α performance charts can be seen in Figure 63 for p=0.9 r=0.9 and p=0.6 r=0.7. It can clearly be seen that there is a range of λ values for which it is better than the MCA estimator (λ=1) in both cases.

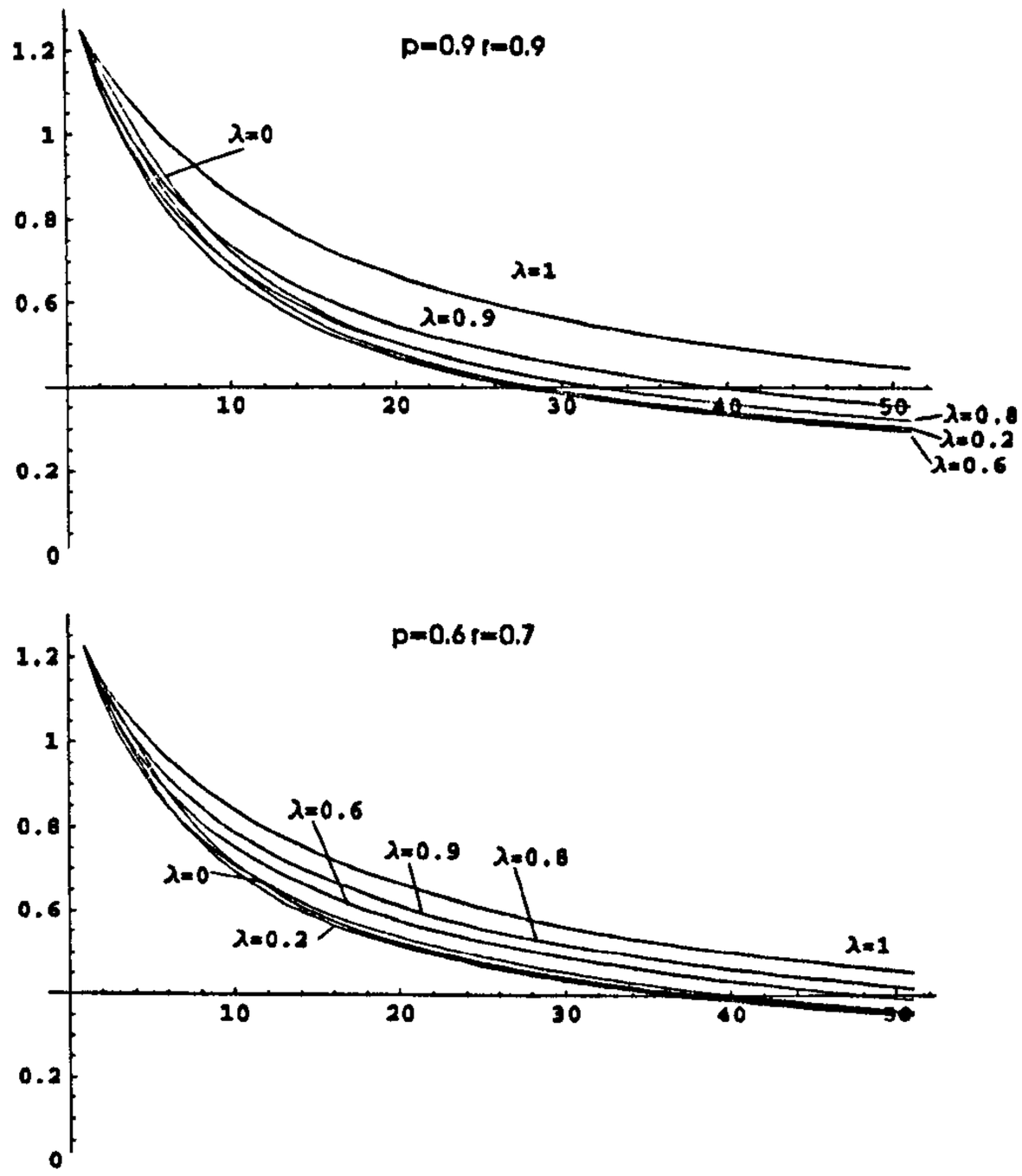


Figure 63: Optimal α TD estimators (using p=0.9 r=0.9 and p=0.6 r=0.7.).

Comparing the average RMS error over the first ten steps for TD and WR reveals that once again they are similar for both models - see Figure 64.

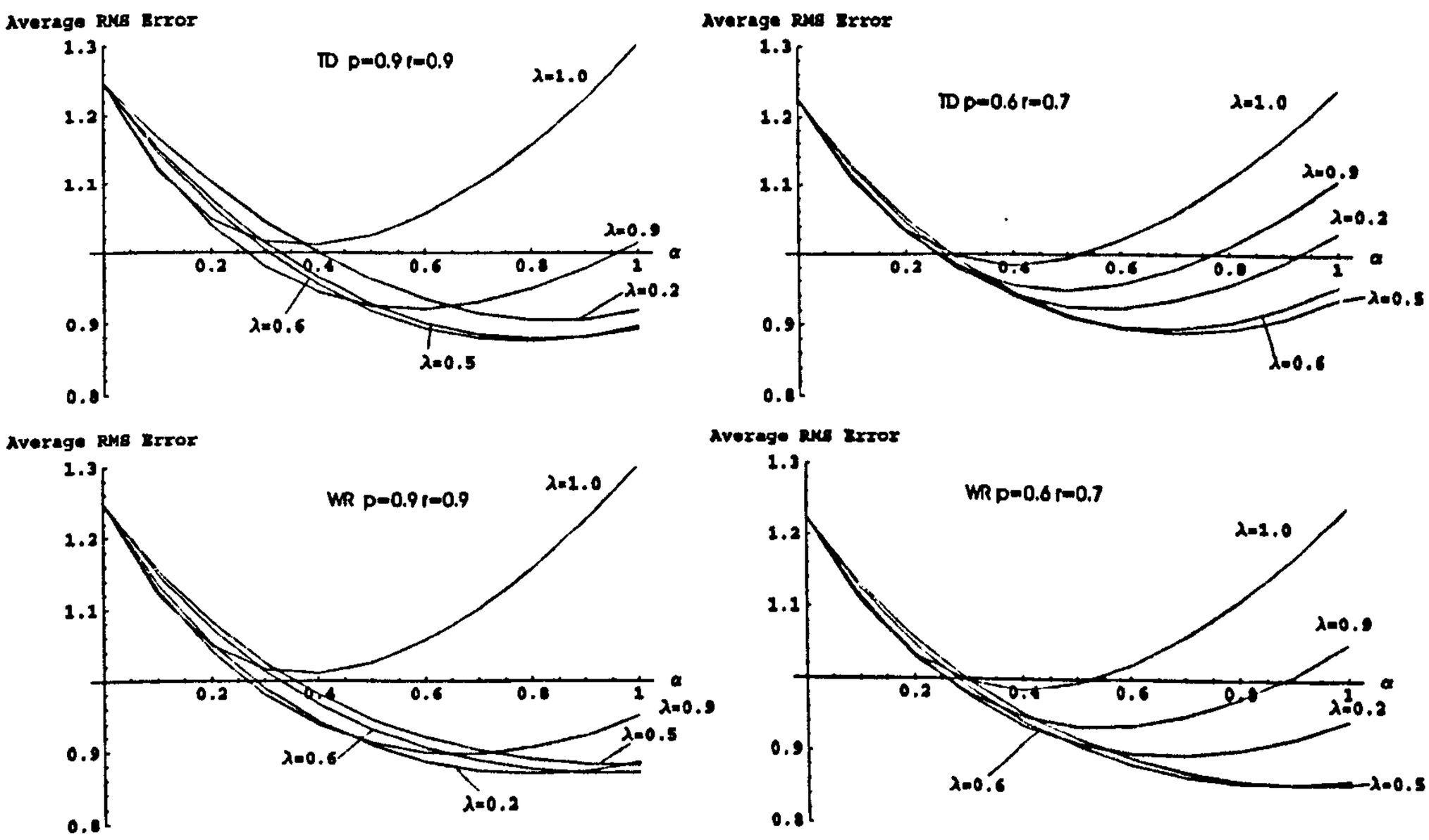
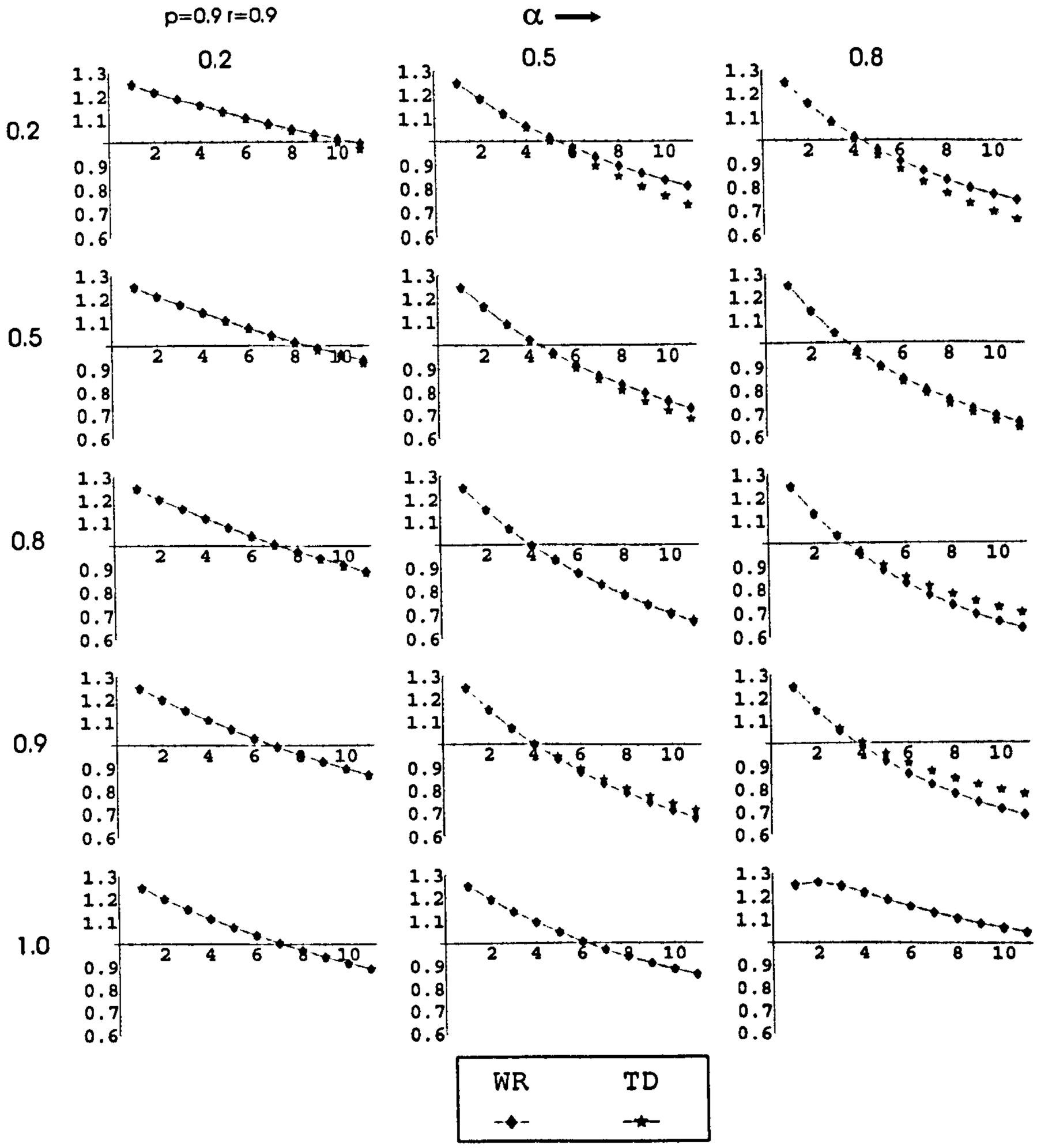


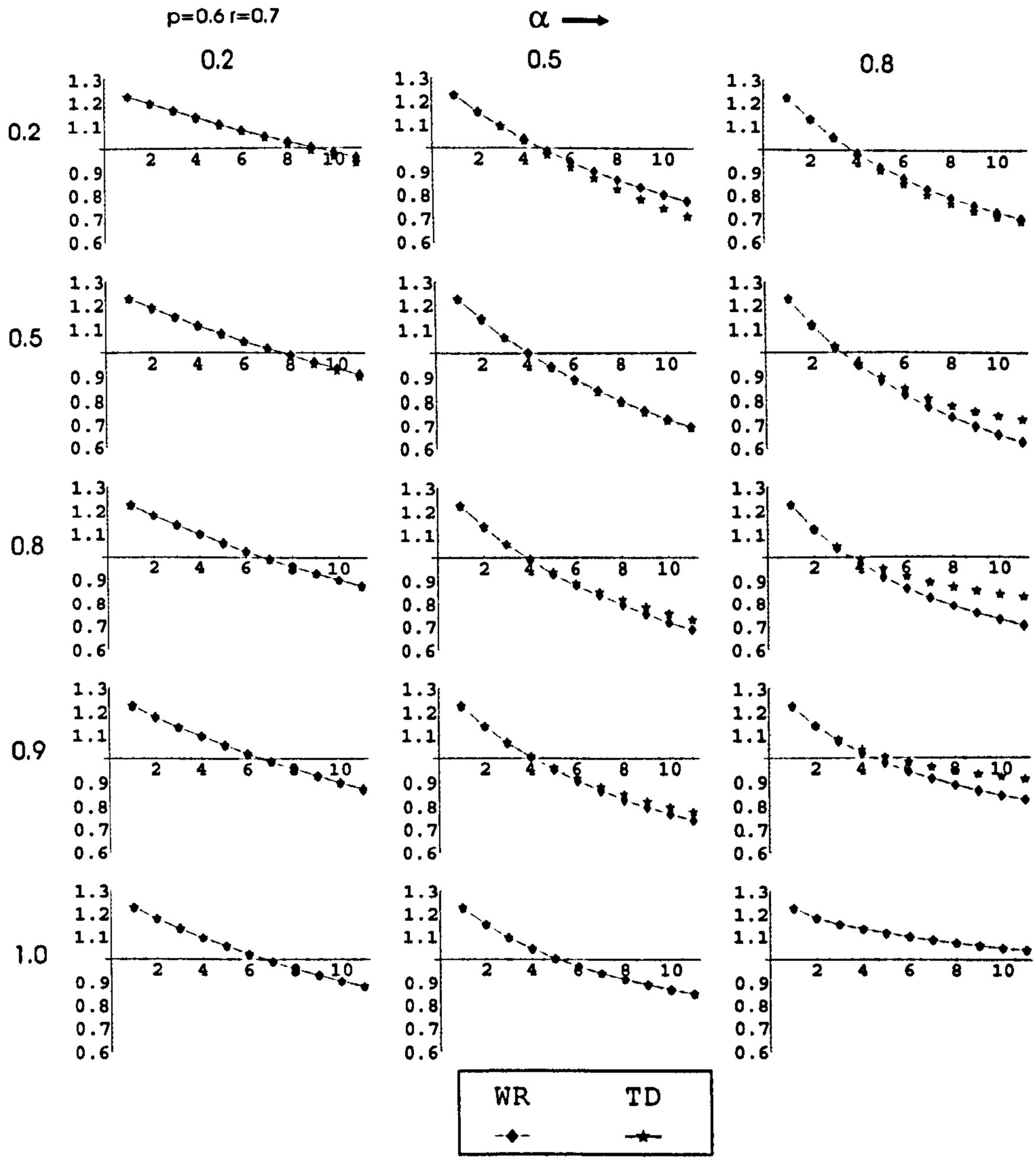
Figure 64: Average RMS error for the first ten steps for TD and WR (using p=0.9 r=0.9 and p=0.6 r=0.7.)

Comparing the detailed RMS behaviour for the first ten steps again reveals how close TD and WR are – see Figure 65 for p=0.9 r=0.9 and Figure 66 for p=0.6 r=0.7. For large λ TD and WR are close even for large α.



NB: vertical axis is the per-state RMS error, horizontal axis is sample size

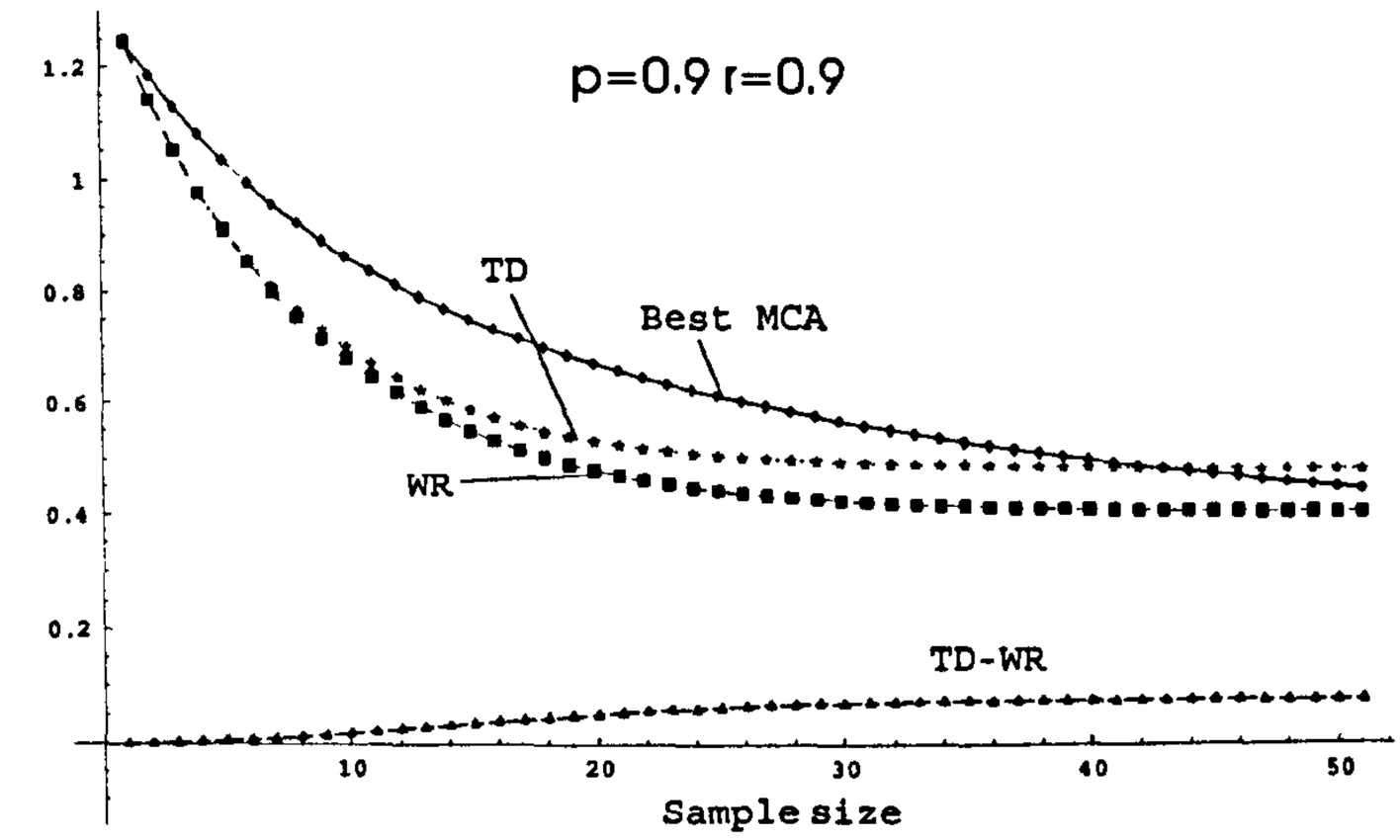
Figure 65: Comparison of WR(λ) with TD(λ) over first 10 steps p=0.9 r=0.9



NB: vertical axis is the per-state RMS error, horizontal axis is sample size

Figure 66: Comparison of WR(λ) with TD(λ) over first 10 steps p=0.6 r=0.7

Large sample performance is also similar to the SRW case with TD and WR close for large values of λ and moving further away due to the difference in asymptotic error as λ decreases. One difference is that WR actually outperforms TD for larger sample sizes as can be seen in Figure 67.



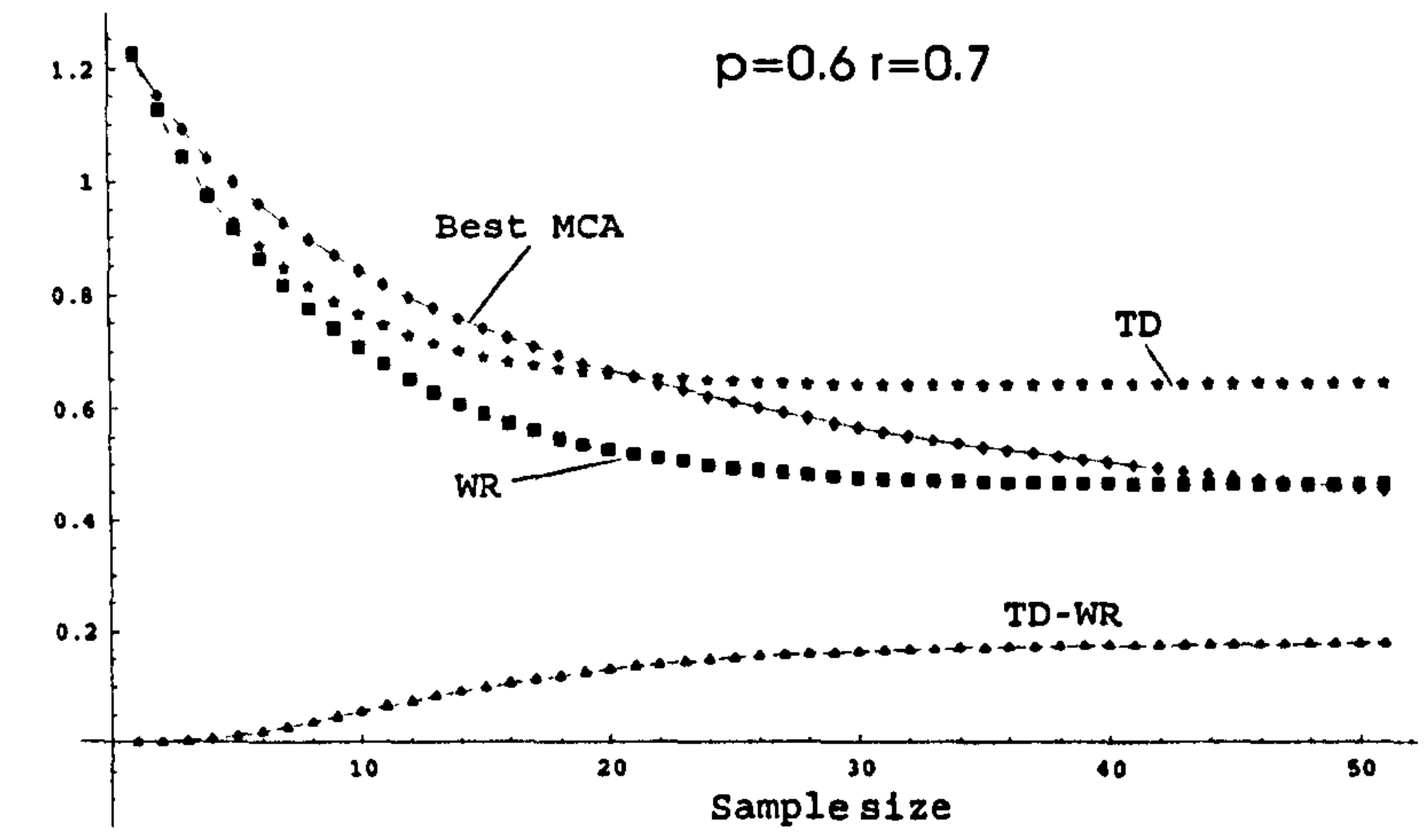


Figure 67: TD and WR for λ=0.8 and α=0.6 (using p=0.9 r=0.9 and p=0.6 r=0.7)

As in the SRW case, using a WR estimator with the same asymptotic error produces a good match over the full range of λ and α. For example, Figure 68 shows the WR RMS for optimal λ and α for TD, using λ =0.8 and α=0.6 for both models.

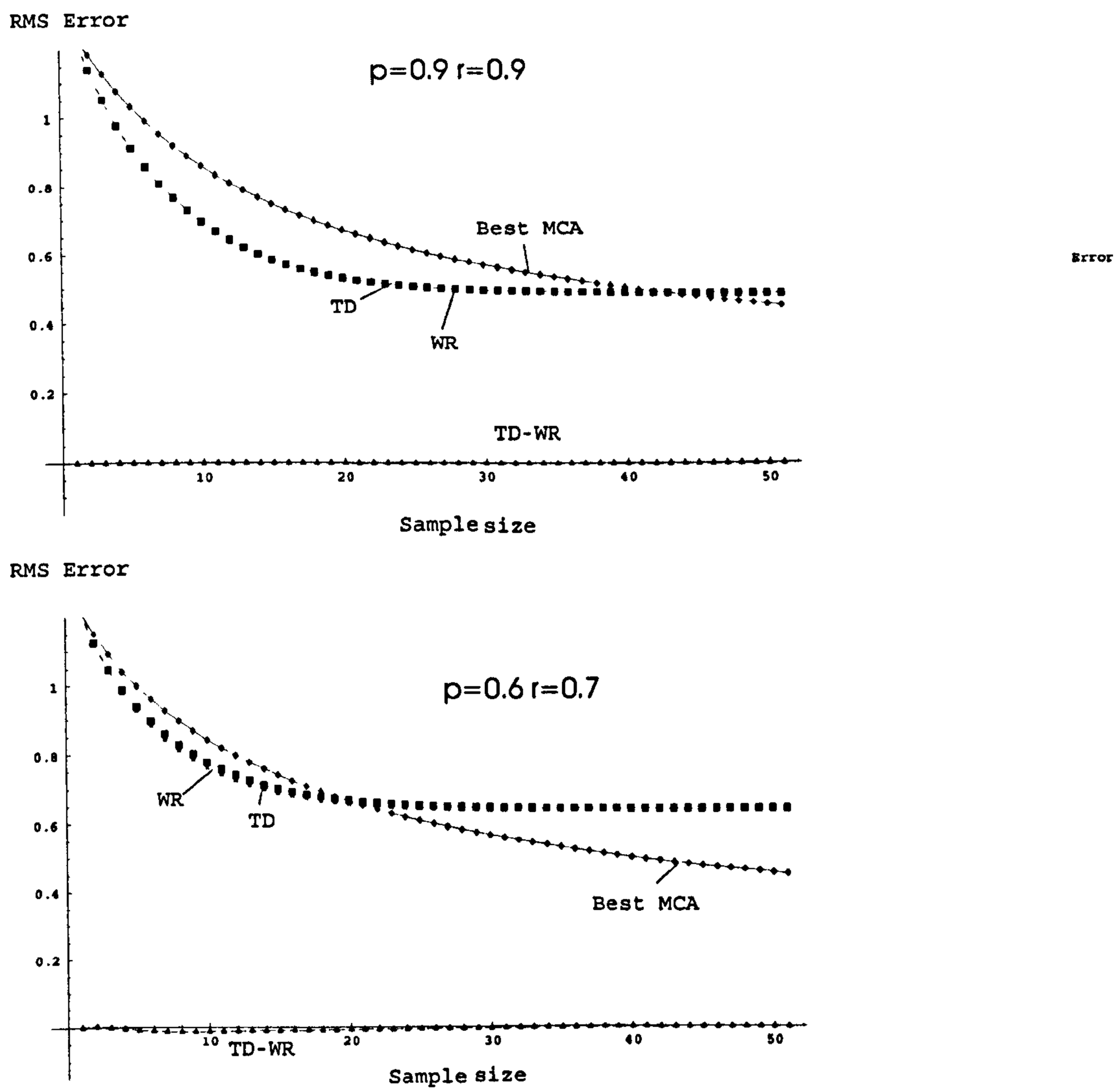


Figure 68: RMS curves for TD for λ =0.8 and α=0.6 and for WR optimal λ and α (using p=0.9 r=0.9 and p=0.6 r=0.7)

TD and WR with non-zero initial estimate

When the estimators are started off with non-zero initial estimates the pattern observed in the case of the SRW model is encountered again. Setting the initial estimate to two doubles the initial bias. It reveals that TD performs worse than the WR estimator, presumably because of its use of poor existing estimates. For example the average RMS error for the first ten estimation steps can be seen in Figure 69.

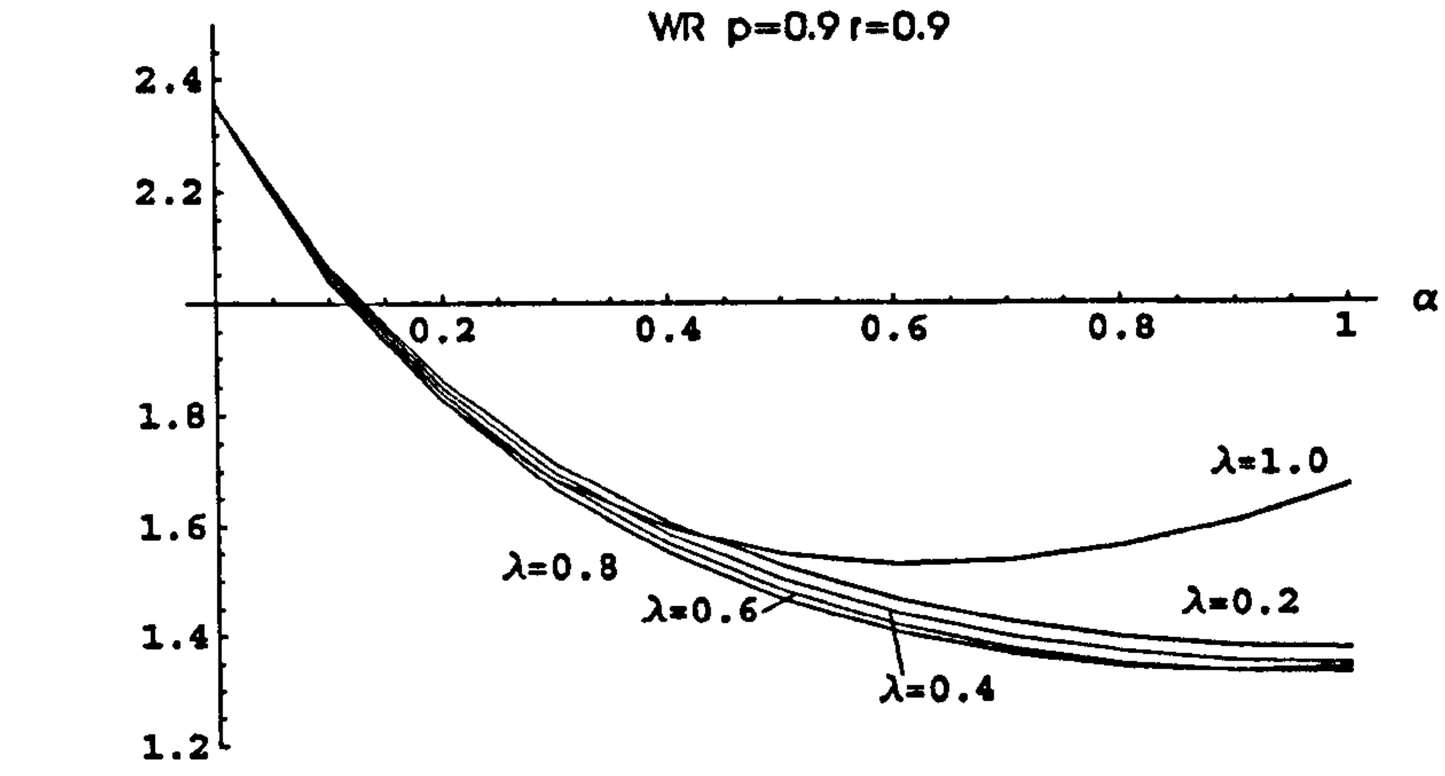


Figure 69: Average RMS error for the first ten steps for TD and WR using p=0.9 r=0.9 with a non-zero initial estimate

The fact that WR is again less affected by the bias can also be seen in the large sample behaviour of the RMS error. Figure 70 shows the RMS curve for λ =0.6 and α=0.2 and comparison with the result in Figure 67 using a zero initial estimate that WR initially outperforms WR and MCA and it takes time for TD to catch up. This behaviour is typical and is again suggestive of the fact that TD takes longer to recover from poor initial estimates than WR or MCA.

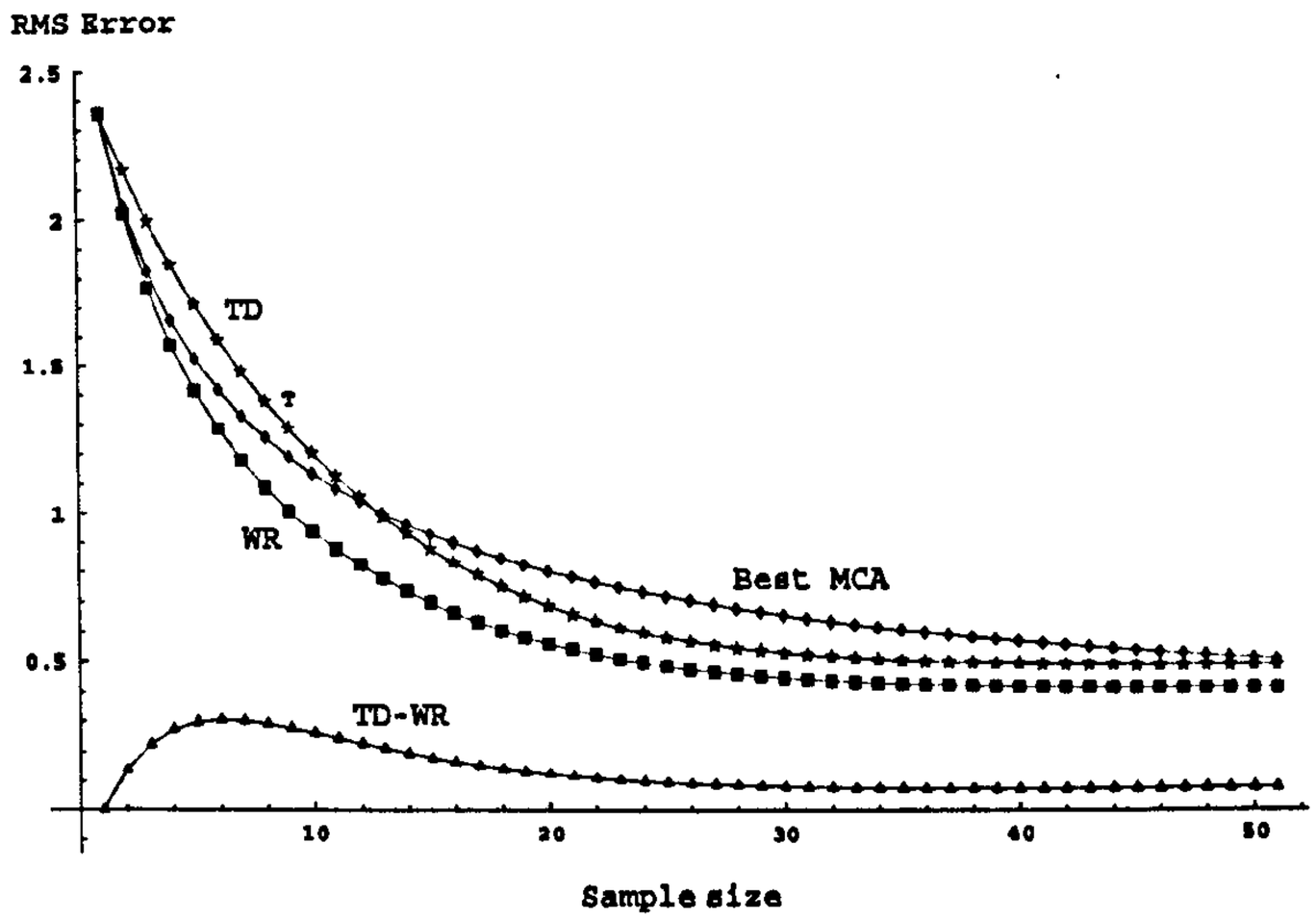


Figure 70: RMS curve for TD and WR for λ =0.8 and α=0.6 with non-zero initial estimate (p=0.9 r=0.9)

Summary of the bottleneck model

The bottleneck model is practically interesting for values of p and r that produce reasonably large differences in expected reward at each state. For these values TD does outperform the simple MCA estimator for a range of λ values. However, irrespective of the model parameters, TD and WR are close for large λ and small α. When TD shows a small sample advantage over the MCA estimator so does the WR estimator. This strongly suggests that any advantages of the TD estimator in the early stages can be attributed to WR learning. As in the case of the SRW and the cyclic model, this effect becomes all the more clear when the estimators are started from a non-zero initial estimate. In this case TD only shows an advantage over MCA later in the procedure and this again tends to confirm that any temporal difference learning is effective later rather than earlier in the estimation procedure.

Conclusion

The additional models, cyclic and bottleneck, confirm the findings about the behaviour of TD and WR in the SRW. In all cases TD and WR are initially close and when TD performs well early on so does WR, and they are close. When the initial estimate is non-zero and the initial bias is higher TD does worse than WR for small samples and only recovers later.

Overall it seems reasonable to suppose that the small sample advantage of the TD estimator is due to the action of weighted rewards rather than temporal difference learning. This is reasonable philosophically, because temporal difference learning needs good estimates to be effective and these are least likely to be available early on. It is also borne out by the behaviour of the estimators in diverse models that show a range of behaviours – cyclicity, path length, path length reward relationship, determinism and bottleneck structure.

Chapter Nine

Replace and Accumulate TD and WR

All of the results obtained so far have compared the First visit form of TD and WR. In this chapter we look at the interaction between TD and WR in the way that the data from a realisation is used. The standard Accumulate and Replace traces form of TD are compared to their equivalent WR estimators, i.e. Every visit WR and Last visit WR, for each of the three models, SWR, Cyclic and Bottleneck.

Analytic RMS curves for Every and Last visit WR

The methods of Chapters Four and Six cannot be used to derive an analytic expression for WR in the Every visit and Last visit forms. This makes the analysis of Every and Last visit WR and their comparison with Accumulating and Replacing traces TD more difficult. However, it is possible to modify the analysis and the resulting program given in Singh and Dayan (1988) to compute the analytical RMS curves for both versions of WR.

Analytic form of RMS error of Replace WR

Replace TD is given by, using the notation in Singh and Dayan (1988):

$$v_i(t) = v_i(t-1) + \alpha\Big(\sum_{d=1}^{\kappa_i(t)-1} \sum_{m=n_i(t;d)+1}^{n_i(t;d+1)} (1-\lambda)\lambda^{m-n_i(t;d)-1} v_{s_m}(t-1) + \sum_{m=n_i(t;\kappa_i(t))+1}^{\tau(t)} (1-\lambda)\lambda^{m-n_i(t;\kappa_i(t))-1} v_{s_m}(t-1) + \lambda^{\tau(t)-n_i(t;\kappa_i(t))} r(t) - \kappa_i(t) v_i(t-1)\Big)$$

The corresponding WR estimator simply ignores the non-reward values:

$$v_i(t) = v_i(t-1) + \alpha(\lambda^{\tau(t)-n_i(t;\kappa_i(t))} r(t) - K_i(t) v_i(t-1))$$

This is clearly a Last visit form of the WR estimator, i.e. the WR estimator is calculated as if the realisation started with the last occurrence of the state. Thus "Replace WR" is the same as "Last visit WR".

Last visit WR is not the same as First visit WR despite the fact that a first visit is the same as a second visit or in general an nth visit due to the Markov property. The reason is simply that a last visit isn't equivalent to an nth visit for any fixed n. At each realisation the last occurrence will have $n = \kappa_i(\tau)$, i.e. the number of occurrences of the state, at the end of the realisation. Also the statistics of the realisation from the last occurrence of a state is different because the state does not recur, i.e. the passage to the terminal state is without state i recurring.

Even though the mean and variance of WR computed by a Last visit estimator aren't easy to derive, it is clear that they follow the same sampling pattern as the First visit estimator. That is, the RMS error of the Last visit WR estimator is:

$$RMS^2_{E[r_i]}[WR^{last}{}_n(s_i)] = \frac{1-\{(1-\alpha f_i(2-\alpha)\}^n}{(2-\alpha)}\alpha VAR[w_i]^{last} + \{(1-\alpha f_i(2-\alpha)\}^n(WR_0(s_i)-E[w_i]^{last})^2 - (E[WR_n(s_i)]-E[w_i]^{last})^2 + (E[WR_n(s_i)]-E[r_i])^2$$

with

$$E[WR_n(s_i)] = (1-\alpha f_i)^n\{WR_0(s_i)-E[w_i]^{last}\} + E[w_i]^{last}$$

It is clear that given values for $E[w_i]^{last}$ and $VAR[w_i]^{last}$, i.e. the mean and variance of the weighted reward for a given value of λ computed from the last occurrence of the state, then the RMS behaviour of the estimator for values of α can be computed.

To do this it is only necessary to set α equal to 1 and the initial estimate to zero in the formula for Replace TD:

$$v_i(1) = \lambda^{\tau(t)-n_i(t,\kappa_i(t))}r(t)$$

That is, the mean and variance of the First visit estimate starting from an initial estimate of zero with $\alpha = 1$ gives the mean and variance of the Last visit weighted reward at each state. These quantities are computed by Singh and Dayan's (1988) program for the analytic RMS of each of the TD estimators and using the program with $v_i(0) = 0$ and $\alpha = 1$ provides the values needed to use the standard First visit formula given above to compute the RMS error curves for the Replace version, i.e.

Last visit, of WR. By modifying the program to provide the mean and variance, rather than the RMS error, it is possible to use it and the formula for the First Visit WR within a Mathematica notebook to compute the RMS error of the Last visit WR estimator at each step.

The agreement between the theoretical predictions for average RMS per state and simulation is excellent. For example, for λ=0.9 and α=0.7 the empirical (20,000 trials) and predicted RMS curves are very close as shown in Figure 71.

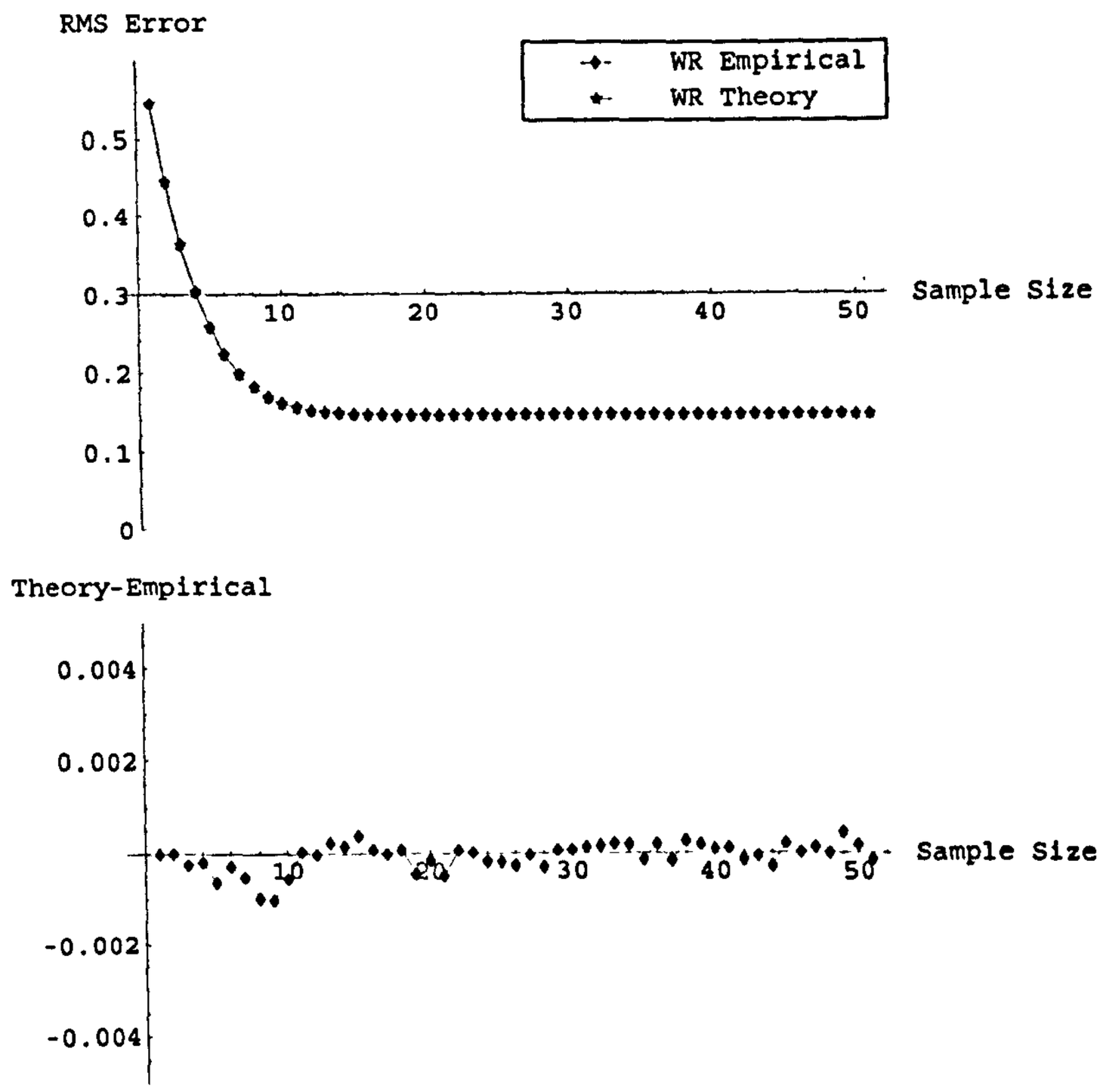


Figure 71: Empirical v theoretical average RMS per state for α=0.4, λ=0.9

Comparison of Replace WR(λ) with Replace TD(λ)

Regions of optimal performance

Using the same Mathematica programs and the modified program derived from Singh and Dayan (1988), analytic RMS error curves for the Replace forms of TD and WR can be compared for each of the models described earlier – SRW, Cyclic and Bottleneck. As Replace TD makes use of a Last visit WR approach we would expect it to do better in any model where the reward was predominantly obtained by direct

transition to a terminal state. If direct reward is more probable than indirect reward the WR weighting would tend to emphasise the short last visit sequences over the less likely longer sequences. As both the Cyclic and Bottleneck models feature single- or two-step transitions to terminal states, Replace TD should perform better than first TD on these models. The other effect to be taken into account is the effect of recurrence or cyclicity causing Replace TD to not converge.

The first thing to investigate is the effect of the replace method on the regions where TD performs better than MCA in the three models. In the case of the SRW the high recurrence rate causes Replace TD to not converge for a large range of α and this results in a very different looking set of optimal RMS curves, see Figure 72. However, it is still clear that there are values of λ in the region 0.8 to 0.9 where it has clear advantage over the MCA $\lambda = 1$ estimator.

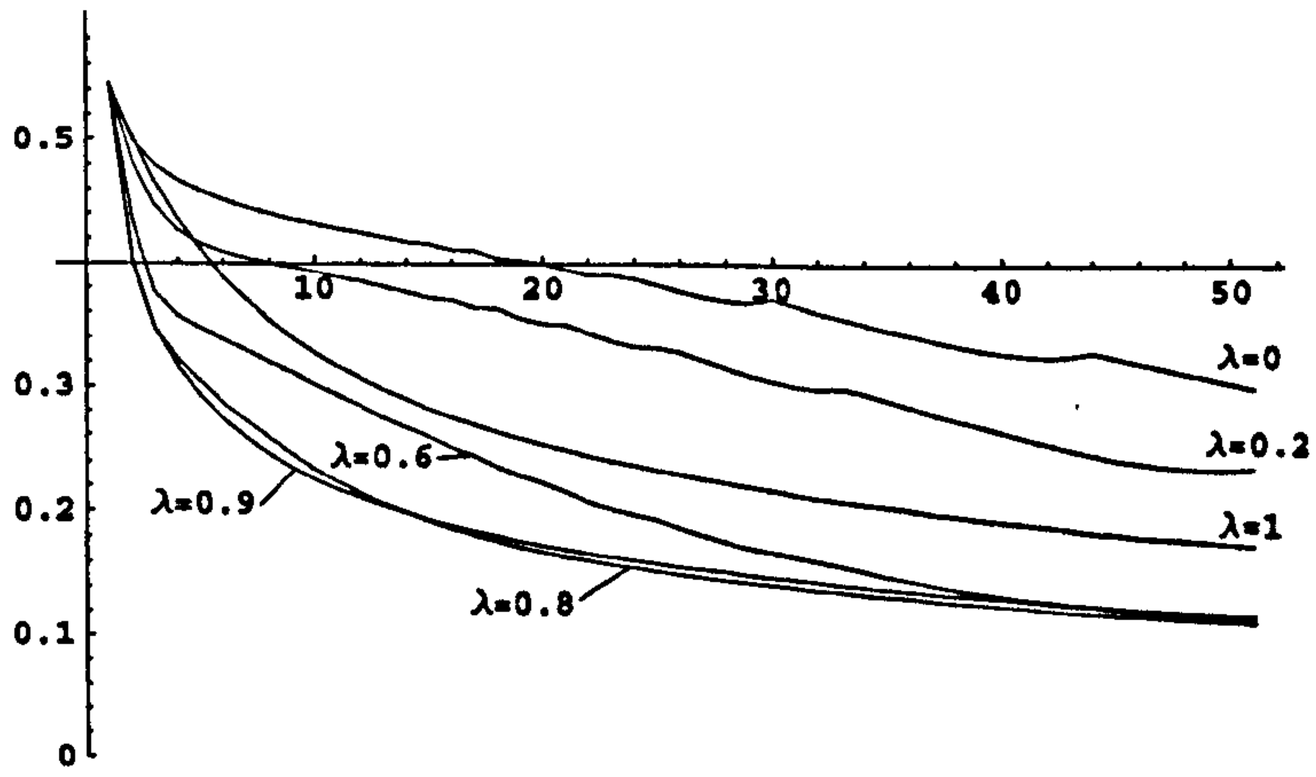


Figure 72: Optimal α Replace TD estimators for the SRW

In the case of the Cyclic model the recurrence rate is lower and Replace TD converges for a much wider range of α. For example, in Figure 73, where c=0.9 and ϕ=1, it is clear that using a Last visit approach has little advantage and Replace TD performs in a similar way to the equivalent MCA estimator for large values of λ.

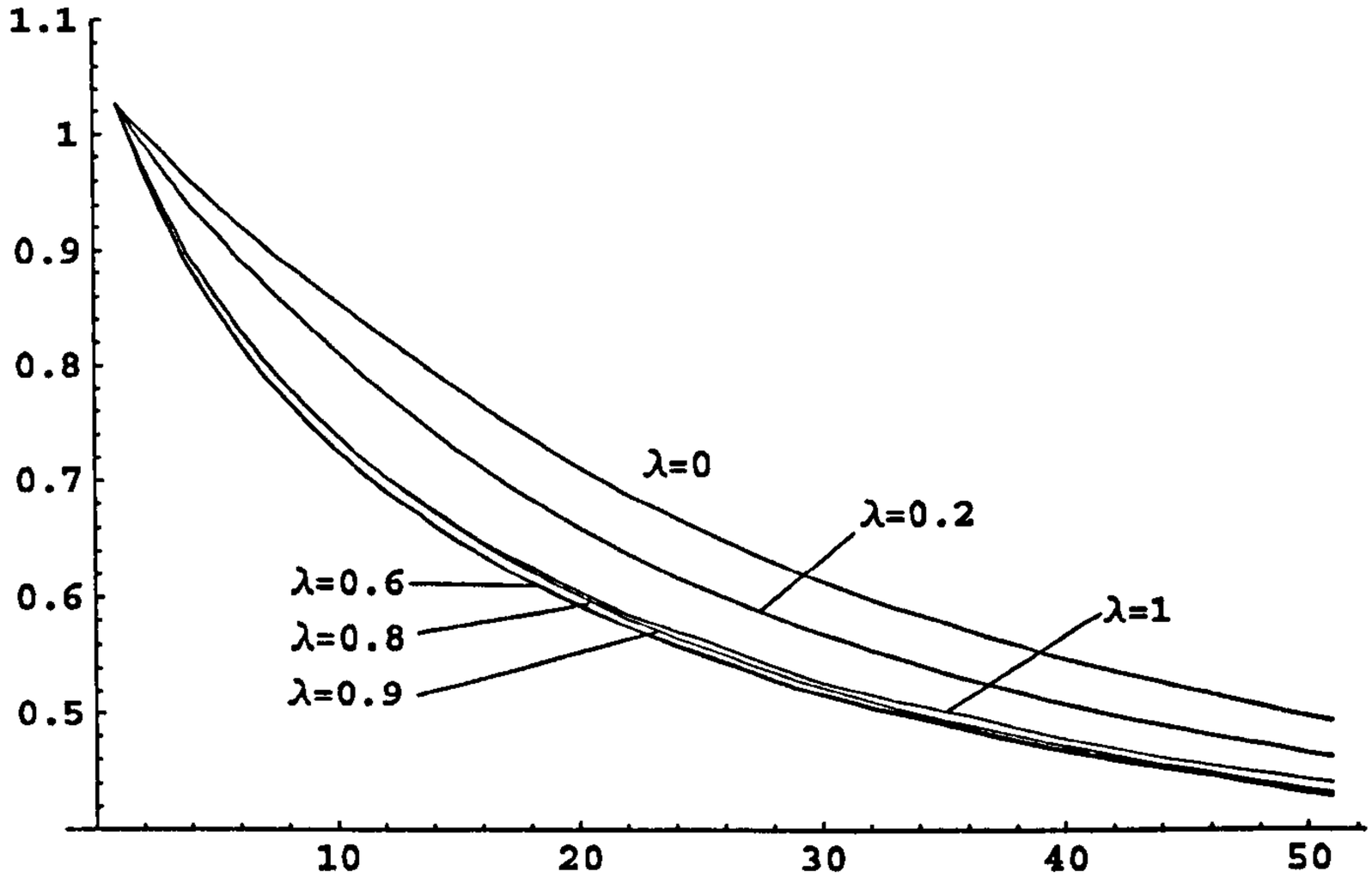


Figure 73: Optimal α TD estimators for the Cyclic model with c=0.9, ϕ=1

In the case of the Bottleneck model the recurrence rate depends on the value of p. For p close to 1 there are large ranges of α for which Replace TD doesn't converge and this can be seen in both charts in Figure 74 as irregularities in the curves. The performance of Replace TD is generally very similar to First visit TD with a small early advantage over the MCA estimator.

Given that the optimal curves are free to use any value of α to obtain a low RMS error the fact that curves for Replace TD and First visit TD are very similar is in agreement with Singh and Dayan's (1988) observation that the effect of cyclicity on differences between First visit, Replacing traces and Accumulating traces is largely a rescaling in the values of λ and α.

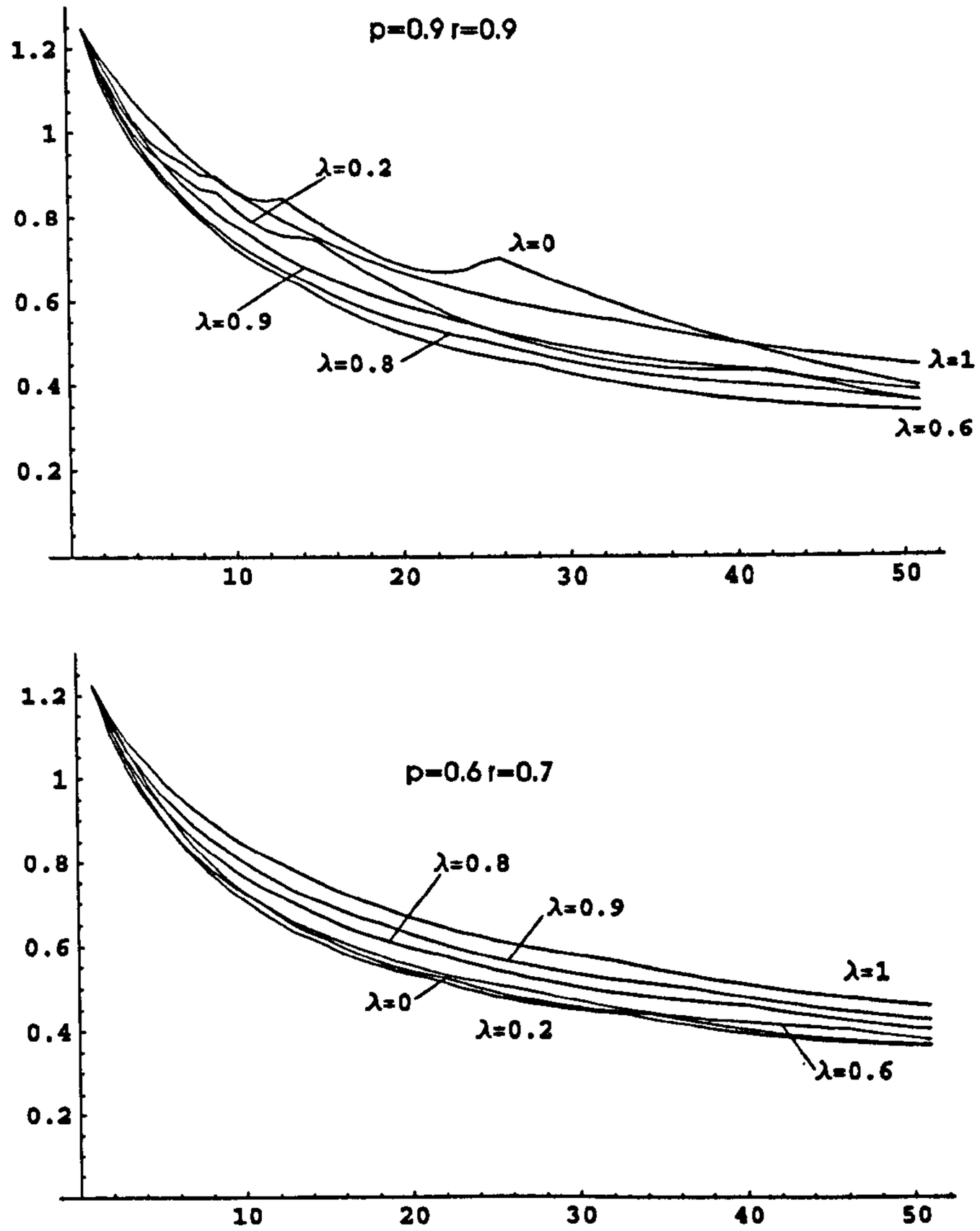


Figure 74: Optimal α Replace TD estimators for the Bottleneck model (using p=0.9, r=0.9 and p=0.6, r=0.7)

Average ten-step RMS error

The average RMS error for a range of λ and α for the Replace (Last visit) WR and TD for the first ten steps for a range of models can be seen in Figure 75. The SRW results suffer from the lack of convergence of Replace TD for a large range of α values. However in the small region where it does converge it can be seen that it is close to Last visit WR. The same observation applies to the other models and we can conclude that where Replace TD does converge it is close to Last visit WR.

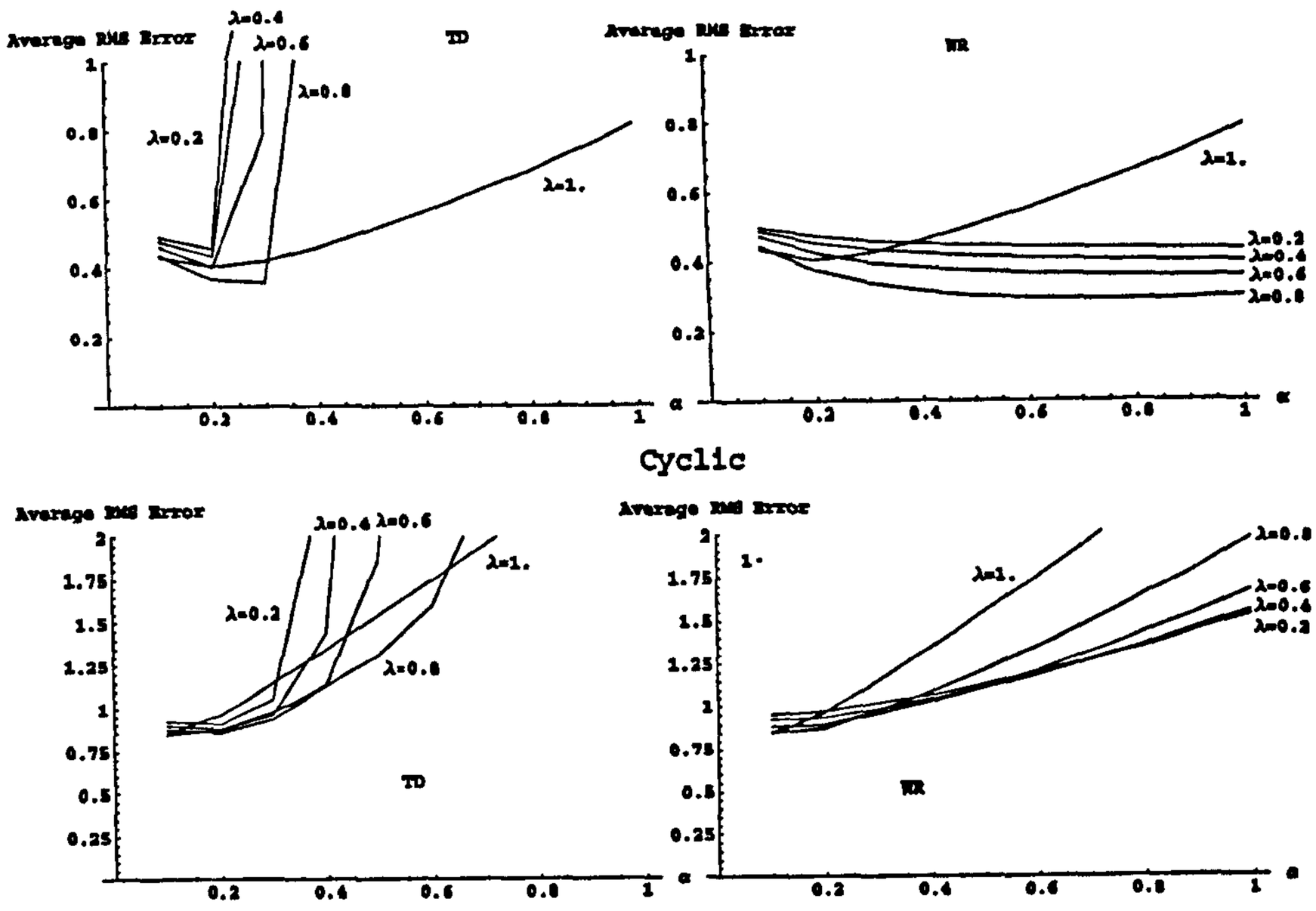
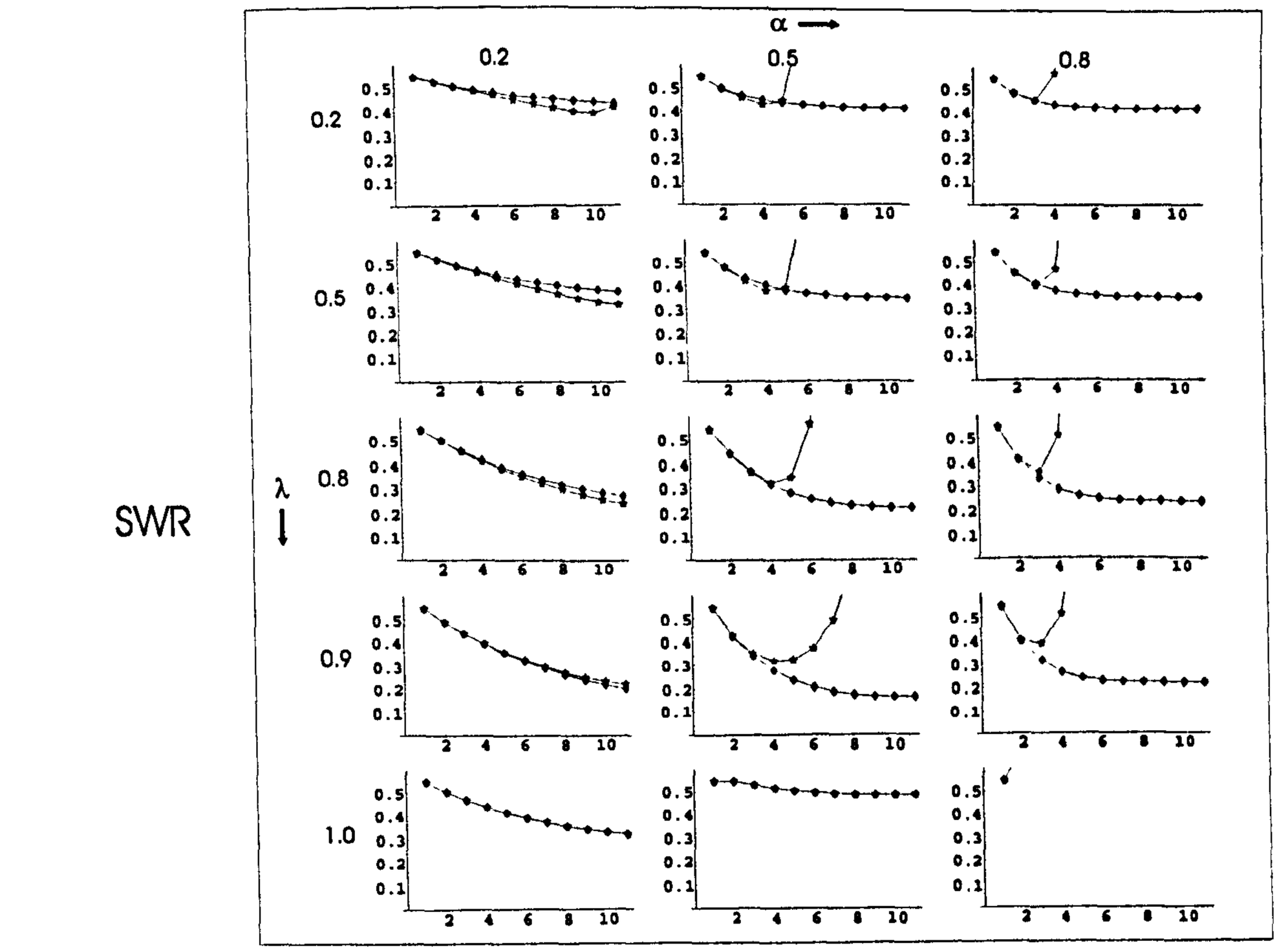


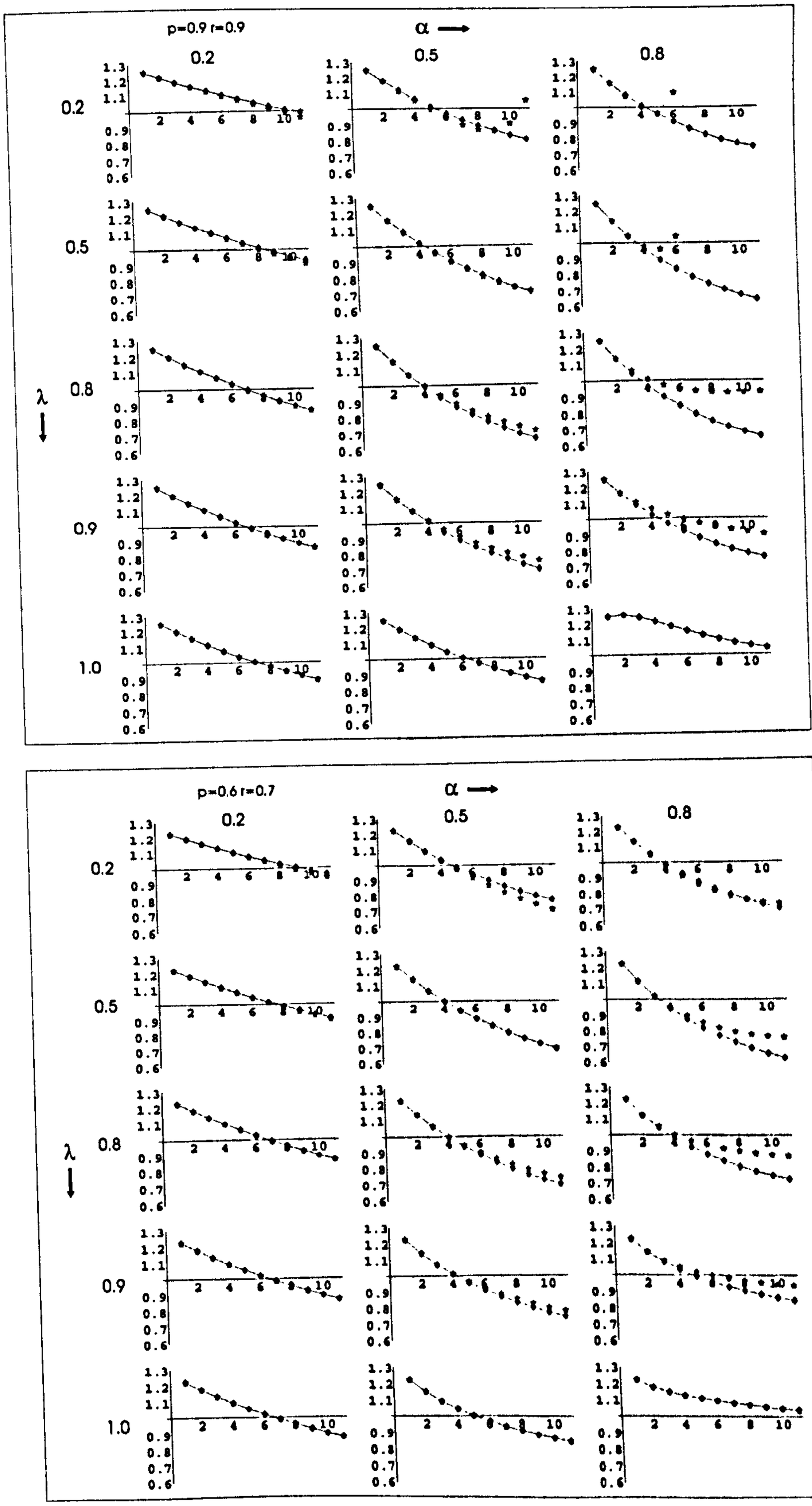
Figure 75: RMS error for Replace TD and Last visit WR for a range of λ and α in SWR, Cyclic and Bottleneck models

Comparing the actual RMS error curves over the first 10 steps of the estimator for each of the models in turn, for a range of λ and α, see Figures 76 and 77, reveals the general behaviour of the two estimators. It shows that, as long as it converges, the Last visit WR estimator is close to the Replace TD estimator.



NB: vertical axis is the per state RMS error; horizontal axis is sample size

Figure 76: Comparison of Replace WR(λ) with TD(λ) over the first 10 steps



NB: vertical axis is the per state RMS error; horizontal axis is sample size

Figure 77: Comparison of Replace WR(λ) with TD(λ) over the first 10 steps

Large sample behaviour

The large sample behaviour in each of the models is made more difficult to study because of the tendency of Replace TD to diverge for some values of λ and α, but in general the behaviour is as described for First visit TD. For example, in Figure 78 Replace TD for λ=0.9 and α=0.2, values for which it converges and performs well, are initially close to the equivalent WR estimator. In this case the WR estimator eventually outperforms the TD estimator, probably because of the additional bias introduced by the non-reward terms.

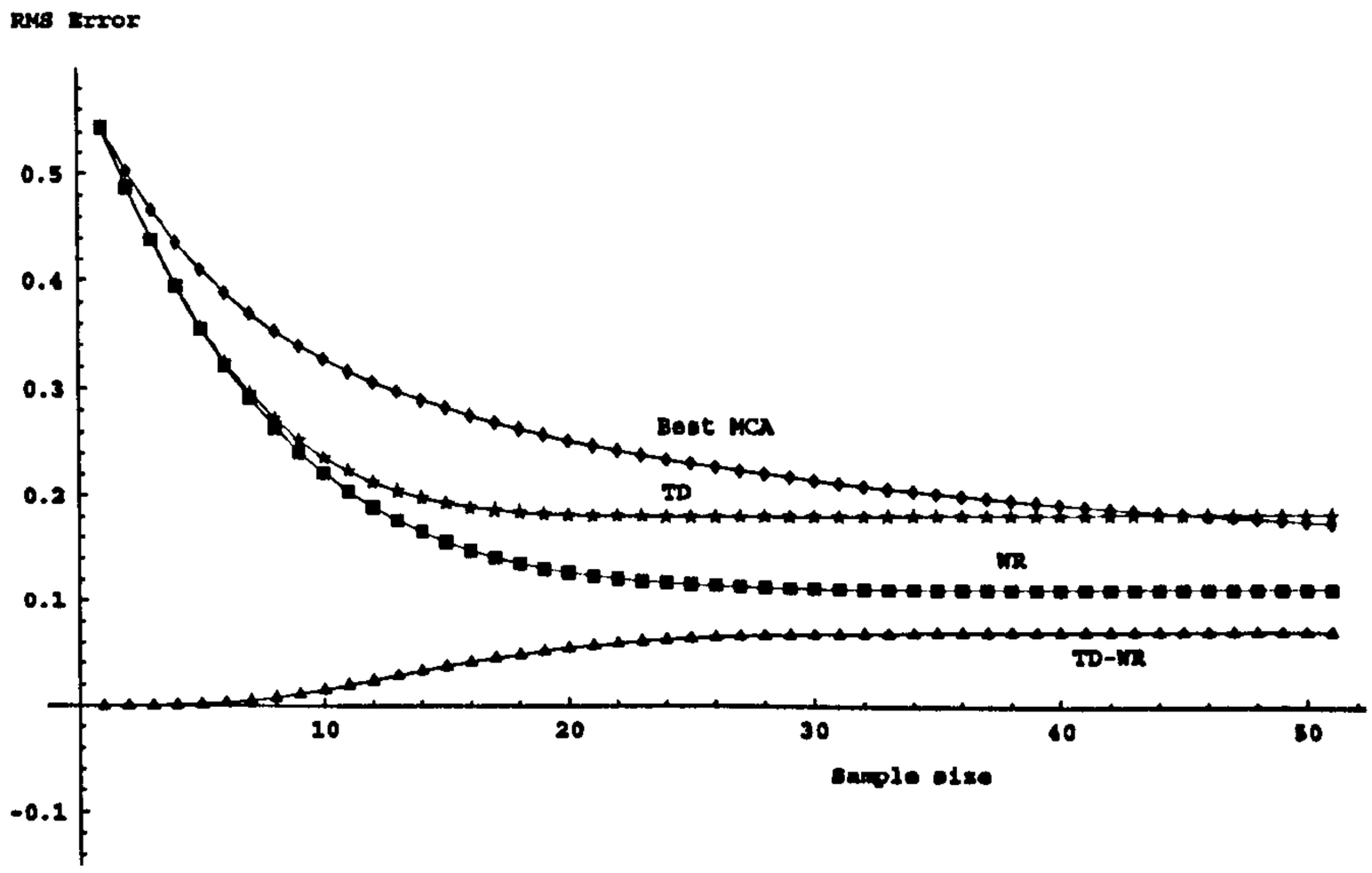


Figure 78: SRW Replace TD and WR for λ=0.9 and α=0.2

For the cyclic model the behaviour is similar but in this case the Replace TD estimator has a smaller asymptotic bias than the WR estimator, see Figure 79. This is presumably because of the reduced recurrence rate compared with the SRW model. As before, the TD estimator is initially close to the WR estimator until they reach their respective asymptotic RMS values.

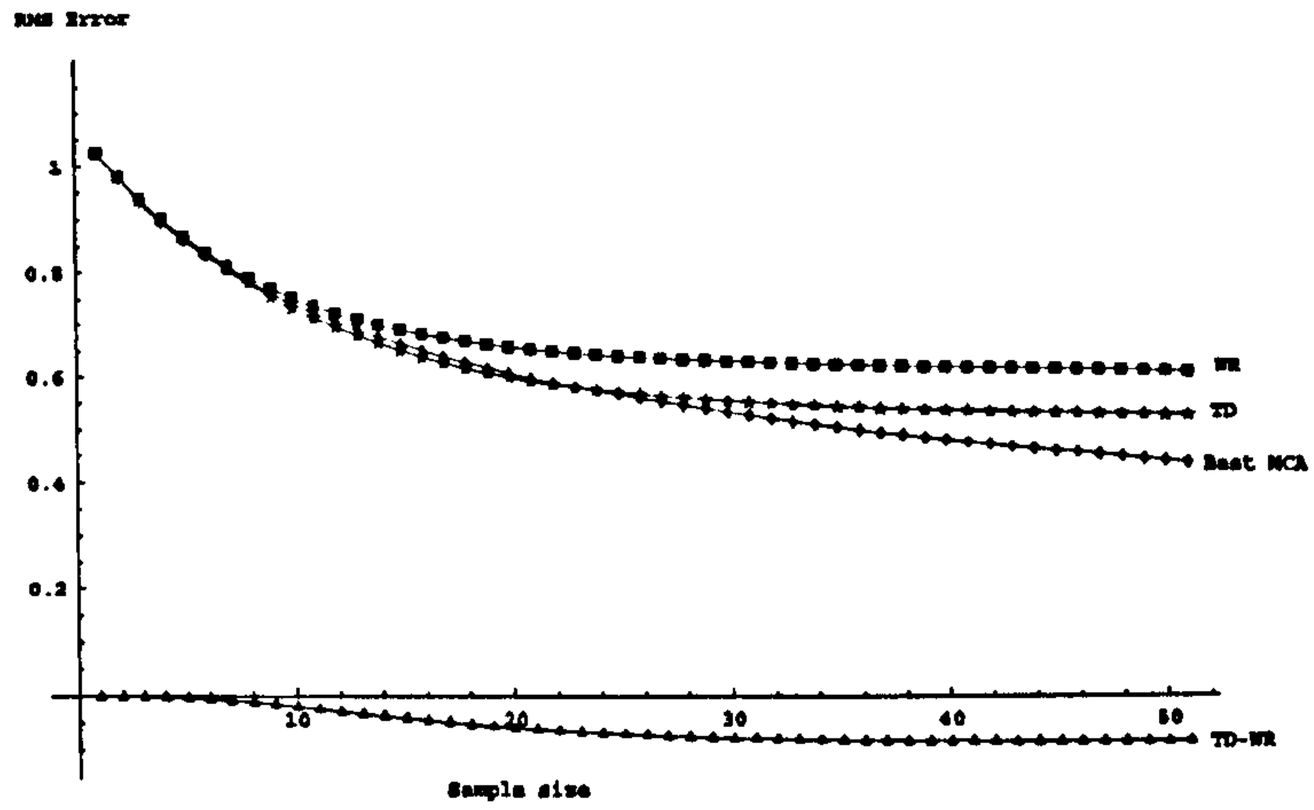


Figure 79: Cyclic Replace TD and WR for λ=0.8 and α=0.2

The Bottleneck model shows the same behaviour. Figures 80 and 81 show that again TD and WR are initially close and then eventually settle to their respective asymptotic RMS values. The example in these figures shows the largest difference between the two estimators.

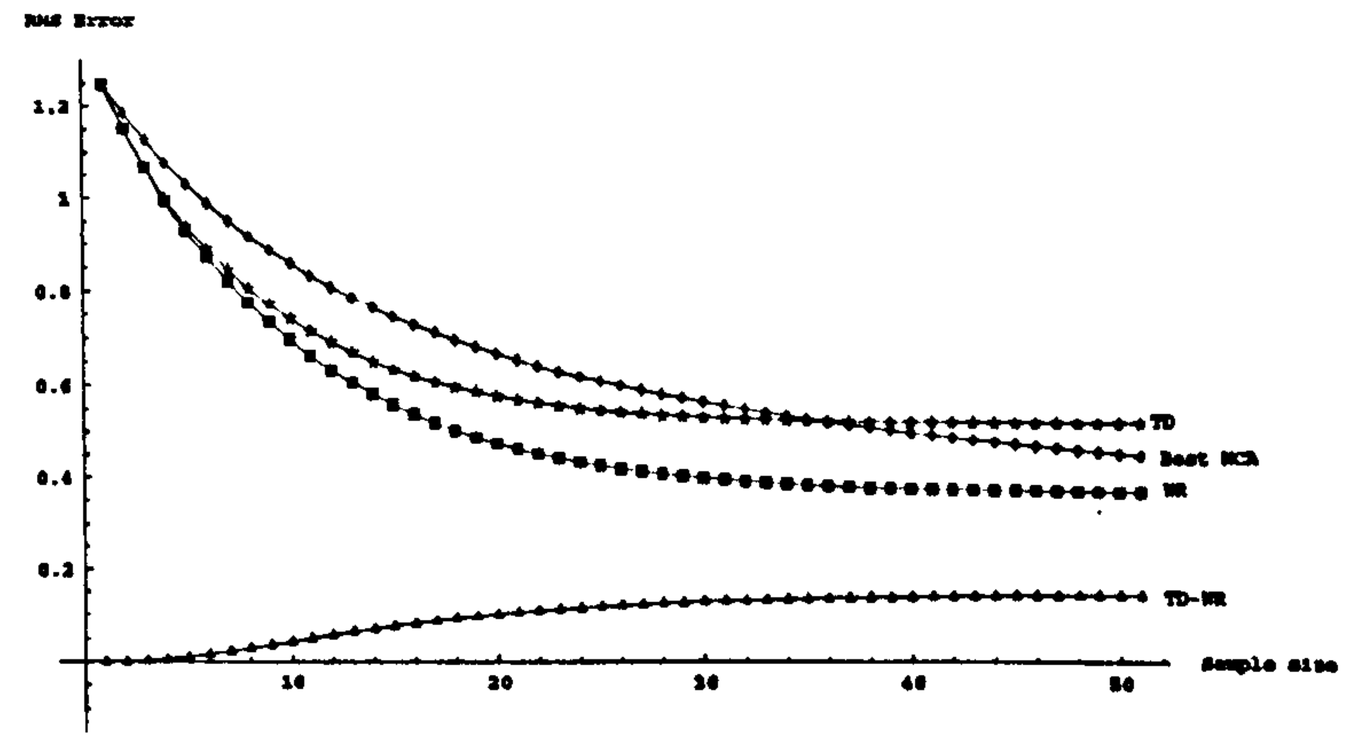


Figure 80: Replace TD and WR for λ=0.8 and α=0.5 (using p=0.9 r=0.9)

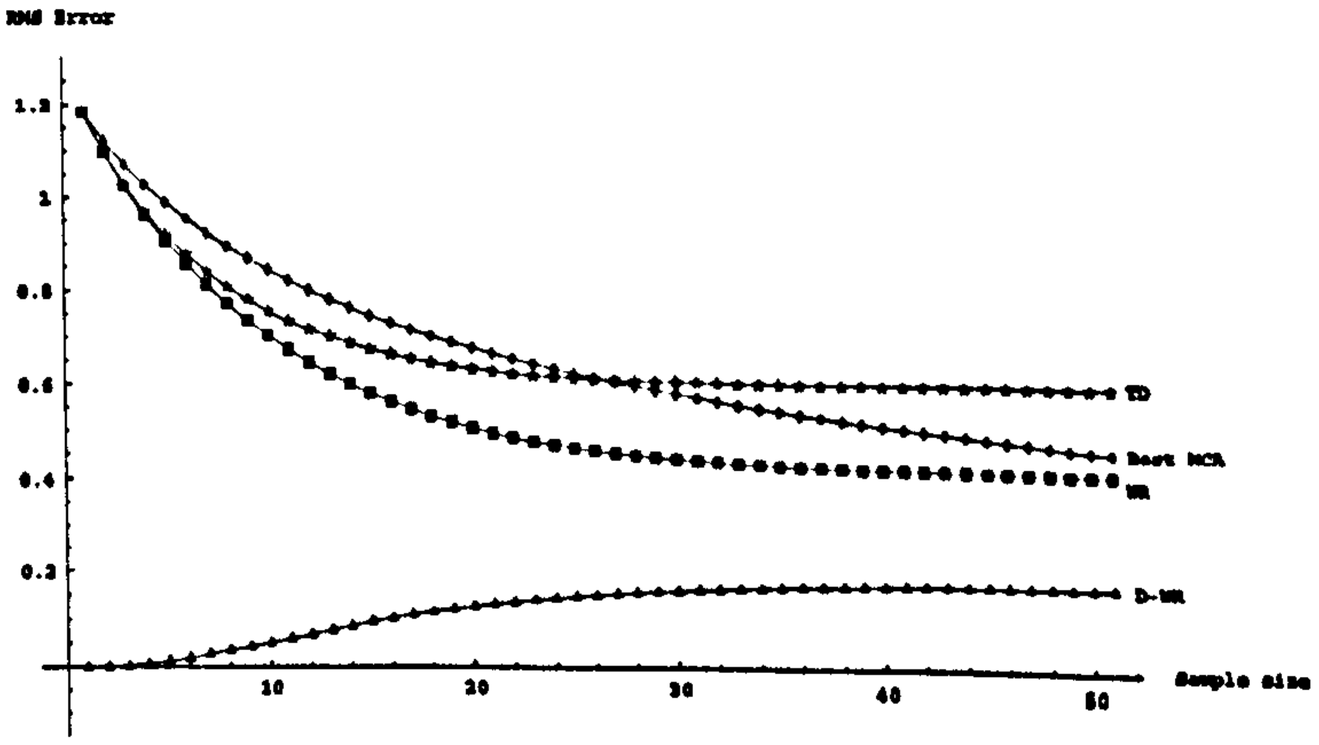


Figure 81: Replace TD and WR for λ=0.8 and α=0.5 (using p=0.6 r=0.7)

Analytic RMS error of Accumulate WR

As already discussed, the methods of Chapters Four and Six cannot be used to derive an analytic expression for WR in the Every visit and Last visit forms. This makes the analysis of Accumulate TD, which corresponds to Every Visit WR when non-reward terms are ignored, more difficult. However, it is possible to modify the analysis and the resulting program given in Singh and Dayan (1988) to compute the analytical RMS curves for Every Visit WR.

Accumulate TD, using the notation in Singh and Dayan (1988), is given by:

$$v_i(t) = v_i(t-1) + \alpha\left(\sum_{n=1}^{\tau(t)} K_i(t;n)\left(\sum_{m=n+1}^{\tau(t)} (1-\lambda)\lambda^{m-n-1} v_{s_m}(t-1) + \lambda^{\tau(t)-n_{\tau(t)}} r(t)\right) - \kappa_i(t) v_i(t-1)\right)$$

The corresponding WR estimator simply ignores the non-reward values:

$$v_i(t) = v_i(t-1) + \alpha\left(\sum_{n=1}^{\tau(t)} K_i(t;n) \sum_{m=n+1}^{\tau(t)} \lambda^{\tau(t)-n_{\tau(t)}} r(t) - \kappa_i(t) v_i(t-1)\right)$$

This is clearly the Every visit form of the WR estimator. Thus Accumulate WR is the same as Every visit WR. The formula for the RMS error of Every Visit WR cannot be obtained by the methods of used earlier and it doesn't follow the same form as for the First visit estimator.

In this case the solution to the problem of obtaining the analytic RMS error is to notice that setting the term $(1-\lambda)$ to zero in Accumulate TD reduces it to Every visit WR:

$$v_i(t) = v_i(t-1) + \alpha\left(\sum_{n=1}^{\tau(t)} K_i(t;n)\left(\sum_{m=n+1}^{\tau(t)} (1-\lambda)\lambda^{m-n-1} v_{s_m}(t-1) + \lambda^{\tau(t)-n_{\tau(t)}} r(t)\right) - \kappa_i(t) v_i(t-1)\right)$$

$$= v_i(t-1) + \alpha\left(\sum_{n=1}^{\tau(t)} K_i(t;n)\lambda^{\tau(t)-n_{\tau(t)}} r(t) - \kappa_i(t) v_i(t-1)\right)$$

As the terms that vanish are all multiplied by $(1-\lambda)$ and these terms appear as $(1-\lambda)$ or $(1-\lambda)^2$ in the final equations for the RMS error of Accumulate TD, the RMS error for Every Visit WR can be obtained by setting these terms to zero even when $\lambda \neq 1$. Applying this procedure to the appropriate version of Singh and Dayan's (1988) program allows us to calculate the RMS error of Every Visit WR.

The agreement between the theoretical predictions for average RMS per state and the simulation is excellent. For example for λ=0.9 and α=0.1, the empirical (20,000 trials) and predicted RMS curves are very close, as shown in Figure 82.

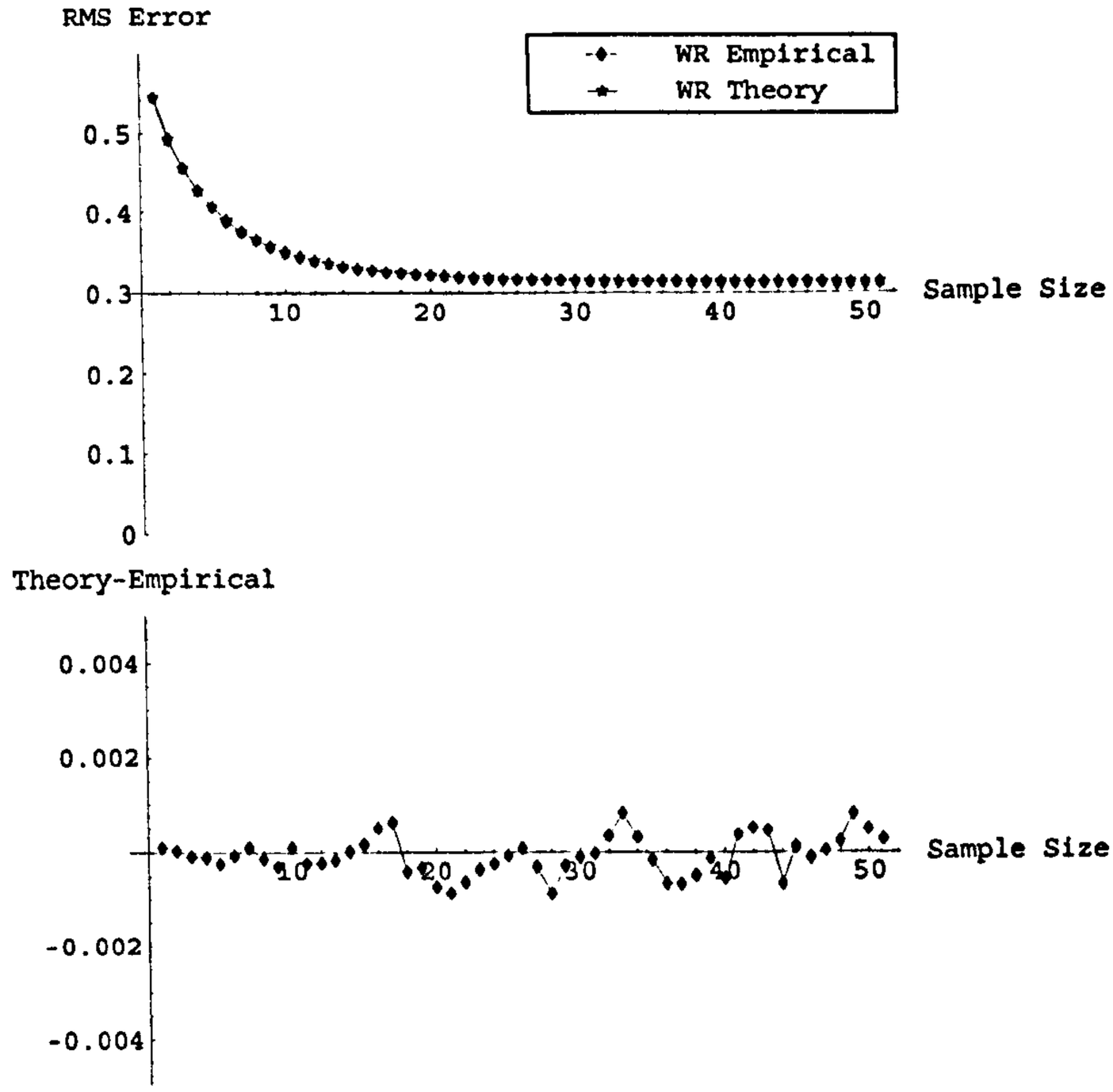


Figure 82: Empirical v theoretical Average RMS per state for α=0.1, λ=0.9

Comparison of Every WR(λ) with Accumulate TD(λ)

Regions of optimal performance

Using the same Mathematica programs and the modified program derived from Singh and Dayan (1988), analytic RMS error curves for the Accumulate forms of TD and WR can be compared for each of the models described earlier – SRW, Cyclic and Bottleneck.

As Accumulate TD makes use of an Every visit WR approach we would expect that any differences between TD and WR to be emphasised due to the multiple updates performed for each realisation. Even so, as the weighted reward component of each update is the same, the estimators should still be close for λ close to 1 and small α. Also, as Accumulate TD tends not to converge for large α in recurrent models and

performs best for values of λ close to 1, any differences that are larger should prove to be in regions where Accumulate TD isn't better than MCA.

The first thing to investigate is the effect of the accumulate method on the regions where TD performs better than MCA in the three models. In the case of the SRW the high recurrence rate causes Accumulate TD to not converge for a large range of α and this results in the optimal RMS curves in Figure 83. However, it is still clear that there are values of λ where it has clear advantage over the MCA $\lambda = 1$ estimator.

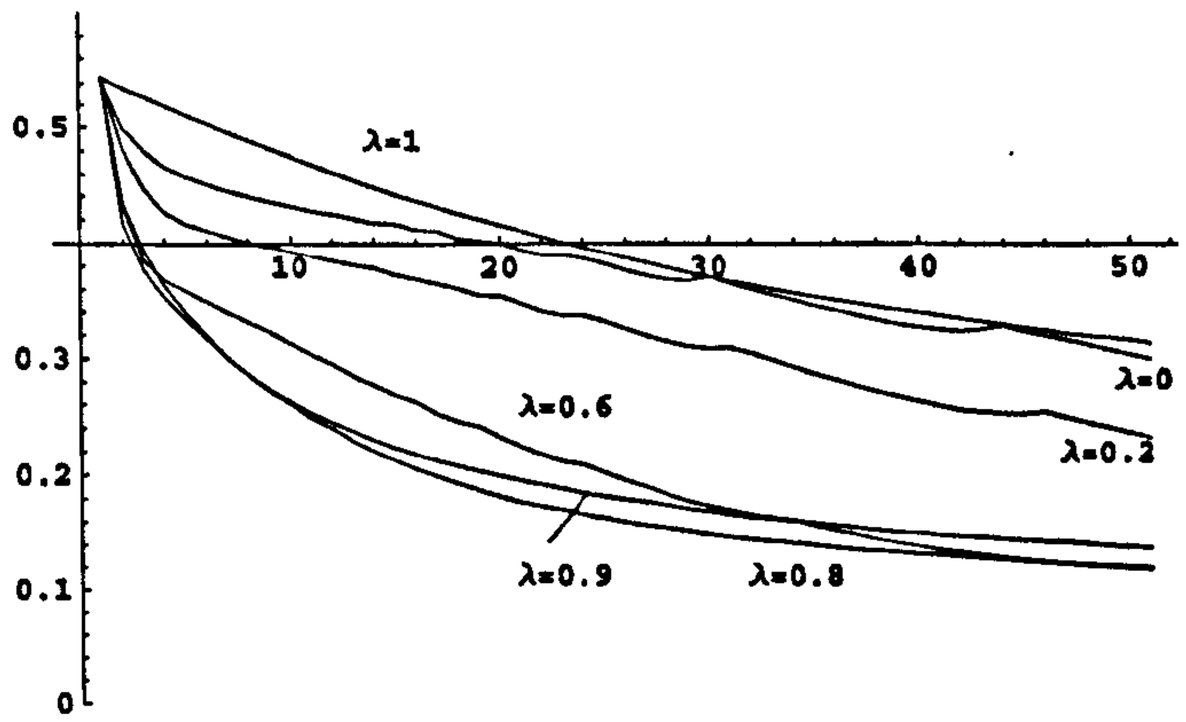


Figure 83: Optimal α Accumulate TD estimators for the SRW

In the case of the Cyclic model the recurrence rate is lower and Accumulate TD converges for a much wider range of α. For example, in Figure 84 where c=0.9 and ϕ=1, it is clear that using an every visit approach has an advantage for this model and Accumulate TD does outperform the equivalent MCA estimator even for small values of λ.

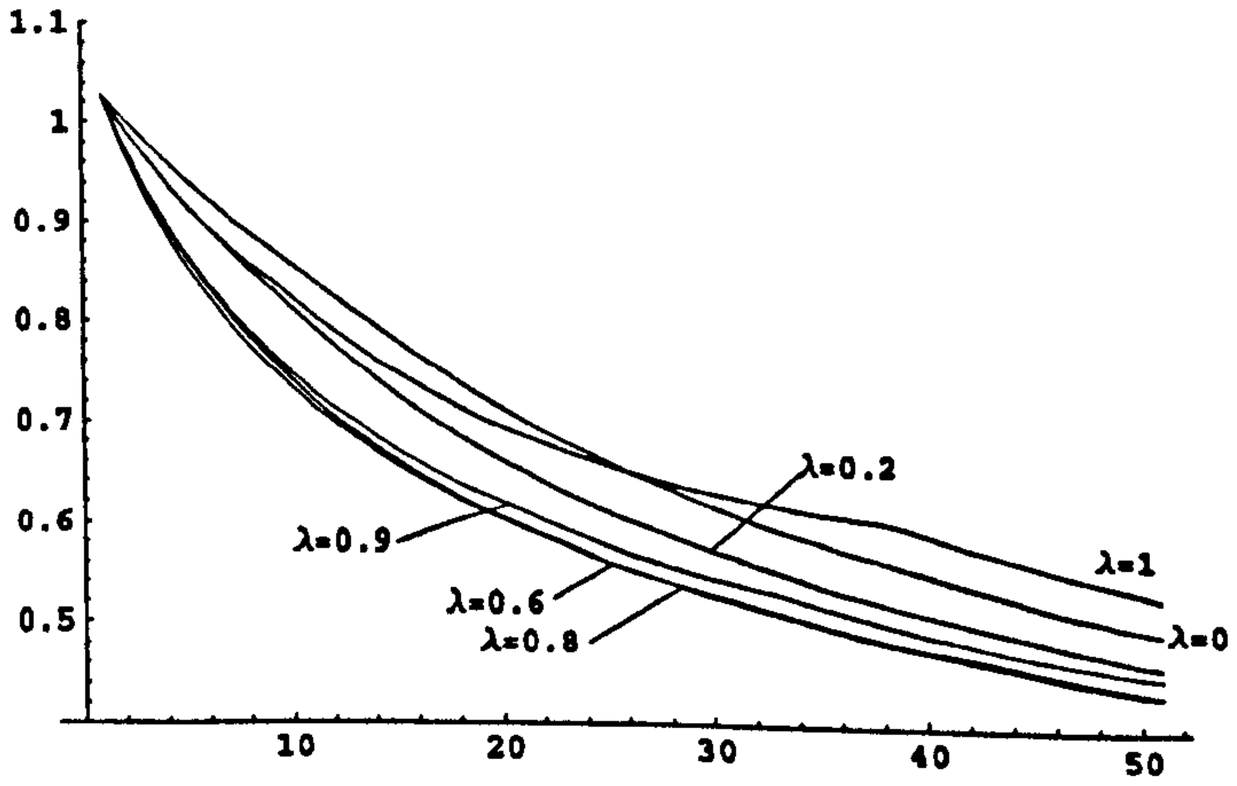
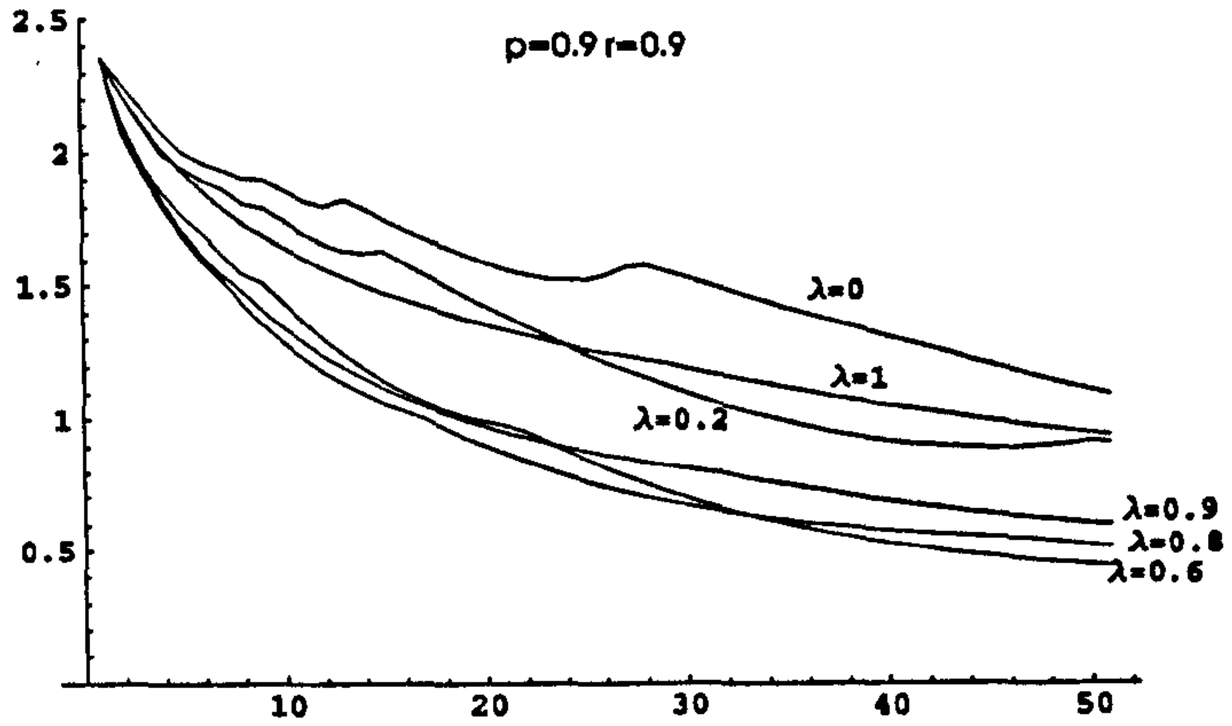


Figure 84: Optimal α TD estimators for Cyclic model with c=0.9, ϕ=1

In the case of the Bottleneck model the recurrence rate depends on the value of p. For p close to 1 there are large ranges of α for which Accumulate TD doesn't converge As can be seen in Figure 85, the performance of Accumulate TD is generally very similar to First and Replace TD with a small early advantage over the MCA estimator.



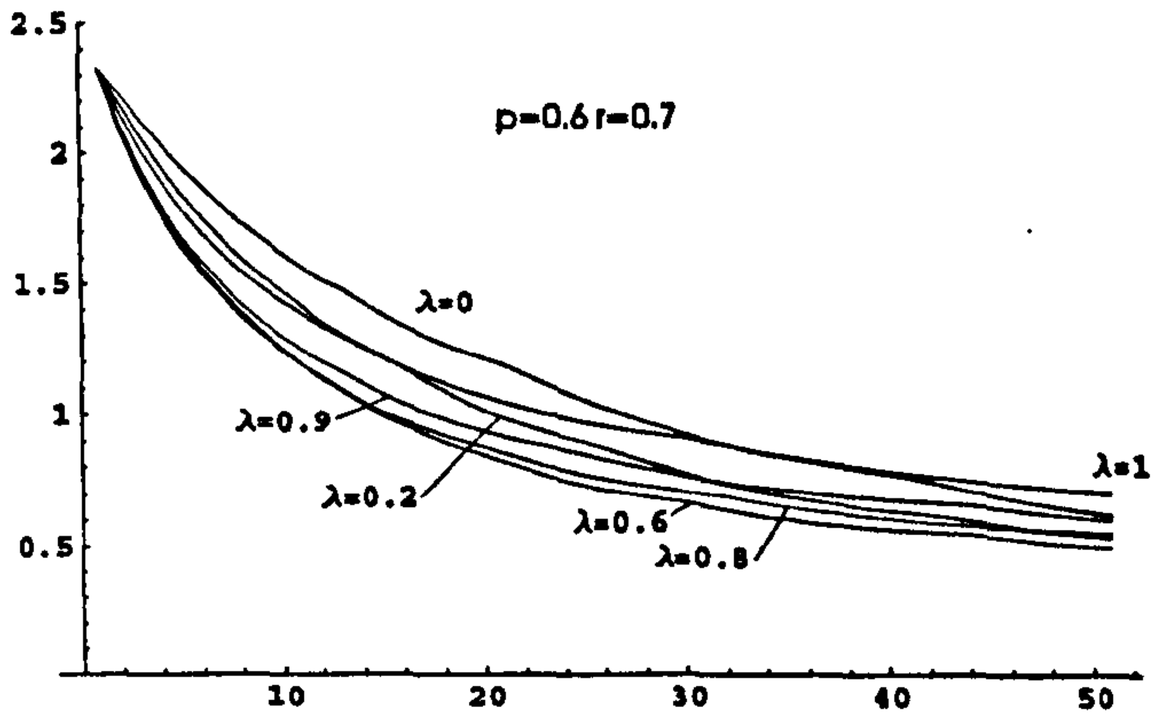


Figure 85: Optimal α Accumulate TD estimators for the Bottleneck (using p=0.9, r=0.9 and p=0.6, r=0.7)

Again the similarities between the optimal RMS curves between Accumulate, Replace and First TD is further confirmation of the observation in Singh and Dayan (1988) that changing the way recurrence is used in TD estimation mainly results in a change in the effective α and λ.

Average ten-step RMS error

The average RMS error for a range of λ and α for the Accumulate (Every visit) WR and TD for the first ten steps for a range of models can be seen in Figure 86. The SRW results suffer from the lack of convergence of Accumulate TD for a large range

of α values and are only plotted for α=0 to 0.3. However, in the small region where they do converge, it can be seen that the trend is similar to Every Visit WR. The same observation applies to the other models and we can conclude that where Accumulate TD does converge it is close to Every Visit WR.

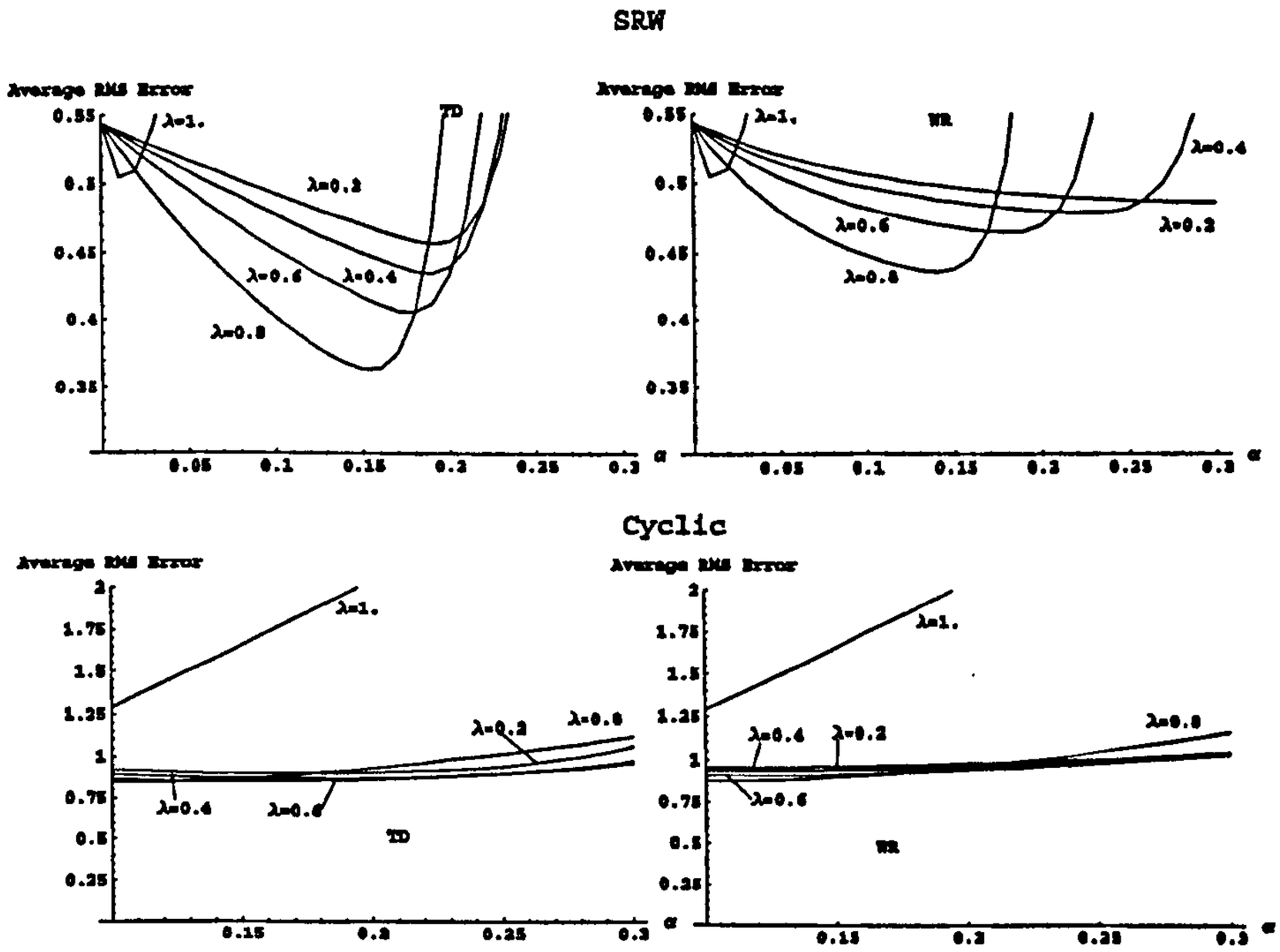
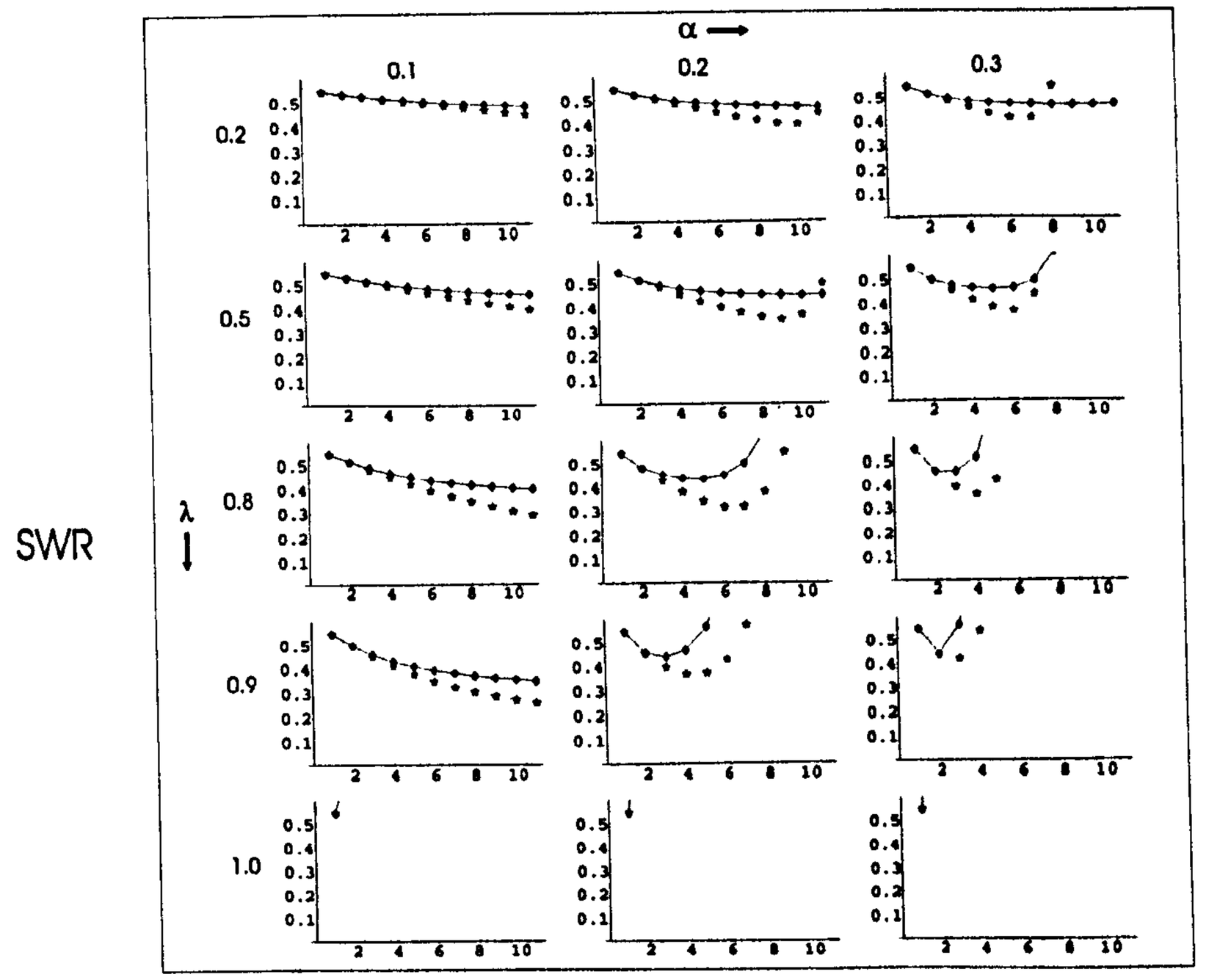
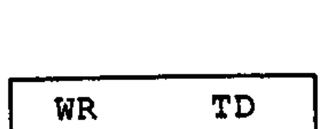
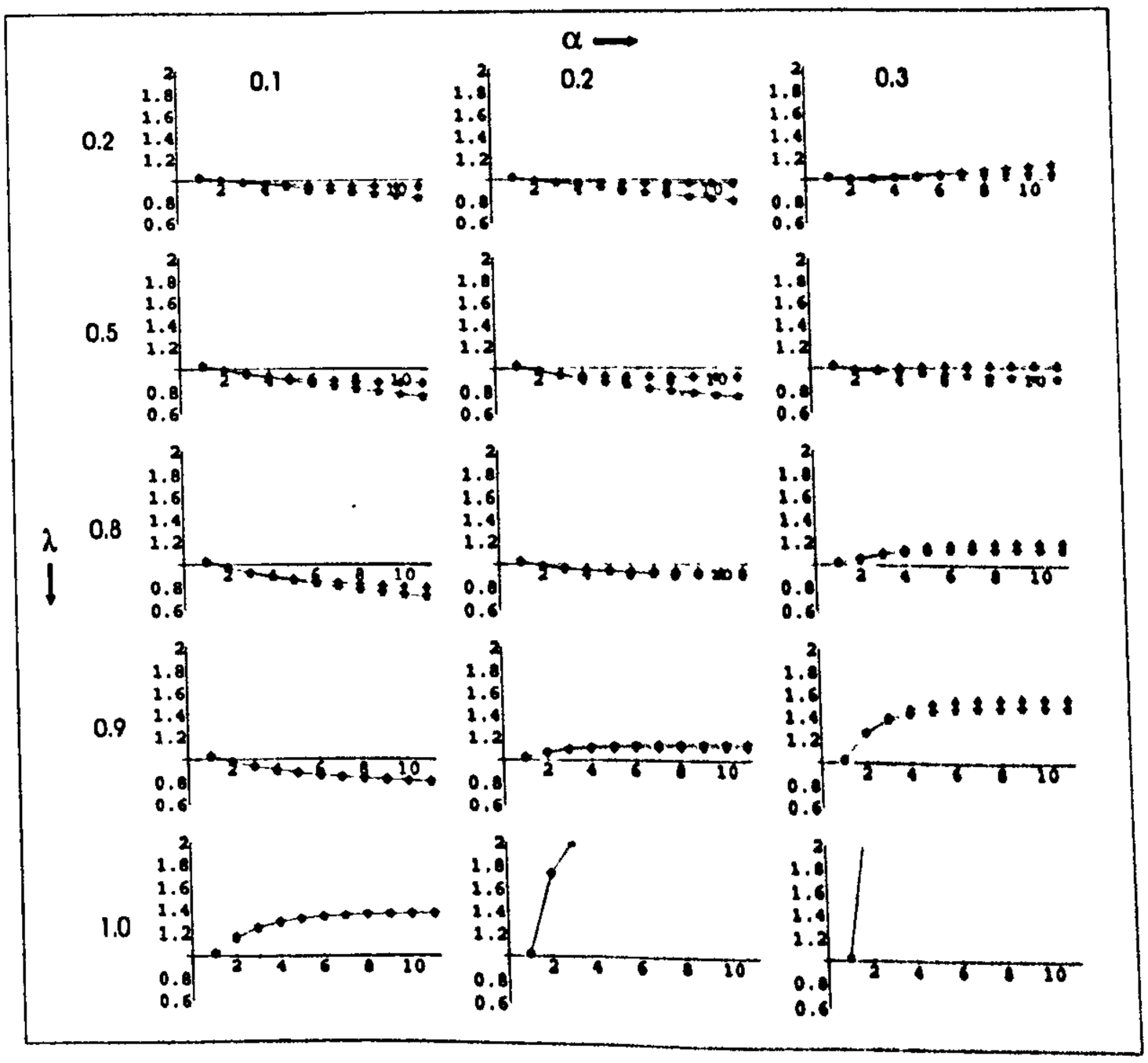


Figure 86: RMS error for Accumulate TD and Every Visit WR for a range of λ and α in SWR, Cyclic and Bottleneck models

Comparing the actual RMS error curves over the first 10 steps of the estimator for each of the models in turn, for a range of λ and α, see Figures 87 and 88, reveals the general behaviour of the two estimators. It shows that, as long as it converges, the Every Visit WR estimator is close to the Accumulate TD estimator.



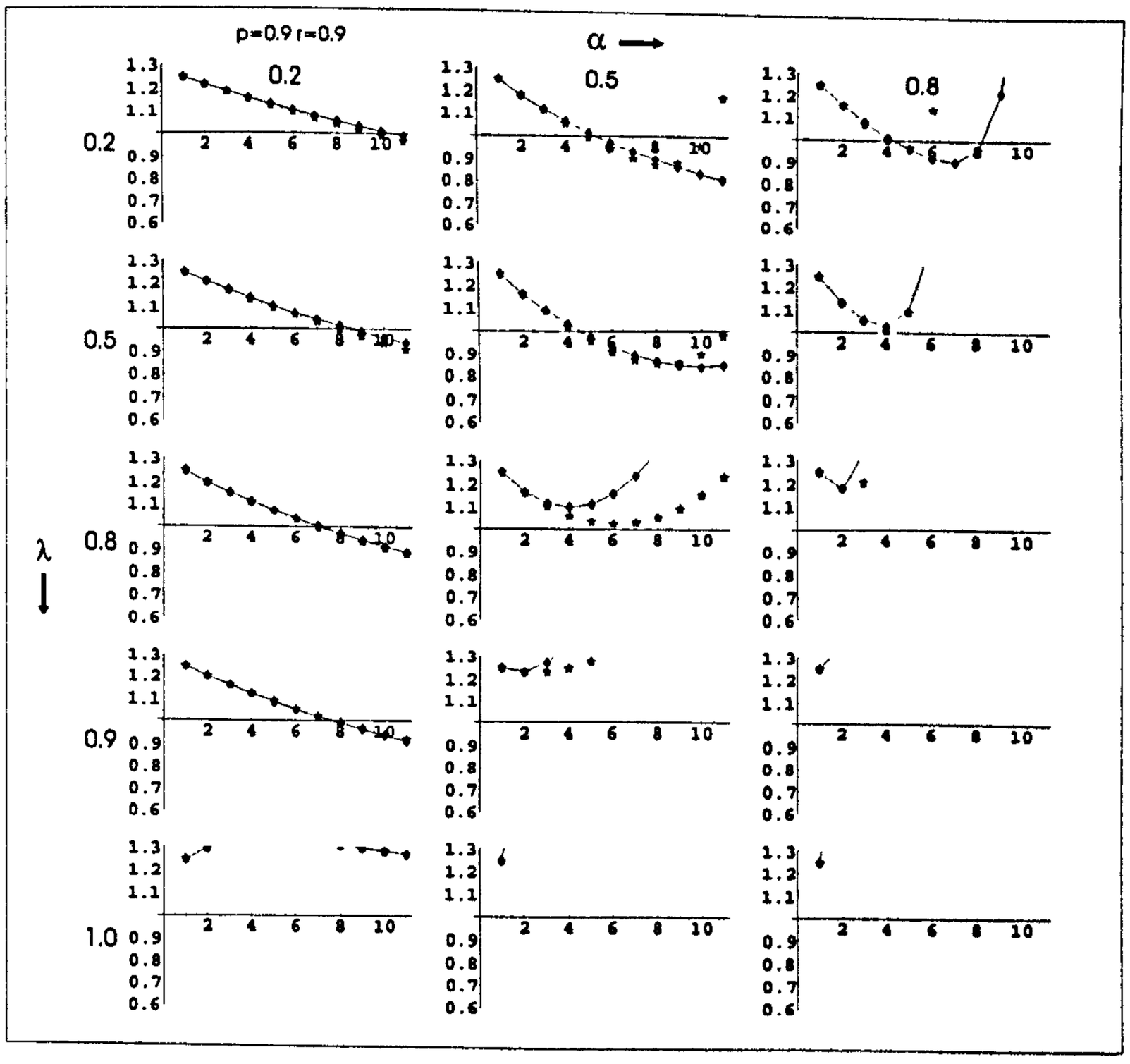




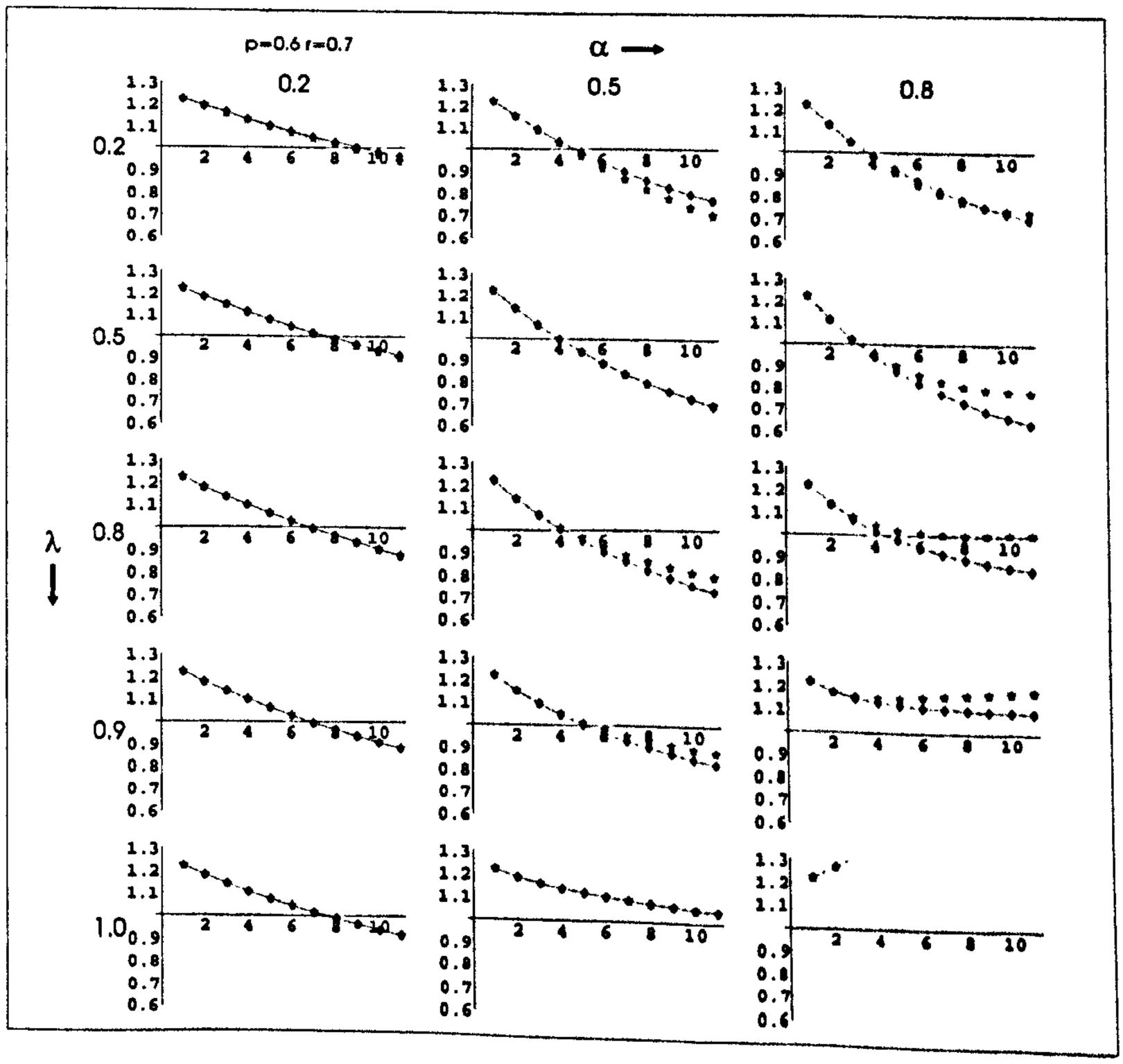
NB: vertical axis is the per-state RMS error; horizontal axis is sample size

Figure 87: Comparison of Accumulate WR(λ) with TD(λ) over the first 10 steps

Bottleneck



Bottleneck



NB: vertical axis is the per-state RMS error; horizontal axis is sample size

Figure 88: Comparison of Accumulate WR(λ) with TD(λ) over the first 10 steps

Large sample behaviour

The large sample behaviour in each of the models is made more difficult to study because of the tendency of Accumulate TD to diverge for some values of λ and α, but in general the behaviour is as described for First and Replace TD if the rescaling of the values of λ and α is taken into account. For example, in Figure 89 Replace TD for λ=0.9 and α=0.1, values for which it converges and performs well, are initially close to the equivalent WR estimator. Unlike the case of the Replace TD estimator, the Accumulate (Every visit) WR estimator has a higher asymptotic RMS error. This could be due to the more effective way that the non-reward terms are averaged to correct the asymptotic bias inherent in using a weighted reward.

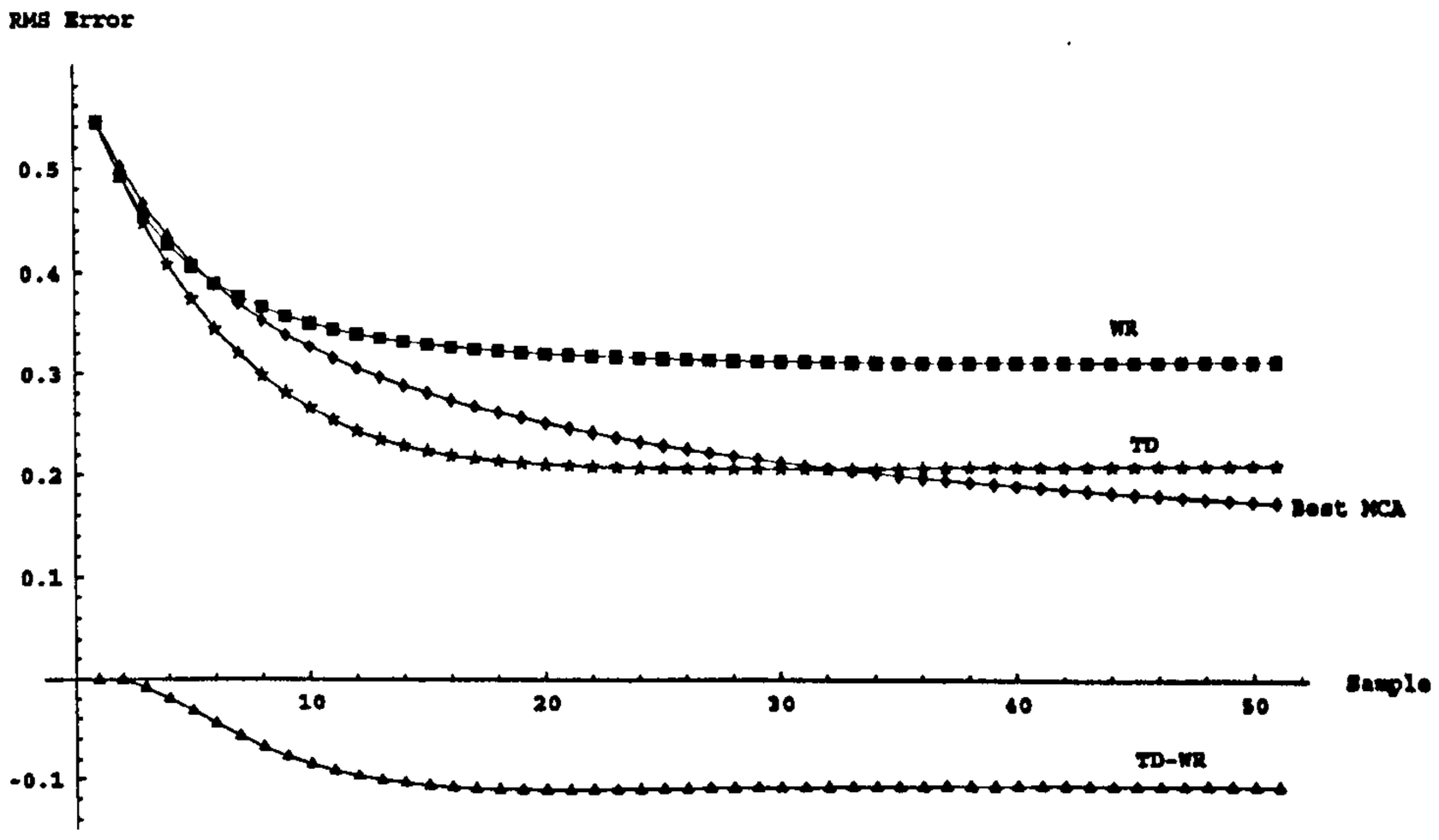


Figure 89: SRW Accumulate TD and WR for λ=0.9 and α=0.1

For the cyclic model the behaviour is similar and again the Accumulate TD estimator has a smaller asymptotic bias than the WR estimator, see Figure 90. As before, the TD is initially close to the WR estimator until they reach their respective asymptotic RMS values.

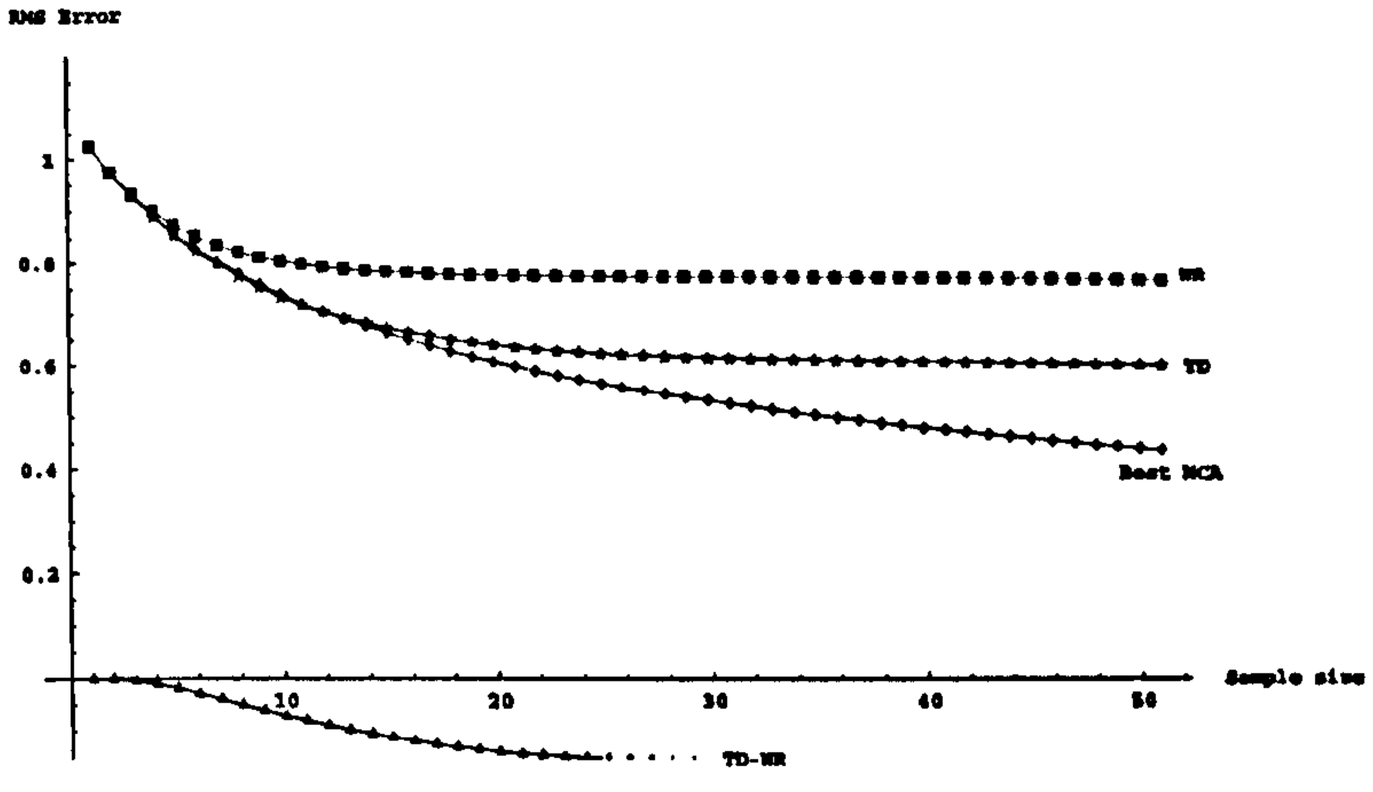


Figure 90: Cyclic Accumulate TD and WR for λ=0.8 and α=0.1

The Bottleneck model shows the same behaviour. Figures 91 and 92 show that again TD and WR are initially close and then eventually settle to their respective asymptotic RMS values.

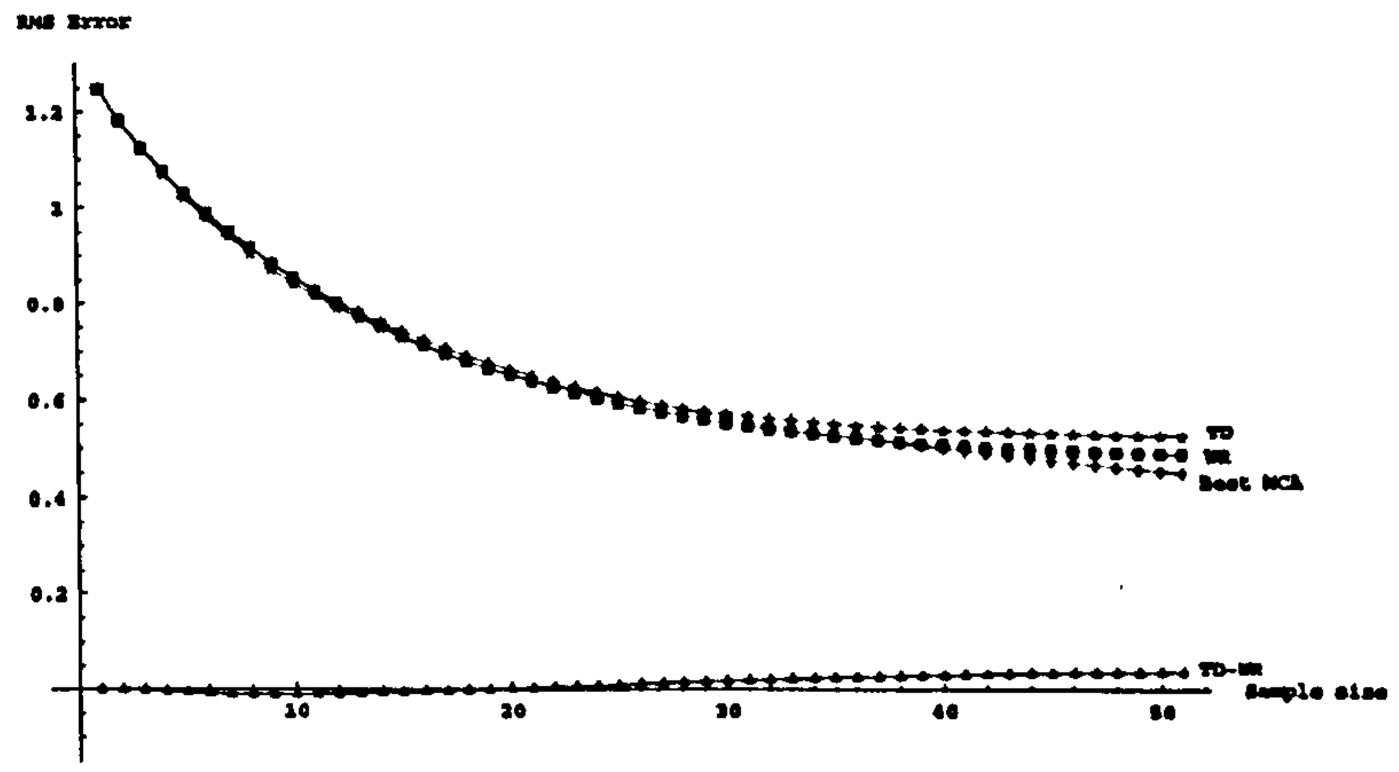


Figure 91: Accumulate TD and WR for λ=0.8 and α=0.3 (using p=0.9, r=0.9)

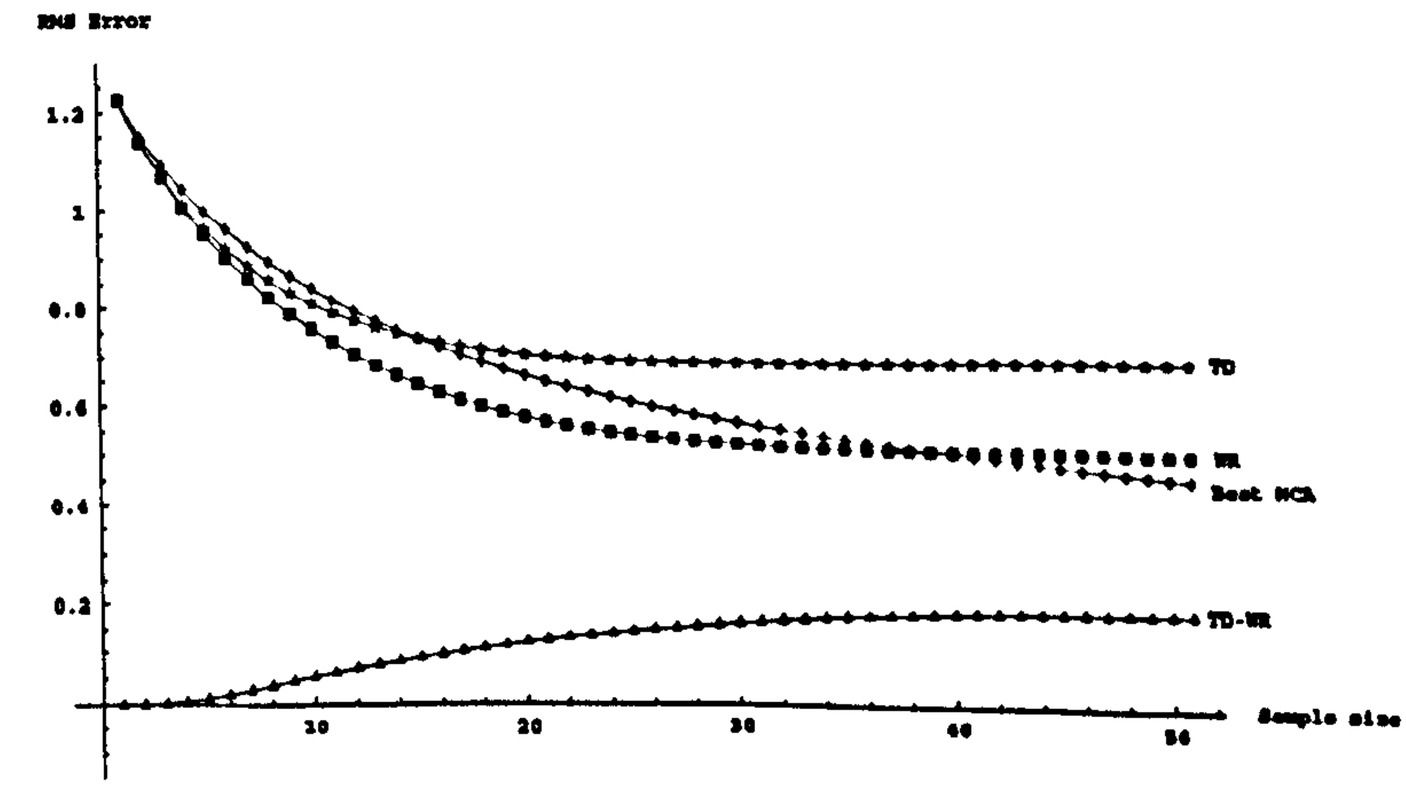


Figure 92: Replace TD and WR for λ=0.8 and α=0.5 (using p=0.6, r=0.7).

Conclusion

The Accumulate TD estimator makes use of an Every visit WR term and Every visit WR is initially close to Accumulate TD, and the two only diverge when either Accumulate TD diverges or when they settle to their asymptotic RMS values. Whatever additional properties using Accumulate TD brings to the estimation procedure, any early advantage it might show over MCA can still be explained by its incorporation of a Every visit WR term. Unlike the Replace TD estimator Accumulate TD tends to achieve a lower asymptotic RMS than its equivalent WR term. This could be due to the fact that it makes better use of the non-reward terms to reduce the bias in the Every visit WR term.

The Replace TD estimator makes use of a Last visit WR term and Last visit WR is initially close to Replace TD and the two only diverge when either Replace TD diverges or when they settle to their asymptotic RMS values. Whatever additional properties using Replace TD brings to the estimation procedure, any early advantage it might show over MCA can still be explained by its incorporation of a Last visit WR term. Indeed in the case of the two models with lower recurrence than the SRW model, the Last visit WR estimator has a lower asymptotic RMS error than Replace TD, suggesting that the inclusion of the "unbalanced" non-reward terms isn't particularly useful in achieving a lower asymptotic RMS error.

Chapter Ten

WR(λ) as an estimator

As the WR(λ) estimator seems to be responsible for the early good performance of TD(λ) it raises the possibility that it might be a useful simplification of the TD estimator and hence have a practical use. In the previous chapters the focus has been on how close WR is to TD when TD has an advantage over the MCA estimator in small samples. In this chapter the emphasis is on how well WR performs as an estimator in its own right. The First visit WR estimator is compared to the First visit MCA estimator. Comparisons with other forms of estimator produce very similar results after allowance has been made for lack of convergence and rescaling of λ and α.

Initially the WR estimator doesn't seem particularly promising because for most values of λ it has a large asymptotic RMS error. The TD estimator has the advantage that it automatically reduces the bias as more data is collected and hence it can be a good large sample estimator as well as small sample estimator. However the asymptotic error in WR decreases as λ increases to 1 and for many applications the variance reduction that it produces might make it worth using. Also, as observed in the previous chapters, there are models for which WR has a lower asymptotic RMS error than TD working at the same λ. For example in the case of the 19-state SRW it is clear that the WR estimator has an advantage over the MCA estimator with the same asymptotic error. This is generally true for the SRW for values of λ close to 1.

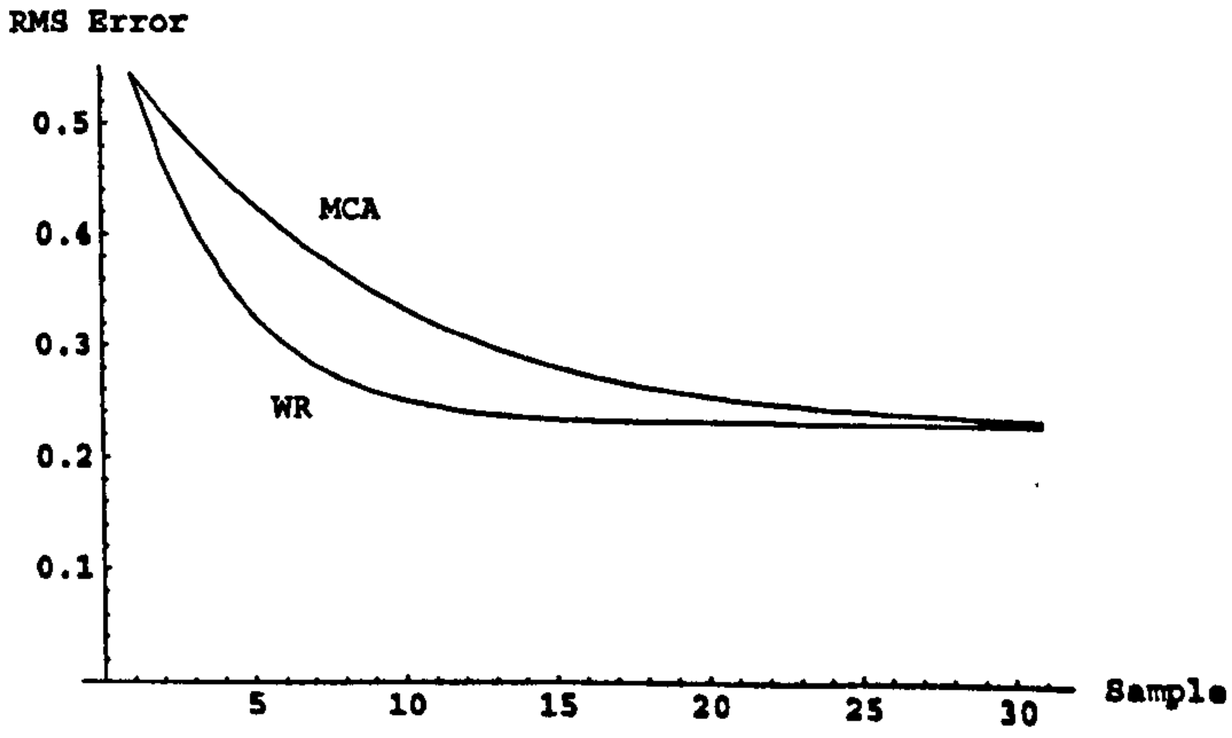


Figure 93: 19-state SRW WR λ=0.95 α=0.4

Figure 94 displays the WR estimators for different values of λ each with its optimum α and shows that WR has an advantage over MCA as long as λ is large enough.

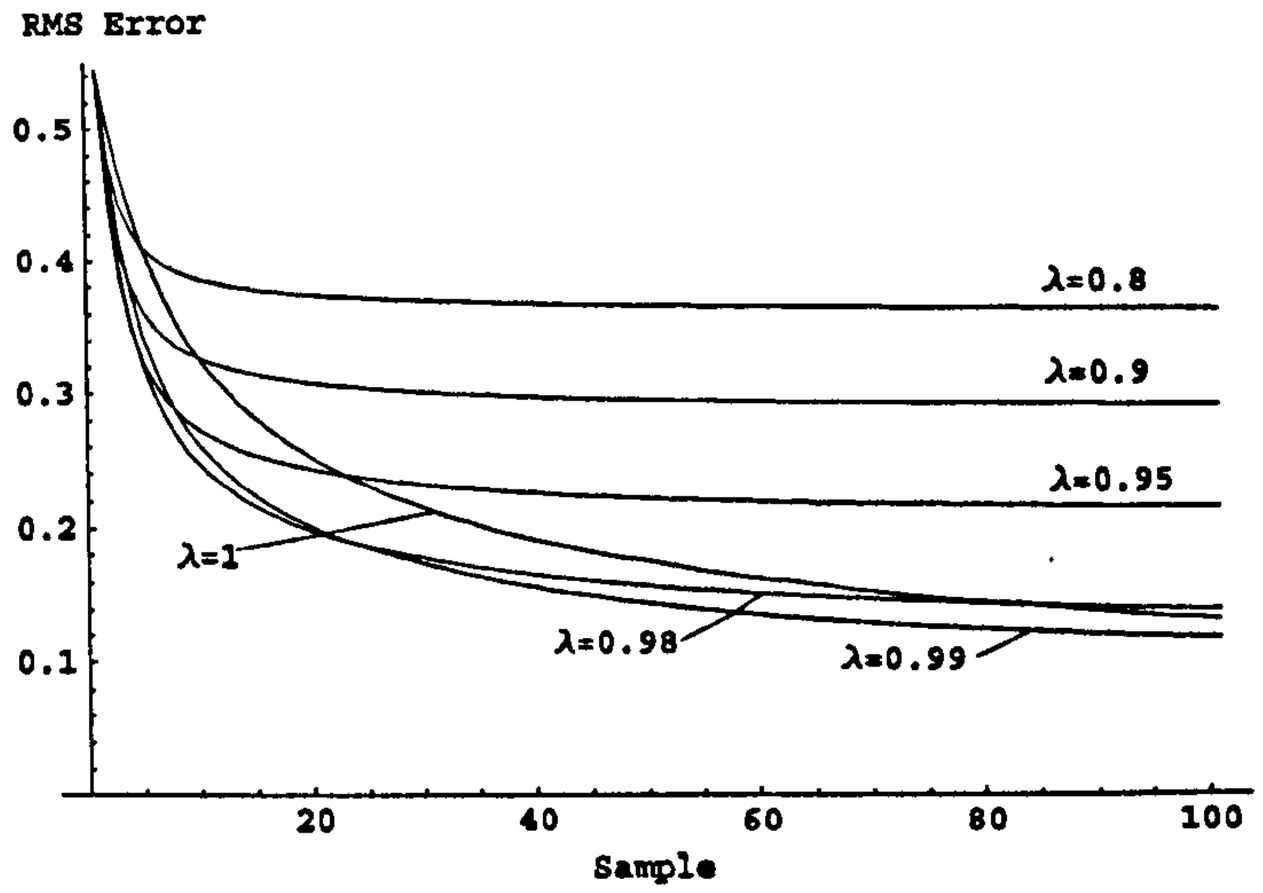


Figure 94: Optimum α WR estimators for the SRW

Notice that as the MCA estimator is asymptotically unbiased there is always a sample size for which the MCA estimator is better than any WR estimator. Smaller values of λ show an initial advantage over the MCA estimator but it is quickly lost due to their much higher asymptotic RMS error.

In the case of the cyclic model the WR estimator is worse than the MCA estimator, as is the TD estimator. This is probably a consequence of their being little relationship between the starting state and the reward received in this particular model. This at least demonstrates a model for which the TD and the WR estimators are out-performed by the simple MCA estimator. The optimum α WR estimators for $c=0.9$ $\phi=1$ can be seen in Figure 95.

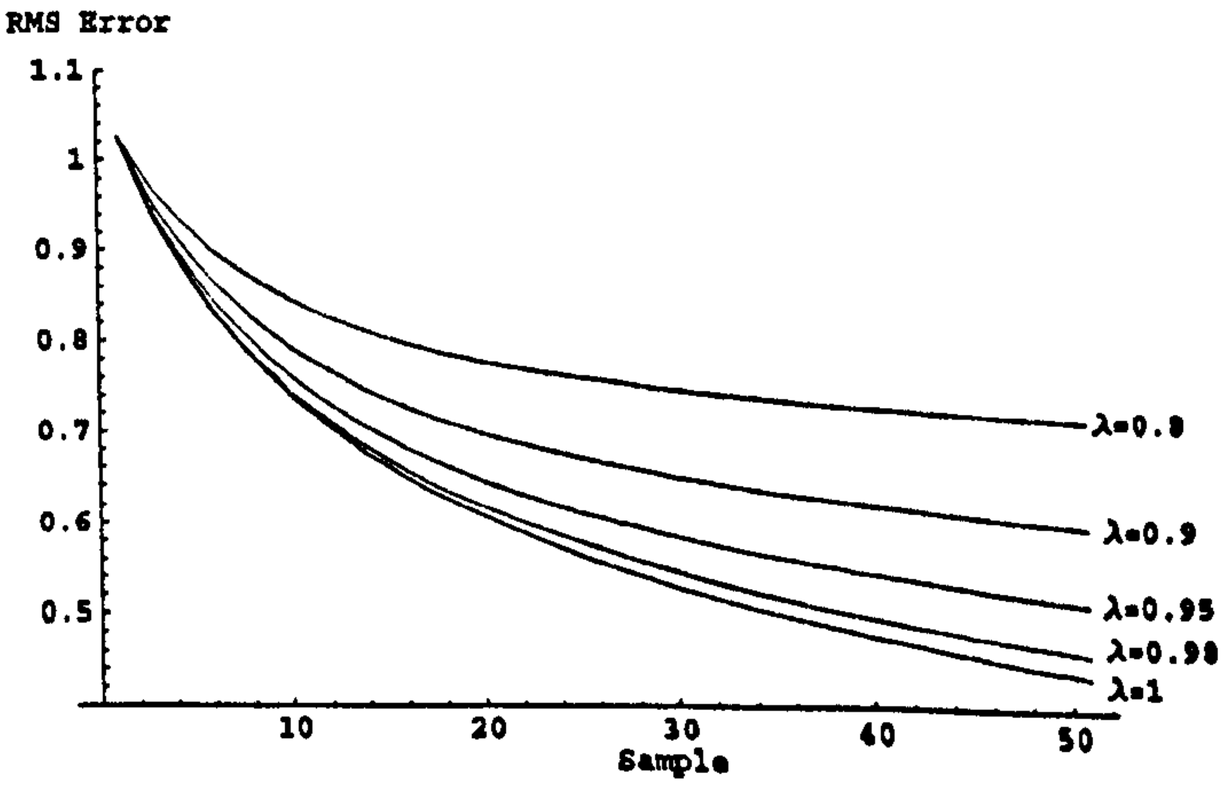


Figure 95: Optimum α WR estimators for the Cyclic model with $c=0.9$ $\phi=1$

In the case of the Bottleneck model the WR estimator performs best, compared with the MCA, when the path length is long p=0.9 and the two halves of the SRW are very different r=0.9 – See Figure 96.

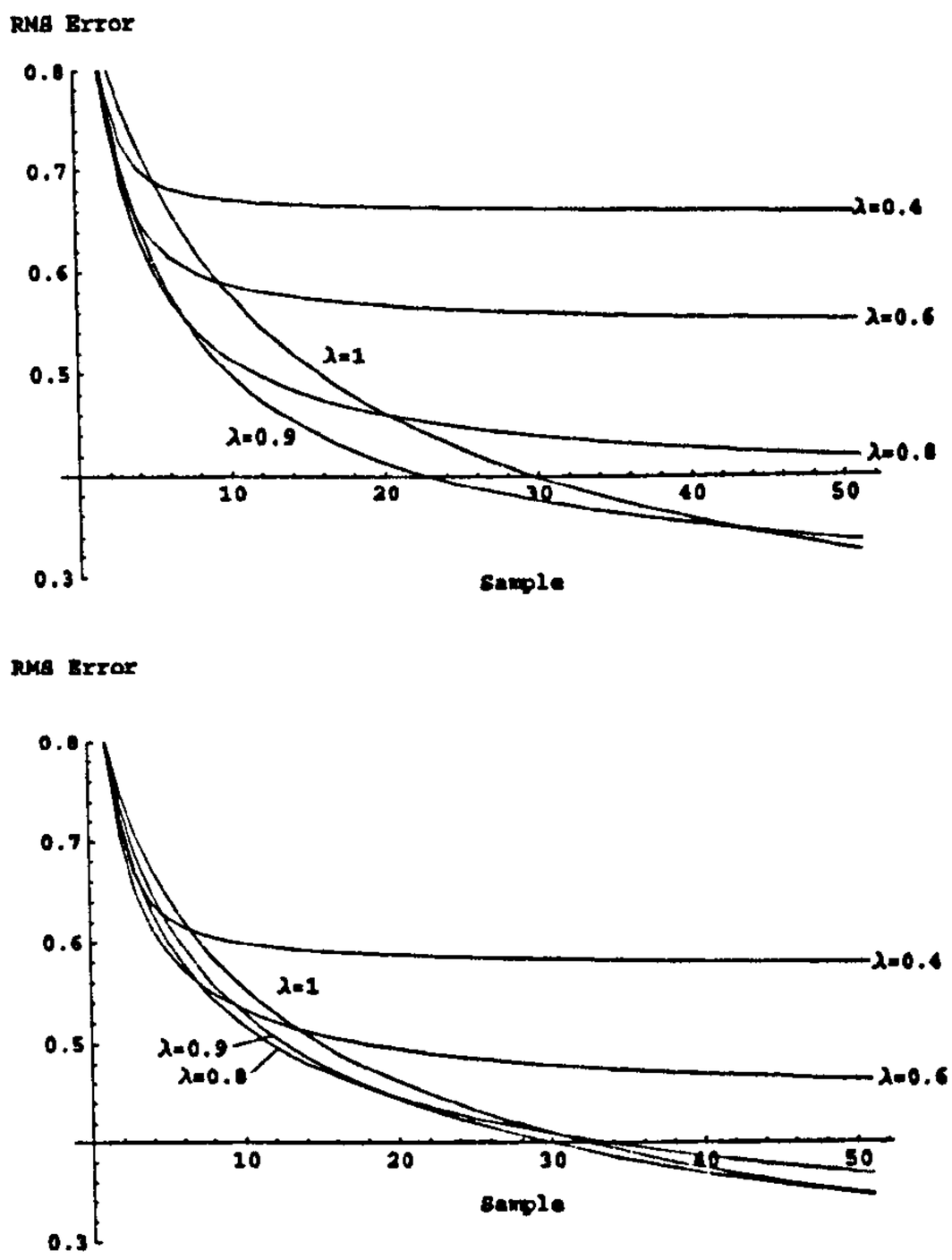


Figure 96: Optimum α WR estimators for the bottleneck model with p=0.9 r=0.9 (top) and p=0.6 r=0.7

The effect of initial bias

The advantage that WR shows over MCA seems to persist with larger initial biases. Indeed, when the additional number of steps needed for the estimator to reach its asymptotic error from the greater initial RMS error is taken into account, the behaviour is more or less the same. For example, Figure 97 shows the WR and MCA estimators for the SRW shown in Figure 93 but with an initial estimate that makes the RMS error 50 i.e. 100 times greater. If allowance is made for scale and the fact that the asymptotic errors are the same in both cases then the behaviour is very similar.

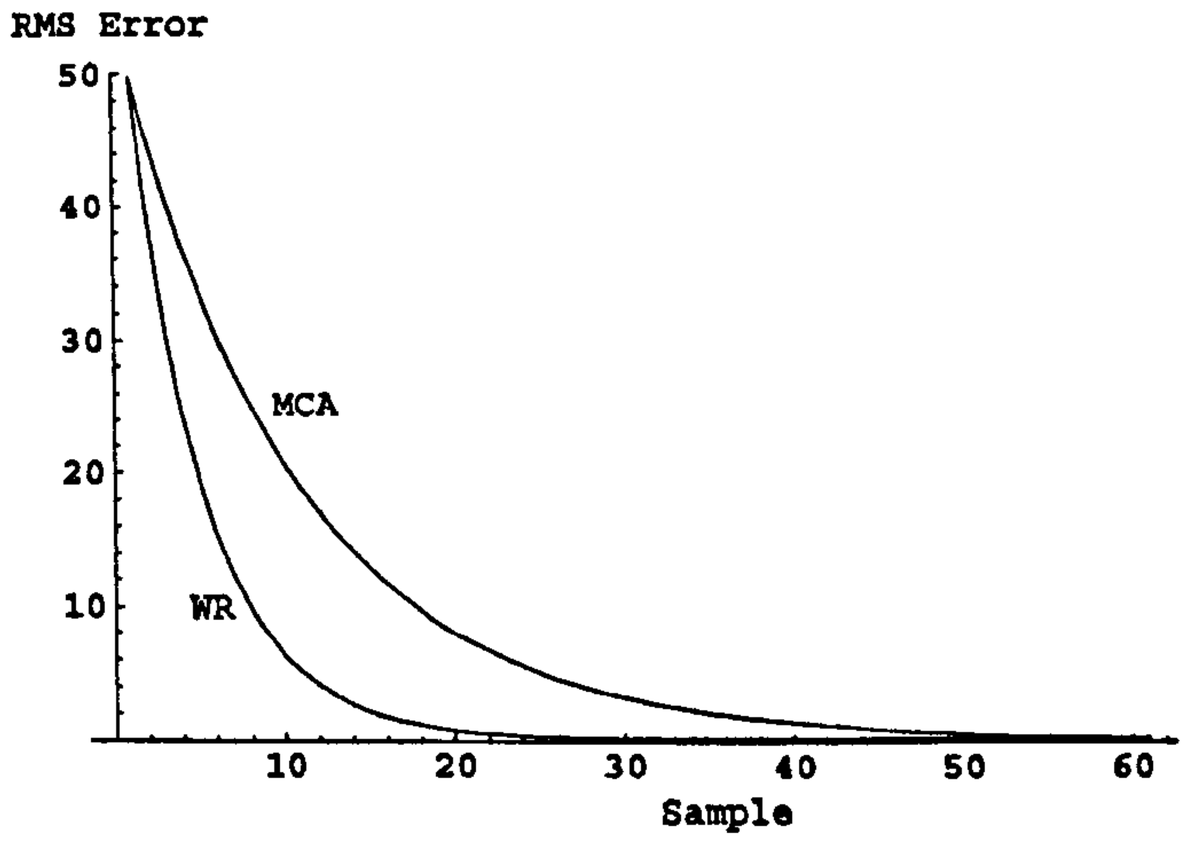


Figure 97: WR λ=0.95 α=0.4 but with initial RMS error =50

In the same way Figure 98 is the corresponding optimum α chart to Figure 94 and again, as long as allowance is made for scale, the behaviour of the WR estimators is very similar.

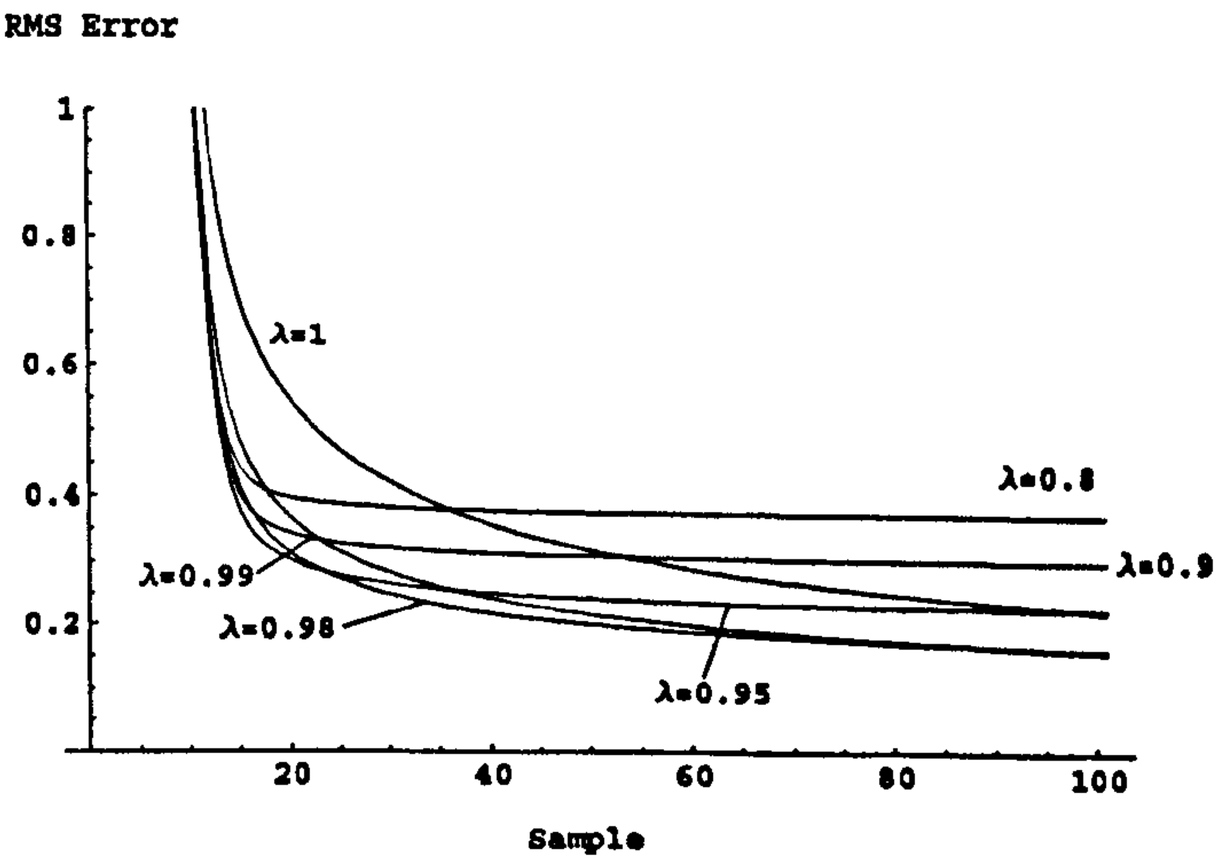


Figure 98: optimum α WR for the SRW with the initial RMS error =50

Analytical optimal values

By using an iterative form of the formulae for the RMS error of the WR estimator:

$$E[WR_t(s_i)] = (1-\alpha f_i)E[WR_{t-1}(s_i)] + \alpha f_i E[w_i]$$

$$\begin{aligned} RMS_t = {} & (1-\alpha f(2-\alpha))RMS_{t-1} + \alpha^2 f \ \ Var[w_i] \\ & -\alpha f(E[r_i]-E[w_i])(2E[WR_{t-1}(s_i)]-2E[r_i]+ \\ & \alpha(E[r_i]+E[w_i]-2E[WR_{t-1}(s_i)])) \end{aligned}$$

we can solve for minimum RMS at each step. This minimisation can be achieved in both λ and α using numerical optimisation (provided by Mathematica). This provides a one-step greedy WR estimator (Singh and Dayan 1998) which while not globally optimum gives an upper limit on the RMS error obtainable by a WR estimator.

The result of optimising λ and α in this way using the 19 state SRW can be seen in Figure 99, which shows the resulting RMS error compared to a one-step greedy optimum MCA (λ=1) estimator and the usual sample average (1/N) estimator.

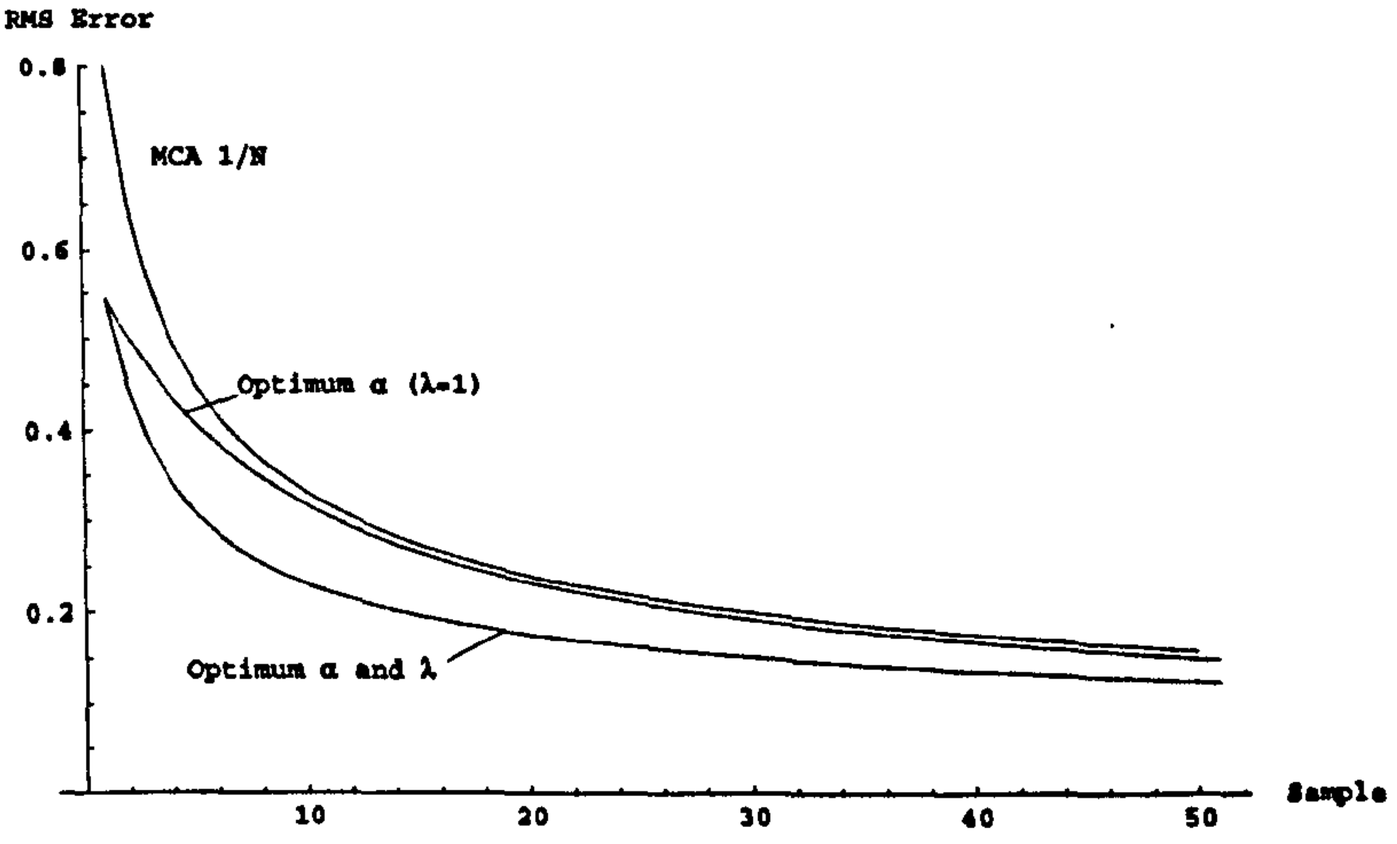


Figure 99: One-step α, λ-greedy WR compared to MCA with α=1/N and one-step optimal α

From this it is clear that with a one-step optimum schedule for λ and α, WR(λ) is a better estimator than the optimum α MCA update estimator which is close in performance to the α=1/N, i.e. the sample mean estimator.

The behaviour of the optimum λ and α can be seen in Figures 100 and 101. From Figure 100 it can be seen that the optimum value of α quickly becomes close to 1/N which is also close to the optimum value of α for the MCA estimator. The reason is simply that as λ becomes closer to 1 then the WR estimator approximates the MCA estimator and the optimal α in each case are close. In the case of λ the only surprise is that the rate at which it approaches 1 relative to sample size is slow.

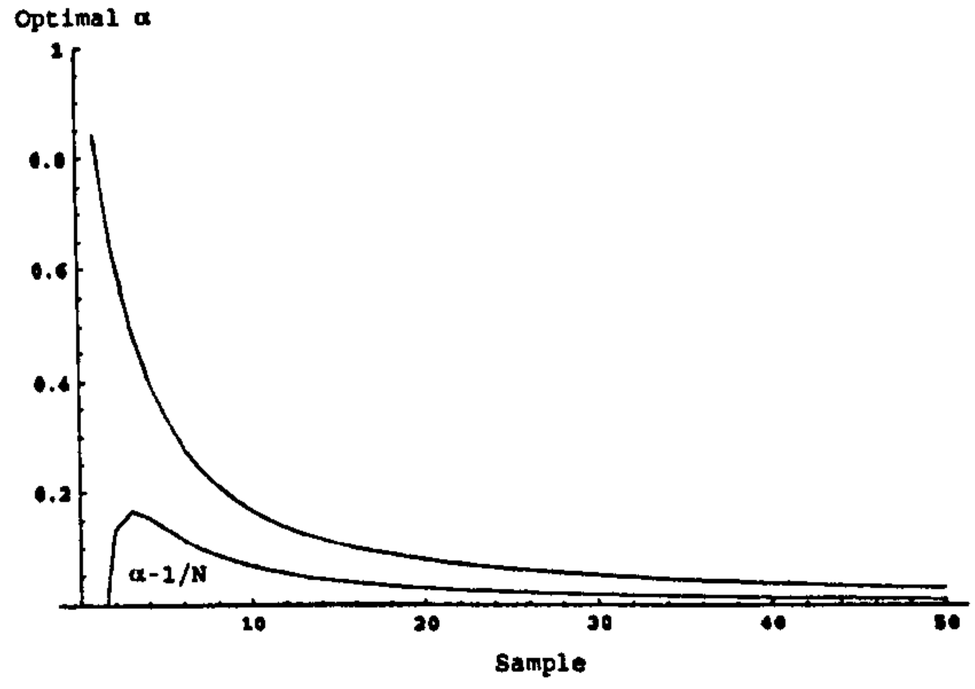


Figure 100: Optimum α compared to α=1/N

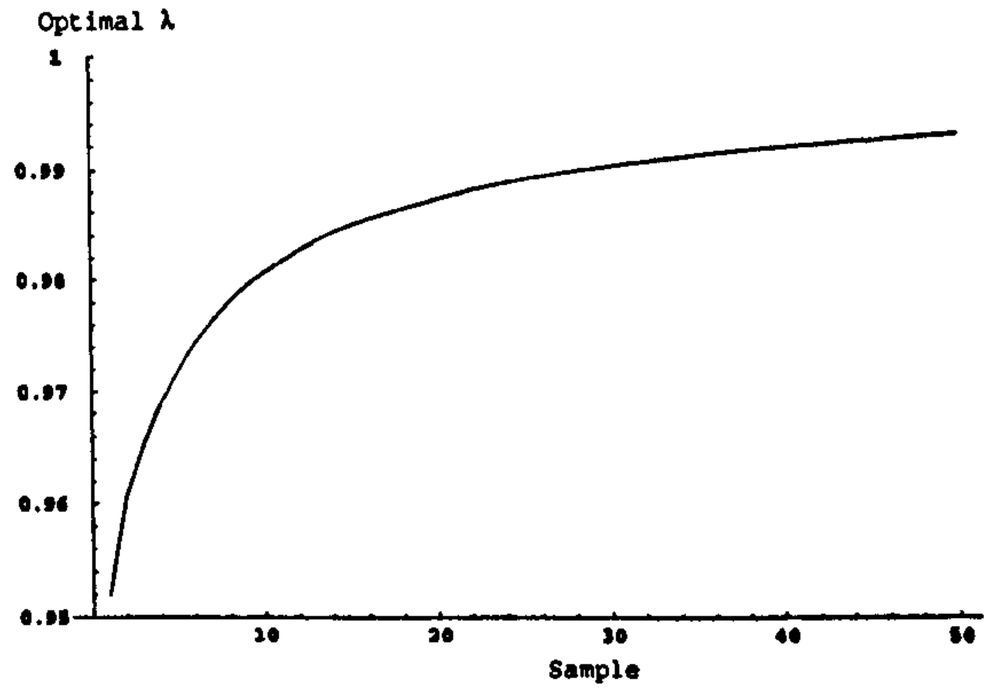


Figure 101: Optimum λ.

In the case of the Cyclic and Bottleneck models discussed in earlier chapters the performance of WR is what would be expected. For the Cyclic model the optimum WR performs exactly like the optimum MCA estimator as the optimum value of λ rapidly approaches 1, revealing that in this case WR can do no better than MCA, see Figure 102.

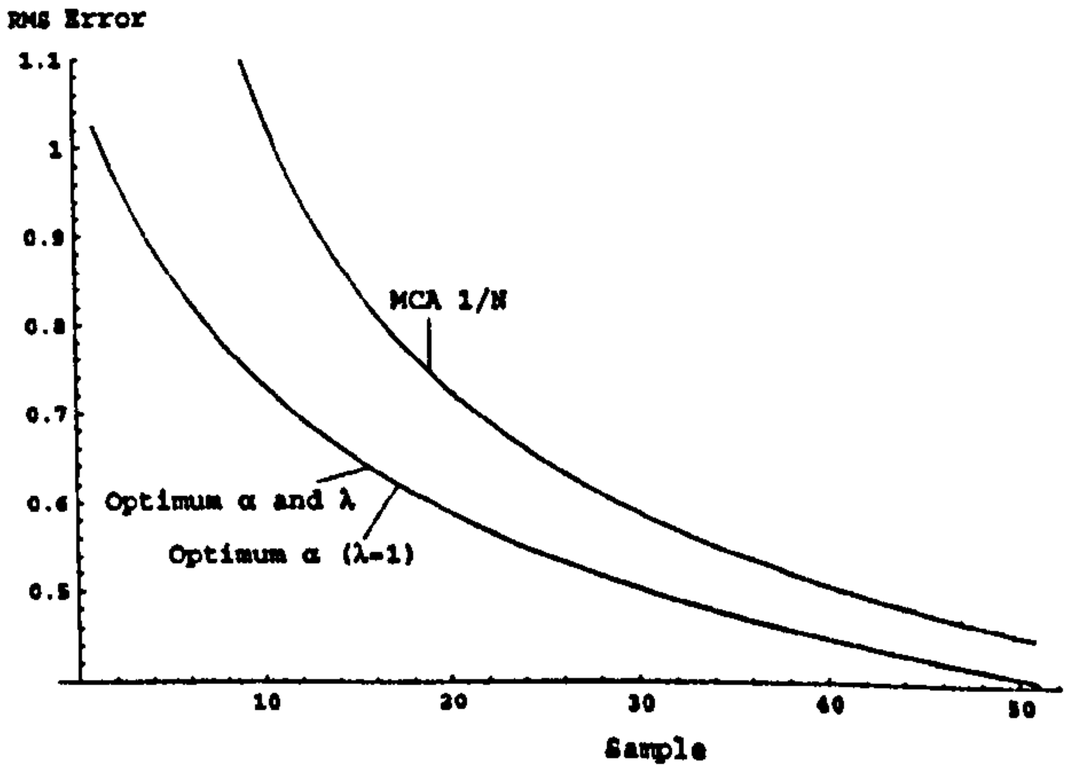


Figure 102: One-step α, λ-greedy WR compared to MCA with α=1/N and optimal α for the Cyclic model with ϕ=1 c=0.9

The optimal values of λ and α can be seen in Figures 103. It can be seen that the optimum value of α is not like 1/N in this case and λ quickly approaches 1.

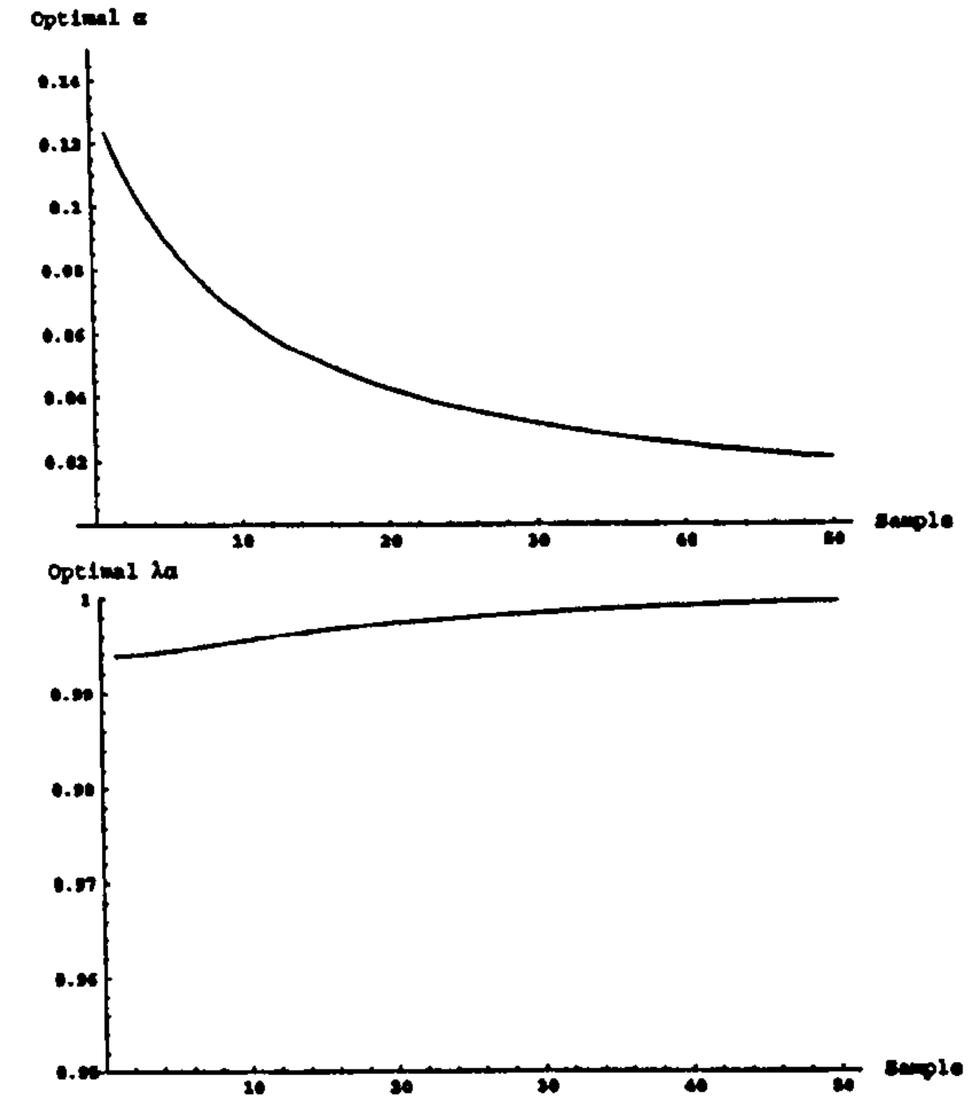


Figure 103: Optimal α and λ for the Cyclic model with ϕ=1 c=0.9

The RMS curves for the Bottleneck model for two sets of parameters can be seen in Figure 104. The optimum performance seems to vary little with changes in the model parameters and this holds for the optimal values of λ and α, as can be seen in Figure 105.

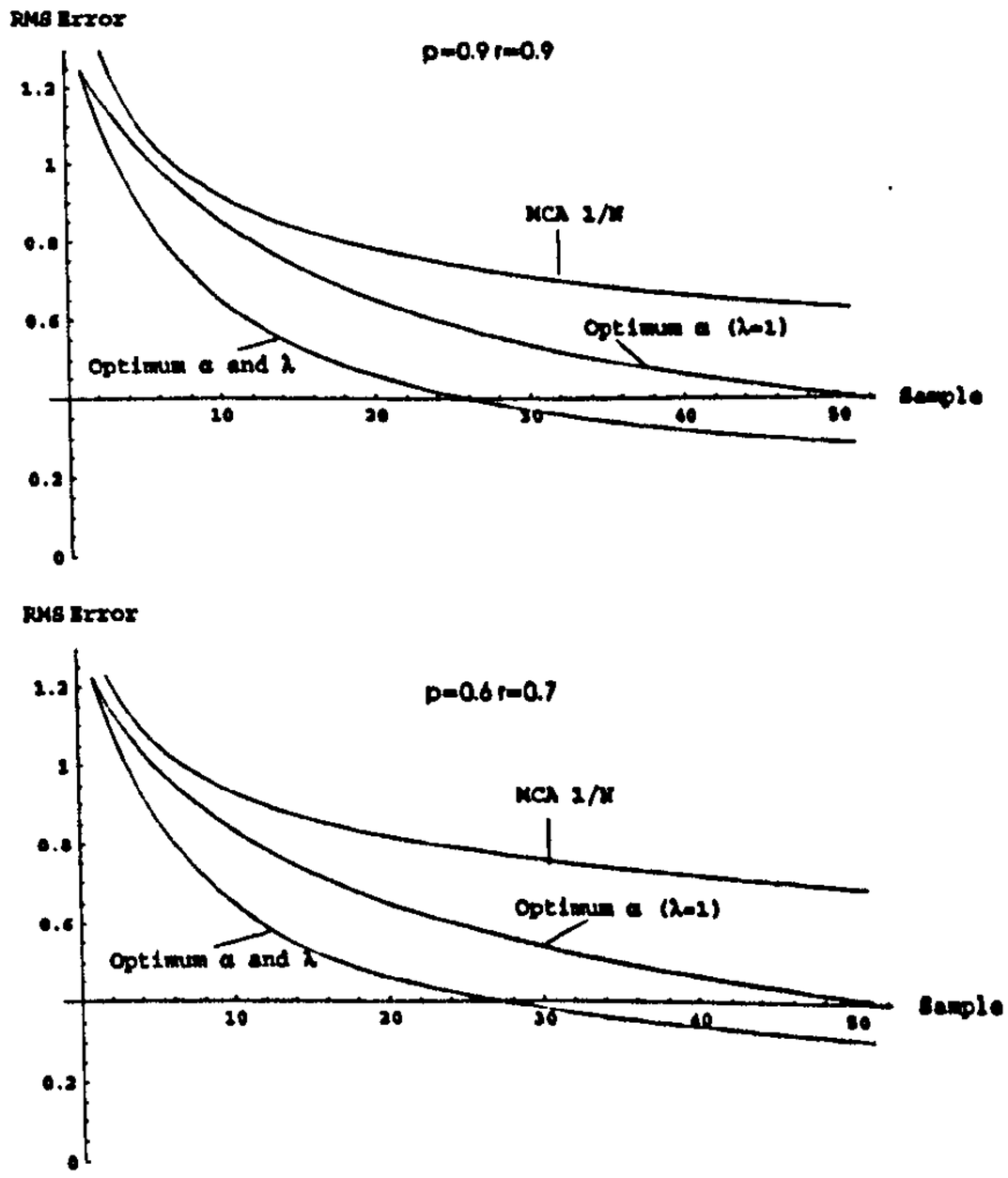


Figure 104: RMS for optimal α and λ for the Bottleneck model

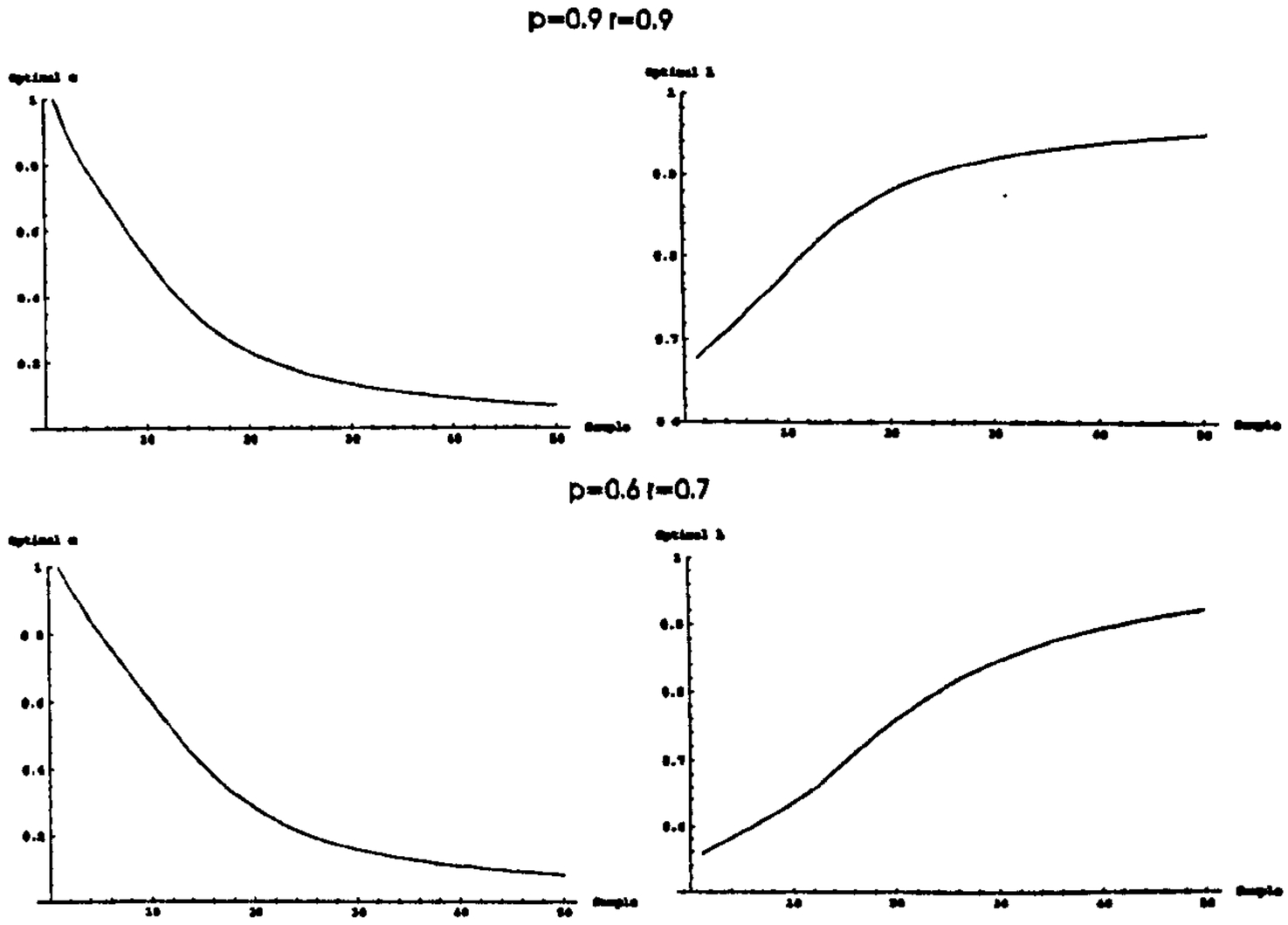


Figure 105: Optimal α and λ for the Bottleneck model

Conclusion

The WR estimator is simple to compute and seems to have advantages over the MCA estimator in some models. However the problem, as with the TD estimator, is selecting good values of λ and α. As the charts of optimal λ and α demonstrate, good choices depend very much on the underlying model. It is also clear that values of λ less than 1 are best early on in estimation and the optimum α isn't changed very much from its optimum MCA value. Clearly more research is needed to characterise the relationship between model type and the behaviour of the WR estimator.

Chapter Eleven

The Performance of Value Estimation

The work outlined in earlier chapters makes it clear that TD(λ) is a better small sample estimator, in the RMS sense. It is still unclear, however, whether this advantage is an important one in practice and this question raised issues about what the estimates are being used for. Are there differences between simple estimators such as MCA and more advanced estimators such as TD and WR that would suggest that they are worth using?

RMS error – an appropriate metric?

Although the single measure of estimation accuracy used so far has been RMS error or something directly based on it, e.g. average RMS error over a given number of samples, this isn't necessarily the only possible or best choice. While the use of RMS error is reasonable from the point of view of classical estimation theory it doesn't really address the problem of making good decisions. The argument is that an estimator that is close to the true value function in the RMS sense is presumably a good estimator from the point of view of decision making. The estimation of a value function is not an end in itself but a step in testing or refining a policy. In this case simply being close to the true value function in the RMS sense may not be the end of the story.

A simple rank order index

If we have an estimate of the value function for each state $\tilde{v}(s)$ then the RMS error is, as discussed, a reasonable approximation of the quality of the estimate in terms of how close it is to the true value function V(s). That is, as:

$$RMS^2 = \sum_s [\tilde{v}(s) - V(s)]^2$$

gets smaller the estimate becomes closer to the true value.

However, this is actually more than is required for decision making based on the value function. If the reinforcement learning agent is currently in state s then the

optimum algorithm is simply greedy with respect to the optimum value function and it should select the state s' that maximises the value function (Sutton & Barto, 1998). That is:

$$s_{n+1}' = \arg\max_{s'}[V(s')]$$

where s' is in the set of states reachable from s in a single step. If we have an estimate of the optimum value function $\tilde{v}(s)$ then it will give the same set of decisions as V(s) if it obeys the same order relations as V(s). That is, if:

$$V(s_i) > V(s_j) \Leftrightarrow \tilde{v}(s_i) > \tilde{v}(s_j)$$

then

$$s_{n+1}' = \arg\max_{s'}[V(s')] = \arg\max_{s'}[\tilde{v}(s')]$$

and the estimated value function results in the same optimum policy as the true optimum value function.

So, even if the estimate of the value function is a long way from the true optimum value function in the RMS sense it could still provide the same optimum policy and hence be as useful as an "accurate" estimate.

As a trivial example consider an estimator of the true value function multiplied by an unknown positive constant:

$$\tilde{v}'(s) = \gamma V(s) \text{ where } \gamma > 0$$

In this case it is clear that the RMS error can be made as large as desired by increasing γ but the scaled estimate still results in the same policy being followed.

This conclusion leads to the desire to quantify the quality of the estimate of the value function in terms more directly related the optimality of the policy. Put simply, it would be better to use a measure that was proportional to the number of correct decisions resulting from the current estimate when used as the value function in a greedy policy. There are a number of possible ways of constructing such a measure but the most direct is simply to count the number of times the estimated value function obeys the same order relations among "connected" states. In general this

involves an upper triangular matrix of order relations between the N states of an MDP. In the case of the the linear SWR MDPs considered in earlier chapters, however, this reduces to a much simpler matrix.

We can define a measure of "concordance" between the estimate and the true value function as:

C= (number of times $\tilde{v}(s)$ gives the same "greedy" decision as V(s))/N

This gives a number between 0 and 1, which can be interpreted as the proportion of times the estimate gives the same decision as the true value. If C is 0 the estimate gives wrong decisions at each state and if C is 1 then it gives correct decisions at each of the states. Clearly there are other possible definitions of concordance between the estimate and the true value, including the standard Spearman rank order correlation, but this definition has the advantage of being simple to interpret and it doesn't include aspects of agreement of rank order that have no bearing on decision making.

A concordance study of the SRW

As the 19-state SRW has provided the proving ground for TD(λ) and WR(λ) estimators to date, it seems sensible initially to investigate this model and the performance of the standard estimators as measured by the concordance coefficient. In the case of the SRW the concordance coefficient takes a simple form. As each state has just two potential successor states C can be written as:

$$C = \sum_{s} \frac{c(s)}{N}$$

where c(s) is defined to be:

$$\begin{aligned} c(s) &= 1 \quad \text{if } \tilde{v}(s-1) \geq \tilde{v}(s+1) \quad \text{and} \quad \text{if } V(s-1) \geq V(s+1) \\ &= 1 \quad \text{if } \tilde{v}(s-1) < \tilde{v}(s+1) \quad \text{and} \quad \text{if } V(s-1) < V(s+1) \\ &= 0 \text{ otherwise} \end{aligned}$$

As the values of V(s) are monotonic increasing from s=0 to s=N, there are a number of simpler ways of writing this expression but this is a general form that works for all MDPs of the same general form.

Using this definition of C, simulations using 1000 repetitions for samples sizes from 1 to 50 for First visit TD(λ) and WR(λ) for a range of values of λ were run. The results are very different from the same comparison using the RMS error. Comparing TD(λ) against the First visit MCA estimator and the equivalent alpha update rule given by TD(1) indicates immediately that TD(λ) arrives at 100% correct decision making within approximately five samples whereas the Monte Carlo and MCA converge slowly to 100% after 40 or 50 samples. For example, for α=0.1, the first visit TD(λ) estimator and the standard First visit MCA estimator give the results in Figure 106.

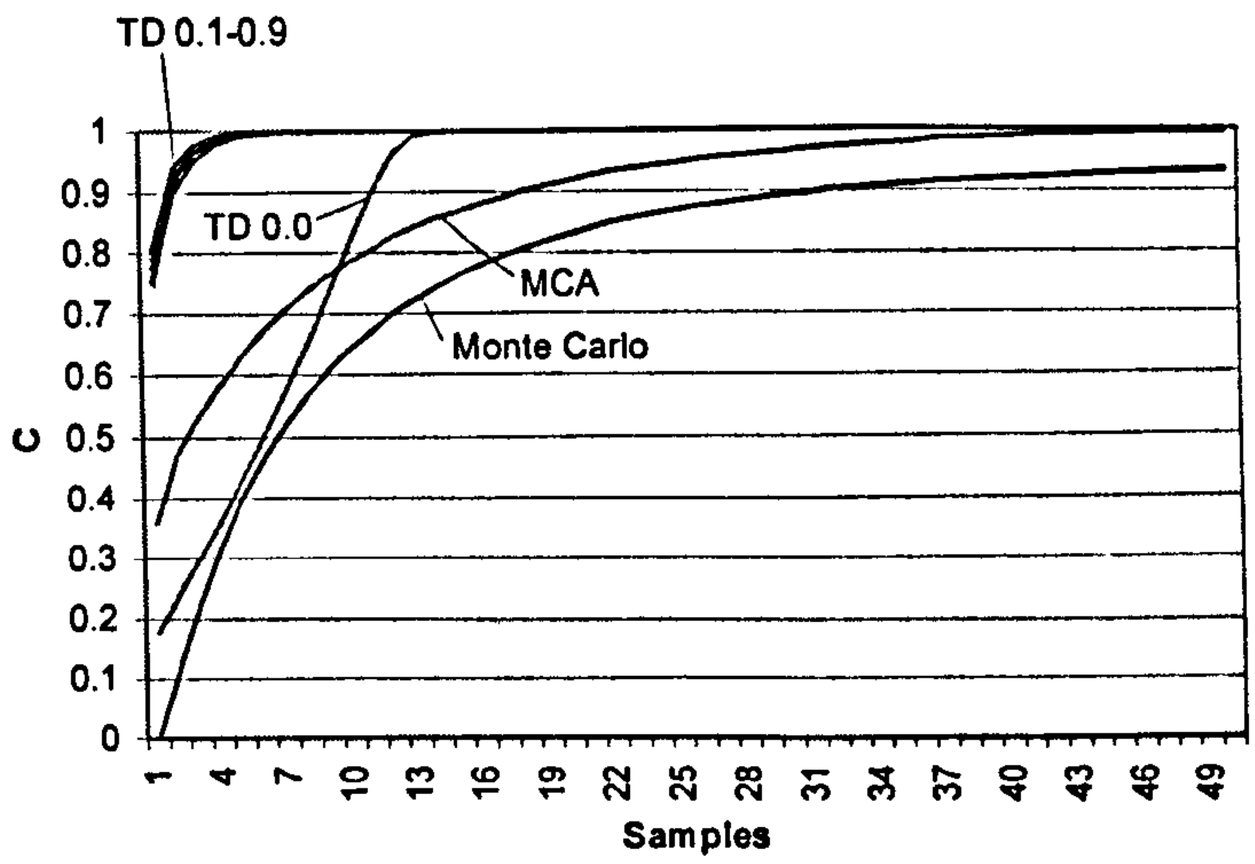


Figure 106: TD(λ) concordance with sample size for α=0.1

Notice that in Figure 106 the Monte Carlo estimator reaches 90% accuracy only after 32 samples whereas TD(λ) for 0<λ<1 reaches 90% accuracy after two samples! Even TD(0), which is slower to converge, reaches 90% accuracy well before the MCA estimator. The MCA rule, also outperforms the Monte Carlo (α=1/N) estimator but it converges at more or less the same rate.

It is also clear that this behaviour is very insensitive to the choice of λ, apart from the extreme values of 0 and 1. The same is true of α if extreme values, i.e. close to 0 or 1, are avoided. As α increases the TD(λ) estimators converge just as quickly but then noise appears to upset the perfect score as additional samples are added. For example, as can be seen in Figure 105, even a value of α as large as .9 doesn't stop

TD(λ) from converging to 1 quickly and only values of λ close to 1 are impaired by the additional noise.

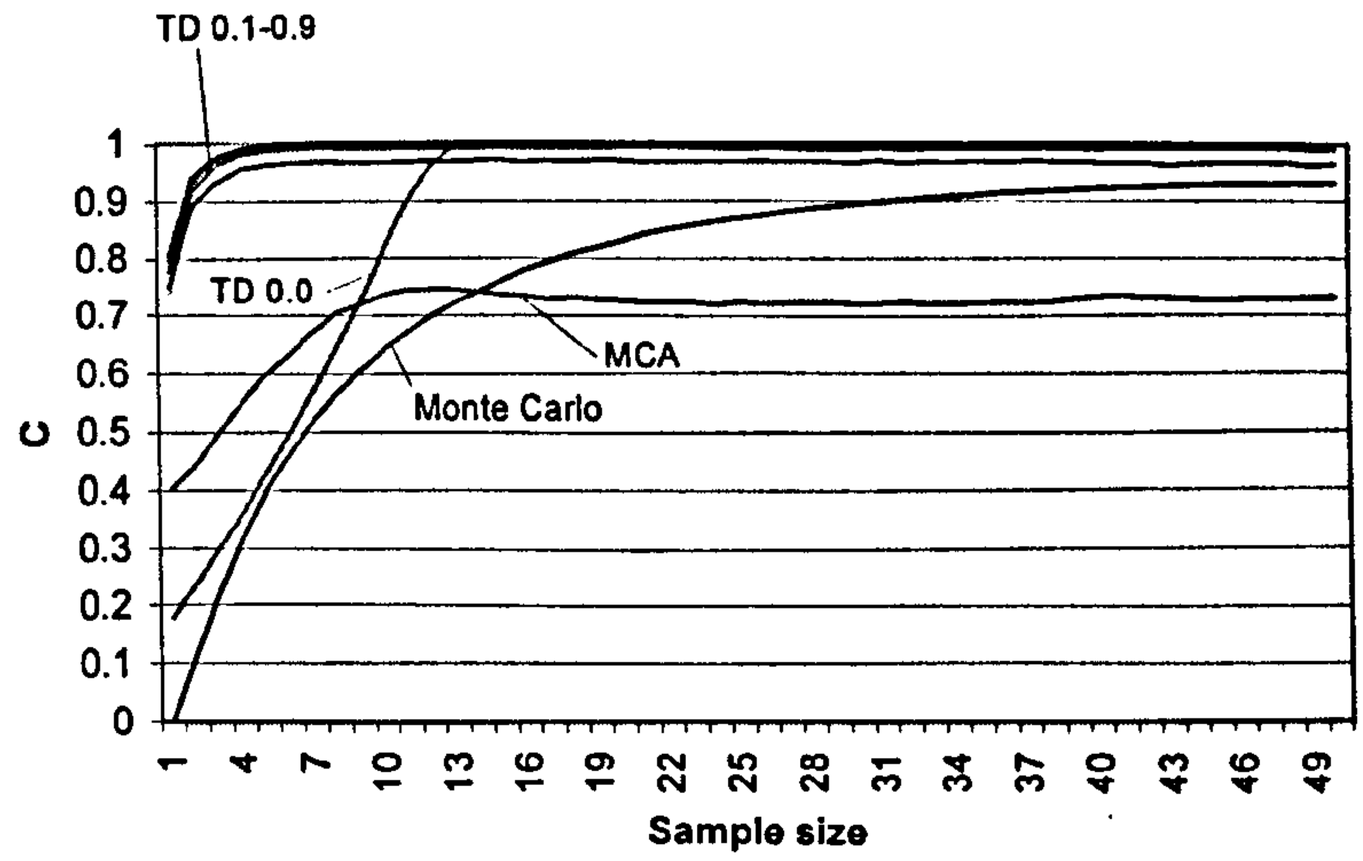


Figure 107: TD(λ) concordance with sample size for α=0.9

The same behaviour can be seen in the experiment using WR(λ), the only real difference being that it doesn't converge quite as rapidly, but given its greater simplicity its performance is good enough to make it a suitable alternative to TD(λ). Notice that as the order of the estimates isn't spoiled by WR(λ) being a biased estimator, there is no need to use a scheme of reducing λ to eliminate the asymptotic bias. If we are only concerned with decision-making performance then asymptotic bias is irrelevant as long as the concordance of the estimate is high. As can be seen in Figure 108, the behaviour of WR(λ) is very similar to TD(λ).

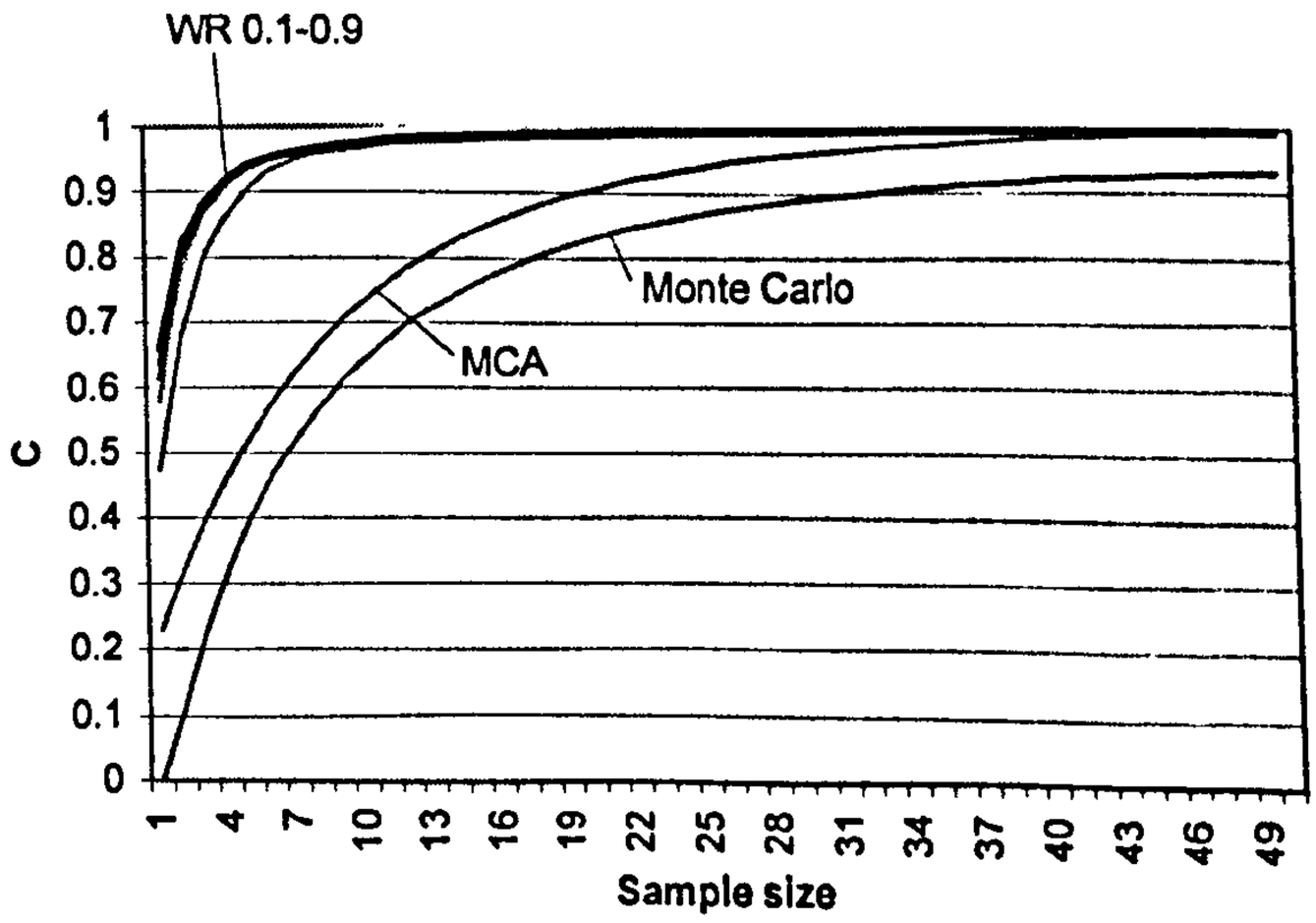


Figure 108: WR(λ) concordance with sample size for α=0.1

As in the case of TD(λ) the performance of WR(λ) also isn't particularly sensitive to the choice of α. For example, Figure 109 shows the concordance for α =0.9, an extreme value. Again the convergence for most values of λ is little changed and the only real effect is the evident noise as the number of samples increases beyond the point of convergence of the estimator.

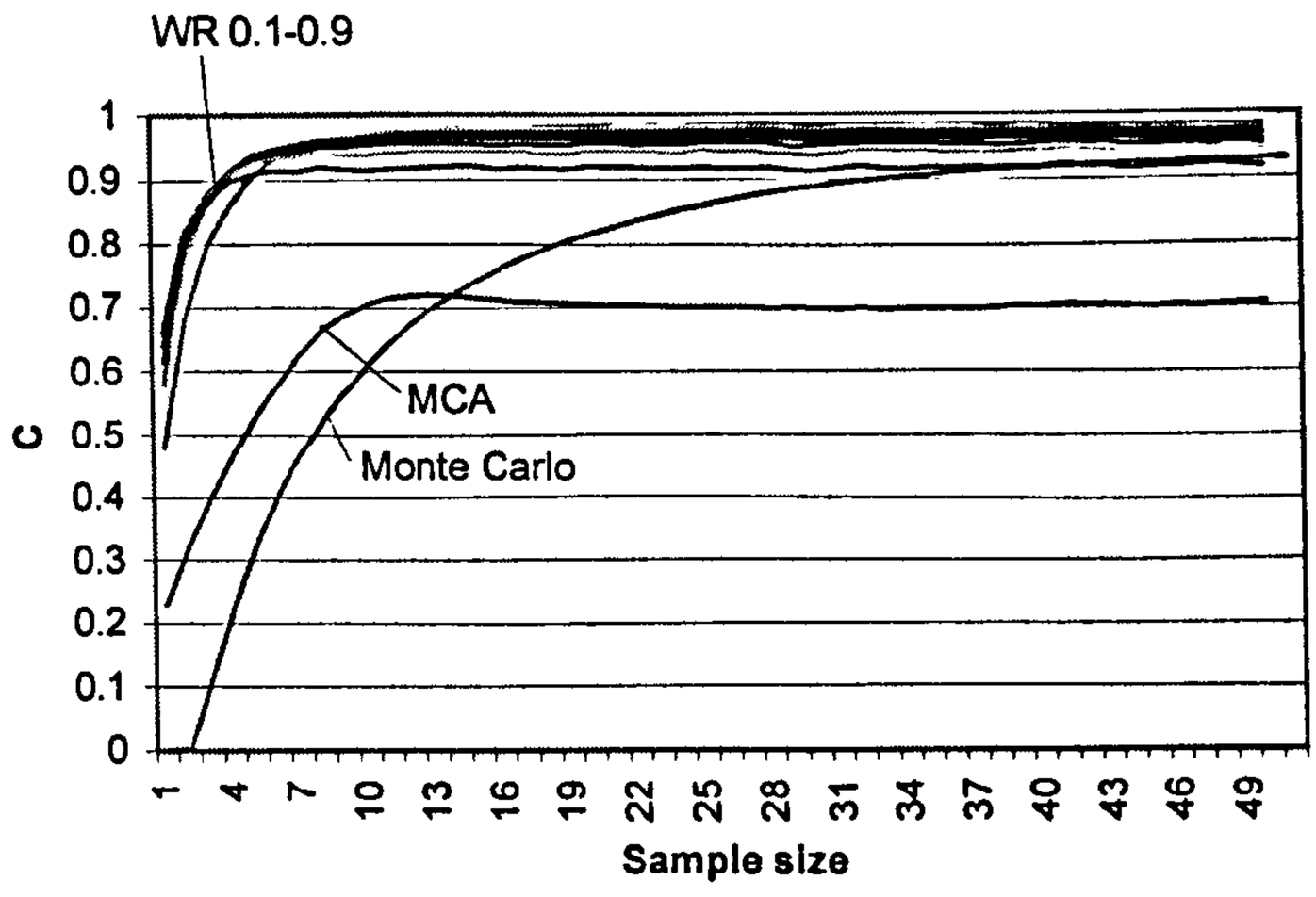


Figure 109: WR(λ) concordance with sample size for α=0.9

Discussion of concordance

The performance of TD(λ) and WR(λ) as measured by concordance with the true value function is very different from the results given by the RMS measure of error.

To summarise:

- The performance of TD(λ) and WR(λ) as measured by concordance is striking. They both show significant advantages over simple estimators such MCA.

- The performance of TD(λ) and WR(λ) is insensitive to the values of λ and α selected as long as extreme values near 0 or 1 are avoided.

Using traditional learning techniques the percentage of correct decisions is still less than 100% after 50 realisations, but both TD(λ) and WR(λ) deliver 100% correct decision making after only 5 to 10 realisations, irrespective of the values of λ and α

as long as they are not extreme. This could be a clear and important advantage and it is certainly a distinct difference between the two types of estimator.

Concordance with non-monotonic value functions

The value function for the simple linear MDP is monotonic and hence it remains a possibility that the advantage seen in TD(λ) and WR(λ) is entirely due to a tendency to produce monotonic estimates, see Figure 110.

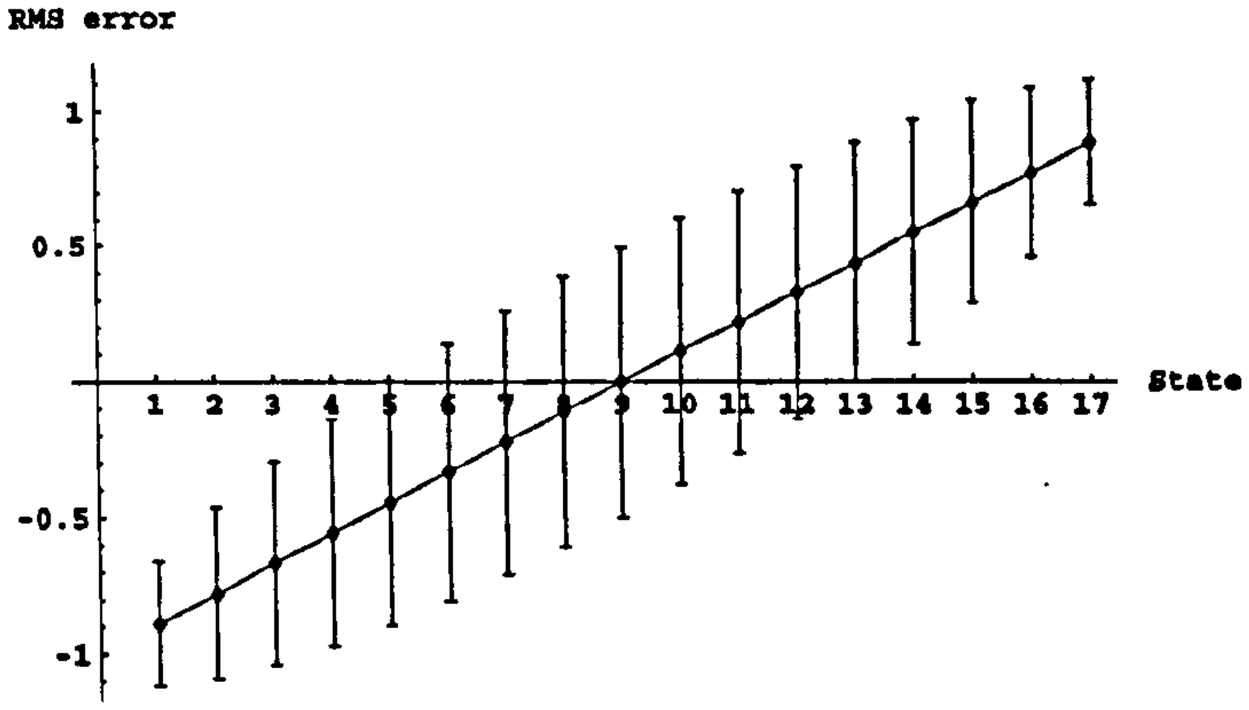


Figure 110: Value function for the 19 state SRW

For example, if an estimator is constructed of the form:

$$\tilde{v}(s) = ms + c$$

where s is the state number, i.e. s=0, N-1, and m is a slope and c an intercept estimated from the data, then the estimates produced always satisfy the inequalities because they are constrained to be monotonic. Of course such an estimator is of little use in practice because the value function encountered in any real situation is unlikely to be monotonic.

The point is that, unless the estimators are tested on an MDP that has a non-monotonic value function, the good performance of TD(λ) and WR(λ) might simply be because of a tendency to give monotonic estimates.

An MDP with non-monotonic value function

It is possible to construct a suitable model which retains most of the characteristics of the linear MDP and has a non-monotonic value function by introducing two states which have probabilities of making direct transitions to the terminal states.

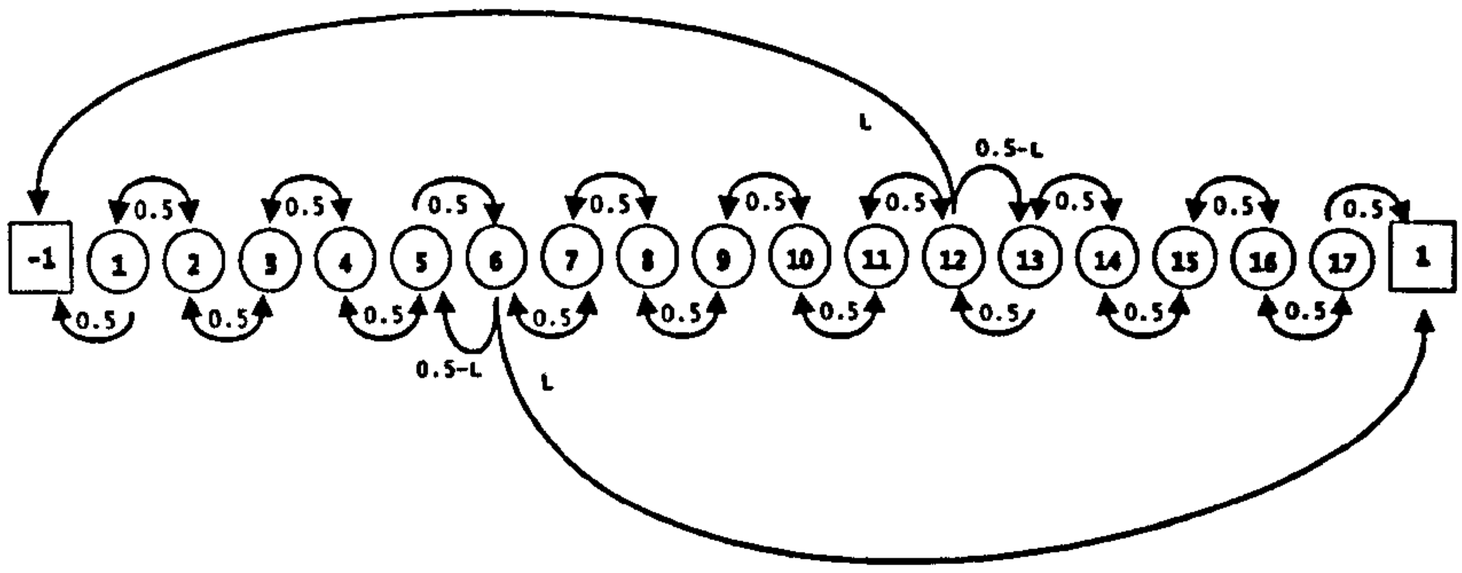


Figure 111: Model with non-monotonic value function

As can be seen from Figure 111, states 6 and 12 provide a symmetric probability of the agent "leaking" out of the usual process and terminating at the "unexpected" end of the chain. To make the theory and testing simpler, and to maintain the similarity to and symmetry of the original linear MDP, both "leaky" states are taken to have the same probability of reaching a terminal state. For small values of the leak probability the value function is still monotonic but as it increases the value function changes to be non-monotonic, see Figures 112 and 113.

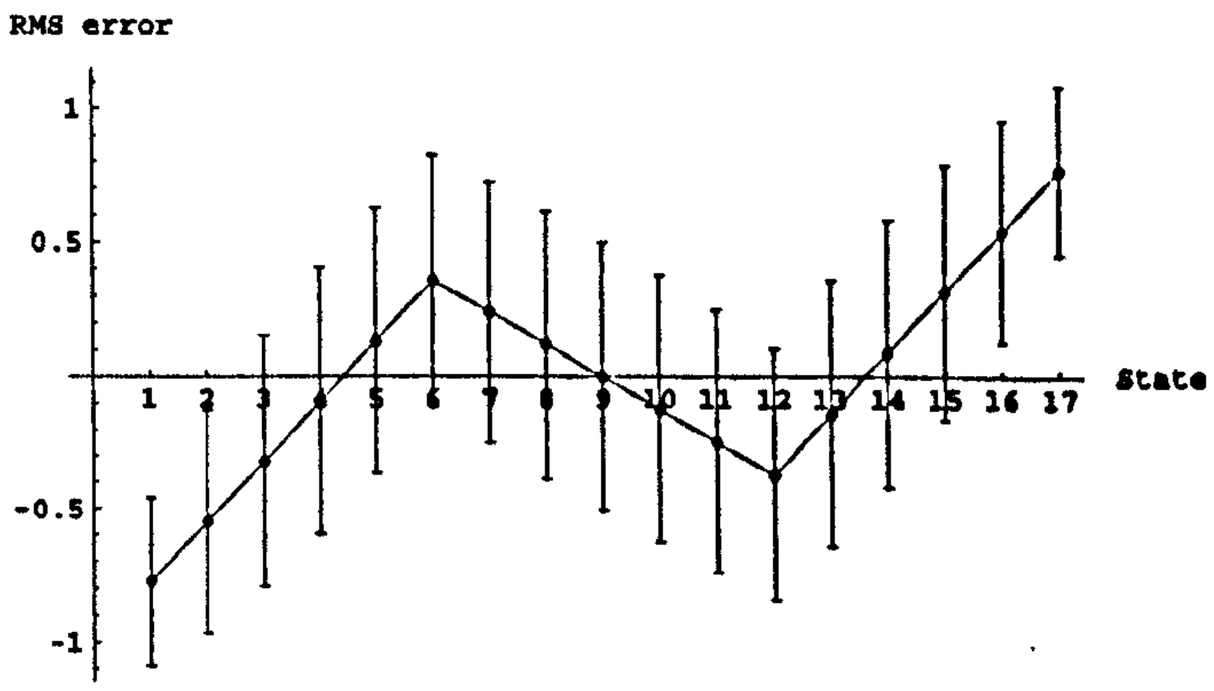


Figure 112: The value function for the "leaky" MDP with leak=0.2

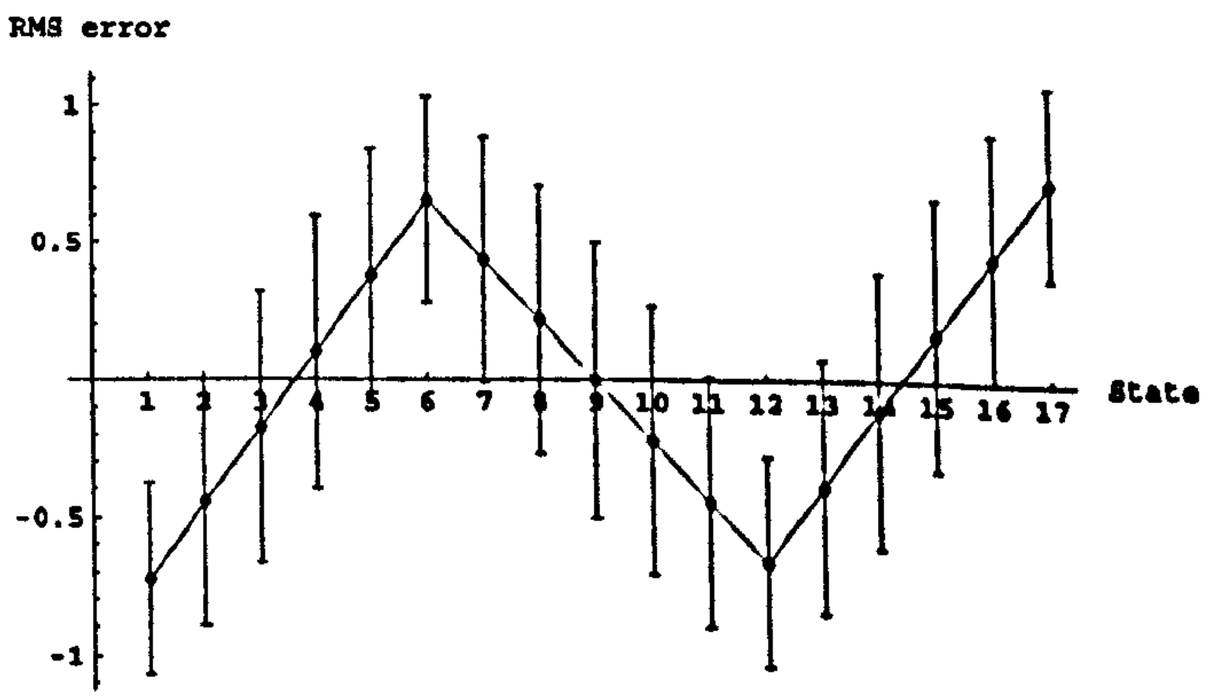


Figure 113: The value function for the "leaky" MDP with leak=0.4

While this MDP is clearly related to the original linear MDP, the SRW, there are a number of differences, apart from its non-monotonic value function, that have to be kept in mind when interpreting results using it. The first is that its overall variance is typically higher than the original model. The introduction of "leaky" states makes the system inherently less predictable and so estimating the value function is a harder task. As can be seen from Figure 114, the variation in variance per state reaches a maximum when the leak probability is approximately 0.2.

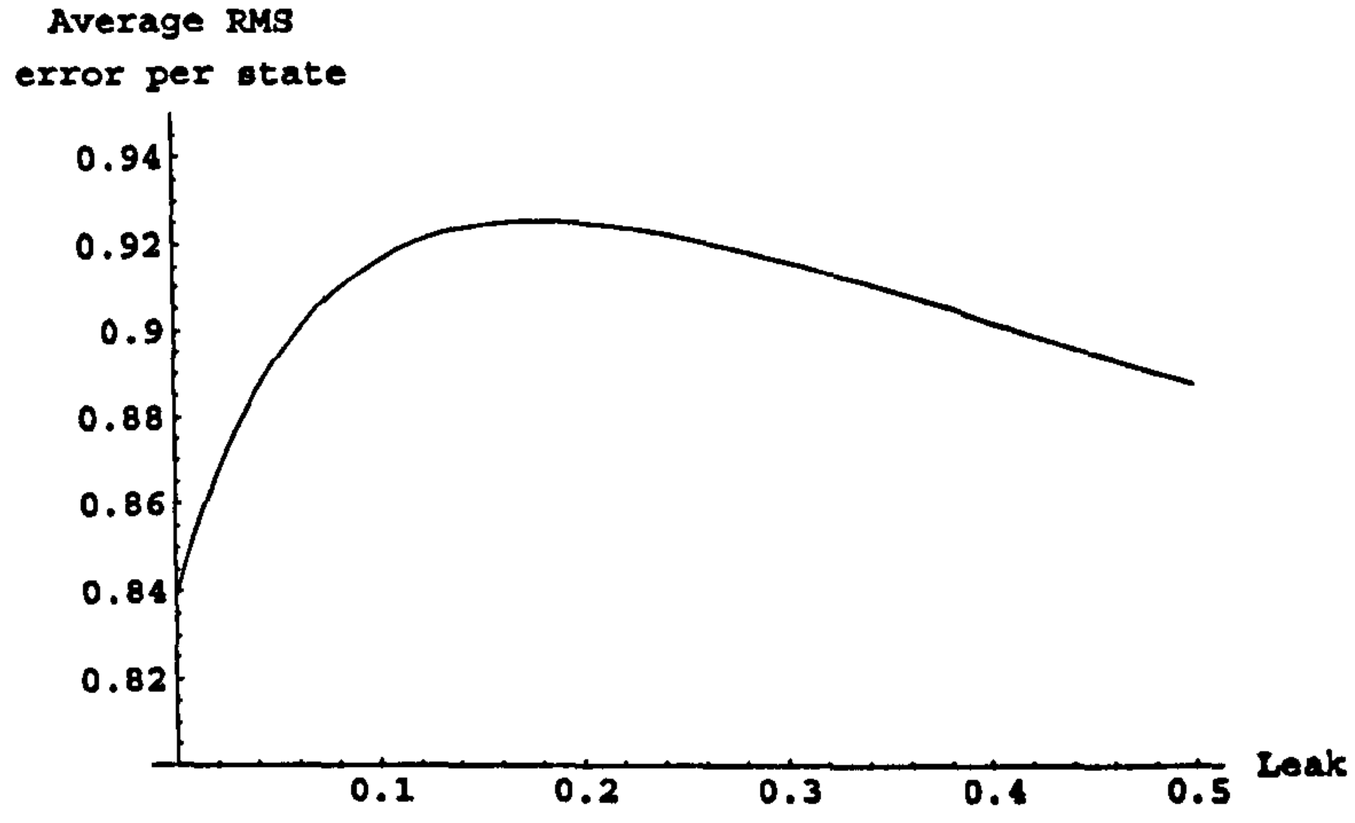


Figure 114: Variation in average RMS per state with "leak" probability

A second difference is in the expected path length. This decreases dramatically as the leak probability increases as can be seen in Figure 115.

Figure 115: Expected path length with "leak" probability

Another important difference in the interpretation of the results of this particular model is the probability of each state occurring in a realisation. It can be seen from Figure 116 that the probability of occurrence of states close to the terminal state is

much reduced by the possibility of a direct jump from states 6 and 12 to the terminal states.

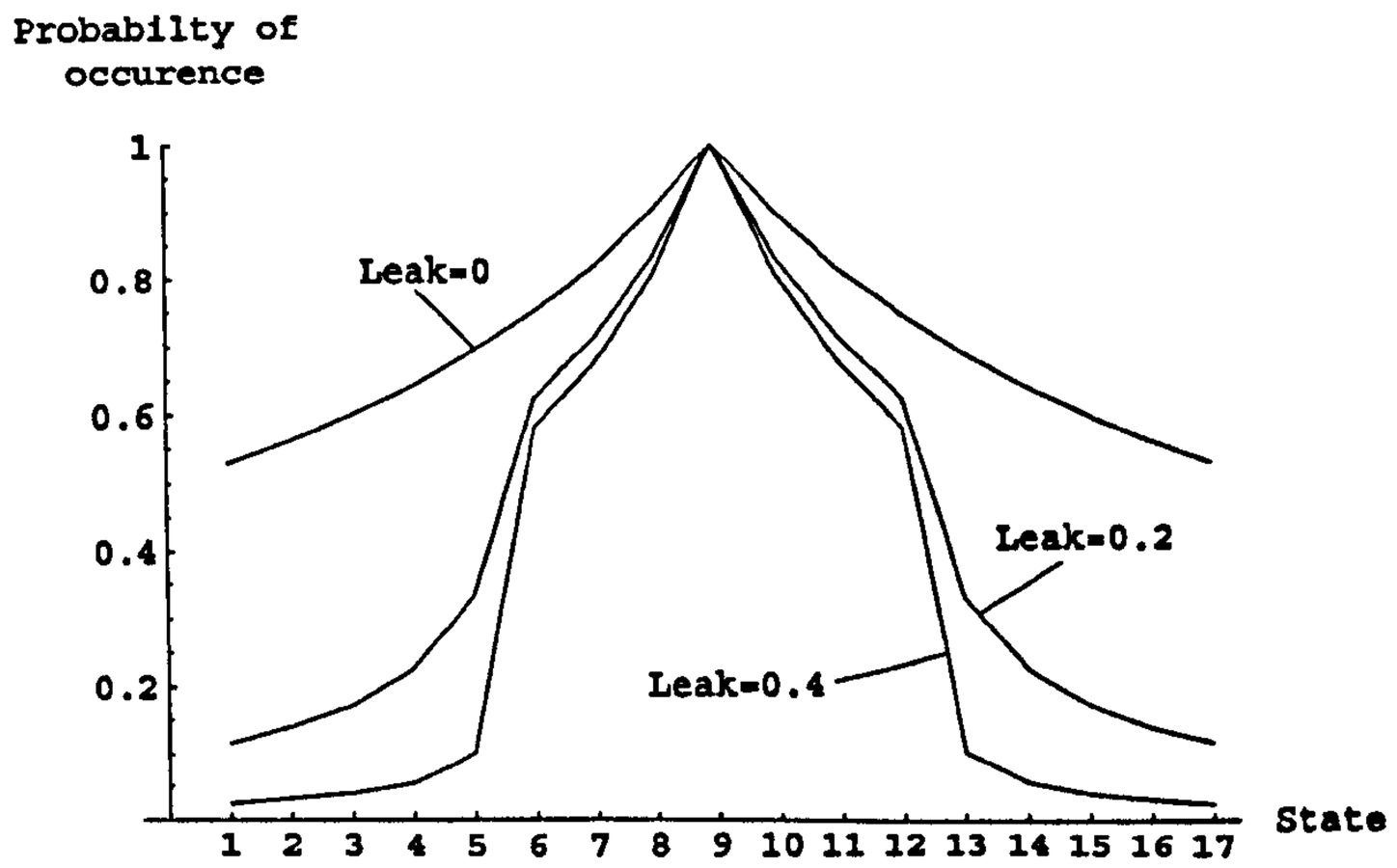


Figure 116: Probability of a state occurring with "leak" probability

In short we should expect the performance of value function estimators to be worse on this model than on the previous linear MDP, but what is of interest is whether or not the TD(λ) and WR(λ) estimators retain their advantage over simpler estimators.

The RMS performance estimator for a non-monotonic MDP

Before moving on to compare the concordance performance of TD(λ), it is worth looking at the RMS performance of the First visit MCA, TD(λ) and WR(λ) estimators as this helps relate the task back to the previous linear MDP with zero leak probability. Using the analytic expressions for TD(λ) from Singh and Dayan (1998), it is possible to compute the first visit estimators using optimal values of α and compare performance with the leak probability, see Figure 117. As can be seen while TD(λ) remains a better estimator than the MCA estimator (λ=1) the asymptotic error increases with the leak probability indicating that the problem is harder.

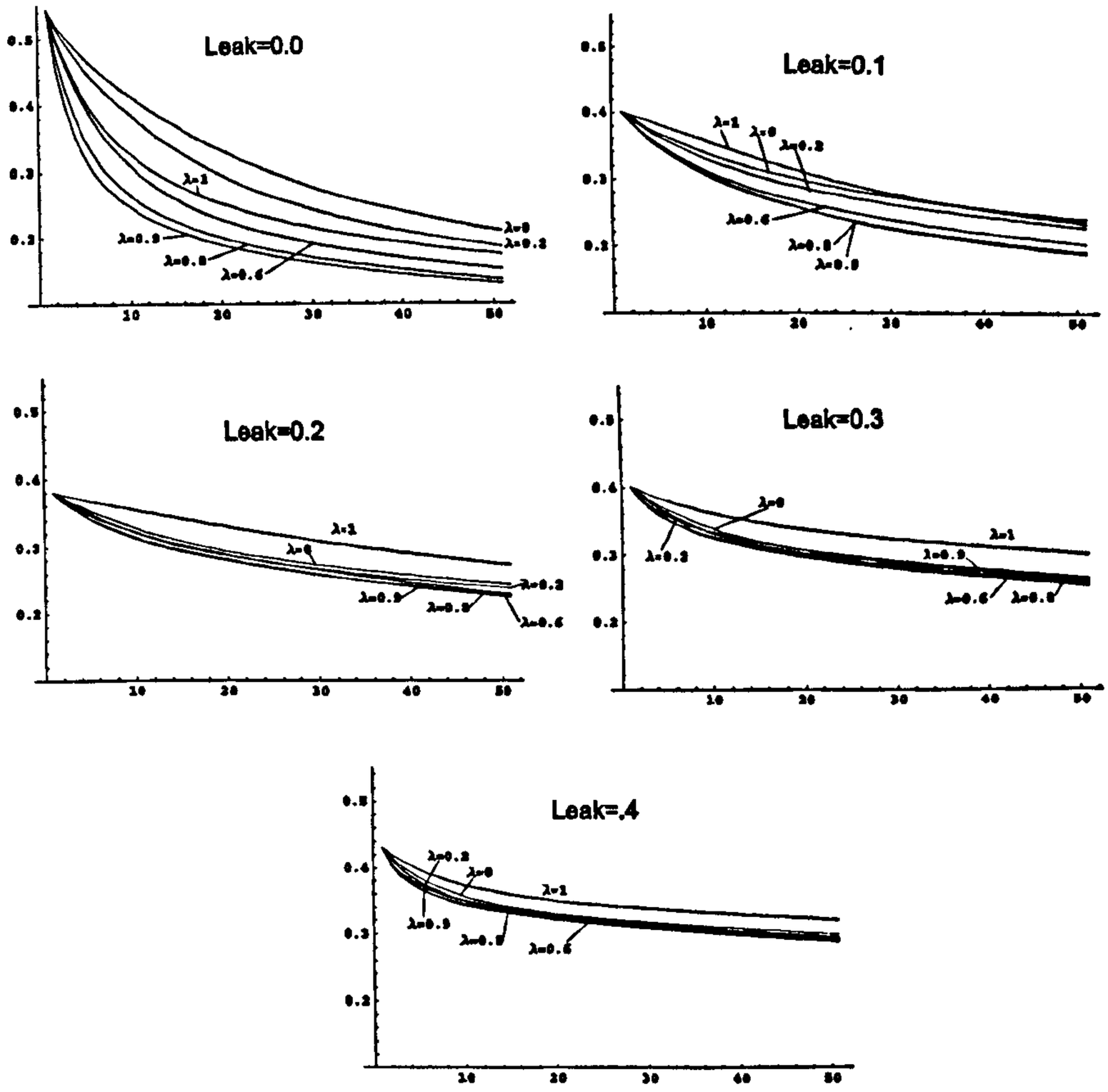
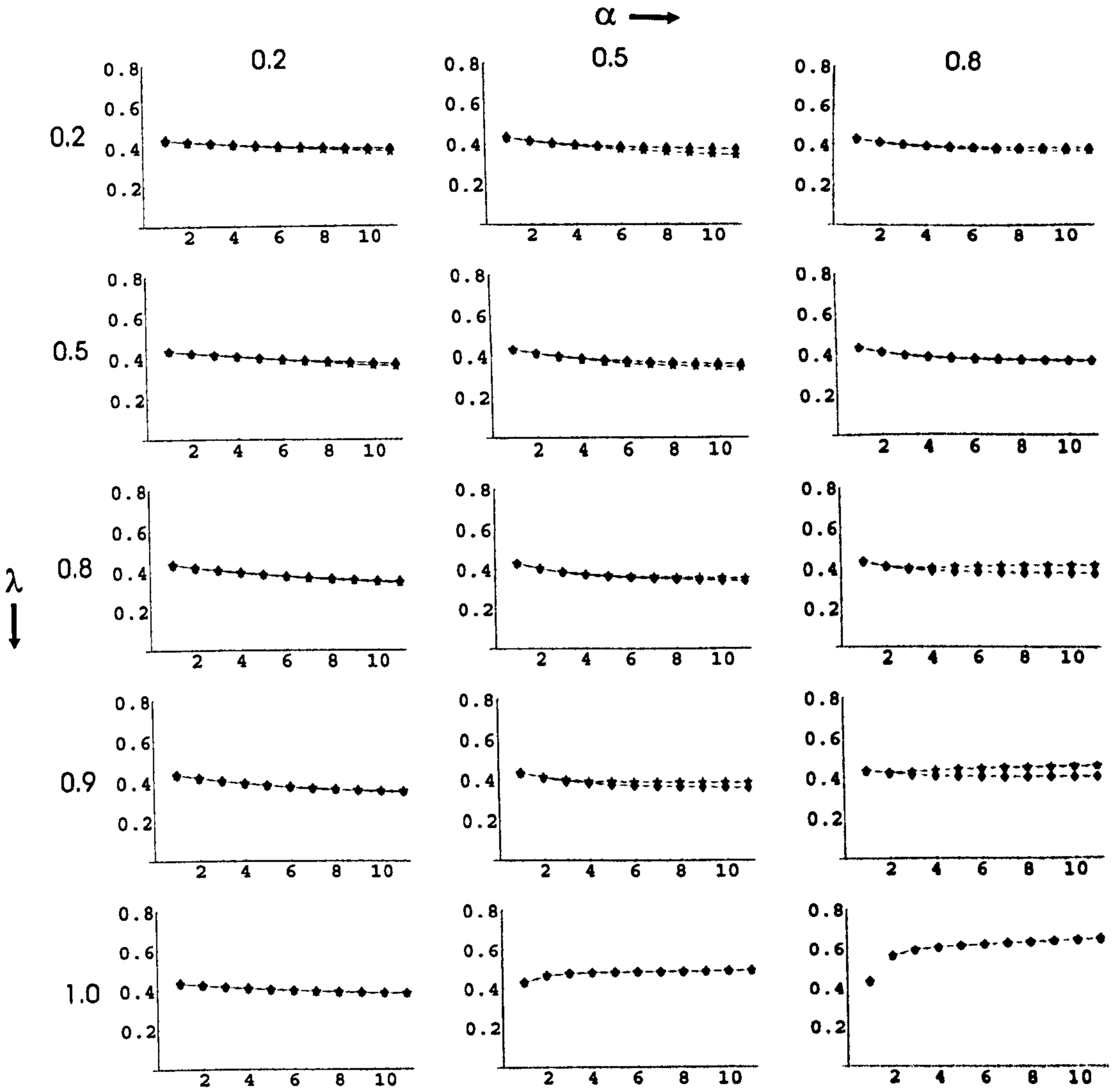


Figure 117: Performance of TD(λ) with "leak" probability

The results for the other variations on the TD(λ) estimator are similar to the first visit case.

It is also worth examining the hypothesis that WR is close to TD in the case of small samples for this new model. As it is closely related to the SRW, there it little change in the small sample results. For example for the case of leak=0.4, which is an extreme case, the small sample performance of TD(λ) and WR(λ) can be seen in Figure 118. As with the SRW, i.e. leak=0, WR and TD are close over the first 10 samples and the results are similar for other values of the leak probability.



NB: vertical axis is the per-state RMS error; horizontal axis is sample size

Figure 118: Comparison of WR(λ) with TD(λ) over first 10 steps with leak=0.4

Concordance performance estimator for a non-monotonic MDP

The results from simulation experiments using the new "leaky" model confirm the fact that TD(λ) and WR(λ) have a noticeable advantage when decision making is the aim of estimating the value function, even when it isn't monotonic. As can be seen in Figures 119 to 122 for a leak probability of 0.1 to 0.4, both TD(λ) and WR(λ)

outperform the First visit Monte Carlo and MCA estimators. It is also clear that performance of the TD(λ) and WR(λ) estimators gets worse, in the sense that they take longer to converge, as the leak probability increases but they still reach good asymptotic performance and are both much better than the corresponding MCA and Monte Carlo estimator.

The performance in these cases is more affected by the value of α than in the case of the MDP with zero leak probability but again the effect is relatively minor with the noise spoiling the performance at larger values of α.

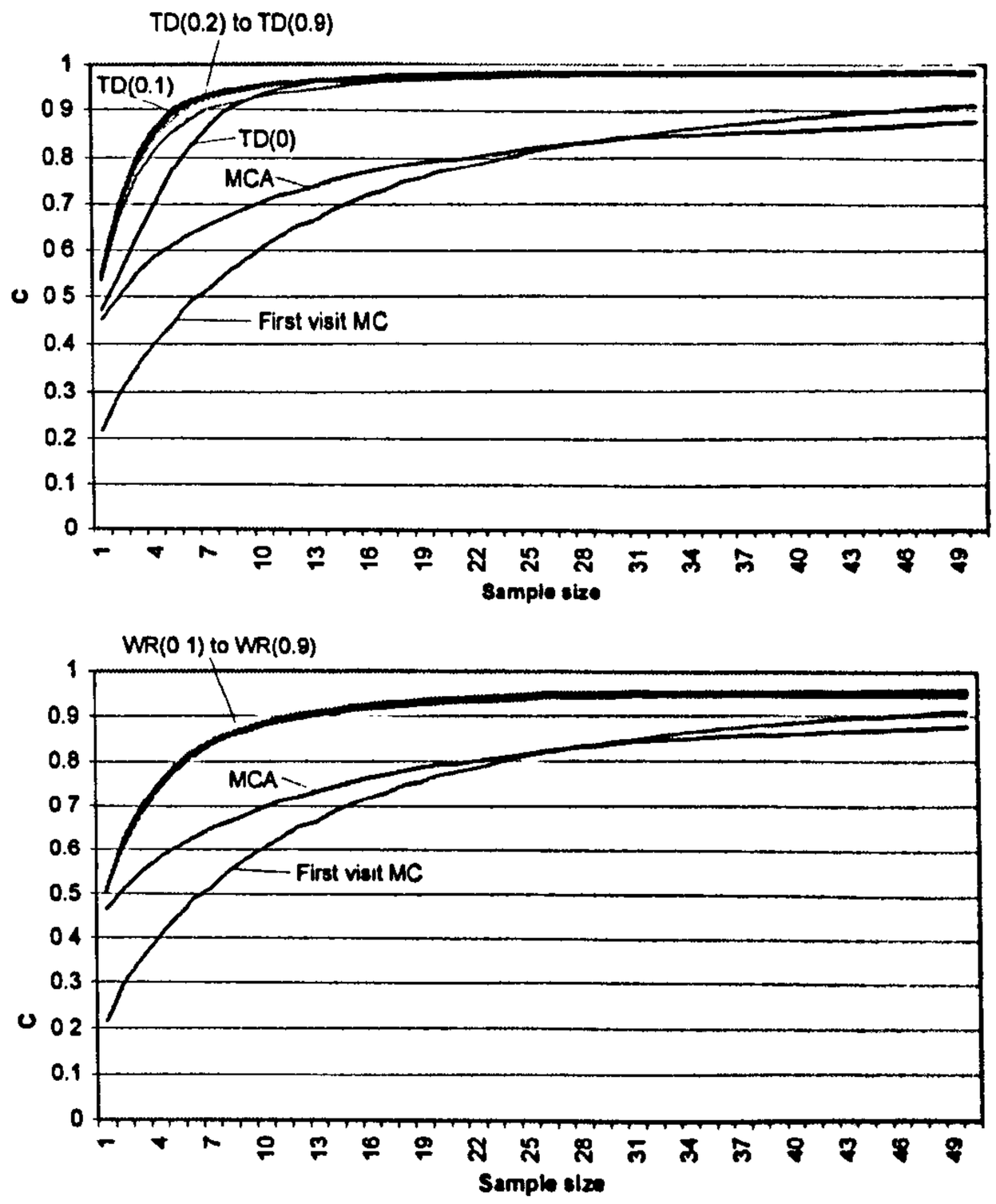


Figure 119: Concordance results for TD(λ) and WR(λ) leak=0.1 for α=0.1

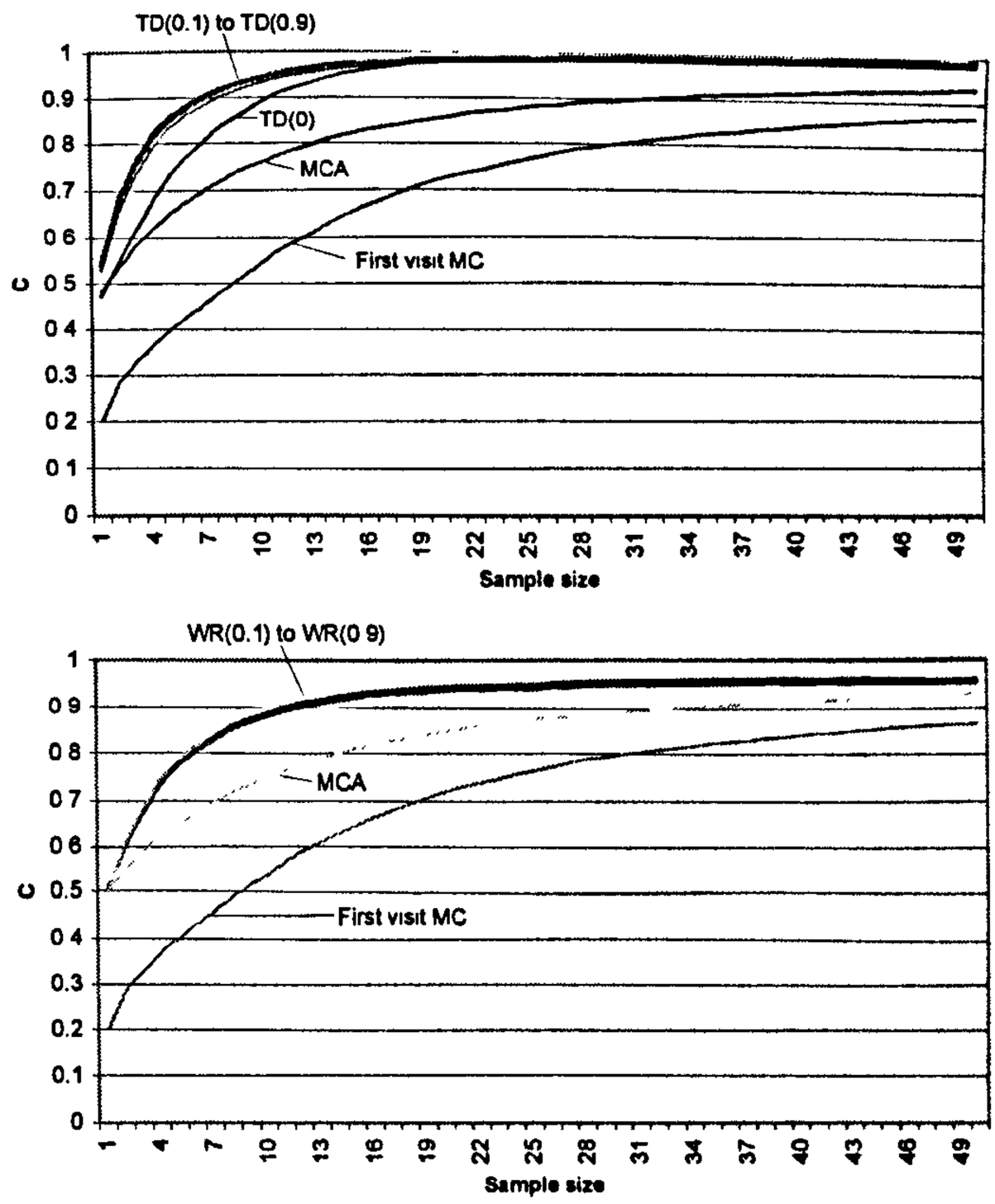


Figure 120: Concordance results TD(λ) and WR(λ) leak=0.2 for α=0.1

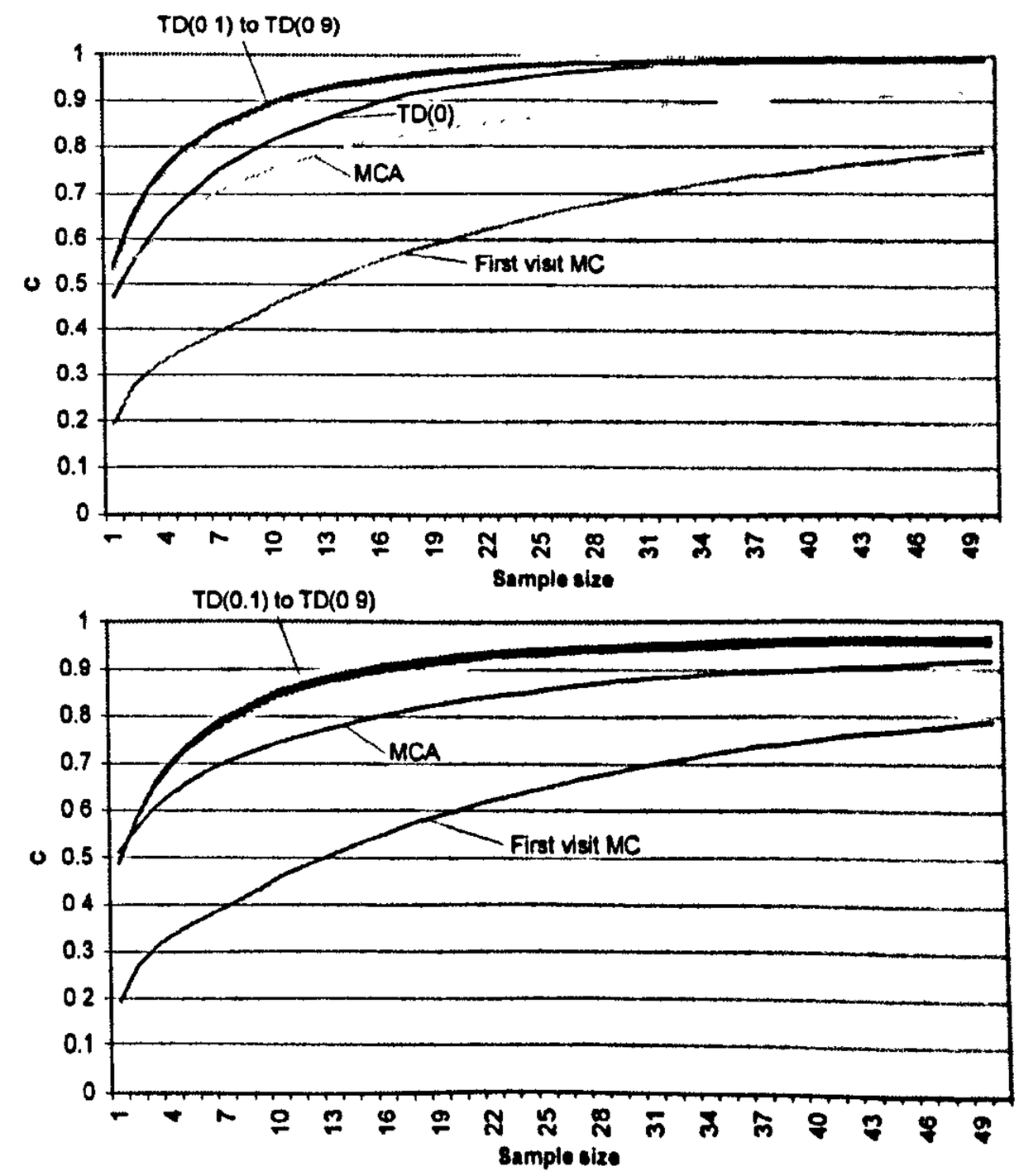


Figure 121: Concordance results for TD(λ) and WR(λ) leak=0.3 for α=0.1

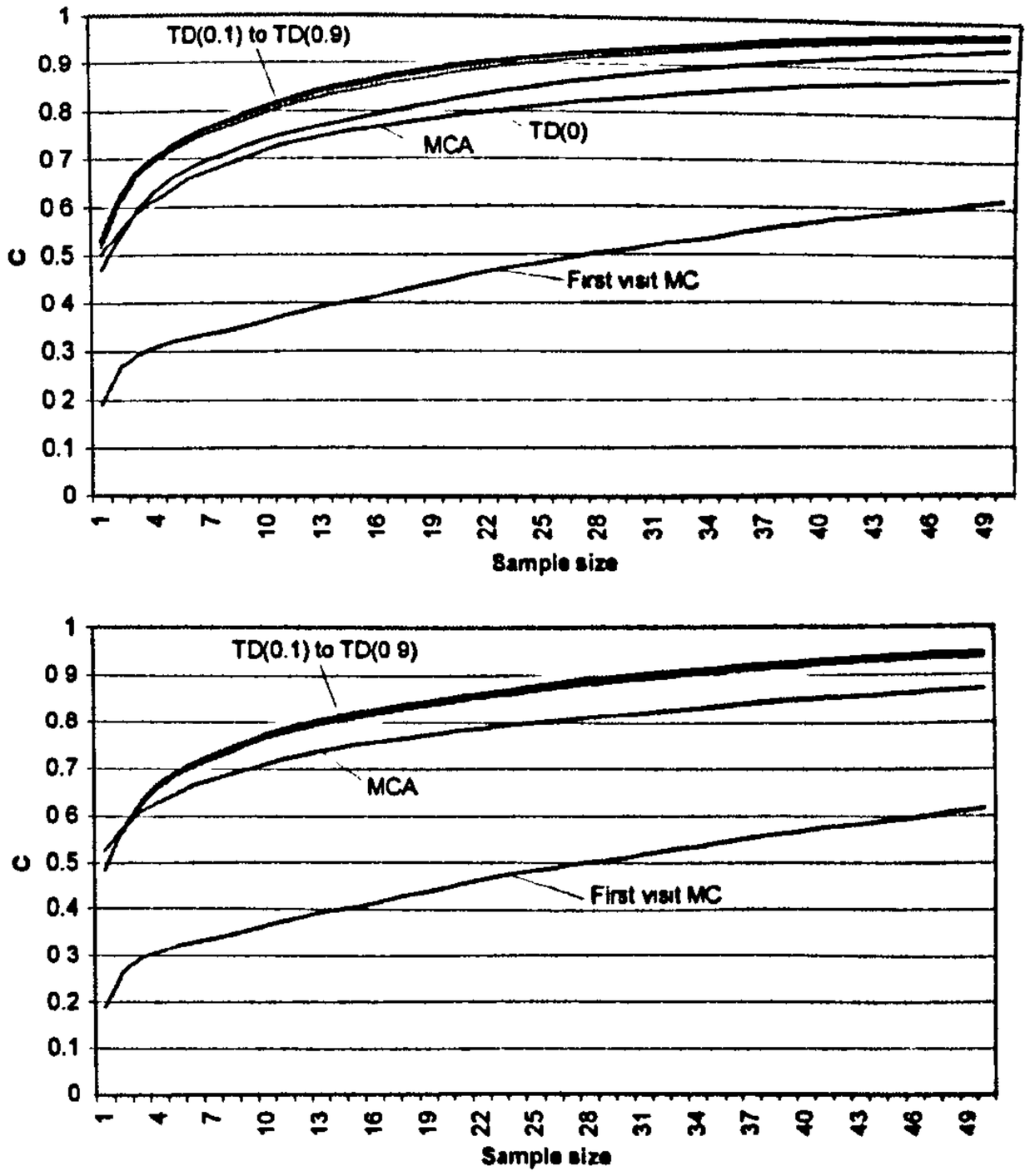


Figure 122: Concordance results for TD(λ) and WR(λ) leak=0.4 for α=0.1

Error pattern

Where the errors occur when using the estimated value function is an interesting question. The pattern of errors turns out to be very similar for a wide range of values of α and λ. For example, Figure 123 shows the concordance measure evaluated at each state for λ=0.5 α=0.1 and leak =0.2 for a sample size of 10.

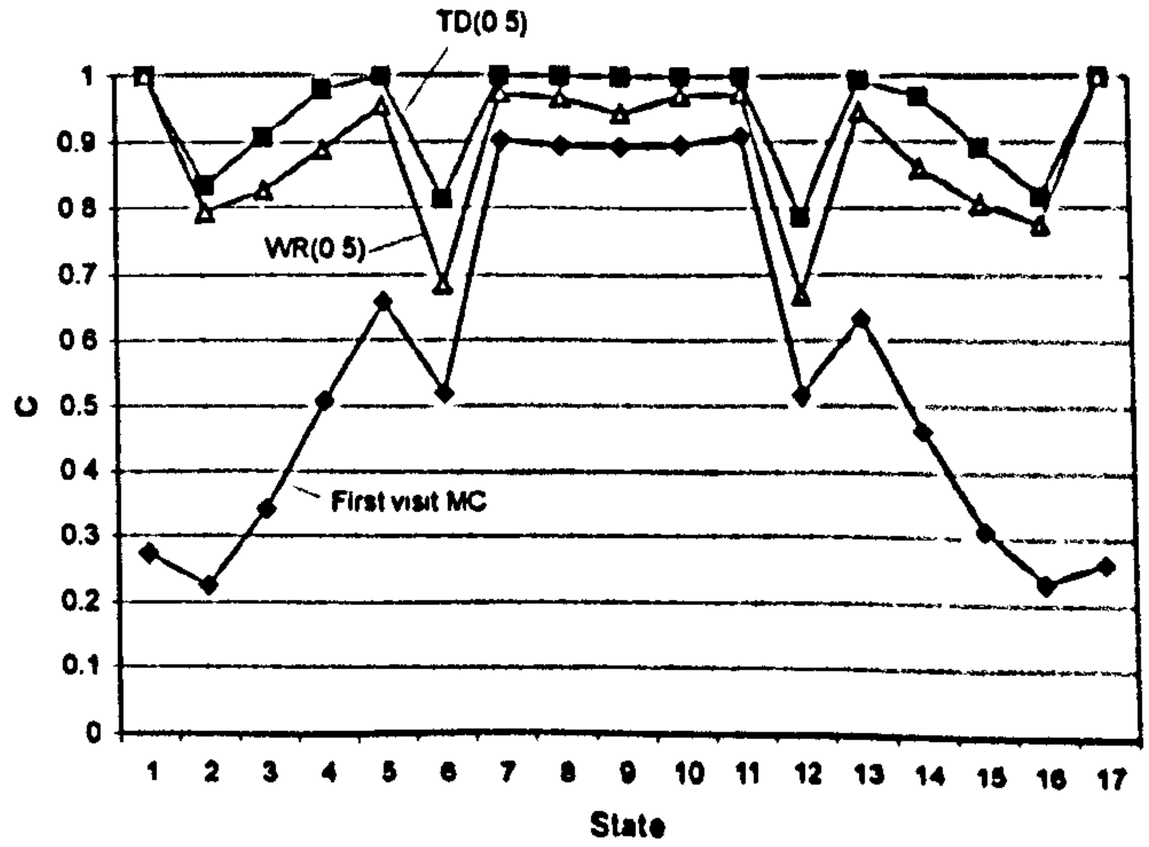


Figure 123: Concordance per state for a sample size of ten λ=0.5 α=0.1 leak=0.2

As can be seen, most of the errors in the TD(λ) and WR(λ) occur at the states where the gradient of the value function changes but the First visit MC estimator makes bigger errors at states closer to the terminal states. The estimators all perform well for the middle states and this is presumably because these have a high probability of occurring in a realisation and hence there is more data available for the estimate. This performance can be compared against a theoretical computation of the statistical distance between the value function of neighbouring states. A suitable measure is a modified F statistic, which indicates how discriminable two groups are on the basis of a single univariate measurement.

The F statistic in this case is:

$$F_i = \frac{\left(V(s_{i-1}) - V(s_{i+1})\right)^2}{\frac{Var\{V(s_{i-1})\}}{Np_{s_{i-1}}} + \frac{Var\{V(s_{i+1})\}}{Np_{s_{i+1}}}}$$

where N is the number of realisations and p_i is the probability of state i occurring in a realisation. The larger the value of F, the further apart are the value functions of the states adjacent to s_i, and hence the more likely it is that a decision based on an estimate of the value function is to be correct. The connection between F and concordance at a given state is not exact because there are other effects not taken into account – specifically the effect of occurrence probability on the sample mean and variance – but the value of F does give an indication of the "ease" of decision making at each state. As can be seen in Figure 124 (for a leak probability of 0.2) the curve is very similar to the pattern of errors encountered for the three estimators.

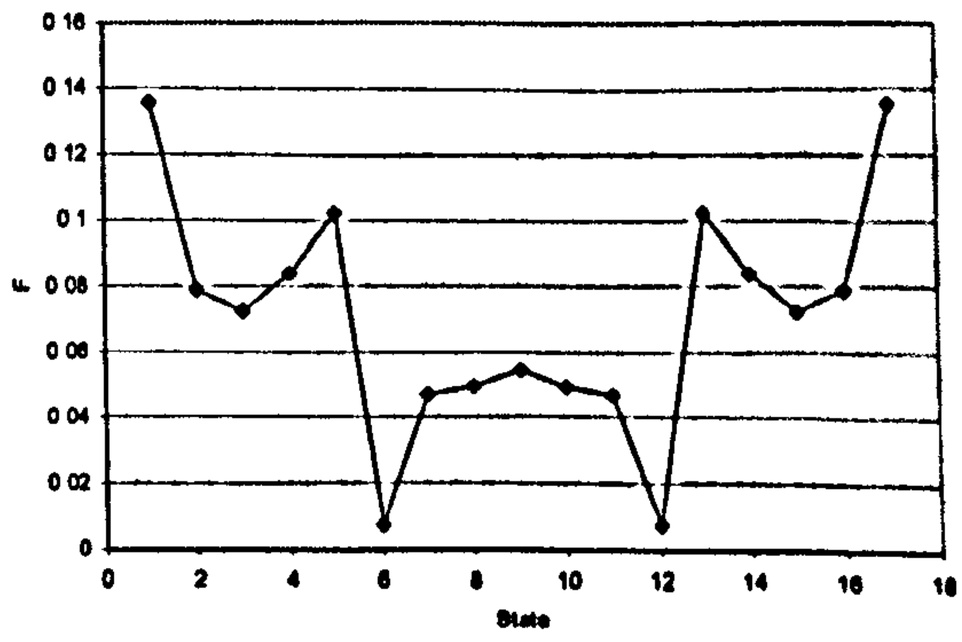


Figure 124: Theoretical indication of "ease" of decision making at each state

What is interesting is the way that all three estimators do better at the middle states than would be expected; presumably due to the higher probability of occurrence of the middle states, and the effect that this has on the estimators' variance. The TD(λ) and WR(λ) estimators also perform better than expected and better than the MCA estimator for the states closer to the reward; the reason for this is less obvious.

As the sample size increases the TD(λ) and the WR(λ) estimators arrive at a stage where the only errors that they are making are at the "leaky" states where the slope of the value function changes. What is interesting is that the errors made at these leaky states are similar to the errors made by the MC estimator at the same sample size, see Figure 125.

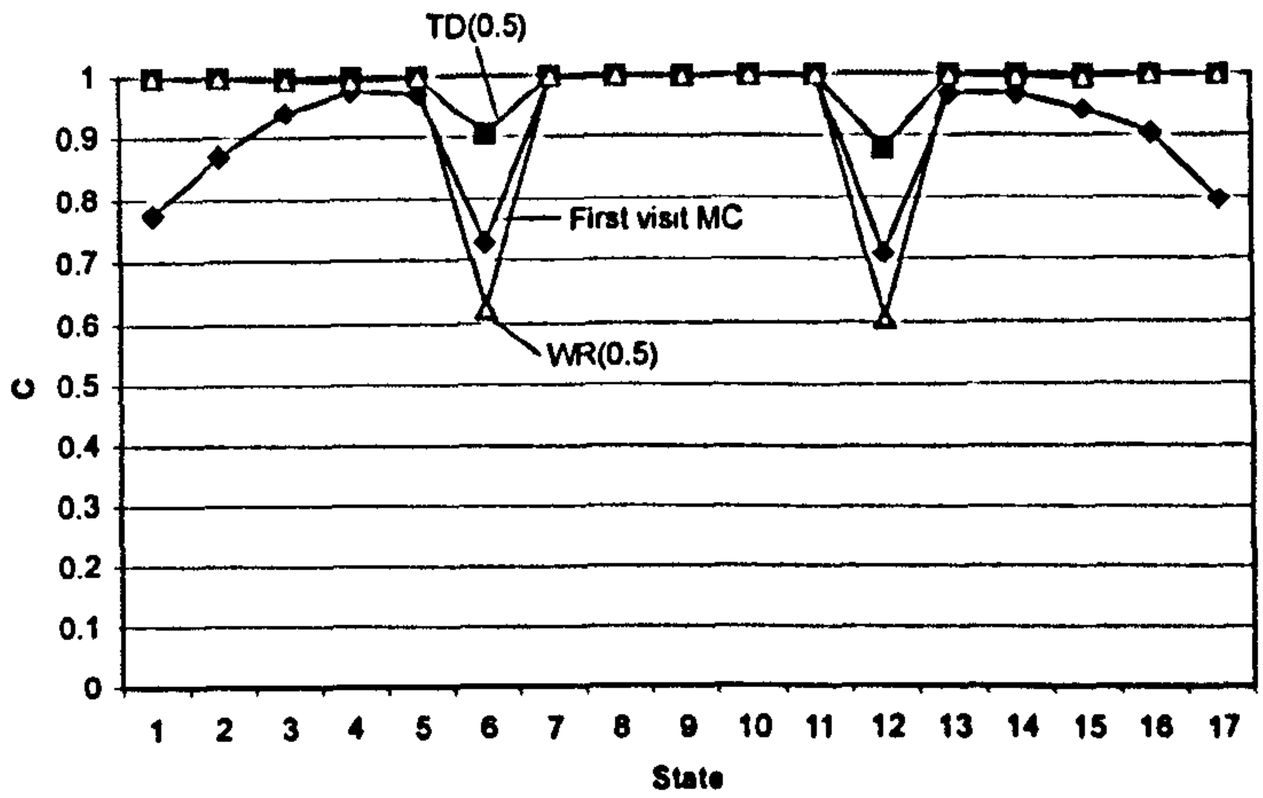


Figure 125: Concordance per state for a sample size of 50 λ=0.5 α=0.1 leak=0.2

What this indicates is that the TD(λ) and WR(λ) estimators are not specially prone to making errors at states where the value function changes slope, which was the original worry concerning the excellent performance of these estimators on the MDP with a monotonic value function. It seems that states either side of a state where the value function changes slope are inherently difficult to estimate and TD(λ) and WR(λ) are no worse than expected at such changes of gradient. This can be taken as yet further evidence that TD(λ) and WR(λ) are not making use of the monotonic properties of the value function to improve their estimates.

Exploration versus Exploitation

The examination of the rank properties of value function estimates it is clear that TD(λ) and WR(λ) are distinctly different to estimators that do not involve weighted rewards such as MCA. Whether or not these differences result in improved performance in reinforcement learning systems depends very much on how they are used. Estimating a value function is usually just a step in a larger procedure concerned with obtaining an optimum policy. What matters is how the value function estimates influence the convergence of algorithms such as policy iteration. There is also the question of "exploration versus exploitation". (For a full account of this topic and of methods for finding optimum policies see Sutton & Barto, 1998).

Given an estimate of a value function then an exploitative policy, or greedy policy, would be to simply select the action that resulted in the maximum reward at each step. However a greedy policy doesn't explore states that haven't been visited or states for which the value function estimates may be unreliable - in a sense this results in a local optimum whereas the aim is to find a global optimum. If the objective is to achieve a long term optimal reward then exploitation has to be traded for some exploration of states to see if a better reward is indeed possible. This raises the question of the effect of a value function estimator on exploration and exploitation. It seems reasonable to suppose that estimators that achieve high concordance with the true value function should provide better exploitation but it isn't clear what effect this would have on exploration.

Clarifying this situation is a large undertaking and a suitable subject for further work but a simple experiment indicates the direction that this further work might take and casts some light on the interpretation of the difference between TD and WR and MCA estimators.

Evaluating an estimate

So far the transition probabilities of the SRW have been regarded as the result of applying some policy but it is also reasonable to regard them as being a stochastic policy applied to the states of the chain. The reward on the left is –1 and +1 on the right and the problem that the reinforcement-learning agent has to solve is to assign

transition probabilities that maximise the reward. It is clear that the solution is to assign 0 to the left-moving probabilities and 1 to the right-moving probabilities – this is the optimum policy. It is also clear that each state has an optimum value of +1.

Starting from a policy that assigns a probability of .5 to both the right and left moving probabilities the value function corresponding to this policy, V, can be estimated from realisations. From this value function estimate a new greedy policy can be derived by setting the probability of transition to the state with the largest estimated value function to 1 and setting it to 0 for all other states. This greedy policy is 100% exploitative and in this situation one would expect estimators with a higher concordance score to produce results closer to the true value function – as only ordinal properties of the estimate are taken into account when the policy is derived. The derived policy can then be evaluated by using it to compute the new value of each state.

To explore the effect of introducing exploration into the policy, a "softmax" (Sutton & Barto, 1998) function can be used to derive the policy from the estimated value function. Softmax assigns the probability:

$$P_{ij} = \frac{e^{\beta V_i}}{\sum_k e^{\beta V_k}}$$

to the transaction probability between state i and j, P_{ij}, based on the estimated value function V. The sum is over states reachable from state i. The parameter β controls the degree of exploration. For sufficiently large values of β, softmax reverts to "hardmax", i.e. it assigns probability 1 to the transition to the state with the largest value. For smaller values of β, the transition to the state with the largest value is assigned a probability <1 and hence other connected states are "explored" by the derived policy. Once again the derived policy can be evaluated by calculating the value of each state under the policy using the usual method. That is, if P is the matrix of transition probabilities derived from V:

$$V^* = (I - S)^{-1} T\mathbf{r}$$

where S is the sub-matrix of P consisting of the transient states, T is the sub-matrix of P consisting of terminal states and **r** is the vector of terminal rewards.

The overall procedure is:

1) Generate a sample and estimate V using the initial policy P_0

2) Derive a softmax policy P using β

3) Evaluate the value of each state under P using $V^{\bullet} = (I - S)^{-1} T\mathbf{r}$

4) Repeat with the next sample

This procedure to evaluate an estimate of a value function was suggested by Dayan (2004) and by varying β the effect of exploration/exploitation on different value function estimators can be examined.

There are a number of practical considerations in implementing this procedure. The first is the effect of the initial value. It is well known that the initial value of the estimate affects the degree of exploration/exploitation (Sutton & Barto, 1998). An initial value greater than the largest reward is "optimistic" and one smaller than the smallest reward is "pessimistic". In some cases an optimistic initial value can keep the system looking for a state with a better value than any it has seen so far, i.e. it causes it to explore the states, and a pessimistic reward can result in the system doing the opposite, i.e. to exploit the states it has seen. In the case of this particular procedure any initial value between the maximum and minimum rewards interacts with the value of α used in the update of the estimates. For example, consider a realisation which terminates on the left with a reward of –1 and an initial estimate of 0. The MCA estimator has the form:

$$(-1,-\alpha,-\alpha,-\alpha,\ldots,0,0,0)$$

The zeros correspond to the states on the right the realisation didn't reach and the –1 is the value of the terminal state. In this case the softmax policy derived from this estimated value function has an equal probability of moving left or right for the states corresponding to $-\alpha$, but a higher right-moving probability for the state with $-\alpha$ on its

left and 0 on its right. This larger right-moving probability results in the estimate of the states' value on the basis of the derived policy being much closer to +1 than it deserves to be on the basis of the evidence. The effect occurs when $-\alpha<c$, where c is the initial estimate. Similarly consider the softmax policy derived from the first state. As it has –1 to its left and $-\alpha$ to its right the result is again a large probability of moving to the right. In fact in most cases this probability is so large, i.e. close to 1, that the calculated value of the derived policy is close to +1 irrespective of the probabilities in the rest of the policy. This effect occurs while α is larger than the smallest reward and a similar effect occurs at the other end of the chain when α is smaller than the largest reward.

One way of removing both effects is to use an initial value that is either so optimistic as to be larger than the maximum terminal reward or so pessimistic as to be smaller than the minimum terminal reward. In this case α has no additional effects over and above the quality of the estimate of the reward and β is the only parameter affecting exploration/exploitation.

Thus in the case of the SRW an initial value of 0 used with terminal rewards of –2 and –1 provides a super-optimistic initial value, and with terminal rewards of 1 and 2 provides a super-pessimistic initial value. In both cases the initial RMS error is the same and so each estimator has to deal with the same initial bias in both cases.

In general it is difficult to quantify the effect of β other than to say that larger values tend to produce an increasingly greedy policy. In this case however we have the exact values, i.e. 0.5 for both moving to the right and the left, for transition matrix P_0 for the model and hence can work out the effect that β has on the derived policy. The true value function corresponding to P_0 for the 19-state SRW is:

-0.89, -0.78,-0.67,-0.55,-0.44,-0.33,-0.22,-0.11, 0,0.11,0.22, 0.33,0.44, 0.55, 0.67,0.78, 0.89

and a hardmax policy derived from this would have a right-moving probability of 1, i.e. it would select the direction of maximum value with probability 1. A softmax policy with β=20 derived from the model's true value function produces a right-

moving probability of 0.90, i.e. a 10% probability of a move to the state with the smaller value or a 90% probability of exploitation. Similarly values of β=12 produce an 80% probability of moving to the largest value, and β=4 produces a 60% probability of moving to the largest value.

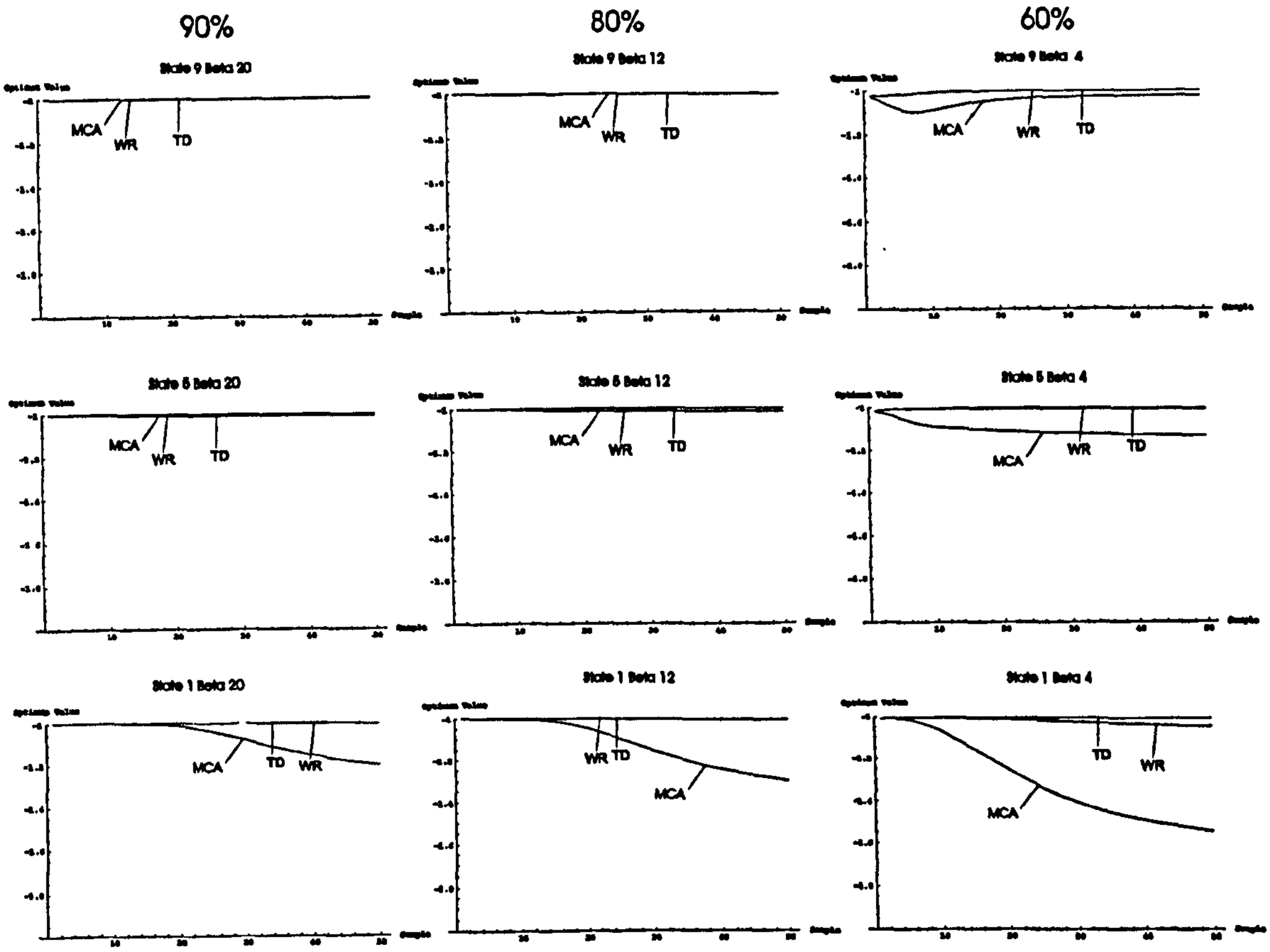


Figure 126: Estimated optimum value of three states using an optimistic initial value for β=20,12 and 4.

The results for an optimistic initial value, i.e. 0, for a reward of –1 or –2 can be seen in Figure 126 for three values of β. Values of α=0.2 and λ=0.9 were used because they give good performance for the three estimators as measured by RMS error and provide a different performance for the MCA versus the TD/WR estimators as measured by concordance. Clearly the estimation problem is harder for state 1 but even here WR and TD give the optimum value, or –1, under as much as 60% exploration. The MCA estimator on the other hand doesn't give the optimum value even at state 9 for 60% exploration. This behaviour is presumably related to the same difference between the estimators as that revealed by the concordance coefficient. It

also reveals that the SRW with 19 states and just two terminal rewards is perhaps a little too easy a problem.

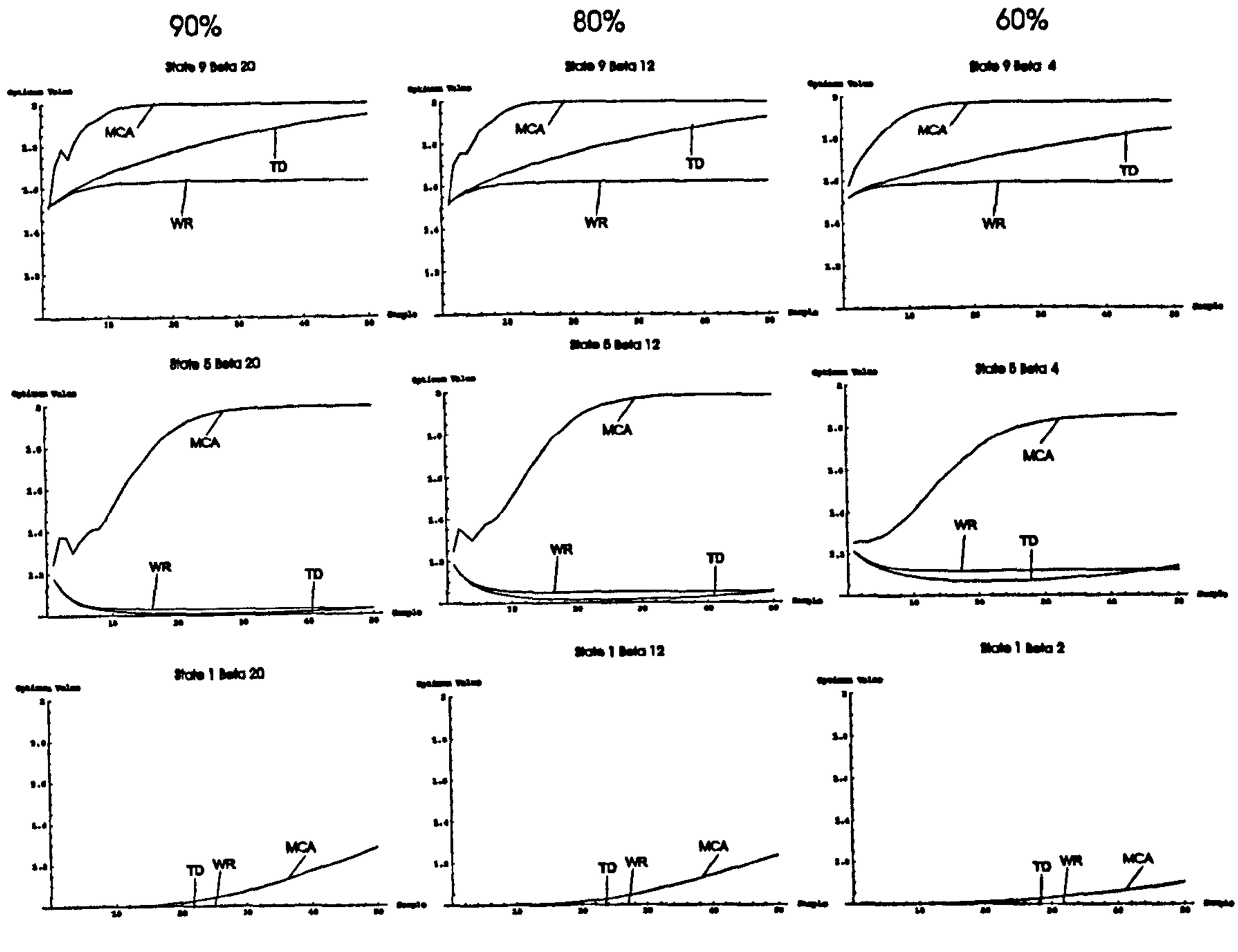


Figure 127: Estimated optimum value of three states using a pessimistic initial value for β=20, 12 and 4.

The results for a pessimistic initial value i.e. 0 for a reward of 1 and 2, can be seen in Figure 127. In this case the MCA estimator gives the optimum value at states 9 and 5 fairly quickly but the WR and TD estimators give a lower value at all states and all levels of exploration. Indeed the WR and TD estimators give a value close to 1 as the optimum value for states 1 and 5 for all three values of β. The reason for this behaviour isn't clear but it doesn't change significantly with the degree of exploration.

In both cases, optimistic and pessimistic initial values, there is a clear difference in the behaviour of the MCA and the TD and WR estimators. In the case of the optimistic initial value this seems to favour the TD and WR estimators but in the case of the pessimistic value it seems to favour the MCA estimator. Thus, while the

concordance coefficient reveals that there is a difference between these estimators, it isn't clear if this difference is important in more complex procedures aimed at finding the optimal policy with exploration.

The rapid convergence, at least in the case of the optimistic starting value, fits in with the proof by Kearns and Singh (1999) that both direct and indirect estimation of the optimal policy has fast convergence to the optimal policy as a function of state transitions observed. They show that the rate of convergence is faster than that of an approach that attempts to estimate model parameters on the way to deriving optimal policies, and that even very poor estimates of the model parameters are good enough to derive a near-optimal policy. These results are in terms of estimates of the transition probabilities, but a similar result should be derivable in terms of value function estimates.

Conclusion

The results outlined in this chapter show that there are distinct differences between the way the MCA estimator and TD/WR estimators behave. The TD and WR estimators achieve a better match with the true value function in terms of ordinal relationships between the values of states than does the MCA estimator. What is more, the performance of TD/WR is relatively insensitive to the values used for λ and α. That such a difference exists is clear, what is less clear is how important it is for the use to which value function estimates are put. In particular, is this difference an advantage when trying to derive the optimal value function by policy iteration or some other method? The results of a simple experiment which estimates the optimal value of states under different degrees of exploration/exploitation trade-off indicates that the answer is far from clear. With optimistic initial values the TD/WR estimators converge rapidly to the optimal values, performing well against the MCA estimator, but with a pessimistic initial value they perform poorly against the MCA estimator.

Chapter Twelve
Discussion, Conclusion and Future Work

This chapter is an overview of results obtained and presents the conclusions which may be drawn from them. Suggestions for further work are considered. These include a deeper examination of the ordinal properties of TD(λ) and WR(λ) estimates and how these influence procedures which estimate the optimal value function.

On-line, off-line, first and every

In Chapter Five the result was presented that a TD(λ) estimator of the form:

$$\mathbf{v}_{t+1} = (\mathrm{I} - \alpha \mathrm{Diag}[\chi])\mathbf{v}_t + \alpha(\mathrm{M}\mathbf{v}_t + \mathbf{n}\mathrm{r}_t)$$

where M is a stochastic matrix and **n** a stochastic vector, does not diverge if M and **n** have total row sums smaller than or equal to one. Putting this in a more directly applicable form it means that convergence is guaranteed as long as the lambda weights in a TD(λ) estimator sum to one and hence form a true weighted average of the reward and none-reward elements.

This was used to examine the way that the various forms of the eligibility traces TD estimator either do or do not converge. The Accumulating and Replacing traces forms do diverge for some values of α and λ and this is a model dependent phenomena. That is, faced with an MDP with unknown parameters it is impossible to select a value of α that can be guaranteed not to cause divergence in the estimate. It is also obvious that given any choice of α it is possible to construct a Markov chain that has sufficient recurrence to make the estimator diverge. Even if a value of α is known to produce a convergent estimator there is always a non-zero probability that a particular sequence of realisations will cause the estimator to diverge. While Replacing traces TD(λ) is less prone to divergence because of the way that the lambda weights reduce the effective value of α it is subject to the same problems as Accumulating traces.

It is clear from this result that it is possible to "correct" the divergence in all forms of TD estimator by forcing the lambda weights to sum to one. An example of this

procedure was provided by the corrected Accumulate trace TD(λ) estimator. The corrected estimators also converge in the mean to the value function, as do the uncorrected estimators, but for a full range of α. It is also clear that the eligibility trace form of the corrected off-line Every visit estimator satisfies the requirements of the convergence theorem of Bertsekas and Tsitsiklis (1996) and so converges not only in the mean but with probability one.

In Chapter Six the subject of the First and Every form of the MCA estimator was examined in detail. Building on the results of Singh and Dayan (1998), analytic forms for the RMS error of the First visit MCA, on-line Every visit MCA and off-line Every visit MCA were presented. These equations are used to prove that for the same asymptotic error the First visit MCA estimator has a larger rate of convergence and hence is presumably to be preferred to the alternatives. This complements the results provided in Sutton and Singh (1994) relating to the case of immediate rewards and $\alpha = 1/N$ where it is proved that the Every visit estimator is eventually worse than the First visit estimator.

Further work

The result that an iteration of the form:

$$\mathbf{v}_{t+1} = (\mathrm{I} - \alpha \mathrm{Diag}[\chi])\mathbf{v}_t + \alpha(\mathrm{M}\mathbf{v}_t + \mathbf{n}\mathbf{r}_t)$$

not only converges but converges in the mean to the value function with only the conditions that they are non-negative and satisfy a relation of the form $\mathbf{n} + \mathrm{M}\mathbf{l} = \chi$ is interesting and should be capable of extension to the proof of convergence with probability one. To do this additional conditions, almost certainly the same as in Bertsekas and Tsitsiklis (1996), are needed on α to ensure convergence in the r^{th} mean as well as the mean. One possible approach would be to simply express the general iteration in an eligibility trace form and then show that the eligibility traces satisfy the necessary conditions.

The question of the superiority or otherwise of the various alternative ways of using recurrent states in estimators could be extended to the case of estimators which use rewards that are functions of the realisation or to the case of immediate rewards. This could be achieved by investigating the ways in which the Markov property restricts

the correlation between rewards within the same realisation. In the case of terminal rewards and the MCA estimator it is clear that the correlation between rewards within the same realisation is one. This is the case where using the recurrent states in an Every visit style of estimator is likely to produce the smallest advantage in terms of RMS reduction It is clear that the Markov property places a restriction on the covariance between occurrences of the reward, irrespective of the reward mechanism as:

$$COV[R_j R_{j+l} \mid k \geq j+l] = COV[R_1 R_l \mid k \geq l]$$

That is, the covariance between rewards obtained at different occurrences depends only on the "distance" between them and the covariance of the first occurrence with the second, third and so on determines all of the covariances. How this could be used to derive either expression for the exact RMS for an estimator with minimal assumptions on the reward mechanism isn't entirely clear.

Temporal Difference philosophy

The task of estimating the value of a state under a given policy is an important part of reinforcement learning. Estimates of state value can be used to improve the current policy and hence the efficiency of estimation is important to the rate of learning. The only sophisticated estimator of the value function that has been extensively promoted (Sutton & Barto, 1998) is TD(λ) and variants based on it. However it still isn't clear when TD is better than alternative estimators. To recall the quote from Sutton and Barto (1998):

> *"If both TD and Monte Carlo methods converge asymptotically to the correct predictions, then a natural next question is "Which gets there first?" In other words which method learns faster? Which makes the more efficient use of limited data? At the current time this is an open question."*

Although there is much evidence that TD can outperform simpler estimators there is currently no proof of optimality and its relationship with model types and parameters is unclear. From experimental studies it seems that TD(λ) generally outperforms simpler estimators early in the estimation procedure and then settles to a similar performance. In practice there is also the issue of the sensitivity to the setting of the λ

and α parameters. How can one be sure that a good choice has been made when the structure of the model is unknown?

The TD(λ) estimator is linked to a wider concept of learning by temporal differences (Sutton & Barto, 1998). In its simplest form this states that we can learn the value of a state from the estimates of the value of the states that follow it in the development of the system. This is most certainly a reasonable idea in that it imposes the structure of the Bellman equations on the estimates of value. That is, if the estimates of the value of all of the subsequent states are exact, the Bellman equations tell us that using the sample mean of these value estimates is a sensible estimator of the value of the current state. This reasoning leads on to the construction of the TD(0) estimator which only involves the use of estimates of value of the subsequent state to update the estimate of value of the current state. So, in TD(0) the value of the subsequent state is used in lieu of the eventual true reward. This scheme is consistent in the sense that under reasonable conditions it converges to the true value function, but it isn't particularly efficient in the early stages of estimation because of the way information propagates.

TD(0) "propagates" knowledge of the eventual reward back to earlier states one state at a time. In this sense it operates like a "bucket brigade" for temporal credit assignment. This "local" architecture has obvious appeal in that it limits the memory requirements of any reinforcement learning system by only ever needing the system to remember the previous state for update. It is also appealing as an explanation of reinforcement learning in biological system where the shifting of the credit to earlier states as knowledge about the reward improves seems to model the natural process (Dayan & Abbott, 2001).

The problems with temporal difference learning philosophy really only start to emerge when the TD(0) estimator is generalised to the TD(λ) estimator. It is clear that TD(0) and TD(1) can be regarded as being the extremes of two approaches to using the information provided by the reward. TD(0) only makes use of the current estimate of value of the subsequent state whereas TD(1), being computationally equivalent to the usual sample mean reward, waits until the process terminates before using the actual reward obtained to update the values of all of the states encountered. The introduction of TD(λ), where λ is allowed to vary between 0 and 1, can be

viewed as an application of a simple mathematical identity or as an extension of the philosophy of temporal difference learning to something much more sophisticated.

As λ moves away from 0 the number of subsequent states included in the estimator effectively increases. This is, in a sense, an extension of the temporal difference method to more than single-step differences. It should be noted that, at the same time as λ increases, an increasing amount of information about the final reward is being incorporated into the estimates. For example, in a single realisation the TD(0) estimator only uses the final true reward to update the penultimate state. In the case of the TD(λ) with λ large enough to effectively average the value of n subsequent states it is clear that the n states that occurred before termination receive a significant amount of information about the actual reward.

Thus TD(λ) has to be seen both as a blending of the temporal difference methods with something that is very much a non-temporal difference method.

The small sample performance of TD(λ)

When the performance of the TD(λ) is examined it is clear that its small sample performance is best for values of λ closer to 1 than 0. The reason for this small sample performance is attributed to the principle of temporal difference learning and to a "bootstrapping" process using the initial estimates to produce better estimates than the using the reward alone. However this raises the problem of how TD methods can provide an early advantage at a time when the initial estimates have little or nothing to do with the expected rewards. It is much more reasonable to see temporal difference learning as being an advantage later in the estimation procedure when the value estimates have started to settle down. Consider the situation of estimating the value of a state by forming an average of the subsequent states' estimated value functions when these are initially set to random values. Clearly no learning at the state in question will take place until the estimates at the subsequent states have been improved to resemble the true values. Obviously if temporal difference learning has an advantage it is unlikely to be in the first few steps of estimation and yet TD(λ) does indeed show its advantage early in the estimation process. This implies that there is indeed some other mechanism in operation during this phase of learning.

Examination of the form of the general TD(λ) estimator suggests that, rather than the averaging of the current estimates of the value of subsequent states, it is the use of the weighted reward which is responsible for the performance. This amounts to using the naïve estimator of the discounted reward to estimate the undiscounted reward – this is the principle of the WR(λ) estimator.

As the discounted reward approaches the undiscounted reward, as λ tends to 1, the bias vanishes and so it is possible to implement an unbiased estimate of the expected reward by using a suitable scheme of values for λ that depend on the sample size.

This said, it is worth noticing that TD(λ) is not a biased estimator and this is clearly due to the action of the temporal difference component of the estimator. By mixing in the estimates of reward at subsequent states the bias is removed from the estimator based on just the discounted reward and hence the TD(λ) estimator reaps the benefits of the variance reduction provided by using the discounted reward and automatically corrects the asymptotic bias without having to change the value of λ.

In Chapters Seven, Eight and Nine the TD(λ) and WR(λ) estimators were compared using a range of models and initial values designed to reveal any differences that might exist. An additional model, the "leaky" SRW, was also used in Chapter Eleven. The various forms of estimator – First, Replacing traces and Accumulating traces – were also compared using the appropriate TD(λ) and WR(λ) on the same range of models. In no case was there an incident of TD(λ) showing a small sample advantage over the MCA estimator without WR(λ) showing the same advantage and being close to TD(λ). Of course the WR(λ), being an asymptotically biased estimator, moved away from the TD(λ) estimator, as the sample size increased. Adjusting the size of the λ to make WR(λ) have the same asymptotic error as TD(λ) is a crude way of examining the way that temporal difference learning removes the bias by increasing the effective value of λ as the sample size increases. In all cases it was possible to match the performance of TD(λ) well enough to suggest that it isn't extracting additional information from the data that WR(λ) is ignoring.

Given these results it is possible to propose an interpretation of the way that TD(λ) works. Initially the estimate acts like a WR estimator and this gives it a small sample advantage – provided the WR estimator has a small sample advantage for the

situation in question. As the estimation procedure progresses the improved estimates of the value function allow temporal difference learning to occur and this can be thought of as increasing the effective value of λ to 1 by forming the weighted averages of these estimates.

Further work

Given that any TD estimator can be written as

$$\mathbf{v}_{t+1} = (I - \alpha \mathrm{Diag}[\chi])\mathbf{v}_t + \alpha(M\mathbf{v}_t + \mathbf{n}\mathbf{r}_t)$$

and the corresponding WR estimator is:

$$\mathbf{v}_{t+1} = (I - \alpha \mathrm{Diag}[\chi])\mathbf{v}_t + \alpha \mathbf{n}\mathbf{r}_t$$

it should be possible to quantify the difference between the two using matrix norms. The fact that $\mathbf{n} + M\mathbf{l} = \chi$ for a convergent estimator should place constraints on the maximum difference between the two estimators independent of the model. For example, it is clear that at the first step of estimation the difference between the two estimators satisfies:

$$\|TD_1(\lambda) - WR_1(\lambda)\| \leq \alpha \|M\| \|\mathbf{v}_0\|$$

It isn't clear how to generalise this to other steps in a way that will yield a useful bound on the difference between the estimators.

The fact that

$$\mathbf{v}_{t+1} = (I - \alpha \mathrm{Diag}[\chi])\mathbf{v}_t + \alpha(M\mathbf{v}_t + \mathbf{n}\mathbf{r}_t)$$

converges in the mean to the value function if $\mathbf{n} + M\mathbf{l} = \chi$ could also be used to create an improved WR estimator. The WR estimator does not converge in the mean to the value function simply because $\mathbf{n} + 0\mathbf{l} \neq \chi$. As in the case of the Replacing and Accumulating traces TD it is possible to correct this "imbalance" by modifying the M matrix so that $\mathbf{n} + M\mathbf{l} = \chi$. Setting M to have the usual lambda weights in TD solves the problem, but this is only one possible choice. For example, $M = \mathrm{Diag}[\chi - \mathbf{n}]$ satisfies the condition and results in the following estimator:

$$\begin{aligned}\mathbf{v}_{t+1} &= (\mathrm{I}-\alpha\mathrm{Diag}[\chi])\mathbf{v}_t + \alpha(\mathrm{Diag}[\chi-\mathbf{n}]\mathbf{v}_t + \mathbf{n}\mathbf{r}_t) \\ &= (\mathrm{I}-\alpha\mathrm{Diag}[\chi]+\alpha\mathrm{Diag}[\chi-\mathbf{n}])\mathbf{v}_t + \alpha\mathbf{n}\mathbf{r}_t \\ &= (\mathrm{I}-\alpha\mathrm{Diag}[\mathbf{n}])\mathbf{v}_t + \alpha\mathbf{n}\mathbf{r}_t\end{aligned}$$

This is effectively an alpha update estimator with $\alpha' = \alpha\mathbf{n}$. In other words, the estimator is:

$$\mathbf{v}_{t+1}(s_i) = (1-\alpha\lambda^{d_i})\mathbf{v}_t(s_i) + \alpha\lambda^{d_i}\mathbf{n}\mathbf{r}_t$$

where λ^{d_i} is the "distance" of the occurrence of state i from the end of the realisation, i.e. the usually WR reward term. This is clearly does not use temporal difference terms and can be regarded as a "corrected" WR estimator. As it satisfies $\mathbf{n} + \mathrm{M}\mathbf{l} = \chi$ it converges in the mean for all α to the value function. A preliminary investigation of this estimator reveals that it works as well or better than either TD or WR on the SRW model and is asymptotically unbiased. Other choices of M are also clearly possible.

Concordance

The use of RMS error as a measure of the quality of value function estimation is very reasonable, but the assessment of quality should also take into account what the value function is going to be used for. As Kearns and Singh (1999) point out, a model with poorly estimated parameters can still provide very good estimates of the optimal value function. The concordance coefficient is an attempt to measure how good an estimate is in terms of deriving better policies. The estimate may be some way from the true value function in RMS terms, but if it is close in concordance terms then it should allow a the optimal value function, and hence optimal policy, to be derived.

The fact that the TD and WR estimators provide estimates which are closer in concordance terms than the MCA estimator is certainly an indication of a qualitative difference between these estimators. The difference is quite large compared the fairly modest RMS advantage over the MCA estimator. The performance of TD and WR is also fairly parameter-insensitive as measured by concordance as compared to the RMS measure.

These differences are also manifest when the estimators are used within a procedure to estimate the optimal value of a state. However, the conclusion that *TD* and *WR* are preferable to the MCA estimator cannot be drawn. The experiment is complicated by the interaction of the initial value and the reward structure. An optimistic initial value results in good performance for TD/WR in line with the concordance coefficient results when there is high exploration, i.e. low values of β. However a pessimistic initial value produces the opposite result, that is MCA is to be preferred to TD/WR. What is more, this is true irrespective of the value of β.

Further work

The idea of quantifying the suitability of a value function estimate in terms of how well it can be used to derive the optimal value function and policy is an area that deserves to be explored further. The results from the 19-state SRW model suggest that more complicated models need to be used to compare this aspect of performance. The influence of optimistic/pessimistic starting values on the quality of value function estimates needs to be explored and to be related to concordance measures. A theoretical and/or practical examination of the relationship between poor value function estimates and the derived optimal value function is also worth pursuing along the lines of Kearns and Singh (1999).

Appendix I

The Difference Form of TD(λ)

The basic TD(λ) update is:

$$V_{k+1}(s_t) = V_k(s_t) + \alpha[R_t^\lambda - V_k(s_t)]$$
$$= V_k(s_t) + \alpha\Delta_t^\lambda$$

with

$$R_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}[\sum_{k=1}^{n-1}\gamma^{k-1}r_{t+k} + \gamma^n V_k(s_{t+n})]$$

Writing:

$$\Delta_t^\lambda = R_t^\lambda - V_k(s_t)$$
$$= (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}[\sum_{j=1}^{n-1}\gamma^{j-1}r_{t+j} + \gamma^n V_k(s_{t+n})] - V_k(s_t)$$
$$= (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}[\sum_{j=1}^{n-1}\gamma^{j-1}r_{t+j}] + (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}\gamma^n V_k(s_{t+n}) - V_k(s_t)$$

or

$$\Delta_t^\lambda = \Delta_r^\lambda + \Delta_V^\lambda$$

The first part of this expression:

$$\Delta_r^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}[\sum_{j=1}^{n-1}\gamma^{j-1}r_{t+j}]$$

is a simply a weighted average of the discounted actual rewards at each step and is not of any great interest in terms of temporal difference learning.

The second term:

$$\Delta_V^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}\gamma^n V_k(s_{t+n}) - V_k(s_t)$$

is the weighted average of the value function at time t+n compared to the current estimate at time t and this is the central part of the philosophy of TD(λ).

This term can be expressed as a weighted average of the difference between estimates at time t and t+1:

$$\Delta_V^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}\gamma^n V_k(s_{t+n}) - V_k(s_t)$$

$$= \sum_{n=1}^{\infty}\lambda^{n-1}\gamma^n(1-\lambda)V_k(s_{t+n}) - V_k(s_t)$$

$$= \sum_{n=1}^{\infty}\lambda^{n-1}\gamma^n V_k(s_{t+n}) - \sum_{n=1}^{\infty}\lambda^{n-1}\gamma^n\lambda V_k(s_{t+n}) - V_k(s_t)$$

$$= \sum_{n=1}^{\infty}\lambda^{n-1}\gamma^n V_k(s_{t+n}) - \sum_{n=1}^{\infty}\lambda^{n}\gamma^n V_k(s_{t+n}) - V_k(s_t)$$

$$= \sum_{n=1}^{\infty}\lambda^{n-1}\gamma^n V_k(s_{t+n}) - \sum_{n=0}^{\infty}\lambda^{n}\gamma^n V_k(s_{t+n}) + \lambda^0\gamma^0 V_k(s_{t+0}) - V_k(s_t)$$

$$= \sum_{n=1}^{\infty}\lambda^{n-1}\gamma^n V_k(s_{t+n}) - \sum_{n=1}^{\infty}\lambda^{n-1}\gamma^{n-1} V_k(s_{t+n-1}) + V_k(s_t) - V_k(s_t)$$

$$= \sum_{n=1}^{\infty}\lambda^{n-1}\gamma^{n-1}[\gamma V_k(s_{t+n}) - V_k(s_{t+n-1})]$$

In other words:

$$\Delta_V^\lambda = \sum_{n=1}^{\infty}\lambda^{n-1}\gamma^{n-1}[\gamma V_k(s_{t+n}) - V_k(s_{t+n-1})]$$

and as γ and λ are arbitrary constants less than one we can write $\beta=\lambda\gamma$

$$\Delta_V^\lambda = \sum_{n=1}^{\infty}\beta^{n-1}[\gamma V_k(s_{t+n}) - V_k(s_{t+n-1})]$$

$$\sum_{n=0}^{\infty}\beta^{n}[\gamma V_k(s_{t+n+1}) - V_k(s_{t+n})]$$

$$\sum_{n=0}^{\infty}\beta^{n}\nabla^{\gamma} V_k(s_{t+n})$$

where

$$\nabla^{\gamma} V_k(s_{t+n}) = \gamma V_k(s_{t+n+1}) - V_k(s_{t+n})$$

which can be considered to be a discounted first difference operator.

If γ=1 and discounting isn't used then:

$$\Delta_V^{\lambda} = \sum_{n=0}^{\infty} \lambda^n [V_k(s_{t+n+1}) - V_k(s_{t+n})]$$

$$= \sum_{n=0}^{\infty} \lambda^n \nabla V_k(s_{t+n})$$

Forward and difference views of simple TD(λ)

As before the form of TD(λ) is much simpler if there are no immediate rewards and no discounting. Although this can be derived from the above expression for TD(λ) using difference operators Sutton (1988) presents a simpler derivation of the difference form of TD(λ) which is worth repeating for the insight it provides.

The standard or forward form of TD(λ) with immediate rewards or discounting is:

$$V_{N(s)+1}(s_t) = V_{N(s)}(s_t)(1-\alpha) + \alpha(1-\lambda)\sum_{n=1}^{T-t-1} \lambda^{n-1} V_{N(s)}(s_{t+n}) + \alpha\lambda^{T-t-1} r_T$$

This can be written in incremental form as:

$$V_{N(s)+1}(s_t) = V_{N(s)}(s_t) + \alpha\left[(1-\lambda)\sum_{n=1}^{T-t-1} \lambda^{n-1} V_{N(s)}(s_{t+n}) + \lambda^{T-t-1} r_T - V_{N(s)}(s_t)\right]$$

The error correction term in the bracket can be expanded as:

$$(1-\lambda)\sum_{n=1}^{T-t-1} \lambda^{n-1} V_{N(s)}(s_{t+n}) + \lambda^{T-t-1} r_T - V_{N(s)}(s_t)$$

$$= \sum_{n=1}^{T-t-1} \lambda^{n-1} V_{N(s)}(s_{t+n}) - \lambda \sum_{n=1}^{T-t-1} \lambda^{n-1} V_{N(s)}(s_{t+n}) + \lambda^{T-t-1} r_T - V_{N(s)}(s_t)$$

The second sum can be converted into a form that allows it to be combined with the first:

$$\lambda \sum_{n=1}^{T-t-1} \lambda^{n-1} V_{N(s)}(s_{t+n}) = \sum_{n=1}^{T-t-1} \lambda^{n} V_{N(s)}(s_{t+n}) = \sum_{n=0}^{T-t-1} \lambda^{n} V_{N(s)}(s_{t+n}) - \lambda^0 V_{N(s)}(s_{t+0})$$

$$= \sum_{n=1}^{T-t-1} \lambda^{n-1} V_{N(s)}(s_{t+n-1}) - \lambda^0 V_{N(s)}(s_{t+0}) + \lambda^{T-t-1} V_{N(s)}(s_{t+T-t-1})$$

Putting this back in the expression and combining summations gives:

$$(1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}V_{N(s)}(s_{t+n})+\lambda^{T-t-1}r_T-V_{N(s)}(s_t)$$

$$=\sum_{n=1}^{T-t-1}\lambda^{n-1}V_{N(s)}(s_{t+n})-\sum_{n=1}^{T-t-1}\lambda^{n-1}V_{N(s)}(s_{t+n-1})+\lambda^0V_{N(s)}(s_{t+0})-\lambda^{T-t-1}V_{N(s)}(s_{t+T-t-1})+\lambda^{T-t-1}r_T-V_{N(s)}(s_t)$$

$$=\sum_{n=1}^{T-t-1}\lambda^{n-1}[V_{N(s)}(s_{t+n})-V_{N(s)}(s_{t+n-1})]+\lambda^{T-t-1}[r_T-V_{N(s)}(s_{T-1})]$$

Finally if we write $V_{N(s)}(s_T)=r_T$ the entire expression can be simplified to a pure sum of differences:

$$\sum_{n=1}^{T-t-1}\lambda^{n-1}[V_{N(s)}(s_{t+n})-V_{N(s)}(s_{t+n-1})]+\lambda^{T-t-1}[r_T-V_{N(s)}(s_{T-1})]$$

$$=\sum_{n=1}^{T-t}\lambda^{n-1}[V_{N(s)}(s_{t+n})-V_{N(s)}(s_{t+n-1})]$$

and the full TD(λ) estimator can be written:

$$V_{N(s)+1}(s_t)=V_{N(s)}(s_t)+\alpha\left[\sum_{n=1}^{T-t}\lambda^{n-1}[V_{N(s)}(s_{t+n})-V_{N(s)}(s_{t+n-1})]\right]$$

This special case reveals most clearly the reason why the method is called Temporal *Difference*. As can be seen the estimator can equally well be thought of as correcting the current estimate of a states value by a weighted sum of the differences between estimates at subsequent states. It is this view that also motivates the temporal difference philosophy.

However it is obvious that the two viewpoints, that is TD(λ) as a weighted average of estimates or as a weighted sum of differences, are identical and equally valid.

Appendix II

Proofs of Asymptotic Convergence of Matrix Estimators

TD(λ) can be written as a matrix estimator since the new estimate is a linear combination of the current vector of estimates and the reward. The coefficients of the linear sums are simply the appropriate lambda weights as described elsewhere in this thesis. That is, for any TD(λ) estimator the update that results at time t can be written in the form:

$$\mathbf{v}_{t+1} = (I - \alpha \mathrm{Diag}[\chi])\mathbf{v}_t + \alpha(M\mathbf{v}_t + \mathbf{n}r_t)$$

where M is the matrix of weights for the non-reward based terms, **n** is the vector of weights for the reward based term, χ is a vector of indicator functions for the event that state i occurred in the realisation, Diag[**x**] is the diagonal matrix with the elements of **x** along the diagonal and r_t is the (scalar) reward actually obtained. In most cases M and **n** are stochastic and based on the realisation obtained at time t, but this doesn't have to be the case and indeed the standard MCA estimator has M=0 and $\mathbf{n}=\chi$.

$$\begin{aligned}\mathbf{v}_{t+1} &= (I - \alpha \mathrm{Diag}[\chi])\mathbf{v}_t + \alpha(0\mathbf{v}_t + \chi r_t) \\ &= (I - \alpha \mathrm{Diag}[\chi])\mathbf{v}_t + \alpha\chi r_t\end{aligned}$$

If state i doesn't occur in the realisation then the by definition the ith row of M and $[\mathbf{n}]_i$ are defined to zero, as is $[\chi]_i$, and so effectively no update is performed for that stats current estimate. We also add the condition that:

$$M\mathbf{l} + \mathbf{n} = \chi$$

This simply means that when the state occurs M and **n** form a true weighted average of the existing estimates and the reward realised. Not all TD estimators satisfy this condition but they can all be adjusted so that it holds.

For example, for a First visit estimator M and **n** are:

$$[M]_{ij} = (1-\lambda)\sum_{d=1}^{\kappa_j(t)} \lambda^{n_j(t,d)-n_i(t;1)-1}$$

$$[\mathbf{n}]_i = \lambda^{\tau(t)-n_i(t;1)}$$

if state i occurred in the realisation where $\tau(t)$ is the number of states in the realisation t and $n_i(t;d)$ is the position of the dth concurrence of state i.

If the state didn't occur we take the corresponding row of M and element of **n** to be zero. The notation may be difficult but the intention is clear. The i,jth element of M consists of the λ weights applied to state j in the calculation of the state i's update.

For the corrected on-line Every visit or Accumulate estimator M and **n** are:

$$[M]_{ij} = \frac{1}{\kappa_i(t)} \sum_{d_i=1}^{\kappa_i(t)} (1-\lambda) \sum_{d_j=1}^{\kappa_j(t)} \lambda^{n_j(t,d_j)-n_i(t;d_i)-1}$$

$$[\mathbf{n}]_i = \frac{1}{\kappa_i(t)} \sum_{d_i=1}^{\kappa_i(t)} \lambda^{\tau(t)-n_i(t;d_i)}$$

if state i occurred in the realisation where $\kappa_i(t)$ is the number of times state i occurs in the realisation at time t.

Again if state i doesn't occur we can take the corresponding row of M and element of **n** to be zero.

Proof that MCA converges in the mean

Although the convergence of the MCA estimator follows from the more general proof of the convergence of the matrix estimator described above, it is instructive to see how the two proofs differ.

To make things simpler but no less general it is assumed that the system is started from state i. The update at each realisation is:

$$\mathbf{v}_{t+1} = (I - \alpha \mathrm{Diag}[\chi])\mathbf{v}_t + \alpha \chi \mathbf{r}_t$$

Taking expectations:

$$\begin{aligned} E[\mathbf{v}_{t+1}] &= E[(I - \alpha \mathrm{Diag}[\chi])\mathbf{v}_t] + \alpha E[\chi \mathbf{r}_t] \\ &= E[(I - \alpha \mathrm{Diag}[\chi])E[\mathbf{v}_t] + \alpha E[\chi \mathbf{r}_t] \\ &= (I - \alpha F)E[\mathbf{v}_t] + \alpha E[\chi \mathbf{r}_t] \end{aligned}$$

where $[F]_{jj}$ is a diagonal matrix of the probabilities that state j will occur when the system is started from state i.

The ith component of the final term is:

$$\begin{aligned} E[\chi \mathbf{r}_t]_i &= P[\chi_i = 0]E[\chi_i \mathbf{r}_t \mid \chi_i = 0] + P[\chi_i = 1]E[\chi_i \mathbf{r}_t \mid \chi_i = 1] \\ &= P[\chi_i = 1]E[\mathbf{r}_t \mid \chi_i = 1] \\ &= [F]_{ii} E[\mathbf{r}]_i \end{aligned}$$

where **r** is the vector of expected rewards for each state i.e. the value function.

Writing this in matrix form gives:

$$E[\chi \mathbf{r}_t] = FE[\mathbf{r}]$$

$$E[\mathbf{v}_{t+1}] = (I - \alpha F)E[\mathbf{v}_t] + \alpha FE[\mathbf{r}]$$

Defining

$$Z = (I - \alpha F)$$
$$\mathbf{c} = \alpha FE[\mathbf{r}]$$

the iteration is of the form:

$$E[\mathbf{v}_{t+1}] = ZE[\mathbf{v}_t] + \mathbf{c}$$

The solution to this first order matrix recursion, using standard methods, is:

$$E[\mathbf{v}_t] = Z^t E[\mathbf{v}_0] + \sum_{k=0}^{t-1} Z^k \mathbf{c}$$

Taking the limit $t \to \infty$ gives

$$\lim_{t\to\infty}[E[\mathbf{v}_t]] = \lim_{t\to\infty}[Z^t E[\mathbf{v}_0]] + \lim_{t\to\infty}[\alpha \sum_{k=0}^{t-1} Z^k \mathbf{c}]$$

Using: $\sum_{k=0}^{\infty} P^k = (I - P)^{-1}$ which applies if the series converges. i.e. if $P^k \to 0$ or equivalently if $\|P\| < 1$ using a suitable matrix norm (Householder, 1964; Bellman, 1960).

The absolute row maximum norm is defined as:

$$\|X\| = \max(\sum_j |x_{ij}|)$$

As $\|Z\| = \|(I - \alpha F)\| \le \|I\| - \alpha\|F\| \le 1 - \alpha$ we have $0 < \alpha < 1 \Rightarrow \|Z\| < 1$. Thus:

$$\lim_{t\to\infty}[Z^t E[\mathbf{v}_0]] = 0 \text{ and}$$

$$\lim_{t\to\infty}[\alpha \sum_{k=0}^{t-1} Z^k \mathbf{c}] = \alpha(I - Z)^{-1}\mathbf{c}$$

$$\lim_{t\to\infty}[E[\mathbf{v}_t]] = \alpha(I - Z)^{-1}\mathbf{c} = (I - I + \alpha F)^{-1}\alpha FE[\mathbf{r}]$$

$$= (\alpha F)^{-1}\alpha F\, E[\mathbf{r}]$$

$$= E[\mathbf{r}]$$

Hence as long as $0 < \alpha < 1$, $\lim_{t\to\infty}[E[\mathbf{v}_t]] = E[\mathbf{r}]$ and the MCA estimator is asymptotically unbiased.

Proof that matrix TD-like estimators do not diverge

In general the estimator is

$$\mathbf{v}_{t+1} = (\mathrm{I} - \alpha(\mathrm{Diag}[\chi] - \mathrm{M})\mathbf{v}_t + \alpha \mathbf{n} r_t$$

Taking the norm (again using the maximum row sum norm) of both sides:

$$\begin{aligned} \|\mathbf{v}_{t+1}\| &= \|(\mathrm{I} - \alpha\mathrm{Diag}[\chi] + \alpha\mathrm{M})\mathbf{v}_t + \alpha \mathbf{n} r_t\| \\ &\leq \|(\mathrm{I} - \alpha\mathrm{Diag}[\chi] + \alpha\mathrm{M})\|\|\mathbf{v}_t\| + \alpha\|\mathbf{n}\|r_t| \end{aligned}$$

The first term can be simplified as:

$$\|(\mathrm{I} - \alpha(\mathrm{Diag}[\chi] - \mathrm{M})\|\|\mathbf{v}_t\| \leq (1 - \alpha\|(\mathrm{Diag}[\chi] - \mathrm{M})\|)\|\mathbf{v}_t\| \leq (1 - \alpha)\|\mathbf{v}_t\|$$

using the fact that

$$0 \leq \alpha \leq 1,\ \|\mathrm{I}\| = 1 \text{ and } \|\mathrm{Diag}[\chi] - \mathrm{M}\| \leq \|\mathrm{Diag}[\chi]\| - \|\mathrm{M}\| = 1 - \|\mathrm{M}\| \leq 1 \text{ as } \|\mathrm{M}\| \geq 0.$$

The second term gives:

$$|r_t|\|\mathbf{n}\| \leq |r_t|\|\mathbf{n} + \mathrm{M}\mathbf{l}\| = |r_t|\|\chi\| \leq |r_t|$$

as $\|\mathrm{M}\mathbf{l}\| \geq 0,\ \mathbf{n} + \mathrm{M}\mathbf{l} = \chi$ and $\|\chi\| \leq 1$

Thus:

$$\|\mathbf{v}_{t+1}\| \leq \|\mathbf{v}_t\|(1 - \alpha) + \alpha|r_t| \Rightarrow \|\mathbf{v}_{t+1}\| \leq \mathrm{Max}(\|\mathbf{v}_t\|, |r_t|)$$

and:

$$\|\mathbf{v}_t\| \leq \mathrm{Max}(\|\mathbf{v}_0\|, |r_0|)$$

So if the initial estimates and the rewards are bounded so is the resulting estimate for all t and the iteration does not diverge as long as $0 \leq \alpha \leq 1$ and $\mathbf{n} + \mathrm{M}\mathbf{l} = \chi$.

Proof that matrix TD-like estimators converge in the mean

Taking the expected value of the update equation gives:

$$\begin{aligned} \mathrm{E}[\mathbf{v}_{t+1}] &= \mathrm{E}[(\mathrm{I} - \alpha\mathrm{Diag}[\chi])\mathbf{v}_t] + \alpha\mathrm{E}[\mathrm{M}\mathbf{v}_t] + \alpha\mathrm{E}[\mathbf{n}r_t] \\ &= (\mathrm{I} - \alpha\mathrm{F})\mathrm{E}[\mathbf{v}_t] + \alpha\mathrm{E}[\mathrm{M}\mathbf{v}_t] + \alpha\mathrm{E}[\mathbf{n}r_t] \end{aligned}$$

The second term is:

$$\begin{aligned}E[M\mathbf{v}_t]_j &= P[\chi_j = 0]E[M\mathbf{v}_t \mid \chi_j = 0]_j + P[\chi_j = 1]E[M\mathbf{v}_t \mid \chi_j = 1]_j \\ &= P[\chi_j = 1]E[M\mathbf{v}_t \mid \chi_j = 1]_j \\ &= [F]_j E[M']_{\bullet j} E[\mathbf{v}_t]_j \\ E[M\mathbf{v}_t] &= FE[M']E[\mathbf{v}_t]\end{aligned}$$

where

$$[M']_{ij} = E[[M]_{ij} \mid \chi_j = 1] \text{ and } E[M']_{\bullet j} \text{ is the jth row of M.}$$

This follows from the independence of $\text{Diag}[\chi], M$ and $E[\mathbf{v}_t]$ as both happen after $\mathbf{v}_t$ has occurred.

The third term is:

$$\begin{aligned}E[[\mathbf{nr}_t]]_j &= P[\chi_j = 0]E[[\mathbf{nr}_t]_j \mid \chi_j = 0] + P[\chi_j = 1]E[[\mathbf{nr}_t]_j \mid \chi_j = 1] \\ &= P[\chi_j = 1]E[[\mathbf{nr}_t]_j \mid \chi_j = 1]\end{aligned}$$

As $M\mathbf{l} + \mathbf{n} = \chi$ can be written $\chi - M\mathbf{l} = (\text{Diag}[\chi] - M)\mathbf{l} = \mathbf{n}$ we also have:

$$\begin{aligned}P[\chi_j = 1]E[[\mathbf{nr}_t]_j \mid \chi_j = 1] &= [F]_{jj} E[[(I - M)\mathbf{l}r_t)]_j \mid \chi_j = 1] \\ &= [F]_{jj} E[[(\mathbf{l}r_t - M\mathbf{l}r_t)]_j \mid \chi_j = 1] \\ &= [F]_{jj} E[[\mathbf{l}\mathbf{l}r_t]_j \mid \chi_j = 1] - [F]_{jj} E[[M\mathbf{l}r_t]_j \mid \chi_j = 1] \\ &= [F]_{jj} E[r_j] - [F]_{jj} E[[M]_{j\bullet}]E[\mathbf{l}r_t]_j \mid \chi_j = 1] \\ &= [F]_{jj} E[r_j] + [F]_{jj} E[[M]_{j\bullet}]E[\mathbf{l}r_t]_j \mid \chi_j = 1] \\ &= [F]_{jj} (1 - E[[M]_{j\bullet}] \mid \chi_j = 1])E[\mathbf{r}]_j\end{aligned}$$

Where is has been assumed that $E[[M\mathbf{l}r_t]_j \mid \chi_j = 1] = E[[M\mathbf{l}]_j \mid \chi_j = 1]E[r_t]_j \mid \chi_j = 1]$

i.e. that M is conditionally independent of the reward.

This is generally true of Markov processes because, conditional on the state occurring at least once, then the reward is independent of interarrival times, and number of occurrences of a state due to the Markov property.

Writing this in matrix form:

$$E[\mathbf{nr}_t] = F(I - E[M'])E[\mathbf{r}]$$

Thus:

$$\begin{aligned}E[\mathbf{v}_{t+1}] &= ((I - \alpha F) + \alpha FE[M'])E[\mathbf{v}_t] + \alpha F(I - E[M'])E[\mathbf{r}] \\ &= ZE[\mathbf{v}_t] + \mathbf{c}\end{aligned}$$

where:

$$Z = (I - \alpha F) + \alpha F E[M']$$
$$c = \alpha F(I - E[M']E[\mathbf{r}]$$

The solution to this first order matrix recursion, using standard methods, is:

$$E[\mathbf{v}_t] = Z^t E[\mathbf{v}_0] + \sum_{k=0}^{t-1} Z^k \mathbf{c}$$

Taking the limit $t \to \infty$ gives

$$\lim_{t\to\infty}[E[\mathbf{v}_t]] = \lim_{t\to\infty}[Z^t E[\mathbf{v}_0]] + \lim_{t\to\infty}[\sum_{k=0}^{t-1} Z^k \mathbf{c}]$$

Using:

$$\sum_{k=0}^{\infty} P^k = (I - P)^{-1}$$

which applies if the series converges, i.e. if $P^k \to 0$ as $k \to \infty$ or $\|P\| < 1$.

$$\begin{aligned}\|Z\| &\le \|(I - \alpha F)\| + \alpha\|E[M']\| \\ &\le 1 - \alpha + \alpha\|E[M']\| \\ &\le 1 - \alpha(1 - \|E[M']\|)\end{aligned}$$

As $M\mathbf{l} + \mathbf{n}' = I$ and the elements of n are positive we can conclude that the row sums of E[M] are less than one. As $0<\alpha<1$ the row sums of Z are also less than 1.

Hence:

$$\|Z\| < 1 \Leftarrow Z^k \to 0 \text{ as } k \to \infty$$

and $$\lim_{t\to\infty}[\sum_{k=0}^{t-1} Z^k \mathbf{c}] = (I - Z)^{-1}\mathbf{c}$$

This allows us to write the limit as:

$$\begin{aligned}\lim_{t\to\infty}[E[\mathbf{v}_t]] &= \lim_{t\to\infty}[Z^t E[\mathbf{v}_0]] + (I - Z)^{-1}\mathbf{c} \\ &= (I - Z)^{-1}\mathbf{c}\end{aligned}$$

Expanding Z and **c**:

$$
\begin{aligned}
(I-Z)^{-1}\mathbf{c} &= (I-(I-\alpha F)-\alpha FE[M'])^{-1}\mathbf{c} \\
&= (\alpha F-\alpha FE[M'])^{-1}\mathbf{c} \\
&= (I-E[M'])^{-1}(\alpha F)^{-1}\mathbf{c} \\
&= (I-E[M'])^{-1}(\alpha F)^{-1}\alpha F(I-E[M'])E[\mathbf{r}] \\
&= E[\mathbf{r}]
\end{aligned}
$$

and hence:

$$\lim_{t\to\infty}[E[\mathbf{v}_t]] = E[\mathbf{r}]$$

and the ith component of the estimate converges in the mean to the expected reward starting from state i for all α ($0<\alpha<1$).

This result applies to any TD estimator, or indeed any estimator which forms a true weighted average of the non-reward and reward terms and so applies to First visit TD and the corrected form of the off-line Every visit TD.

In the more general case where $M\mathbf{l}+\mathbf{n}=\mathbf{k}$ where $\|\mathbf{k}\|>1$ it is clear that the proof follows the same steps but now we need a condition similar to $\alpha\|\mathbf{k}\|<1$ to ensure convergence for at least a limited range of α.

Symbol Definitions

α	The step size parameter in the alpha update rule $(0 \le \alpha \le 1)$
γ	The discounting factor $(0 < \gamma \le 1)$
$\kappa_i(t)$	Number of times state i occurs in the realisation t
λ	The temporal difference weighting factor $(0 \le \lambda \le 1)$
$\pi(s) \rightarrow a \quad s \in S \quad and \quad a \in A$	The deterministic policy which maps state *s* to action *a*
$\pi(s,a) \rightarrow [0,1] \quad s \in S \quad and \quad a \in A$	The stochastic policy which maps state *s* to the probability of selecting action *a*
$\pi^* = \arg\max_{\pi} E(\sum_{t=0}^{\infty} \gamma^t r_t)$	The optimal policy starting from state *s*
$\tau(t)$	The number of steps in realisation t
χ	A vector of indicator functions for the event that state i occurred in the realisation
A	Set of actions
$[A]_{ij}$	The i,j^{th} element of the matrix A, i.e. $[A]_{ij}=a_{ij}$
C	Concordance coefficient – proportion that an estimated value function agrees with the decisions made by a population value function
$E(\sum_{t=0}^{\infty} \gamma^t r_t)$	The expected discounted reward
F	A diagonal matrix of the probabilities $[F]_{jj}$ that state j will occur when the system is started from state i
$i = \arg\max_{k}(f(k))$	Returns the value of the parameter *k* that maximises *f*, i.e. max(f(k))=f(i)
$K_i(t;n)$	Indicator function for state i occurring at position n in realisation t
1	A vector of ones
$n_i(t;d)$	The position in the realisation at which the d^{th} occurrence of state i occurs
$N_k(s_j)$ $\kappa_i(t)$	The number of visits to state s_j in *k* samples.
p^*_1	Probability of absorption at right given the current state is s_i

p_{ij}	Markov transition probabilities for state *i* to state *j*
$P^* = QT$	The matrix of absorption probabilities. i.e. the probabilities of being absorbed in each of the terminal states
$P = \begin{bmatrix} S & T \\ 0 & I \end{bmatrix}$	The transition matrix of the entire absorbing Markov chain, where S is the transition matrix of the non-absorbing states and T is the transition matrix from the non-absorbing to the absorbing states
$P(s,s') = \sum_a \pi(s,a)T(s,a,s')$	The transition probabilities on the Markov chain induced by following the policy $\pi(s,a)$ on the MDP with state action. That is the probability that state *s'* will follow state *s* while following policy π.
$Q^*(s,a)$	The optimal action value function. That is the value of taking action *a* and then following the optimal policy
$Q^\pi(s,a) = E_\pi(\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a)$	The action value function, i.e. the expected discounted reward given that action *a* is taken in state *s* and policy π is followed from this point on
$Q = \sum_{k=1}^{\infty} S^k = (I-S)^{-1}$	The fundamental matrix of the absorbing Markov chain
R	Set of reinforcement signals
r_t	Reward received at time t
r_T or r	Reward received at time T – the terminal reward.
R(*s,a*)	Reward received on taking action a in state s
R(s,a,s')	Reward received on taking action a in state s which results in state *s'*
$R_t = r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots + \gamma^{T-1} r_T$	The discounted reward on the t^{th} sample
$R_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)}$	The temporal difference weighted sum of the *n*-step truncated rewards
$R_t^{(n)} = \sum_{j=1}^{n-1} \gamma^{j-1} r_{t+j} + \gamma^n V_k(s_{t+n})$	The corrected *n*-step truncated return
S	Set of states
T(*s,a*) also written T(*s,a,s'*)	State action transition function which gives a probability distribution of each state *s'* being the result of *s,a*, i.e. Π(*s'*) is the probability of *s'* being the next state after action *a* in state *s*.
$V_k(s_j)$	The estimate of the value of state s_j after *k* samples

$V^{\pi}(s) = \underset{\pi}{E}(\sum_{t=0}^{\infty} \gamma^t r_t)$	The value of a state *s* while following policy π, i.e. the expected discounted reward as a result of following policy π after being in state *s*
$V^{*}(s) = \max_{\pi} E(\sum_{t=0}^{\infty} \gamma^t r_t)$	The optimal value, i.e. the value of the state following the optimal policy
$V_k(s_j) = V_{k-1}(s_j)(1-\alpha_k) + R_k\alpha_k$	The alpha update rule for the new estimate of the value function given the discounted reward R_k received on the k^{th} trial.
$\tilde{v}(s)$	The population average of a value function estimate.

References

Barnard, E. (1993). Temporal-difference methods and Markov models. *IEEE Transactions on Systems, Man, and Cybernetics,23.*

Barto, A. G. and Sutton, R. S. (1981a). Goal seeking components for adaptive intelligence: An initial assessment. *Technical Report AFWAL-TR-81-1070*, Air Force Wright Aeronautical Laboratories/Avionics Laboratory, Wright-Patterson.

Barto, A. G. and Sutton, R. S. (1981b). Landmark learning: An illustration of associative search. *Biological Cybernetics,42*,1-8.

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics, 13*, 835- 846. Reprinted in Anderson, J. A. and Rosenfeld, E. (1988). *Neurocomputing: Foundations of Research*, MIT Press.

Baxter, J., Tridgell, A., and Weaver, L. (1997). KnightCap: A chess program that learns by combining TD(λ) with minimax search. *TechnicalReport, Learning Systems Group, Australian National University.*

Baxter, J., Tridgell, A., and Weaver, L. (1998). TDLeaf(lambda): Combining Temporal Difference Learning with Game-Tree Search. *Australian Journal of Intelligent Information Processing,* 39-43.

Baxter, J., Tridgell, A., and Weaver, L. (2000). Learning to play Chess using temporal differences. *Machine Learning,40*, 243-263.

Beal, D.F. and Smith, M.C. (1997) Learning piece values using temporal differences. *Journal of The International Computer Chess Association*, 147-151.

Beal, D.F. and Smith, M.C. (2000) Temporal difference learning for heuristic search and game playing. *Information Sciences, 122*, 3-21.

Beal, D.F. and Smith, M.C. (2001) Temporal difference learning applied to game playing and the results of application to Shogi. *Theoretical Computer Science, 252*, 105-119.

Bellman, R. (1957). *Dynamic Programming.* Princeton University Press.

Bellman, R. (1960). *Introduction to Matrix Analysis.* McGraw Hill.

Bertsekas, D.P. (1987) *Dynamic Programming: Deterministic and Stochastic Models.* Prentice-Hall.

Bertsekas, D.P., and Tsitsiklis, J. (1996). *Neuro-Dynamic Programming.* Athena Scientific.

Billingsley, P. (1961a). *Statistical Inference for Markov Processes.* University of Chicago Press.

Billingsley, P. (1961b). Statistical Methods in Markov Chains. *Annals of Mathematical Statistics, Vol 32.*

Birkhoff, G., and MacLane S. (1953). *A Survey of Modern Algebra.* Macmillan.

Bouzy, B. and Cazenave, T. (2001). Computer Go: An AI oriented survey. *Artificial Intelligence,13*, 39-103.

Boyan, J.A. (1999). Least-squares temporal difference learning, *Proceedings of the 16th International Conference on Machine Learning*. Morgan Kaufmann, 49-56.

Bradtke, S.J. and Barto, A.G. (1996). Linear least-squares algorithms for temporal difference learning. *Machine Learning, 22* ,33-57.

Chan, H.W., King, I. and Lui, J.C.S, (1996). Performance Analysis of a New Updating Rule for TD(λ) Learning in Feedforward Networks for Position Evaluation in Go. *Proceedings of the IEEE International Conference on Neural Networks, vol.3*, 1716-1720.

Chung, K.L. (1974). *Elementary Probability Theory with Stochastic Processes.* Springer-Verlag, New York.

Cichosz, P. and Mulawka, J.J. (1995). Fast and efficient reinforcement learning with truncated temporal differences. *International Conference on Machine Learning*, 99-107.

Cox, D.R. and Hinkley, D.V. (1974). *Theoretical Statistics*. Chapman and Hall.

Darwen, P.J. (2001). Why co-evolution beats temporal difference learning at backgammon for a linear architecture, but not a non-linear architecture. *Proceedings of the 2001 Congress on Evolutionary Computation (CEC 2001)*, IEEE Press, 1003-1010.

Dayan, P. (1992). The convergence of TD(λ) for general λ. *Machine Learning,8.*

Dayan, P. (1993). Improving generalisation for temporal difference learning: The successor representation. *Neural Computation,5*, 13-624.

Dayan, P. (2001). Reinforcement learning in Gallistel, C.R., *Steven's Handbook of Experimental Psychology,* Wiley.

Dayan, P. (2004), Private communication.

Dayan, P. and Abbott, L.F. (2001). *Theoretical Neuroscience.* MIT Press.

Dayan, P., Schraudolph, N.N. and Sejnowski, T.J. (2001). Learning to evaluate Go positions via temporal difference methods. *Computational Intelligence in Games*, 74-96. Springer Verlag.

Dayan, P. and Watkins, C.J.C.H. (2001). Reinforcement learning. *Encyclopedia of Cognitive Science*, MacMillan Press.

Dayan, P. and Sejnowski, T. (1994). TD(λ) converges with probability 1. *Machine Learning,14.*

Ekker, R., van der Werf, E.C.D. and Schomaker, L.R.B. (2004) Dedicated TD-learning for stronger gameplay: applications to Go. *Benelearn'04: Proceedings of the Thirteenth Belgian-Dutch Conference on Machine Learning.*

Enzenberger, M. (2003). Evaluation in Go by a Neural Network Using Soft Segmentation. *Proceedings of the 10th Advances in Computer Games Conference (Graz)*, University of Alberta.

Frobenius, G. (1912). Über matrizen aus positiven elementen, *S-B. Preuss. Akademic Wissenshaft*, 456-477.

Ghory, I. (2004). Reinforcement learning in board games. *Technical Report CSTR-04-004*, Department of Computer Science, University of Bristol.

Goldberg, S. (1958). *Introduction to Difference Equations*. John Wiley & Sons.

Goodwin, G.C. and Sin, K.S. (1984). *Adaptive Filtering Prediction and Control.*

Gosavi, A., Bandla, N. and Das, T.K. (2002). A Reinforcement Learning Approach to Airline Seat Allocation for Multiple Fare Classes with Overbooking, *IIE Transactions*, *34*, 729-742.

Grigoriadis, A. and Paliouras, G. (2004). Focused Crawling using Temporal Difference-Learning. *Proceedings of the Panhellenic Conference in Artificial Intelligence (SETN), Lecture Notes in Artificial Intelligence*. Springer Verlag,

Gu, D and Hu, H. (2002). Reinforcement learning of fuzzy logic controller for quadruped walking robots, *Proceedings of 15th IFAC World Congress*, Barcelona.

Guo, C. and Kuh, A. (1997). Temporal difference learning applied to sequential detection, *IEEE Trans. on Neural Networks Vol 8 No 2.*

Gurvits, L., Lin, L.J., and Hanson, S. J. (1994). Incremental learning of evaluation functions for absorbing Markov chains: New methods and theorems.

Hansen, E.A. and Cohen, P.R. (1992).Learning a decision rule for monitoring tasks with deadlines. *UM-CS Technical Report.*

Haykin, S. (1994). *Neural Networks: a comprehensive foundation.* Macmillan Publishing Company.

Hinton, G.E., McClelland, J.L., and Rumelhart, D.E. (1986). Distributed representations, in Rumelhart, D.E., McClelland, J.L. and the PDP Research Group, *Parallel distributed processing: Explorations in the microstructure of cognition. Vol 1: Foundations*. MIT Press/Bradford Books.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.

Holland, J. H. (1976). Adaptation. In Rosen, R. and Snell, F. M., editors, *Progress in Theoretical Biology,4*, 263-293. Academic Press, NY.

Householder, A.S. (1964). *The Theory of Matrices in Numerical Analysis*, Dover.

Howard, R. (1960). *Dynamic Programming and Markov Processes*. MIT Press.

IBM AlphaWorks (2004) Agent Building and Learning Environment http://www.alphaworks.ibm.com/tech/able

Jaakkola, T., Jordan, M. I., and Singh, S. P. (1994). On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation,6*.

Kaelbling, L.P, Littman M. L. and Moore A.P. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research,4*, 237-285.

Kearns, M. and Singh, S. (1999). Finite sample convergence rates for Q-learning and indirect algorithms. *Neural Information Processing Systems, 12*, MIT Press.

Kearns, M. and Singh, S. (2000). Bias-Variance Error Bounds for Temporal Difference Updates. *Proceedings of the 13th Annual Conference on Computer Learning Theory*, 142-147, Morgan Kaufmann.

Keerthi, S. and Ravindran, B. (1995). A tutorial survey of reinforcement learning, *Sadhana*, Indian Academy of Sciences.

Kemeny, J.G. and Snell, L. (1983). *Finite Markov chains*. Springer-Verlag.

Kendall, M.G. and Stuart, A. (1973). *The Advanced Theory of Statistics, Volume 2 Inference and Relationship, Third Edition*, Charles Griffin and Co.

Klopf, A. H. (1972). Brain function and adaptive systems - A heterostatic theory. *Technical Report AFCRL-72-0164*, Air Force Cambridge Research Laboratories, Bedford, MA.

Lagoudakis, M.G. and Parr, R. (2002). Learning in zero-sum team Markov games using factored value functions. *Proceedings of NIPS*2002: Neural Information Processing Systems: Natural and Synthetic*, Vancouver. 1659-1666.

Littman, M.L. (1994). Markov Games as a Framework for Multi-Agent Reinforcement Learning. *Proceedings of the 11th International Conference on Machine Learning*, 157-163, Morgan Kaufmann.

Mannen, H. and Wiering, M. (2004). Learning to play chess using TD(λ)-learning with database games. *Benelearn'04: Proceedings of the Thirteenth Belgian-Dutch Conference on Machine Learning*.

Michie, D. and Chambers, R.A. (1968). Boxes: An experiment in adaptive control, in *Machine Intelligence,2*, Oliver and Boyd.

Moser, M.C. (1986). Rambot: A connectionist expert system that learns by example, *Technical Report 8610*, Institute for Cognitive Science, UCSD.

Myers, C. (1992). *Delay learning in artificial neural networks*, Chapman and Hall.

Nau, D.S. (1980). Pathology on game trees: A summary of results. *Proceedings of the First National Conference on Artificial Intelligence*, 102-104.

Nau, D.S. (1981). Pearl's Game is pathological. *Technical Report TR-999*, Computer Science Department, University of Maryland.

Nie, A.G, Honemann. A, et al. (2001). *The Osnabrueck RoboCup Agents Project.* Institute of Cognitive Science, Osnabrueck.

Patist, J.P. and Wiering, M. (2004), Learning to play Draughts using temporal difference learning with neural networks and databases. *Benelearn'04: Proceedings of the Thirteenth Belgian-Dutch Conference on Machine Learning.*

Peng, J. (1993). *Efficient Dynamic Programming-Based Learning for Control.* PhD thesis, Northeastern University, Boston.

Perron, O. (1907). Zur theorie der matrizen, *Mathematical Annals 64*, 248-263.

Pollack, J.B. and Blair, A.D. (1997) *Why did TD-Gammon work.* Advances in Neural Information Systems Vol 9.

Precup, D., Sutton, R.S, and Dasgupta, S. (2001). Off-policy temporal-difference learning with function approximation. *Proceedings of the 18th International Conference on Machine Learning.* Morgan Kaufmann, 417-424.

Puterman, M.L. (1994). *Markov Decision Problems.* Wiley.

Puterman, M.L. and Shin, M.C. (1978). Modified policy iteration algorithms for discounted Markov decision problems. *Management Science,21*, 11127-1137.

Ragg, T., Braunn, H. and Feulner, J. (1994). Improving Temporal Difference Learning for Deterministic Sequential Decision Problems. *Proceedings of the International Conference on Artificial Neural Networks,* 117-122. ICANN.

Rao, R. P. N.(2000) Learning to Maximise Rewards. *Neural Networks, 13,* 135-137.

Samuel, A.L. (1959). Some studies in machine learning using the game of checkers, *IBM Journal on Research and Development 3*, 210-299.

Schaeffer J., Hlynka M., and Jussila V. (2001). Temporal Difference Learning Applied to a High-Performance Game-Playing Program, *International Joint Conference on Artificial Intelligence (IJCAI)*, 529-534.

Schapire, R.E. and Warmuth, M.K. (1994), On the worst-case analysis of temporal-difference learning algorithms. *International Conference on Machine Learning*, 266-274

Schraudolph, N. N., Dayan, P. and Sejnowski, T. J. (1994). Temporal Difference Learning of Position Evaluation in the Game of Go, *Advances in Neural Information Processing,6*, Morgan Kaufmann, 817-824.

Schraudolph, N. N, Dayan, P. and Sejnowski, T. J. (2000). Learning to evaluate Go positions via temporal difference methods. In Jain, L.C. & Baba, N. *Soft Computing Techniques in Game Playing.* Springer-Verlag.

Singh, S. and Dayan, P. (1998). Analytical Mean Squared Error Curves for Temporal Difference Learning. *Machine Learning,32.*

Singh, S. and Sutton, R. (1996). Reinforcement learning with eligibility traces. *Machine Learning,22.*

Sutton, R. S. (1978a). Learning theory support for a single channel theory of the brain.

Sutton, R. S. (1978b). Single channel theory: A neuronal theory of learning. *Brain Theory Newsletter,4,* 72-75.

Sutton, R. S. (1978c). A unified theory of expectation in classical and instrumental conditioning.

Sutton, R. S. (1984). *Temporal Credit Assignment in Reinforcement Learning.* PhD thesis, University of Massachusetts.

Sutton, R.S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning 3.*

Sutton, R.S. (1997). On the Significance of Markov Decision Processes, *Proceedings of the International Conference on Artificial Neural Networks*, ICANN.

Sutton, R. S. and Barto, A. G. (1981). An adaptive network that constructs and uses an internal model of its world. *Cognition and Brain Theory*, 3:217--246.

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

Sutton, R.S, and Singh, S (1994). On bias and step size in temporal-difference learning. *Proceedings of the Eighth Yale Workshop on Adaptive and Learning Systems,* 91-96. Center for Systems Science, Dunham Laboratory, Yale University.

Szepesvári, C. and Littman, M.L. (1996). Generalized Markov Decision Processes: Dynamic-programming and reinforcement-learning algorithms, *Technical Report, Brown University, Number CS-96-11.*

Tadic, V. (2001). On the convergence of temporal difference learning with linear function approximation. *Machine Learning, 42*, 241-267.

Tesauro, G. and Sejnowski T.J. (1989). A parallel Network that learns to play backgammon. *Artificial Intelligence 39*, 357-390.

Tesauro, G. (1992). Practical Issues in Temporal Difference Learning. *Machine Learning 8,* 257-277.

Tesauro, G. (1994). TD-Gammon, a self-teaching backgammon program achieves master level play. *Neural Computation 6*, 215-219.

Thrun, S. (1995). Learning to Play the Game of Chess, in Tesauro, G., Touretzky, D. and Leen, T. *Advances in Neural Information Processing Systems*, MIT Press.

Trinh, T.B, Bashi, A. S. and Deshpande, N. (1998). Temporal Difference Learning in Chinese Chess. *Proceedings of the 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems: Tasks and Methods in Applied Artificial Intelligence*, 612-618.

Tsitsiklis, J. N. (1994). Asynchronous stochastic approximation and q-learning. *Machine Learning 16*, 185-202.

Varga, R.S. (2000). *Matrix Iterative Analysis, Second Edition*, Springer.

Watkins, C.J.C.H. (1989). *Learning from Delayed Rewards*. Ph.D. thesis, Cambridge University.

Watkins, C.J.C.H. and Dayan P. (1992). Q-learning. *Machine Learning 8*, 279-292.

Wetherill, G.B. (1975). *Sequential Methods in Statistics*, Chapman and Hall.

White, L.M. (1995). *Temporal Difference Learning: Eligibility Traces and the Successor Representation for Actions*, MSc Thesis, University of Toronto.

Wyatt, J. (2002). Reinforcement Learning: a brief overview in Stamatescu, I.O. *Perspectives on Adaptivity and Learning*, Springer, 243-264.

Zhang, W. and Dietterich, T. G. (1995), A reinforcement learning approach to Job-shop Scheduling. *Proceedings of the International Joint Conference on Artificial Intellience.*