brief version

Learning resembles Evolution Even when using Temporal Difference.

Tom Westerdale

September 8, 2023

Abstract

A learning system can be said to learn strings of actions. An evolving population can be said to evolve strings of genes. In a broad class of adaptive plans, the string frequencies change according to the same simple formula whether the strings are strings of actions in time or strings of genes in space, just as John Holland foresaw. Time symmetric analysis shows this in the basic case where the Markov property holds in the learning system. In this paper we extend that analysis to simple actor critic systems where the actor and critic use the same feature detectors and the critic uses a simple temporal difference method. Much rule based work in evolutionary computation is based on a special case of such systems. The system-environment complex can be viewed in two equivalent ways, in a reinforcement learning way, and in a hierarchical string population way. This is the case even when there is no explicit hierarchy in the reinforcement learning view. Some fields of study emphasize the first view; others emphasize the second. Our equivalence result helps connect problems and investigations across the fields. A major problem is freeloaders that cause bias in value estimates.

1 Value Estimation and Adaptation

A brain learns. A population evolves. In his 1975 book, *Adaptation in Natural and Artificial Systems* [3], John Holland placed these adaptive systems and others in a common formal framework. He believed that learning and evolution are similar adaptive processes.

But a formal approach highlighted problems. Reinforcement Learning [6] deals with conditional probabilities of actions, whereas evolution deals with unconditional probabilities of genes and gene strings. These different probabilities seeemed to change and adapt differently. Learning and evolution seemed different. Reinforcement Learning and Evolutionary Computation separated.

Our work shows that if we take a time symmetric aproach in the formal framework, then the probability changes are not really different. Learning and evolution are similar. Reinforcement Learning and Evolutionary Computation can now get back in touch. Holland was right.

Through interaction with the environment, a learning system learns what to do, which actions to take when and in what combinations. An evolving population, through interaction with the environment, learns which genes are appropriate when and in what combinations. A learning system can be said to learn strings of actions. An evolving population can be said to learn strings of genes. In a broad class of adaptive plans, which I call Natural Plans, string frequencies change according to the same simple formula whether the strings are strings of actions in time or strings of genes in space. The frequency changes preserve the hierarchical structure of strings and substrings. Learning and evolution are similar adaptive processes. We have shown this by first analyzing the basic case where the Markov property holds in the learning system and then extending the analysis to cases where it doesn't. We analyzed the basic case in the paper *Learning Resembles Evolution – the Markov Case* [12]. In the present paper we extend that analysis to cases where the Markov property fails.

In this paper, the adaptive system takes an action in each time step. Each action causes a change in the environment state. The system makes a probabilistic choice of which action to take, so the whole system-environment complex is a Markov chain. We assume it has a finite number of states. We assume that attached to each state of the chain is a fixed number that we will call the state's payoff.[1] When the chain enters a state, the system receives payoff equal to the payoff number attached to that state. I write $\bar{m}$ for the average payoff received per time step.

The adaptive system continually adjusts the probabilities with which it chooses actions. It does this in an attempt to increase $\bar{m}$. The chain transition probabilities are functions of the action probabilities, so the system is adjusting the Markov chain transition probabilities. It wants the probabilities of high value transitions to go up.

[1]In reinforcement learning, the word *reward* means payoff, whereas in evolutionary computation, reward usually means reinforcement.

A transition's total value is the sum of its pre-value, which reflects payoff received in time steps before the transition, and its post-value, which reflects payoff received in time steps after the transition. Two transitions from the same state have the same pre-value, so machine learning procedures ignore pre-values. But here we analyse those procedures using total values. The procedures haven't changed. Only the time symmetric analysis is new.

It is often useful to conceptually separate the adaptive system into two parts, the *critic* and the *actor*. [2] The critic observes the sequence of state transitions and payoffs, and it prepares estimates of state post-values, which it passes to the actor. The actor takes actions and it adjusts the transition probabilities on the basis of the post-value estimates it receives from the critic.

In many systems the separation of critic and actor is not clean, the two parts are rather merged, and more information flows between the two parts. In that case, the separation into actor and critic is rather contrived. Nevertheless, attempting the separation can be informative. In this paper we speak often of the critic and actor. The critic does value estimation, whereas the actor does adaptation. The critic adjusts value estimates using a *value estimation method*. The actor adjusts probabilities using an *adaptive plan*.

Before we discuss value estimation and adaptation, we will examine the Markov chain. In subsections 2.1 and 2.2 we introduce our notation and precisely define the chain quantities. In particular we make precise what we mean by state value and transition value. (Section 2 is a summary of [12] with proofs removed.)

2 Adaptive Markov Chains

2.1 Notation

All vectors in this paper are row vectors. Their transposes are column vectors. Each of our vectors will be a row of N complex numbers. A vector is non-negative if all its entries are non-negative real numbers. A vector is called *stochastic* if its entries are all non-negative real numbers and the entries sum to 1. Of course a vector is nonzero if it has a nonzero entry.

In most cases we will use the following notation. A vector will be a bold lower case letter, for example, $\mathbf{v}$. Each entry in the vector will be written using the corresponding italic letter thus, v_i . A matrix will be an upper case italic letter, for example, M . Each of its entries will be written using that same letter thus, M_{ij} .

The vector $\mathbf{e}$ is the vector of all ones. The vector $\mathbf{e}_i$ is the vector whose i'th entry is 1 and whose other entries are all 0.

Our matrices will be $N \times N$ matrices of real numbers. A matrix is *non-negative* if all of its entries are non-negative real numbers. A matrix is row stochastic if each of its rows is a stochastic vector. So we see that a matrix A is row stochastic of and only if A is non-negative and $A\mathbf{e}^\top = \mathbf{e}^\top$. Of course a row is nonzero if it has a nonzero entry. Similarly for nonzero column.

To *normalize* a non-negative vector means to change its entries, dividing each entry by the sum of all the entries. So the normalized vector is stochastic. To *row normalize* a non-negative matrix means to normalize each of its rows, to divide each entry by the corresponding row sum.

If $\mathbf{v}$ is a vector, then $\mathrm{diag}(\mathbf{v})$ is the diagonal matrix whose ii'th entry is v_i , for all i .

We will be discussing a Markov chain that has N states. When I say Markov chain, I will mean a Markov chain with N states. ($N \geq 2$)

Attached to each legal chain transition $i \to j$ is a positive real number s_{ij} that I call its *availability*. If there is no legal transition from i to j then $s_{ij} = 0$. The matrix S is the $N \times N$ matrix of availabilities s_{ij} . Adaptation will change the availabilities.[2]

The matrix P is the row normalized S . The number $\bar{s}_i$ is the i'th row sum of S . So $P_{ij} = s_{ij}\bar{s}_i^{-1}$.

The matrix P is the chain's transition probability matrix. If $i \to j$ is a legal chain transition, then P_{ij} is the transition probability. By that I mean that P_{ij} is the conditional probability that the next state is j , given that the current state is i . If $i \to j$ is illegal then of course $P_{ij} = 0$. The transition probability of every legal transition is positive. We see that P is row stochastic.

Our Markov chain will be *strongly connected*, which means that for any ordered pair of states $\langle i, j \rangle$, there is a sequence of legal transitions that takes the chain from state i to state j .

Attached to each state i is a fixed unchanging real number m_i called its *payoff*. As S changes, P changes, but the payoff vector $\mathbf{m}$ remains the same.

[2]The matrix S is the only matrix whose entries I write with a lower case letter, s_{ij} .

2.2 Chain Values

We now define the probabilities, the values, and $\bar{m}$ as functions of the availabilities S. The actor adjusts S, and this changes the probabilities P and all the other quantities including $\bar{m}$. The hope is that $\bar{m}$ will rise. I use the term *excess payoff* to mean payoff minus $\bar{m}$.
(I write $\sum_{\mathrm{C}}$ for Cesaro summation.)

Symbol	Meaning	Defining Equation
$\bar{s}_i$	availability row sum	$\bar{s}_i = \sum_j s_{ij}$ or $\bar{\mathbf{s}} = \mathbf{e}\, S^\top$
$\bar{S}$		$\bar{S} = \mathrm{diag}(\bar{\mathbf{s}})$
P_{ij}	probability of $i \to j$	$P_{ij} = s_{ij}\bar{s}_i^{-1}$ or $P = \bar{S}^{-1}S$
$\tilde{p}_i$	probability of state i	
D		$D = \mathrm{diag}(\tilde{\mathbf{p}})$
F_{ij}	frequency of $i \to j$	$F_{ij} = \tilde{p}_i P_{ij}$ or $F = DP$
$\hat{P}$		$\hat{P} = I - P + \mathbf{e}^\top \tilde{\mathbf{p}}$
$\bar{m}$	average payoff	$\bar{m} = \tilde{\mathbf{p}}\, \mathbf{m}^\top$
a_i	excess payoff of state i	$a_i = m_i - \bar{m}$
c_i	post-value of state i	$\mathbf{c}^\top = \hat{P}^{-1}\mathbf{a}^\top = \sum_{\mathrm{C}\,k=0}^{\infty}(P^k \mathbf{a}^\top)$
h_{ij}	choice value of $i \to j$	$h_{ij} = c_j - c_i + a_i$
q_i		$q_i = \tilde{p}_i \bar{s}_i^{-1}$

We know that every $\tilde{p}_i$ is positive, and that the inverse and Cesaro sum in the definition of c_i exist and make sense.[3] It is straightforward to show these facts.

$$\tilde{\mathbf{p}}\, \mathbf{a}^\top = 0 \qquad \text{and} \qquad \tilde{\mathbf{p}}\, \mathbf{c}^\top = 0 \tag{1}$$

$$\tilde{\mathbf{p}} = \tilde{\mathbf{p}}P = \tilde{\mathbf{p}}\hat{P} = \tilde{\mathbf{p}}\hat{P}^{-1} \qquad \text{and} \qquad \mathbf{e}^\top = P\,\mathbf{e}^\top = \hat{P}\,\mathbf{e}^\top = \hat{P}^{-1}\mathbf{e}^\top \tag{2}$$

$$(I-P)\,\mathbf{c}^\top = \mathbf{a}^\top \qquad \text{and} \qquad c_i - a_i = \sum_j P_{ij} c_j \tag{3}$$

$$\sum_j P_{ij} h_{ij} = 0 \tag{4}$$

We note that P is the row normalized F. Let the matrix B be the row normalized $F^\top$. The entry B_{ij} is the conditional probability that the previous state was j, given that the current state is i. The matrix B is the transition probability matrix of a different chain. I call it the *backward chain* and call the original chain the forward chain.[4] The state probabilities in the backward chain are the same as in the forward chain. So are the state payoffs.

Post-value in the backward chain is *pre-value* in the forward chain.
Total value is pre-value plus post-value. Total value is time symmetric.
So we have the following additional definitions

Symbol	Meaning	Defining Equation
B_{ij}	backward probability	$B_{ij} = F_{ji}\,\tilde{p}_i^{-1}$ or $B = D^{-1}F^\top$
$\hat{B}$		$\hat{B} = I - B + \mathbf{e}^\top \tilde{\mathbf{p}}$
b_i	pre-value of state i	$\mathbf{b}^\top = \hat{B}^{-1}\mathbf{a}^\top = \sum_{\mathrm{C}\,k=0}^{\infty}(B^k \mathbf{a}^\top)$
v_{ij}	total value of $i \to j$	$v_{ij} = b_i + c_j$
$\bar{v}_i$	total value of state i	$\bar{v}_i = b_i - a_i + c_i$

It is straightforward to show the following unsurprising facts.

$$(I-B)\,\mathbf{b}^\top = \mathbf{a}^\top \qquad \text{and} \qquad b_i - a_i = \sum_j B_{ij} b_j \tag{5}$$

$$\sum_j F_{ij} v_{ij} = \tilde{p}_i \bar{v}_i \qquad \text{and} \qquad \sum_i F_{ij} v_{ij} = \tilde{p}_j \bar{v}_j \tag{6}$$

$$\sum_i \tilde{p}_i \bar{v}_i = 0 \qquad \text{and} \qquad \sum_{ij} F_{ij} v_{ij} = 0 \tag{7}$$

We can extend the notion of the total value of a transition and define what we can call the total-value of a string of transitions. We write v_σ to mean the total-value of string σ. To avoid introducing unnecessary notation, I shall avoid giving a formal definition. I shall merely illustrate with an example.

[3] For discussion and proofs see [12]. In reinforcement learning, values are often discounted; in evolutionary computation this is rarely done. Discounted post-value is $\mathbf{c}^\top = \sum_{n=0}^{\infty}(x^k P^k \mathbf{a}^\top)$, where $0 < x < 1$. I regard the discount as a distortion to be avoided if possible, but this paper is not the place for that discussion.

[4] The backward chain is sometimes called the time reversed chain.

We choose an illustrative string σ .

$$\sigma \;=\; (\alpha \to \beta)(\beta \to i)(i \to j)(j \to \delta)(\delta \to \varepsilon) \tag{8}$$

The frequency φ_σ of string σ is this.
$\varphi_\sigma = \tilde{p}_\alpha P_{\alpha\beta} P_{\beta i} P_{ij} P_{j\delta} P_{\delta\varepsilon}$.
The *total value* of σ is this.

$$v_\sigma \;=\; b_\alpha + a_\beta + a_i + a_j + a_\delta + c_\varepsilon \tag{9}$$

The pre-value b_α and the post-value c_ε embody the interaction of σ with the environment, with the preceding and following strings.

These definitions make sense. For example, suppose we select a random five transition string, selecting each string σ with probability φ_σ . Then F_{ij} is the probability that the third transition in the selected string is $i \to j$. To see that this is the case, note that in general we have $\tilde{p}_i P_{ij} \;=\; F_{ij} \;=\; B_{ji} \tilde{p}_j$, so if σ is the example string above, then we can write the string frequency φ_σ as
$B_{\beta\alpha} B_{i\beta} F_{ij} P_{j\delta} P_{\delta\varepsilon}$.
Now summing over α, β, ε, and δ gives us F_{ij} .
The transition frequency is the sum of the frequencies of the strings it is in, just as in population genetics. Furthermore, the total value of a substring is the average of the total values of the strings it is in, just as in population genetics. [12]

All the quantities defined in the this subsection are rational functions of S .
In particular, $\bar{m}$ is a rational function of S , and for all ij we have

$$\frac{\partial \bar{m}}{\partial s_{ij}} \;=\; q_i h_{ij} \quad . \tag{10}$$

(Proofs of these statements are in [12].)

2.3 Natural Adaptive Plans

We now suppose that each entry s_{ij} in S is a continuous and differentiable function of some real parameter t , which we can think of as time. It follows that all our quantities are continuous and differentiable functions of t . We write $'$ to mean derivative with respect to t . So the entries in the matrix S' are the derivatives s'_{ij} , and the entries in the vector $\tilde{\mathbf{p}}'$ are the derivatives $\tilde{p}'_i$, and so forth.

Consider the simple scenario in which the actor directly controls each availability s_{ij} and in which the critic passes to the actor the correct post-value c_i of the current state i . The actor follows its adaptive plan and makes small updates to the availabilities in an attempt to increase $\bar{m}$. Here is a simple example.

If the current transition is $i \to j$, the actor adds $K q_i^{-1} c_j$ to s_{ij} .

The constant K is the learning rate. We see that $s'_{ij} \;=\; F_{ij}(K q_i^{-1} c_j) \;=\; s_{ij} K c_j$. The adaptive plan here is $s'_{ij} \;=\; s_{ij} K c_j$. An adaptive plan specifies the desired time derivatives s'_{ij} of the availabilities.

We now define a class of adaptive plans that I call *Natural Plans.*

Natural Adaptive Plan
There are numbers $\beta_1, \beta_2, \beta_3, ..., \beta_N$ such that
for all ij we have $s'_{ij} = s_{ij} K(\beta_i + c_j)$.

Plan $s'_{ij} \;=\; s_{ij} K c_j$ and plan $s'_{ij} \;=\; s_{ij} K h_{ij}$ are both Natural Plans.

In a natural plan we have
$$\bar{m}' \;=\; K \sum_{ij} F_{ij} h_{ij}^2 \quad . \tag{11}$$

This is the Markov chain analog of Fisher's fundamental theorem of natural selection. It is proved in [12]. I call $\bar{m}'$ the climb rate. Equations (11) and (10) tell us that in a natural plan, the climb rate $\bar{m}'$ is positive unless the ground is level.

Theorem 1
The following four conditions are equivalent. Either all four conditions hold or none of them do.
(1) *There are numbers* $\beta_1, \beta_2, \beta_3, ..., \beta_N$ *such that* $s'_{ij} = s_{ij} K(\beta_i + c_j)$ *for all* ij .
(2) $P'_{ij} = P_{ij} K h_{ij}$ *for all* ij .
(3) $F'_{ij} = F_{ij} K v_{ij}$ *for all* ij .
(4) $\varphi'_\sigma \;=\; \varphi_\sigma K v_\sigma$ *for every transition string* σ .

Theorem 1 gives us four equivalent definitions of Natural Plan. Definitions *(3)* and *(4)* are time symmetric. Although obvious in retrospect, the time symmetric theorem 1 long eluded us. Reference [12] is its proof.

Definition *(4)* is also the basic equation for an evolving population of strings σ of genes. The number of copies of a string σ in the population is φ_σ . The reproductive rate of σ is $\varphi'_\sigma \varphi_\sigma^{-1}$. We have value proportional reproductive rate.[5] Strings of transitions in time are behaving like strings of genes in space. The high value strings are becoming more prevalent. Reinforcement Learning is behaving like Evolutionary Computation. There are differences of course, but many of the fundamental principles operate in both fields.

3 What This Paper Tries to Do

In theorem 1, conditions *(1)* and *(2)* look like learning systems; conditions *(3)* and *(4)* look like evolving systems. Theorem 1 connects learning and evolution.

But there is a big problem. Conditions *(1)* and *(2)* can't really hold in an actual learning system. Conditions *(1)* and *(2)* require the critic to keep track of the post-value of every state, and they require the actor to keep track of the availability or probability of every transition. No learning system, no brain, is big enough to do that, and there is not enough time in the universe to do all the updates. We need to re-design the critic to amalgamate states and transitions, but then the post-value estimates will be wrong. And what about the connection to evolution?

In this paper, we will take a machine learning approach to problems with the learning system. We solve each problem through re-design, but each re-design introduces new problems. With each re-design, we examine how this affects the learning-evolution connection in theorem 1. We hope that the whole process eventually leads to a decent learning system, perhaps a system like the actor-critic on page 399 of [6]. In this short paper we can only start that process.

Of course the successive re-design is just a simplified re-statement of machine learning history that is well known. It's all been done before, one way or another. What is new here is that with each re-design we take time to examine the connection to evolution.

In this paper we only begin the re-design process, but even here we see fundamental problems that pervade learning systems, and we see that similar problems pervade evolving populations. The freeloader problem in particular has a long history in evolutionary biology. We shall see that the learning system has a hierarchical structure similar to that of evolving populations and characterized by the same simple equation.

Our re-design will involve amalgamation of transitions, and that will complicate our notation. So let's postpone that amalgamation until section 5, and first do as much as we can using the present simpler notation. Our re-design will introduce biases into the post-value estimates, so let's first look at how much damage such biases can do.

4 Faulty Critics and Faulty Actors

4.1 False Payoffs and False Values

In machine learning, a critic prepares post-value estimates, which it passes to the actor. The estimates typically suffer from bias and from sampling error noise, and these distort what the actor does. Reinforcement Learning and Evolutionary Computation have both developed many techniques to reduce that distortion. Generally speaking, however, the distortion does not go away. We will not discuss any of these techniques in this paper. What we need to discuss is the nature and effect of the bias.

Of course reducing learning rate K reduces the noise, but has no effect on the bias, so in this subsection let's assume that K is small enough that we can ignore the noise. Then we can concentrate on the bias. Let $\bar{w}_i$ be the critic's estimate of the post-value of state i . Then the bias is $\bar{w}_i - c_i$.
The natural plan $s'_{ij} = s_{ij} K c_j$ then becomes

$$s'_{ij} \;=\; s_{ij} K \bar{w}_j \quad . \tag{12}$$

It turns out that theorem 1 has a lot to tell us about what happens when the actor uses this plan.

Let's define a vector $\hat{\mathbf{a}}^\top$ of what I shall call false payoffs.

$$\hat{\mathbf{a}}^\top = (I - P)\,\bar{\mathbf{w}}^\top \qquad\qquad \hat{a}_i \;=\; \bar{w}_i - \sum_j P_{ij} \bar{w}_j \tag{13}$$

[5]Using questionable terminology, John Holland called this fitness proportional selection. The word "fitness" sometimes means value and sometimes means reproductive rate. This confusion is a big subject in its own right. I avoid the word.

The average false payoff $\tilde{\mathbf{p}}\,\hat{\mathbf{a}}^\top$ is zero, so the false excess payoffs are the same as the false payoffs. Let's now pretend that the excess payoffs are the false ones $\hat{a}_i$ rather than the true excess payoffs a_i . Using the false excess payoffs, we define false post-values $\hat{c}_i$, false pre-values $\hat{b}_i$, false transition total values $\hat{v}_{ij}$, false transition choice values $\hat{h}_{ij}$, and false string total values $\hat{v}_\sigma$. The definitions are just as for the true values, but using $\hat{\mathbf{a}}^\top$ rather than $\mathbf{a}^\top$.

For example, here is the definition of false post-values.

$\hat{\mathbf{c}}^\top = \hat{P}^{-1}\hat{\mathbf{a}}^\top$.

Since $\tilde{\mathbf{p}}\hat{P}^{-1} = \tilde{\mathbf{p}}$, we have $\tilde{\mathbf{p}}\,\hat{\mathbf{c}}^\top = 0$. We also have

$\hat{P}\,\bar{\mathbf{w}}^\top = (I - P)\,\bar{\mathbf{w}}^\top + \mathbf{e}^\top\tilde{\mathbf{p}}\,\bar{\mathbf{w}}^\top = \hat{\mathbf{a}}^\top + \mathbf{e}^\top\tilde{\mathbf{p}}\,\bar{\mathbf{w}}^\top$.

Multiplying on the left by $\hat{P}^{-1}$ and using $\hat{P}^{-1}\mathbf{e}^\top = \mathbf{e}^\top$ gives us

$$\bar{\mathbf{w}}^\top = \hat{\mathbf{c}}^\top + (\tilde{\mathbf{p}}\,\bar{\mathbf{w}}^\top)\,\mathbf{e}^\top \qquad\qquad \bar{w}_i = \hat{c}_i + \sum_j \tilde{p}_j \bar{w}_j \tag{14}$$

So the plan (12) that the actor is using can be written

$$s'_{ij} = s_{ij}K((\tilde{\mathbf{p}}\,\bar{\mathbf{w}}^\top) + \hat{c}_j) \quad . \tag{15}$$

This looks like a natural plan, except that we have false post-values in place of true post-values. I'll call it a false natural plan.

Excess payoffs might be any real numbers whose average is zero. So the excess payoffs *could* be $\hat{\mathbf{a}}^\top$, but they aren't. They're $\mathbf{a}^\top$. If they were $\hat{\mathbf{a}}^\top$ then theorem 1 would still hold. So theorem 1 holds with all the values in it replaced by false values. Definition *(4)* says that in a false natural plan, strings of transitions are still behaving like strings in an evolving population, but now the strings with high *false* value are becoming more prevalent. The population still evolves as normal, but it evolves as if the payoffs were the false ones.

In our false natural plan, equation (11) becomes $\bar{m}' = K\sum_{ij} F_{ij}h_{ij}\hat{h}_{ij}$, so $\bar{m}'$ can be negative.[6]

How bad this is depends on how different the false payoffs are from the true ones, and this depends on how badly the critic is doing its job. To investigate this, we need to look at the value estimation method that the critic is using, and it is to value estimation methods that we now turn.

4.2 Markov Chain Value Estimation Using TDz

In this paper the critic uses temporal difference methods. These excllent value estimation methods are much less noisy than trace methods and so are particularly useful in learning. But they introduce damaging biases that complicate the simple learning-evolution equivalence of theorem 1. In this paper we address those biases.[7]

Let's re-design the system so that the critic uses the temporal difference method called undiscounted linear TD(0). I shall call it TDz for short. It works like this.[8]

There are ℓ functions,

$\psi_1, \psi_2, \psi_3, ..., \psi_\ell$,

from the set of states to the real numbers. These functions are fixed and unchanging. We call these functions feature detectors. That terminology reflects the following view. Each state is deemed to have ℓ different features. If i is a state, then $\psi_k(i)$ is the value of its k'th feature. For convenience, I shall write $\psi_k(i)$ as ψ_{ik} . Thanks to the feature detectors, when i is the current state, all the numbers

$\psi_{i1}, \psi_{i2}, \psi_{i3}, ..., \psi_{i\ell}$

are available to the critic.

The critic holds ℓ real parameters

$\mathbf{z} = \langle z_1, z_2, z_3, ..., z_\ell \rangle$.

I'll call these parameters cash balances.[9] We define

$$\tilde{z}_i = \sum_k \psi_{ik} z_k \quad . \tag{16}$$

The number $\tilde{z}_i$ can be used as a guess at the value of state i .

[6]Use equation (15) in the proof of equation (11) in reference [12].

[7]Usually when trace methods are used, the critic and actor are merged. The policy gradient theorem [6] shows that trace methods are unbiased, but they must be discounted to reduce noise, and the discount introduces bias-like distortions. Adaptation using trace methods has long been seen as similar to evolution [7].

[8]The description here is basically Sutton and Barto's description in [6] with the discount removed.

[9]Because in the bucket brigade, a special case, they are treated like cash. [4]

It is the job of the critic to adjust the parameter vector $\mathbf{z}$ to improve the state value guesses.

When transition $i \to j$ occurs, the critic calculates what Sutton and Barto call the TD-error, $\bar{z}_j - \bar{z}_i + a_i$.
It then increments each z_k by the amount

$$\varepsilon\,(\bar{z}_j - \bar{z}_i + a_i)\,\psi_{ik} \quad . \tag{17}$$

The number ε is a small fixed constant that we can think of as the step size.[10]
The increment can of course be negative. We would like to believe that this process converges in some sense.

Given the current $\mathbf{z}$, there is conceptually the average of the possible increments to z_k in the next step. That's $\sum_{ij} F_{ij}\,\varepsilon\,(\bar{z}_j - \bar{z}_i + a_i)\,\psi_{ik}$. The average change in $\mathbf{z}$ I call the average step. We can't actually take an average step, but suppose we could. Holding the probabilities constant and taking a sequence of average steps gives us a sequence of cash balance $\mathbf{z}$ vectors. That sequence always converges.[11] We say that TDz converges in the average step sense.[12] The vector to which it converges I will call $\mathbf{w}$.

Let's define

$$\bar{w}_i \;=\; \sum_k \psi_{ik} w_k \quad . \tag{18}$$

Since the sequence of $\mathbf{z}$ vectors converges to $\mathbf{w}$, the sequence of $\bar{\mathbf{z}}$ vectors converges to $\bar{\mathbf{w}}$. The vector $\bar{\mathbf{w}}$ is the vector of value estimates.

The idea is that when the current state is i , the critic should pass the value estimate w_i to the actor. Well, if the critic has been running for a while, the $\mathbf{z}$ vector should be near its limit,[13] and $\bar{z}_i$ should be near $\bar{w}_i$, so the critic can use equation (16) to obtain a number very close to w_i and pass it to the actor.

Of course the actor is meanwhile goofing things up by changing the probabilities and so changing the limit vector $\mathbf{w}$, but we believe that if the learning rate K is small enough the critic should be able to keep $\mathbf{z}$ close to the limit.

Now as in equation(13), we define the false payoffs $\hat{\mathbf{a}}^\top$.

The value estimate vector $\bar{\mathbf{w}}^\top$ is not unique. It depends on the starting cash balance vector. If we start with two different cash balance vectors, we get two different sequences and probably two different limit vectors $\bar{\mathbf{w}}^\top$, but we have shown[14] that the difference between those limit vectors will be a scalar multiple of $\mathbf{e}^\top$. So we see from the definition of $\hat{\mathbf{a}}^\top$ that the false payoff vector is unique and independent of the starting cash balance vector. So all the false values are also independent of the starting vector.

Now we come back to the question at the end of section 4.1. How badly is the critic doing? How much do the false payoffs $\hat{a}_i$ differ from the true excess payoffs a_i ? How closely do the false payoffs approximate the true ones?

Actually, not very closely at all. The limit cash balance vector $\mathbf{w}$ does not give us a good approximation. There are other cash balance vectors that give us better approximations. But we do have the following result.[15] For all k we have

$$\sum_i \tilde{p}_i \psi_{ik} \hat{a}_i \;=\; \sum_i \tilde{p}_i \psi_{ik} a_i \quad . \tag{19}$$

We can think of $\sum_i \tilde{p}_i \psi_{ik} a_i$ as the average payoff allocated to feature k . I call it the feature's payoff. A feature's false payoff is the same as its true payoff. A change from true to false payoffs conserves payoff at every feature.

4.3 Linear Availability Determination

Now let's look at the actor's problem. We said that typically the actor can't keep track of every availability s_{ij} . There are too many. One approach to this problem is for the actor to also use feature detectors to produce availabilities when needed. Let's re-design the actor to do this. For simplicity, let's have it use the same feature detectors as the critic.

[10] Actually, Sutton and Barto use a TD-error of $\bar{z}_j - \bar{z}_i + a_j$. The difference is not significant. The formulation I've given fits in better with the notation in our discussion.

[11] provided the step size ε is small enough. Remember, ε is constant.

[12] The convergence proof is in [11]. That proof is an extension of the discounted case proof that is in [6]. In [11] we give a usable formula for the limit vector. Of course what the limit vector is depends on the probabilities. It doesn't depend on the step size ε . In the literature, the naming of this kind of convergence seems confusing to me. I call it average step convergence since at least that's clear.

[13] We haven't actually proved this. I have proved what amounts to this for the bucket brigade on Markov chains.[10] That's a special case of TDz.

[14] See [11].

[15] See [11] for proof.

The actor keeps a vector of parameters.
$\boldsymbol{\theta} \;=\; \langle \theta_1, \theta_2, \theta_3, ..., \theta_\ell \rangle$
For each state j we define[16]

$$\phi_j \;=\; \exp\left(\sum_k \psi_{jk}\theta_k\right) \quad . \tag{20}$$

Then for every legal transition $i \to j$ we define
$s_{ij} \;=\; \phi_j$.
(Of course if there is no transition from i to j then $s_{ij} = 0$.)

The availabilities are functions of the parameter vector $\boldsymbol{\theta}$. The actor updates the availabilities by updating $\boldsymbol{\theta}$. What should the updates be? Let's try having the actor use the simplest update. Let's assume the critic passes to the actor the whole cash balance vector, and let's for the moment assume that the critic has in effect converged, so what the actor is getting is a copy of the vector $\mathbf{w}$. Then the critic simply increments every θ_k by the amount Kw_k in every time step.

This gives us
$\theta'_k \;=\; Kw_k$.
Taking time derivatives in (20) and using the previous equation gives us
$\phi'_j \;=\; \phi_j K\bar{w}_j$.
Then for every legal transition $i \to j$ we have
$s'_{ij} \;=\; s_{ij} K\bar{w}_j$,
and of course that equation holds trivially if there is no transition from i to j .
This is the false natural plan (12).

One problem with this approach is that it requires the feature detectors to be able to look one step into the future. Suppose the current state is i . For every legal transition $i \to j$ coming from i , the actor needs to compute s_{ij} , so it needs to compute ϕ_j using equation (20). So the actor needs to know the numbers $\psi_{j1}, \psi_{j2}, \psi_{j3}, ..., \psi_{j\ell}$. One way to attack this problem is to amalgamate transitions into what we call rules.

5 Simple Rule Based Actor-Critic

5.1 Rules

Let's amalgamate our transitions into rules so that we have a rule based system in which just one rule fires in each time step. I'll describe formally what I mean.

Consider a subset x of the set of transitions in our Markov chain. We can think of x as a relation on the set of states. Its domain $\mathrm{Dom}(x)$ is a subset of the set of states. A *rule* is such a relation x that is a function from $\mathrm{Dom}(x)$ into the set of states. The domain is called the rule's *condition.* If the current state is in the rule's condition, then we say the condition is met and the rule is *eligible* to fire. (In this paper, "eligible" means eligible to fire, whereas in reinforcement learning, "eligible" means eligible to be reinforced.)

A rule based system consists of a collection of such rules. Every transition is in at least one rule. In each time step, the actor selects one of the eligible rules and it fires, causing a state transition in the Markov chain. If the current state is i and rule x fires then the transition it causes is the transition from i that is in rule x . We often think of a rule as a condition plus an action. It is the action that causes the state transition.

We will assume that the ranges of the rules are disjoint. (The ranges are bound to be disjoint if the notion of state includes a specification of which rule has just fired.) This means that the rules form a partition of the set of transitions. And since the chain is strongly connected, the ranges form a partition of the set of states.

Associated with each state-rule pair $\langle i, x \rangle$ is an availability number ϕ_{ix} . If $i \in \mathrm{Dom}(x)$ then ϕ_{ix} is a positive number. If $i \notin \mathrm{Dom}(x)$ then ϕ_{ix} is zero. We define
$\bar{\phi}_i \;=\; \sum_x \phi_{ix}$ and $\pi_{ix} \;=\; \bar{\phi}_i^{-1}\phi_{ix}$.
In each time step, the actor selects an eligible rule to fire. It selects the rule probabilistically. The probability of a rule being selected is in proportion to its availability. That is, if i is the current state, then the probability that x is selected to fire is π_{ix} .

We see that the availabilities determine the state transition probability matrix P . If transition $i \to j$ is in rule x , then $s_{ij} = \phi_{ix}$. This gives us the S matrix and we see that P is the row normalized S , just as it ought to be.

[16]Equation(20), or rather equation (20*) of subsection 5.2, is given in [6] as equations (13.2) and (13.3) in subsection 13.1.

We think of the Markov chain as the environment. The environment state determines which rules are eligible to fire. When a rule fires it causes a state transition in the environment chain.

Putting the environment and system together, we can think of the pair $\langle i, x\rangle$ as the state of the *system-environment complex*. Intuitively, if the complex is in state $\langle i, x\rangle$, that means that the environment is in state i and the actor has decided that the eligible rule x is the next one to fire. The states $\langle i, x\rangle$ that can occur are those for which $i \in \mathrm{Dom}(x)$. Each of these states has what I call its target. The *target* of state $\langle i, x\rangle$ is the environment state j such that $i \to j$ is a transition in rule x , (In other words, the target is the next environment state.)

The system-environment complex is a Markov chain. I will call it the pairs chain since each state is a pair $\langle i, x\rangle$. The pair $\langle i, x\rangle$ is a state if and only if $i \in \mathrm{Dom}(y)$. A transition $\langle i, x\rangle \longrightarrow \langle j, y\rangle$ is legal just if j is the target of $\langle i, x\rangle$. If it is legal, then its transition probability is π_{jy} .

5.2 Shared-Features-Actor-Critic System

Let's re-design our system so that it is based on the pairs chain. For reference, I'll call it a shared-features-actor-critic system because I haven't been able to think of a better name. In a shared-features-actor-critic, the actor and critic use the same features in just the way we have described. The critic uses TDz and passes the cash balance vector to the actor. The actor uses equation (20) to determine availabilities and uses the parameter update we described. However, the underlying Markov chain the system is dealing with is not the environment chain; it's the pairs chain. The features are features of pairs chain states. The k'th feature value of state $\langle i, x\rangle$ is ψ_{ixk} . The availability of transition $\langle i, x\rangle \longrightarrow \langle j, y\rangle$ is s_{ixjy} . All of our discussion in section 4 holds, but we need to change our notation throughout so that the states are pairs.

Here is a list of some of the notation changes we make. The top row is items in our previous discussion, and the bottom row gives the corresponding new pairs chain version. The rest of this section will use the pairs chain notation.

environment	i	j	m_i	a_i	c_i	b_i	$\bar{z}_i$	$\bar{w}_i$	$\hat{a}_i$	$\hat{c}_i$	$\hat{b}_i$	h_{ij}	v_{ij}	ψ_{ik}
pairs chain	$\langle i, x\rangle$	$\langle j, y\rangle$	m_{ix}	a_{ix}	c_{ix}	b_{ix}	$\bar{z}_{ix}$	$\bar{w}_{ix}$	$\hat{a}_{ix}$	$\hat{c}_{ix}$	$\hat{b}_{ix}$	h_{ixjy}	v_{ixjy}	ψ_{ixk}

Of course if j is the target of $\langle i, x\rangle$ then a_{ix} is simply a_j . So we see that if two pairs chain states $\langle i, x\rangle$ and $\langle j, x\rangle$ have the same target then they have the same excess payoff. $a_{ix} = a_{jx}$ But they don't necessarily have the same false excess payoff. We can have $\hat{a}_{ix} \neq \hat{a}_{jx}$. The same holds true for payoffs, post-values, and pre-values.

Here are the notation changes that relate to the transition given on the left:

environment	$i \to j$	s_{ij}	ϕ_j	P_{ij}	$\tilde{p}_i$	F_{ij}
pairs chain	$\langle i, x\rangle \longrightarrow \langle j, y\rangle$	s_{ixjy}	ϕ_{jy}	π_{jy}	$\tilde{p}_i\pi_{ix}$	$\tilde{p}_i\pi_{ix}\pi_{jy}$

Let's make sure the availabilities make sense. For the environment chain, we had $s_{ij} = \phi_j$. For the pairs chain transition $\langle i, x\rangle \longrightarrow \langle j, y\rangle$ this becomes $s_{ixjy} = \phi_{jy}$. For the transition to be legal, j must be the target of $\langle i, x\rangle$, so we see that our analog of $\bar{s}_i$ is $\bar{s}_{ix}$ defined as
$\bar{s}_{ix} = \sum_y s_{ixjy} = \sum_y \phi_{jy} = \bar{\phi}_j$,
where the sums are over all rules y such that $j \in \mathrm{Dom}(y)$.
Then of course the transition probability is
$\bar{s}_{ix}^{-1}\, s_{ixjy} = \bar{\phi}_j^{-1}\, \phi_{jy} = \pi_{jy}$.
It all makes sense.

With the changed notation, the equations in our discussion now look like this.

$$s'_{ixjy} = s_{ixjy} K \bar{w}_{jy} \tag{12*}$$

if j is the target of $\langle i, x\rangle$, then
$$\hat{a}_{ix} = \bar{w}_{ix} - \textstyle\sum_y \pi_{jy}\bar{w}_{jy} \tag{13*}$$

$$\bar{w}_{ix} = \hat{c}_{ix} + \textstyle\sum_{jy} \tilde{p}_j \pi_{jy} \bar{w}_{jy} \tag{14*}$$

$$s'_{ixjy} = s_{ixjy} K \left(\left(\textstyle\sum_{ix} \tilde{p}_i \pi_{ix} \bar{w}_{ix}\right) + \hat{c}_{jy} \right) \tag{15*}$$

$$\bar{z}_{ix} = \textstyle\sum_k \psi_{ixk} z_k \tag{16*}$$

z_k increment for $\langle i, x\rangle \longrightarrow \langle j, y\rangle$ is
$$\varepsilon\left(\bar{z}_{jy} - \bar{z}_{ix} + a_{ix}\right)\psi_{ixk} \tag{17*}$$

$$\bar{w}_{ix} = \textstyle\sum_k \psi_{ixk} w_k \tag{18*}$$

$$\textstyle\sum_{ix} \tilde{p}_i \pi_{ix} \psi_{ixk} \hat{a}_{ix} = \textstyle\sum_{ix} \tilde{p}_i \pi_{ix} \psi_{ixk} a_{ix} \tag{19*}$$

if $j \in \mathrm{Dom}(y)$ then
$$\phi_{jy} = \exp\left(\textstyle\sum_k \psi_{jyk}\theta_k\right) \tag{20*}$$

Remember that $\phi_{jy} = 0$ if $j \notin \mathrm{Dom}(y)$.

Let's briefly summarize the shared-features-actor-critic plot line using the pairs chain notation.

The critic uses TDz on the pairs chain.
The relevant equations are (16*), (18*), and the z_k increment (17*).

The actor uses (20*) and $s_{ixjy} = \phi_{jy}$ to obtain the availabilities.
It makes adjustments $\theta'_k = Kw_k$ to parameter vector $\boldsymbol{\theta}$, and this results in adaptive plan (12*).
This plan can be written as (15*), and we see that it is a false natural plan.

That plot line is the same as it was with the environment chain except that the pairs chain notation is more complicated. So why are we looking at this more complicated pairs chain?

It's because in the pairs chain, the feature detectors can do that necessary look into the future. Suppose the current state is $\langle i, x \rangle$, and suppose its target is j . For each possible next state $\langle j, y \rangle$, the actor needs to know the feature values $\psi_{jy1}, \psi_{jy2}, \psi_{jy3}, ..., \psi_{jy\ell}$. The actor can wait until x has fired. Then the current state will be j . The actor now needs those feature values for j and each eligible rule y . It knows which rules are eligible. A feature detector looks at the current state j and the eligible rule y to calculate the feature value.

The actor's adaptive plan is a false natural plan, so strings of transitions in the pairs chain are behaving like strings in an evolving population. The population is evolving as if the payoffs were the false ones. How different are the false payoffs from the true ones? Well, equation (19*) tells us that the average payoff allocated to a feature is the same whether the payoffs are all true or all false.

5.3 Bucket Brigade

Now let's look at a particularly simple special case. A *rule-bucket-brigade* system is a shared-features-actor-critic system in which ℓ is the number of rules, and in which for all ixk we have
$\psi_{ixk} = \delta_{xk}$ (Kronecker delta).
Let's look at the critic. Equations (16*) and(18*) now read $\bar{z}_{ix} = z_x$ and $\bar{w}_{ix} = w_x$. There is one cash balance for each rule.

We ask how the actor chooses which rule to fire. We define $\vartheta_y = e^{\theta_y}$. I'm going to call ϑ_y the availability of rule y . If $\langle j, y \rangle$ is a pair state, that is, if $j \in \mathrm{Dom}(y)$, then we obtain ϕ_{jy} from equation (20*), which in our special case reads $\phi_{jy} = \vartheta_y$.

Suppose the current state is j . Then
$\phi_{jy} = \begin{cases} \vartheta_y & \text{if } y \text{ is eligible.} \\ 0 & \text{if } y \text{ is ineligible.} \end{cases}$
Now $\bar{\phi}_j = \sum_y \phi_{jy}$, so we can write $\bar{\phi}_j = \sum_y \vartheta_y$, where the sum is only over the eligible rules y . Then we have $\pi_{jy} = \bar{\phi}_j \vartheta_y$. The probability of an eligible rule being chosen is proportional to its availability.

The availability adjustment is $\theta'_x = Kw_x$. In other words, $\vartheta'_x = \vartheta_x K w_x$.

TDz operation is as follows. Suppose the transition is $\langle i, x \rangle \longrightarrow \langle j, y \rangle$. The increment to z_k is given by (17*). Since a_{ix} is a_j , the increment can be written $\varepsilon\,(z_y - z_x + a_j)\,\delta_{xk}$. In other words, when the firing sequence is $x \to y$, the cash balance z_x is incremented by $\varepsilon\,(z_y - z_x + a_j)$, where a_j is the excess payoff following the firing of x . The other cash balances are unchanged.

So in summary, the system works like this. I'll write ϑ_x and z_x for the availability and cash balance of rule x . The rule to fire is chosen probabilistically from among the eligible rules with probabilities proportional to the rule availabilities. When a rule x fires, its cash balance z_x is incremented by ε times the resulting excess payoff. In addition, rule x passes proportion ε of its cash to the rule that fired in the previous time step. John Holland called this method of cash adjustment a *bucket brigade*.[17] In each time step, every rule's availability ϑ_x is incremented by $\vartheta_x K z_x$. (That's supposed to be close to $\vartheta_x K w_x$.)

This system is a simplified version of systems that have long been in use in evolutionary computation.

In a rule-bucket-brigade system, equation (19*) becomes

$$\sum_i \tilde{p}_i\,\pi_{ix}\,\hat{a}_{ix} \quad = \quad \sum_i \tilde{p}_i\,\pi_{ix}\,a_{ix} \quad . \tag{21}$$

Now suppose we are in the current state and then a rule fires. The conditional probability that the rule is x given that the state is i , that conditional probability is π_{ix} . But what is the conditional probability that the state is i given that the rule is x ? A little Beyesian reasoning tells us that that conditional probability is
$\tilde{p}_i \pi_{ix} (\sum_j \tilde{p}_j \pi_{jx})^{-1}$.

So suppose x fires. What is the excess payoff *after* x fires? If i was the environment state before x fired, then the excess payoff afterwards is a_{ix} . And the probability it was i before x fired is

[17] This is a simplified bucket brigade. Holland describes a more complicated version. [4]

$\tilde{p}_i\pi_{ix}(\sum_j \tilde{p}_j\pi_{jx})^{-1}$. So the average excess payoff immediately after x fires is $\sum_i \tilde{p}_i\pi_{ix}(\sum_j \tilde{p}_j\pi_{jx})^{-1}a_{ix}$.
Equation (21) tells us that this average is the same whether we use true or false payoffs.

In our special case, equation (13*) becomes $\hat{a}_{ix} = w_x - \sum_y \pi_{jy}w_y$, where j is the target of $\langle i,x\rangle$. So if $\langle i,x\rangle$ and $\langle j,x\rangle$ have the same target then they have the same false payoff. Therefore, we can assign the false payoffs directly to the environment states. Take any environment state j . Let x be the unique rule in whose range j is located. The false payoff of j is the false payoff of any $\langle i,x\rangle$ whose target is j . So now let's look at the states in the range of some rule x . The previous paragraph tells us that the average of the excess payoffs of those states is the same whether we use true or false payoffs. I'll write $\hat{a}_j$ for the false payoff of j .

The situation with false post-values is more straightforward. Let β be the average value estimate. That is, $\beta = \sum_{jy} \tilde{p}_j\pi_{jy}w_y$. Then equation (14*) reads $\hat{c}_{ix} = w_x - \beta$. We see that the false post-value of $\langle i,x\rangle$ is independent of i . As we did with false payoffs, we can assign the false post-values to the environment states. The states in a rule's range will all have the same false post-values. So we can assign the false post-values to the rules themselves. Let's write $\hat{c}_x$ for the false post-value of rule x . We have $\hat{c}_x = w_x - \beta$, so the rule false post-values are simply the value estimates normalized by subtraction of the average.

I call a rule *level* if every state in its range has the same post-value. A change from true payoffs to false payoffs makes every rule level without changing the average excess payoff of any range.

5.4 Bucket Brigade on a Markov Chain

Let's take a moment to look at a very special rule-bucket-brigade case. In this special case, the number of rules is the same as the number of states. For each state there is a rule that consists of all the transitions to that state. We can think of the rule availability as the availabilty of that state. We simply have a Markov chain with state availabilities. The actor chooses the next state transition.

Now we said that if the critic uses the bucket brigade, then in each rule's range, the average of the excess payoffs is the same whether we use true or false payoffs. But here, every range is a singleton, so the true and false excess payoffs are the same, and the true and false post-values are the same. We said that $w_x - \beta$ was the false post-value of x so here it is also the true post value. On a Markov chain, the bucket brigade converges to the correct post-values.[18]

5.5 The Rule Chain and Leveling the Rules

Now let's return to rule-bucket-brigade systems in general.

Let $\mathcal{P}_{xy}$ be the conditional probability that the next rule to fire will be y given that x is the rule that just fired. We can imagine a Markov chain whose states are the rules and whose transition probabilities are the $\mathcal{P}_{xy}$ probabilities. Let's call this the rule chain. (The Markov property holds on the rules in the rule chain, but it doesn't hold on the rules in the system.) We note that the absolute probability of state x in the rule chain is the same as the probability that rule x fires in the rule-bucket-brigade system, namely $\sum_i \tilde{p}_i\pi_{ix}$.

Let the payoff attached to each rule chain state be the average of the payoff elicited by the firing of that rule. If we run the bucket brigade on the rule chain, its average step is the same as when we run the bucket brigade on the rule based system, so it converges (in the average step sense) to the same value estimates w_x . And on a Markov chain, the bucket brigade converges to the correct post-values. It follows that the true post-value of state x in the rule chain is the same as what we earlier called $\hat{c}_x$, the false post-value of rule x in the rule based system. The bucket brigade is giving us rule chain post-values, not true post-values.

Another way of thinking of it is this. Suppose the environment chain could change state within a rule's range. In each time step, before the proper state transition, the environment looks at the rule range that the current state is in. It then selects a new state at random from that range, and the state transition proceeds from that new state. The selection is probabilistic with probabilities proportional to the absolute state probabilities $\tilde{p}_i$. This makes each rule level. In fact, it makes the state post-values equal to the false post-values.

[18] Bucket brigade average step convergence to the correct post-values is proved directly in [9].

5.6 Parallelism

Look at the shared-features-actor-critic system and note the inherent parallelism in equation (20*). If we define $\vartheta_k = e^{\theta_k}$ then equation (20*) can be written $\phi_{ix} = \prod_k \vartheta_k^{\psi_{ixk}}$.
Let j be the target of $\langle i, x \rangle$. Now let's pretend we have ℓ separate rules, and that ϑ_k is the probability of rule k firing. The symbol x stands for a salvo of rules firing. The number ψ_{ixk} is the number of shots rule k fires in salvo x . When the actor fires salvo x , that causes transition $i \to j$. So we see that $\phi_{ix} = P_{ij}$. The adaptive adjustment $\theta'_k = Kw_k$ becomes $\vartheta'_k = \vartheta_k K w_k$. This all resembles the situation in evolving populations of gene strings, where a gene string (organism) causes changes (transitions) in its environment.[19]

6 The Adaptive Hierarchy – Freeloaders

As we have said, the shared-features-actor-critic system uses a false natural plan. This means that for all transition strings σ , the fourth condition of Theorem 1 holds, but with v_σ replaced by the false total string value, which we write as $\hat{v}_\sigma$. We have $\varphi'_\sigma = \varphi_\sigma K \hat{v}_\sigma$.

The definition of $\hat{v}_\sigma$ here is like equation (9) except that the payoffs and values are false and the states are pairs. For example, in place of $(i \to j)$ in equation (8) we might have $\langle i, x \rangle \longrightarrow \langle jy \rangle$. In that case, a_i and a_j in equation (9) would be replaced by $\hat{a}_{ix}$ and $\hat{a}_{jy}$.

The equation $\varphi'_\sigma = \varphi_\sigma K \hat{v}_\sigma$ holds for any string of any length, and so it holds for all of its substrings. Just as in evolving populations of gene strings, the strings and substrings form a hierarchy, and the adaptation equation $\varphi'_\sigma = \varphi_\sigma K \hat{v}_\sigma$ holds at every level of the hierarchy.[20]

It is instructive to look at rule-bucket-brigade systems because they are comparatively simple, so some of the issues are clearer. The false payoffs and false values of the pairs are the same as of their targets, so in equations we can often replace the pairs with their targets. In the last paragraph we replaced i with $\langle i, x \rangle$ and replaced a_i with $\hat{a}_{ix}$. But the target of $\langle i, x \rangle$ is j , so we can replace i with j and replace a_i with $\hat{a}_j$. The string σ is then a string of environment state transitions, and its false total value is what the true value would be if the payoffs were the false ones. The false value of the string in equation (8) is simply

$$\hat{v}_\sigma = \hat{b}_\alpha + \hat{a}_\beta + \hat{a}_i + \hat{a}_j + \hat{a}_\delta + \hat{c}_\varepsilon \tag{22}$$

The string frequencies are changing much as they do in an evolving population of gene strings. There is a virtual population of transition strings. The population structure is hierarchical, with substring frequencies changing in the same way as do the frequencies of the longer strings. But the fly in the ointment is that success is measured in terms of false payoff, not true payoff.

Consider a string that has low true total value, but high false total value. Such a culprit string will unfairly increase its prevalence in the virtual population, and this can cause $\bar{m}$ to drop.

Equation (9) shows us that much of a string's value is the excess payoffs received when the string is executed. In (22) these payoffs are replaced by false payoffs. If a false payoff is higher than the corresponding true payoff, this can give the string an unfairly high false value. For example, $\hat{a}_i$ might be higher than a_i . A switch from true payoffs to false payoffs is giving $\hat{a}_i - a_i$ undeserved extra payoff to the culprit string.

How big can this undeserved payoff $\hat{a}_i - a_i$ be? Let x be the rule in whose range i is located. We have seen that the switch from true to false payoffs doesn't change the average of the payoffs of the states in $\mathrm{Ran}(x)$. So the undeserved payoff $\hat{a}_i - a_i$ is not created out of thin air. It's stolen from the other states in $\mathrm{Ran}(x)$. The culprit steals that undeserved payoff from victim strings that pass through $\mathrm{Ran}(x)$. The victims produce the payoff, which the culprit then steals. In Evolutionary Computation we call the culprit a *freeloader*.[21]

Freeloaders distort the adaptive process. They can cause $\bar{m}$ to head continually downhill. They afflict adaptive systems everywhere, in evolutionary biology, in evolutionary computation, in reinforcement learning, and of course in economics. They are easy to see when the critic uses the bucket brigade, but with few exceptions they occur whenever a critic uses a temporal difference method. Their presence means that the critic produces *biased* value estimates. There are many important ways of combatting freeloaders, but

[19] Holland often had rules firing in parallel, but his parallel bucket brigades were rather ad hoc. He should have tried TDz on the parallel formulation we have given here. Evolutionary computation can learn from reinforcement learning.

[20] In evolution, sexual recombination distorts the equation for the longer strings, and there are many complications.

[21] A simple explanation of freeloaders in the bucket brigade is in [8]. In that early paper, bucket brigade freeloaders were called cannibals. Thanks to Dave Goldberg, we now call them freeloaders, the term used in other fields.

generally speaking the freeloaders do not go away. Further discussion of freeloading is beyond the scope of this paper.

Temporal difference methods are excellent because they take advantage of underlying Markov structure to implicitly increase sample size and so reduce sampling error noise. This is crucial. Otherwise, to reduce noise, adaptation would have to slow to a snail's pace. Since freeloading is inherent in these excellent methods, understanding its detrimental effects is important.

7 Problems

We said that in this paper we can only begin the re-design process. The next re-design of our system is beyond the scope of this paper. So are the problems and inadequacies the re-design should address. But let's just touch upon some of them.

The big problem is of course the freeloading inherent in any temporal difference critic, but there are other problems more amenable to attack. One is an actor problem, the possible spreading out over time of the entries in the $\boldsymbol{\theta}$ vector. Entries could become too large to be practical. The problem can be particularly bad if there is redundancy in the feature detectors.

For example, suppose $\psi_{ix2} = \psi_{ix3}$ for all $\langle i, x\rangle$. Then θ_2 and θ_3 could diverge without even affecting the probabilities since the probabilities depend only on $\theta_2 + \theta_3$. The redundancy doesn't occur in rule-bucket-brigade systems, but it does occur in other shared-features-actor-critic systems.

If there is redundancy, that means that the function from cash balance vectors to post-value estimate vectors (equation (18*)) has a non-trivial kernel. It turns out that TDz with a non-zero but vanishing discount gives us an appropriate cash balance vector that is orthogonal to the kernel. That removes most of the redundancy. If there is a cash balance vector $\boldsymbol{\eta}$ that gives us a post-value estimate of 1 for every state, then there is still one remaining dimension of redundancy. Subtracting a multiple of $\boldsymbol{\eta}$ from $\boldsymbol{\theta}$ leaves the probabilities unchanged and combats that redundancy. Producing a copy of $\boldsymbol{\eta}$ looks tricky, but possible. (See [11] for proofs relevant to all this.)

Sutton and Barto have a different approach. [6, page 399] Their critic uses a discount, so that gets rid of the redundancy associated with the kernel.[22] And their actor uses false choice value where ours uses false post-value, so their actor is trying for condition *(2)* of theorem 1 rather than for condition *(1)*. That gets rid of the remaining dimension of redundancy.

Conceptually, their actor uses its own cash balance vector for its $\boldsymbol{\theta}$. The actor produces its cash balance vector using the false choice values it gets from the critic, but it uses an increment different from that used by the critic. The result is a vector giving value estimates that resemble what population genetics calls genic values. Remember our redundancy example where it was the sum $\theta_2 + \theta_3$ that determined the probabilities. Here, the θ_2 and θ_3 entries are both replaced by the sum. Problem gone. Actually, the actor's cash balance vector doesn't explicitly appear, and the increments go directly to $\boldsymbol{\theta}$.

Often features relevant to the actor are different from features relevant to the critic. Sutton and Barto's actor uses different features from the those the critic uses. Sutton and Barto's actor has other complications, and my comments about it may not be totally correct. It's certainly more general than I've implied.

Returning to our shared-features-actor-critic system, we note that just because θ_k is increasing, that doesn't mean that it will necessarily continue to increase without bound. As $\boldsymbol{\theta}$ changes, the probabilities change, and so do the true values, the false values, and the limit cash balance vector $\mathbf{w}$.

In the shared-features-actor-critic that we have described, the environment itself is deterministic. Given the current state and the rule that fires, the next environment state is determined. But in the systems described by Sutton and Barto [6], the next environment state is chosen probabilistically. It's as if the actor and environment take turns making probabilisitic choices. Our analysis doesn't extend directly to such systems because the probabilities used by the environment are kept constant (clamped).

One way of extending the analysis to such systems is by introducing a new kind of false payoff that makes the environment think there is no point in changing its probabilities. The new false payoffs and old false payoffs are on different states.

8 Thoughts

In this paper we used the time symmetric approach to describe strings of actions in time in much the same way that evolutionary biology describes strings of genes in space. Here we have virtual populations of

[22]See footnote 3.

strings of actions just as evolutionary biology has populations of strings of genes. The populations evolve in similar but not identical ways. Both populations are hierarchical, both suffer from freeloading, and the basic equation of frequency change of strings and substrings is the same.

Different fields of study see adaptive systems from different points of view, but the basic issues are the same. In this paper we looked at the basic issues from at least two equivalent viewpoints, which we might call the reinforcement learning viewpoint and the evolution viewpoint. From the reinforcement learning viewpoint we see a system choosing successive actions probabilistically and modifying those probabilities through learning. From the evolution viewpoint we see a hierarchical population of strings and substrings and we see the string frequencies changing through evolution. Two equivalent views, one flat and one hierarchical, but it's the same system.

All this was outlined by John Holland in his 1975 book, *Adaptation in Natural and Artificial Systems.* [3] The problem back then was that crucial steps in the formalization of the equivalence were missing. To obtain the basic equivalence in theorem 1, we have to think of action values as including pre-values as well as post-values. Without that time-symmetric insight, the equations of learning and evolution remained stubbornly different, and the fields of reinforcement learning and evolutionary computation separated and drifted apart. Each field quite properly ignored the other, because each had little to contribute to the other. Until now. Now it is possible to re-establish contact.

There is an underlying hierarchy in learning that is in some respects formally equivalent to the hierarchy in evolving populations. Some reinforcement learning systems have an explicit hierarchy. [1] In what ways does this differ from a hierarchy that may be already implicit in an evolution view of the system? Is a successful explicit hierarchy sometimes one that emphasizes and elaborates on a hierarchy that is already there?

The *Group Selection* literature has shown us how an evolving population can be equivalently described in both a hierarchical fashion and flat fashion.[23] It has much to say about the role of freeloading and altruism (negative freeloading).[24]

In establishing the equivalences that Holland foresaw, it is the temporal difference methods that have been the most recalcitrant. But in this paper we have seen that even they yield to the time symmetric approach.

Holland was right.

References

[1] Andrew G. Barto and Sridhar Mahadevan. Recent Advances in Hierarchical Reinforcement Learning. *Discrete Event Dynamic Systems: Theory and Applications*, 13:343–379, 2003.

[2] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Looking back on the actor-critic architecture. *IEEE Trans. Syst., Man, Cybern.*, SMC-51(1):40–50, 2021.

[3] J. H. Holland. *Adaptation in Natural and Artificial Systems.* Univ. of Michigan Press, Ann Arbor, 1975.

[4] J. H. Holland, K. J. Holyoak, R. E. Nisbett, and P. R. Thagard. *Induction: Processes of Inference, Learning and Discovery.* MIT Press, Cambridge, MA, 1986.

[5] Elliott Sober and David Sloan Wilson. *Unto Others, the Evolution and Psychology of Unselfish Behavior.* Harvard University Press, Cambridge, Massachusetts, 1999.

[6] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction, Second edition.* MIT Press, Cambridge, Mass., 2018.

[7] T. H. Westerdale. A Reward Scheme for Production Systems with Overlapping Conflict Sets. *IEEE Trans. Syst., Man, Cybern.*, SMC-16(3):369–383, 1986.

[8] T. H. Westerdale. A defense of the bucket brigade. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 282–290, San Mateo, CA, 1989. Morgan Kaufmann.

[23] The flat value is Hamilton's *inclusive fitness.*

[24] See Part 1 of [5]

[9] T. H. Westerdale. Quasimorphisms or queasymorphisms? Modeling finite automaton environments. In Gregory J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 128–147, San Mateo, CA, 1991. Morgan Kaufmann.

[10] Tom Westerdale. Bucket Brigade Convergence on Markov Chains. pages 1–50, 2017. unpublished.

[11] Tom Westerdale. Average Step Convergence of Undiscounted Linear-TD(0). pages 1–16, 2023. unpublished.

[12] Tom Westerdale. Learning Resembles Evolution – the Markov Case. pages 1–12, 2023. unpublished.