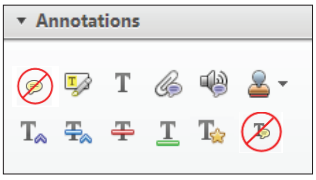
Page Proof Instructions and Queries

Journal Title: ADB
Article Number: 589355

Greetings, and thank you for publishing with SAGE. We have prepared this page proof for your review. Please respond to each of the below queries by digitally marking this PDF using Adobe Reader.

Click "Comment" in the upper right corner of Adobe Reader to access the mark-up tools as follows:

For textual edits, please use the "Annotations" tools. Please refrain from using the two tools crossed out below, as data loss can occur when using these tools.



For formatting requests, questions, or other complicated changes, please insert a comment using "Drawing Markups."

Detailed annotation guidelines can be viewed at: http://www.sagepub.com/repository/binaries/pdfs/AnnotationGuidelines.pdf
Adobe Reader can be downloaded (free) at: http://www.adobe.com/products/reader.html.

Sl. No.	Query
1	AQ1: Please small photographs and mini biographies for all authors.

Original Paper

Adaptive Behavior

Adaptive Behavior
1–9

DOI: 10.1177/1059712315589355
adb.sagepub.com
SAGE

Back to optimality: A formal framework to express the dynamics of learning optimal behavior

Eduardo Alonso[1,2,3], Michael Fairbank[1,2] and Esther Mondragón[2,3]

Abstract

Whether animals behave optimally is an open question of great importance, both theoretically and in practice. Attempts to answer this question focus on two aspects of the optimization problem, the quantity to be optimized and the optimization process itself. In this paper, we assume the abstract concept of cost as the quantity to be minimized and propose a reinforcement learning algorithm, called Value-Gradient Learning (VGL), as a computational model of behavior optimality. We prove that, unlike standard models of Reinforcement Learning, Temporal Difference in particular, VGL is guaranteed to converge to optimality under certain conditions. The core of the proof is the mathematical equivalence of VGL and Pontryagin's Minimum Principle, a well-known optimization technique in systems and control theory. Given the similarity between VGL's formulation and regulatory models of behavior, we argue that our algorithm may provide psychologists with a tool to formulate such models in optimization terms.

Keywords

Optimality, Principle of Least Action, bliss point, reinforcement learning, Value-Gradient Learning

1 Background

Of all the possible trajectories that an object thrown into the air can follow, why does it follow a parabola? Why doesn't it go up, stay a while at its highest point and then rush down?

Typically, we explain the object's motion in terms of the force of gravity and Newton's second law. Starting with a description of the initial state (the object's initial position and velocity). we will obtain a complete specification of the path the object takes by generating it, one infinitesimal piece at a time. It will be a cumbersome exercise, but we know we will get the correct answer. And yet, we will be wrong. Modern physics teaches us that the true trajectory is the one that makes the *action* "least", or, more precisely, stationary (Goldstein, 1974).

Consider the trajectory of the object: on the one hand, it wants to spend as much time as possible where the kinetic energy is least and the potential energy is greatest, that is, at the top of its trajectory; on the other hand, if it spends too much time at the top of its trajectory, it will need to rush to get up there and get back down and this will take a lot of action. The perfect compromise is a parabolic path (Baez, 2005).

Formally, the *action* to be minimized is the integral of the *Lagrangian function*, over time. The Lagrangian itself describes completely the dynamics of the system under consideration as the difference between its kinetic energy (the energy due to its motion, how much "is happening") and its potential energy (the energy due to its position or configuration, how much "could happen"). The object follows a particular trajectory not because of the effect of gravitation *per se*, but because Nature *minimizes* the action by taking the shortest path. This is the Principle of Least Action (de Maupertuis, 1744; Euler, 1744; although first formulated by Pierre de Fermat as the principle of least time in the 17th century).

Crucially, this principle is equivalent to the Euler–Lagrange equation of motion. This equation is more elegant (it works with energy scalars rather than force vectors) and universal (it applies to any generalized

[1]Department of Computer Science, City University London, London EC1V 0HB, UK
[2]Systems and Control Research Centre, City University London, London EC1V 0HB, UK
[3]Centre for Computational and Animal Learning Research, St. Albans AL1 1RQ, UK

Corresponding author:
Eduardo Alonso, Department of Computer Science, City University London, London EC1V 0HB, UK
Email: E.Alonso@city.ac.uk

co-ordinates) than Newton's, and, when transformed into its Hamiltonian form, reflects, via Noether's theorems, the symmetries of Nature. These are fundamental concepts upon which modern physics is founded, including electromagnetism, general relativity and the Standard Model (Feynman & Hibbs, 1965; Landau & Lifshitz, 1975; Taylor & Wheeler 2000).[1]

It has been suggested recently that the Principle of Least Action is not circumscribed to physics but rather that it is a general law that also applies to biological phenomena and even to natural selection (Kaila & Annila, 2008).

The question is: can we export this analysis to the study of behavior? Consider the curves depicting the cumulative number of responses under simple reinforcement schedules as described in Skinner (1959). They show a correlation between the rate of responding and the rate (and magnitude) of the reinforcement. Of all the possible curves, why do animals follow these curves in particular?

In a way, we are facing the same questions as with the trajectories of objects in motion. And, as with Newton's laws of motion, several models have been proposed to explain these patterns of behavior, from the Matching Law (Herrnstein, 1961) to the Mathematical Principles of Reinforcement (Killen & Sitomer, 2003). Yet, we are still missing fundamental principles that guide our research in the area (although see Anttila & Annila, 2011, for a strong argument for the consumption of free energy in the least time for such as principle).

In the rest of the paper, we present a meta-model of learning and behavior based on optimal control theory that fills this gap. In this sense, our proposal is close to Hanson (1977) and Killeen (1992). First we introduce the underlying formal framework, described as a reinforcement learning process in its broadest sense; in Section 3 we formulate our model, that we call Value-Gradient Learning (VGL); in Section 4, we demonstrate the relation between VGL and Pontryagin's Minimum Principle and its convergence to optimality; VGL's convergence to optimality is proved in Section 5; finally, a parallelism between regulatory theories of behavior and our algorithm is drawn.

2 Basic framework

The basic assumption of our proposal is that animals behave (choose actions, make decisions) to optimize a given quantity, e.g., to maximize an accumulative reward signal in reinforcement learning, to minimize the distance between a distribution of behaviors and the bliss point in regulatory models, or to maintain a steady ratio of energetic gain to cost in foraging theories. We acknowledge that it is controversial whether behavior obeys an optimality principle (e.g., Staddon, 2007). Nevertheless, we endorse the idea that Nature follows one form or another of the Principle of Least Action, and that, in any case, such principle is a valid tool to formulate the laws of behavior dynamics.

Given its generality (Wörgötter & Porr, 2005), we take reinforcement learning as our basic framework and express it in terms of optimal control systems. At the end of the day, animals are behavior systems—sets of behaviors that are organized around biological functions and goals, e.g., feeding (Timberlake & Silva, 1995), defense (Fanselow, 1994), or sex (Domjan, 1994). More specifically, a behavior system is a closed-loop control system: The control (response) affects the system's output (whether it is reinforced), which in turn is measured and fed back to alter the control.

A typical scenario is an animal inhabiting a state space $S \subset \mathbb{R}^n$, such that at t time it "lives" in state vector $\vec{x}_t \in \mathbb{S}$. The state space represents any features the modeler considers relevant, typically a collection of stimuli but can also include internal constructs. At each time, the animal chooses an action $\vec{u}_t$ (from an action space $\vec{u}_t \in \mathrm{A}$), which takes it to the next state according to a *model function*:

$$\vec{x}_{t+1} = f(\vec{x}_t, \vec{u}_t) \tag{1}$$

and gives it an immediate scalar U_t, given by the *cost function*:

$$U_t = U(\vec{x}_t, \vec{u}_t) \tag{2}$$

The animal keeps acting, forming a trajectory of states $(\vec{x}_0, \vec{x}_1, ...)$ indefinitely or until a given terminal state is reached (for instance, the end of a trial). In this problem the animal must learn to choose actions that minimize the expectation of the total long-term cost received from any given start state $\vec{x}_0$. Formally, the problem is to find an *action network* $A(\vec{x}, \vec{z})$, where $\vec{z}$ is the parameter vector of a function approximator (typically, a neural network), which calculates actions:

$$\vec{u}_t = A(\vec{x}_t, \vec{z}) \tag{3}$$

such that the following *cost-to-go function* (aka *value function*) is minimized:

$$J(\vec{x}_0, \vec{z}) = \sum\nolimits_t \gamma^t U_t \tag{4}$$

subject to Equations 1, 2 and 3, where $\gamma \in [0, 1]$ is a constant discount factor that specifies the relative importance of long-term costs over short-term ones. Throughout the rest of the paper, we assume that the action network is differentiable with respect to all of its arguments, and similar differentiability conditions apply to the model and cost functions.

At this point, it must be emphasized that the model function, Equation 1, can represent any given model of

behavior (Staddon & Cerutti, 2003), and that the quantity to be optimized can be any quantity.

In order to *learn* the optimal trajectory we use a second function approximator, $\tilde{J}(\vec{x},\vec{w}) \in \mathbb{R}$, with weight vector $\vec{w}$, known as the *critic*. The idea is to train the critic to approximate the cost-to-go function, so that $\tilde{J}(\vec{x},\vec{w}) \simeq J(\vec{x},\vec{z})$ for all $\vec{x} \in \mathbb{S}$. If this is achieved perfectly, and if simultaneously the action network always chooses actions according to the *greedy policy*:

$$\vec{u} = \arg\min_{\vec{u}\in A}\left(U(\vec{x},\vec{u}) + \gamma\tilde{J}(f(\vec{x},\vec{u}),\vec{w})\right) \quad \forall\vec{x} \qquad (5)$$

then Bellman's Principle of Optimality (Bellman, 1957) establishes that the trajectories produced will be optimal. Choosing actions by Equation 5 is called the greedy policy, as it chooses actions that the critic rates as best.

The question is to ascertain the conditions under which convergence to optimality is guaranteed and whether such conditions are behaviorally plausible. In particular, Temporal Difference (TD)[2] comprises a collection of reinforcement learning algorithms that have become popular in modeling real-time error-correction learning. TD is model-free, that is, it does not assume knowledge of Equations 1 and 2. As a consequence, in order to meet Bellman's equation, it predicates that animals learn optimal policies by exploring the whole state space by repeatedly trying all possible actions at every possible state. Unfortunately, in most biologically relevant problems the space state is too large (Ludvig, Bellemare, & Pearson, 2011), and approximations can diverge (Fairbank & Alonso, 2012a).

3 Value-Gradient Learning

The Value-Gradient Learning algorithm (VGL) that we present as an alternative addresses the issue of the Bellman equation needing to be solved over the whole state space, in that it turns out to be only necessary to fully learn the value gradient along a single trajectory, under a greedy policy, for it to be locally extremal, and often locally optimal (Fairbank & Alonso, 2012b; Fairbank, Prokhorov & Alonso, 2013). In describing VGL, we use what we call *trajectory shorthand notation*, according to which subscripted t variables attached to a function name indicates that all arguments of the function are to be evaluated at time step t of a trajectory. For example, $\tilde{J}_{t+1} \equiv \tilde{J}(\vec{x}_{t+1},\vec{w})$, $U_t \equiv U(\vec{x}_t,\vec{u}_t)$ and $\tilde{G}_t \equiv \tilde{G}(\vec{x}_t,\vec{w})$. A convention is also adopted that all defined vector quantities are columns, whether they are coordinates, or derivatives with respect to coordinates, using the transpose of the usual Jacobian notation.

The approximate value gradient, or *critic gradient*, is defined to be

$$\tilde{G}(\vec{x},\vec{w}) = \frac{\partial\tilde{J}(\vec{x},\vec{w})}{\partial\vec{x}} \qquad (6)$$

and the *target value gradient* is defined as

$$G'_t = \left(\frac{DU}{D\vec{x}}\right)_t + \gamma\left(\frac{Df}{D\vec{x}}\right)_t\left(\lambda G'_{t+1} + (1-\lambda)\tilde{G}_{t+1}\right) \qquad (7)$$

where $\left(\frac{D}{D\vec{x}}\right)_t$ is a shorthand for

$$\frac{D}{D\vec{x}} = \frac{\partial}{\partial\vec{x}} + \frac{\partial A}{\partial\vec{x}}\frac{\partial}{\partial\vec{u}} \qquad (8)$$

and λ is a fixed "bootstrapping" constant, described further below.

The target value-gradients, G', are so called because the VGL objective is to achieve $\tilde{G}_t = G'_t$ for all t along a trajectory. The VGL system must also make sure the greedy policy Equation 5 is satisfied at every time step. Achieving these two goals simultaneously will achieve optimality, as discussed and proven in Section 5.

In order to achieve these two goals, the VGL system is comprised of two weight updates. The first is a weight update to the critic $\tilde{G}(x,w)$, given by:

$$\Delta\vec{w} = \alpha\sum_t\left(\frac{\partial\tilde{G}}{\partial\vec{w}}\right)_t\Omega_t\left(G'_t - \tilde{G}_t\right) \qquad (9)$$

where $\alpha>0$ is a learning-rate constant. The objective of Equation 9 is to make each critic gradient $\tilde{G}_t$ match its target, G'_t. The Ω_t matrix is any positive-definite matrix, included for generality, and chosen by the researcher. Its presence (and positive-definiteness) will force every component of $\tilde{G}_t$ to move towards its corresponding component of G'_t. It is common to just choose Ω_t as the identity matrix.

The second weight update is to the action network $A(x,z)$, given by

$$\Delta\vec{z} = \beta\sum_t\left(\frac{\partial A}{\partial\vec{z}}\right)_t\left(\left(\frac{\partial U}{\partial u}\right)_t + \left(\frac{\partial f}{\partial u}\right)_t\tilde{G}_{t+1}\right), \qquad (10)$$

where $\beta>0$ is the learning rate for the action network. This weight update aims to make the action network more "greedy", i.e., to become closer to Equation 5.

The parameter $\lambda \in [0,1]$ from Equation 7 determines how much the target is based on the critic's estimation, and how much on the actual future total trajectory cost. That is, the parameter λ specifies the degree to which an estimate is updated towards itself—what is called *bootstrapping* in reinforcement learning parlance. In practice, λ gives the modeler flexibility in adjusting the speed of learning to the dynamics of the environment. When $\lambda = 1$, the above recursion simplifies down to $G'_t = G_t = \left(\frac{\partial J}{\partial\vec{x}}\right)_t$, i.e., in this extreme the target equals the true cost-to-go of the trajectory.

Table 1 presents the on-line version of the VGL algorithm, where $e \in \mathbb{R}^{\dim(\vec{w})\times\dim(\vec{x})}$ is an eligibility trace workspace matrix and $\vec{\delta} \in \mathbb{R}^{\dim(\vec{x})}$ is a workspace vector. The derivation of this algorithm from Equations 9 and

Table 1. Actual characteristics of the fabricated module.

1. $e \leftarrow 0$
2. $t \leftarrow 0$
3. *while* $\vec{x}_t$ *not final do*
4. $\vec{u}_t \leftarrow A(\vec{x}_t, \vec{z})$
5. $\vec{x}_{t+1} \leftarrow f(\vec{x}_t, \vec{u}_t)$
6. $\vec{\delta} \leftarrow \left(\frac{\partial U}{\partial \vec{x}}\right)_t + \left(\frac{\partial A}{\partial \vec{x}}\right)_t \left(\frac{\partial U}{\partial \vec{u}}\right)_t + \gamma\left(\left(\frac{\partial f}{\partial \vec{x}}\right)_t + \left(\frac{\partial A}{\partial \vec{x}}\right)_t \left(\frac{\partial f}{\partial \vec{u}}\right)_t\right)\tilde{G}_{t+1} - \tilde{G}_t$
7. $e \leftarrow e + \left(\frac{\partial \tilde{G}}{\partial \vec{w}}\right)_t \Omega_t$
8. $\vec{w} \leftarrow \vec{w} + \alpha e \vec{\delta}$
9. $e \leftarrow \lambda\gamma e\left(\left(\frac{\partial f}{\partial \vec{x}}\right)_t + \left(\frac{\partial A}{\partial \vec{x}}\right)_t \left(\frac{\partial f}{\partial \vec{u}}\right)_t\right)$
10. $\vec{z} \leftarrow \vec{z} - \beta\left(\frac{\partial A}{\partial \vec{z}}\right)_t \left(\left(\frac{\partial U}{\partial \vec{u}}\right)_t + \gamma\left(\frac{\partial f}{\partial \vec{u}}\right)_t \tilde{G}_{t+1}\right)$
11. $t \leftarrow t + 1$
12. *end while*

10 is described in Fairbank (2014). This implementation works forwards in time and can be continually applied as trajectories are expanded. Modelers can use an alternative batch mode implementation, which is applicable to completed trajectories (Fairbank & Alonso, 2012b). Both algorithms are mathematically equivalent.

VGL is an extension of well-known methods in adaptive (a.k.a. approximate) dynamic programming, Dual Heuristic Programming and Generalized Dual Heuristic Programming in particular (Prokhorov & Wunsch, 1997; Wang, Zhange, & Liu, 2009; Werbos, 1992). VGL extends these methods, that can be seen as VGL(0) methods, with a lambda parameter, as TD(λ) does with TD(0) (Sutton & Barto, 1998). See the Appendix for a detailed argument on the relationship between TD(λ) and VGL(λ).

To summarize, we have restored optimality. If we learn the gradient of the value function and choose greedy actions that follow the full model of the system, the trajectory so built is guaranteed to be locally optimal.[3] Why is that? Can we relate this result to general laws such as the Principle of Least Action? Yes, we can—as VGL is equivalent to Pontryagin's Minimum Principle.

4 Back to the Principle of Least Action

Pontryagin's Minimum Principle (PMP) (Pontryagin, Boltyanskii, Gamkrelidze, & Mishchenko, 1962) can be seen as the application of the Principle of Least Action to optimal control. More precisely, the Hamiltonian of a control system is defined as $H(\vec{x}, \vec{p}, \vec{u})_t = L(\vec{x}, \vec{u})_t + \vec{p}_t^T f(\vec{x}, \vec{u})_t$, where L stands for the Lagrangian of the system such that, if the state given by the function represents constraints in the minimization problem, the costate $\vec{p}_t$ (also known as adjoint state or Lagrange multiplier) represents the cost of violating those constraints. In other words $\vec{p}_t$ is the rate of change of the Hamiltonian as a function of the constraint. Intuitively, the constraint f can be thought of as competing with the desired function to pull the system to its minimum or maximum (or to a steady state), and $\vec{p}_t$ can be thought of as measure of how hard f has to pull in order to make those forces balance out in the constraint surface.

In its discrete-time version (Todorov, 2006), PMP means that the following three equations must be satisfied for all time steps t of a trajectory for the trajectory to be optimal:

$$\vec{x}_{t+1} = f(\vec{x}_t, \vec{u}_t) \tag{11a}$$

$$\vec{p}_t = \left(\frac{\partial U}{\partial \vec{x}}\right)_t + \gamma\left(\frac{\partial f}{\partial \vec{x}}\right)_t \vec{p}_{t+1} \tag{11b}$$

$$\vec{u}_t = \min_{\vec{u} \in A}\left(U(\vec{x}_t, \vec{u}_t) + \gamma f(\vec{x}_t, \vec{u}_t)^T \vec{p}_{t+1}\right) \tag{11c}$$

Equation 11b specifies how the costate vectors are recursively calculated, by working backwards from the end of the trajectory. Equation 11a states how the state vector $\vec{x}_t$ is recursively calculated, by working forwards along the trajectory. If these forward and backward passes match up according to Equation 11c, then PMP is satisfied.

Bellman's and Pontryagin's principles are related by the fact that the costate vector $\vec{p}_t$ in PMP satisfies $\left(\frac{\partial J^*}{\partial \vec{x}}\right)_t \equiv \vec{p}_t$, where $J^*(\vec{x})$ is Bellman's optimal value function. However, PMP applies to local trajectories rather than to whole state spaces and avoids the "curse of dimensionality".

VGL is related to PMP in that both attempt to learn an optimal value gradient.[4] If the VGL objective is attained all along a trajectory found by a greedy policy, i.e., if $\tilde{G}_t = G_t'$ for all t, then $\tilde{G}_t$ will become equal to $\vec{p}_t$ for all t along that trajectory. This means that VGL can gain the efficiency benefits of PMP if just one trajectory is learned, i.e., meeting the VGL objective along one greedy trajectory is a *necessary* condition for the trajectory to be extremal.

Finally, note that in PMP, and likewise in VGL, in order to achieve optimality, it is not sufficient to take a fixed trajectory found by the greedy policy and fully learn the costate vectors (value-gradients) along it. By the time the costate vectors have changed, the third condition of PMP will be violated. We need to learn the costate vectors along the trajectory while *simultaneously* making the trajectory greedy with respect to those costate vectors.

5 Proof of optimality

In order to prove the optimality result we first introduce a syntactic sugar for greedy policies and

reformulate total trajectory-cost functions in terms of actions (rather than in terms of the action network's weight vector).

We define $\tilde{Q}(\vec{x},\vec{u},\vec{w}) = U(\vec{x},\vec{u}) + \gamma\tilde{J}(f(\vec{x},\vec{u}),\vec{w})$ and rewrite the greedy policy accordingly as

$$\vec{u} = \arg\min_{\vec{u}\in A}\left(\tilde{Q}(\vec{x},\vec{u},\vec{w})\right), \qquad \textit{for any } \vec{x}\in\mathbb{S} \tag{12}$$

A *greedy action* is any action $\vec{u}$ that satisfies Equation 12. A *greedy trajectory* is a trajectory in which all the actions that parameterize it are greedy actions. The function $\tilde{Q}$ is differentiable, and its derivative with respect to $\vec{u}$ is,

$$\left(\frac{\partial\tilde{Q}}{\partial\vec{u}}\right)_t = \left(\frac{\partial U}{\partial\vec{u}}\right)_t + \gamma\left(\frac{\partial f}{\partial\vec{u}}\right)_t \tilde{G}_{t+1} \tag{13}$$

The total trajectory-cost function, which depends upon a full list of actions $(\vec{u}_0,\vec{u}_1,...)$, is defined as

$$J(\vec{x}_t,\vec{u}_t,\vec{u}_{t+1},...) = U(\vec{x}_t,\vec{u}_t) + \gamma J(f(\vec{x}_t,\vec{u}_t),\vec{u}_{t+1},\vec{u}_{t+2},...) \tag{14}$$

We define a trajectory parameterized by $(\vec{x}_0,\vec{u}_0,\vec{u}_1,\vec{u}_2,...)$ to be a *locally extremal trajectory* (LET) if, for all t and all action components i,

$$\left(\frac{\partial J}{\partial\vec{u}^i}\right)_t = 0 \tag{15}$$

which is a standard sufficient condition for a stationary point.

Lemma 1. For a greedy trajectory and any fixed $\lambda\in[0,1]$, *if* $\tilde{G}_t = G'_t$ *for all* t, *then* $\tilde{G}_t = G'_t = \left(\frac{\partial J}{\partial\vec{x}}\right)_t$ *for all* t.

Theorem 1. Any greedy trajectory satisfying $\tilde{G}_t = G'_t$ *for all* t, *where actions are chosen from an unbound action space, must be locally extremal.*

Proof. As a greedy trajectory minimizes $\tilde{Q}(\vec{x},\vec{u},\vec{w})$ with respect to $\vec{u}_t$ at each time step t, and we have assumed that the action space is unbound, we know that at each t and for each action component i,

$$\left(\frac{\partial\tilde{Q}}{\partial\vec{u}^i}\right)_t = 0 \tag{16}$$

Therefore, as

$$\begin{aligned}\left(\frac{\partial J}{\partial\vec{u}}\right)_t &= \left(\frac{\partial U}{\partial\vec{u}}\right)_t + \gamma\left(\frac{\partial f}{\partial\vec{u}}\right)_t\left(\frac{\partial J}{\partial\vec{x}}\right)_{t+1} \text{(by differentiating Eq.(14))}\\ &= \left(\frac{\partial U}{\partial\vec{u}}\right)_t + \gamma\left(\frac{\partial f}{\partial\vec{u}}\right)_t \tilde{G}_{t+1} \text{(by Lemma 1)}\\ &= \left(\frac{\partial\tilde{Q}}{\partial\vec{u}}\right)_t \text{ (by Eq.(13))}\end{aligned}$$

we have $\left(\frac{\partial J}{\partial\vec{u}}\right)_t = \left(\frac{\partial\tilde{Q}}{\partial\vec{u}}\right)_t$, for all t. Therefore the consequences of a greedy trajectory, Equation 16, become equivalent to the sufficient condition of LET, Equation 15, which implies the trajectory is a LET.

Q.E.D

The above proof was applicable to unbound action spaces. An extension to bound action spaces is given by Fairbank (2014).

The conclusion of this proof is that merely learning the value-gradients and a greedy policy all along a trajectory will make the trajectory locally extremal. Simply learning these two quantities along a trajectory is sufficient to bend that trajectory into a locally extremal shape, without having needed to perform any explicit exploration. This is because the target value-gradients are vectors that "point" in the direction of the best neighboring states. When the VGL system learns those target vectors, and when coupled with a greedy policy, the system will automatically exploit better actions whenever a better action choice appears. This explanation is summarized as follows:

- The target vectors G'_t act as arrows, which point in the best direction.
- The VGL system learns those target vectors (i.e., it makes $\tilde{G}_t$ match G'_t).
- The greedy policy follows the learned vectors $\tilde{G}_t$.

Hence VGL addresses the classic "exploration versus exploitation" dilemma, in that it is possible to behave greedily and learn about better alternatives at the same time.

This is a remarkable efficiency saving for VGL as opposed to critic learning by scalar values across the state space. In those scalar methods, which include the TD learning method, the relevant optimality condition is Bellman's, which requires full knowledge (and therefore full exploration) of the whole state space.

An explicit comparison in learning speed between these two different methods shows the speed up in learning can be of several orders of magnitude when VGL methods are used (Fairbank & Alonso, 2012c), and this confirms the automatic exploration and trajectory bending of VGL.

Caveats must be highlighted in that the VGL method only addresses local exploration of the value function. In practice, it may be necessary to supplement this automatic local exploration with some explicit exploration to find a global optimum. Also, the VGL method does not address the exploration of the functions $f(x,u)$ and $U(x,u)$, which would need learning by a separate explicit exploration if they were not given a priori.

Another caveat is that it is hoped that Equation 9 will eventually converge. Whether this eventually happens is a separate issue, but convergence has been

proven in one major case by Fairbank, Alonso, and Prokhorov (2013).

Although the optimality result of this section requires a greedy policy, any approximation to a greedy policy would produce approximately optimal trajectories. The learning Equation 10 is not a greedy policy, but it is something that aims to move towards greediness, and, in practice, this is sufficient to have a robust learning system.

6 Behavioral optimal systems

Behavioral regulation theory is perhaps the most intuitive instantiation of how our approach can be applied to the study of behavior (Allison, 1983; Hanson & Timberlake, 1983; Timberlake, 1984). When behavioral systems, that is, animals, are free to act as they please, their preferred or optimal distribution of activities defines a behavioral bliss point (BBP) or baseline level of activity. In dynamic terms the BBP is a natural, steady and stable, attractor.

This view encapsulates behavioral regulation theory and generalizes the concept of homeostasis and negative feedback from cybernetics to physiology and psychology (Ashby, 1952). Physiological homeostasis keeps physiological parameters such as body temperature close to an optimal or ideal level. This level is "defended" in that deviations from the target temperature trigger compensatory physiological mechanisms that return the system to its homeostatic levels. In behavioral systems, what is defended is the animal's BBP against instrumental contingencies that create disturbances to which the system adapts. Other metaphors are possible, after all the bliss point represents an equilibrium in a population of behaviors—pretty much as the equilibrium observed in the number of different types of ants in a colony or between competing (prey–predator) species in an environment.

More specifically, operant behavior can be explained in terms of time constraints and feedback constraints, the reinforcement schedule to which the animal is subjected (Staddon, 1979). Starting from its BBP, the animal finds the optimal equilibrium between instrumental and contingent responses—the one that minimizes the cost involved. Instrumental conditioning procedures are response constraints. They disrupt the free choice of behavior and interfere with how an organism makes choices among the available responses. The instrumental conditioning procedure does not allow the animal to return to its BBP. But the organism achieves a contingent optimization by approaching its bliss point under the constraints of the instrumental conditioning procedure. Put it this way, the analysis of operant behavior is an optimal control problem and thus we should be able to express it in terms of VGL: L, the Lagrangian, can be defined as the cost to be minimized, f would model time and feedback constraints and $\vec{p}$, the multiplier or conjugate momentum, can be explicitly represented as G'. Not surprisingly, this formulation matches Staddon's term by term (Staddon, 1979).

7 Conclusions

In this paper, we have presented a meta-model of optimal behavior in the form of a learning algorithm, VGL. The main insight is that, assuming that animals use a greedy policy and that they use models to calculate the gradients of a cost-to-go function at every time step, the resulting behavior is optimal. This property follows from VGL's equivalence to PMP, a well-known optimality principle in control theory. As a meta-model, VGL does not commit to any particular model or quantity to be optimized. Instead, it establishes the conditions under which a given behavioral model will be optimal.

Let us summarize our approach: In Nature, the dynamics of a system minimizes the integral of its Lagrangian over time. This quantity is known as the action, thus systems follow the Principle of Least Action. Animals cannot be considered as passive systems rather they interact with their environment and receive feedback from it. That is, animals are control systems. Control systems follow a variation of the Principle of Least Action, PMP. This approach is alien to psychological and biological models of behavior—although see Staddon (1979) for a formulation of operant behavior as adaptation to constraints. In this paper, we bridge this gap, introduce VGL, and prove that is equivalent to PMP—from which it inherits its optimality results.

Following David Marr's Tri-Level Hypothesis (Marr & Poggio, 1976), our proposal addresses the question of how does the system do what it does, rather than what does the system do. The former refers to the algorithmic level, focusing on how learning works, and, following the PLA, how does it in an optimal way. We have provided an answer to this question. The latter, which action is minimized as described by VGL, refers to the computational level and is beyond the scope of the present study. In order to provide precise predictions, both levels are required. This paper presents a formal, rigorous framework that may help in this direction. We have intentionally kept the computational level open, so that our algorithmic proposal is not biased. Future work in this direction includes two distinct approaches: using a richer representation of payoffs, including the environment and additional aspects of an individual's behavior, metabolism, and lifetime traits, which are usually abstracted away in reinforcement learning (both in TD and VGL), on the one hand, and adapting the VGL algorithm to Darwinian fitness, on the other.

Acknowledgments

We would like to thank Professor Arto Annila and anonymous reviewers for their comments on the manuscript as well as Professor Peter Killen for his feedback on previous versions.

Notes

1. This exposition refers to the classical view of stationary state dynamics; dissipation is, of course, ubiquitous and must be taken into account in the final form of the PLA and added to the corresponding equations.
2. Originally defined in control theory by Paul Werbos (1977) as Heuristic Dynamic Programming, TD(0) was extended and presented as a model of conditioning in Barto and Sutton (1982).
3. VGL guarantees *global* optimality if applied to all trajectories.
4. In fact, our re-formulation of PMP is somehow simpler, as PMP conditions are reduced to two, namely, the costate and the min function that defines the greedy policy. PMP can be described as $\vec{u}^*(\vec{x},\vec{p}) = \arg\min_{\vec{u}\in A} H(\vec{x},\vec{p},\vec{u})$, which is a form of the greedy policy, and the adjoint equation $\vec{p}_t = \left(\frac{\partial J}{\partial \vec{x}}\right)_t \in \mathbb{R}$, VGL's gradient.

References

Allison, J. (1983). *Behavioral economics*. New York: Praeger.

Anttila, J, & Annila, A. (2011). Natural games. *Physics Letters, 375*, 3755–3761.

Ashby, W. R. (1952). *Design for a brain: The origin of adaptive behavior*. London, UK: Chapman and Hall.

Baez, J. C. (2005). *Lectures on classic mechanics*. Department of Physics and Astronomy, Louisiana State University.

Barto, A. G., & Sutton, R. S. (1982). Simulation of anticipatory responses in classical conditioning by a neuron-like adaptive element. *Behavioral Brain Research, 4*, 221–235.

Bellman, R. (1957). *Dynamic programming*. Princeton, NJ: Princeton University Press.

de Maupertuis, P. L. M. (1744). Accord de différentes lois de la nature qui avaient jusqu'ici paru incompatibles. *Mém. As. Sc. Paris*, 417.

Domjan, M. (1994). Formulation of a behavior system for sexual conditioning. *Psychonomic Bulletin & Review, 1*, 421–428.

Euler, L. (1744). *Methodus inveniendi Lineas curvas maximi minive proprietate gaudentes*. Lausanne, Switzerland: Bousquet [reprinted in Leonhardi Euleri Opera Omnia. Series 1, vol. 24. Zurich, Switzerland: Orell Fuessli, 1952].

Fairbank, M. (2014). *Value-gradient learning*. Doctoral dissertation, City University London. http://openaccess.city.ac.uk/3438/

Fairbank, M., & Alonso, E. (2012a). The divergence of reinforcement learning algorithms with value-iteration and function approximation. In *Proceedings of the IEEE International Joint Conference on Neural Networks* 2012 (IJCNN'12) (pp. 3070–3077). New York: IEEE Press.

Fairbank, M., & Alonso, E. (2012b).Value-gradient learning. In *Proceedings of the IEEE International Joint Conference on Neural Networks* 2012 (IJCNN'12) (pp. 3062–3069). New York: IEEE Press.

Fairbank, M., & Alonso, E. (2012c). A comparison of learning speed and ability to cope without exploration between DHP and TD(0). In *Proceedings of the IEEE International Joint Conference on Neural Networks* 2012 (IJCNN'12) (pp. 1478–1485). New York: IEEE Press.

Fairbank, M., Alonso, E., & Prokhorov, D. (2013). An equivalence between adaptive dynamic programming with a critic and backpropagation through time. *IEEE Transactions on Neural Networks and Learning Systems, 24*(12), 2088–2100.

Fairbank, M., Prokhorov, D., & Alonso, E. (2013). Approximating optimal control with value gradient learning. In F. L. Lewis, & D. Liu (Eds.), *Reinforcement learning and approximate dynamic programming for feedback control* (pp. 142–161). New York: Wiley–IEEE Press.

Fanselow, M. S. (1994). Neural organization of the defensive behavior system responsible for fear. *Psychonomic Bulletin & Review, 1*, 429–438.

Feynman, R. P., & Hibbs, A. R. (1965). *Quantum mechanics and path integrals*. New York: McGraw-Hill.

Goldstein, H. (1974). *Classical mechanics*. Cambridge, MA: Addison-Wesley.

Hanson, S. J. (1977). *The Rescorla–Wagner model and the temporal control of behavior*. Master's thesis, Arizona State University.

Hanson, S. J., & Timberlake, W. (1983). Regulation during challenge: A general model of learned performance under schedule constraint. *Animal Learning & Behavior, 14*, 435–442.

Herrnstein, R. J. (1961). Relative and absolute strength of responses as a function of frequency of reinforcement. *Journal of the Experimental Analysis of Behaviour, 4*, 267–72.

Kaila, V. R. I., & Annila, A. (2008). Natural selection for least action. *Proceedings of the Royal Society A, 464*, 3055–3070.

Killeen, P. R. (1992). Mechanics of the animate. *Journal of the Experimental Analysis of Behavior, 57*, 429–463.

Killeen, P. R., & Sitomer, M. T. (2003). MPR. *Behavioural Processes, 62*, 49–64.

Landau, L. D., & Lifshitz, E. M. (1975). *The classical theory of fields*. London, UK: Pergamon.

Ludvig, E. A., Bellemare, M. G., & Pearson, K. G. (2011). A primer on reinforcement learning in the brain: Psychological, computational, and neural perspectives. In E. Alonso, & E Mondragón (Eds.), *Computational neuroscience for advancing artificial intelligence: Models, methods and applications,* pp. 111–144. Hershey, PA: IGI Global.

Marr, D., & Poggio, T. (1976). *From understanding computation to understanding neural circuitry*. Artificial Intelligence Laboratory. AI Memo, MIT, AIM-357.

Pontryagin, L. S., Boltyanskii, V. G., Gamkrelidze, R. V., & Mishchenko, E. F. (1962). *The mathematical theory of optimal processes*. New York: Gordon and Breach.

Prokhorov, D., & Wunsch, D. (1997). Adaptive critic designs. *IEEE Transactions on Neural Networks*, September: 997–1007.

Skinner, B. F. (1959). *Cumulative record*. New York: Apple-Century-Crofts.

Staddon, J. E. (1979). Operant behavior as adaptation to constraint. *Journal of Experimental Psychology: General, 108*, 48–67.

Staddon, J. E. (2007). Is animal behavior optimal? In A. Bejan, & G. W. Merkx (Eds.), *Constructal theory of social dynamics*. New York: Springer.

Staddon, J. E., & Cerutti, D. T. (2003). Operant behavior. *Annual Review of Psychology, 54*, 115–144.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Taylor, E. F., & Wheeler, J. A. (2000). *Exploring black holes: An introduction to general relativity*. New York: Addison-Wesley Longman.

Timberlake, W. (1984). Behavior regulation and learned performance: Some misapprehensions and disagreements. *Journal of the Experimental Analysis of Behavior, 41*, 355–375.

Timberlake, W., & Silva, K. M. (1995). Appetitive behavior in psychology, ethology, and behavior systems. In N. Thompson (Ed.), *Perspectives in ethology: Vol. 11. Behavioral design* (pp. 212–254). New York: Plenum.

Todorov, E. (2006). Optimal control theory. In K. Doya, S. Ishii, A. Pouget, & R. P. N. Rao (Eds.), *Bayesian brain: Probabilistic approaches to neural coding* (pp. 269–298). Cambridge, MA: MIT Press.

Wang, F.-Y., Zhange, H., & Liu, D. (2009). Adaptive dynamic programming: An introduction. *IEEE Computational Intelligence Magazine*, 39–47.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. Ph.D. thesis, Cambridge University.

Werbos, P. J. (1977). Advanced forecasting for global crisis warning and models of intelligence. In *General Systems Yearbook*.

Werbos, P. J. (1992). Approximating dynamic programming for real-time control and neural modeling. In D. White, & D. A. Sofge (Eds.), *Handbook of intelligent control* (pp. 493–525). New York: Van Nostrand Reinhold.

Wörgötter, F., & Porr, B. (2005). Temporal sequence learning, prediction and control—A review of different models and their relation to biological mechanisms. *Neural Computation, 17*, 245–319.

Appendix

Temporal Difference (TD) is a special case of VGL. TD is model-free VGL, that is, VGL without requiring explicit knowledge of Equations 1 and 2. In such cases, instead of the gradient of the cost function, animals would learn only the cost function. As a consequence, in order to meet Bellman's condition and achieve optimality, animals would need to explore the whole state space. This is an unrealistic assumption in most biologically relevant cases, where the state space is too large. VGL, on the other hand, works with models and, in so doing, learns local optimal trajectories without extensive exploration. The TD algorithm can be summarized in the following weights update rule:

$$\Delta\vec{w} = \alpha \sum_t \left(\frac{\partial \tilde{J}}{\partial \vec{w}}\right)_t \left(J'_t - \tilde{J}_t\right) \tag{17}$$

where the target J' is equivalent to Watkin's "λ-Return" (Watkins, 1989). If we compare Equation 9 against its TD equivalent, Equation 17, we see that they are analogous except for the introduction of the model. This may give the wrong impression that VGL(λ) is just a differentiated form of TD(λ). Indeed, whereas TD(λ) attempts to learn the values of the Bellman equation, VGL(λ) attempts to learn the gradient of these values. However, if the VGL weight update is at a fixed point at every time step along a trajectory generated by a greedy policy, for any λ, then that trajectory is locally extremal, and often locally optimal. This contrasts to TD methods in that it is possible for the TD weight update to be at a fixed point at every time step along a trajectory generated by a greedy policy, without the trajectory being optimal. VGL methods have a much lesser requirement for exploration than TD methods do, as the local part of exploration comes for free with them. What we mean by this is that provided the VGL learning algorithm makes progress towards achieving $\tilde{G}_t = G'_t$ all along a greedy trajectory, then provided the trajectory remains greedy (through continued updating of the actor or use of the greedy policy), it will make progress in *bending* itself towards a locally optimal shape, and this will happen without the need for any stochastic exploration. In comparison, the failure of TD without any exploration in a deterministic environment is dramatic and common, even when the value-function is perfectly learned along a single trajectory.

It is worth highlighting that in order to compute VGL it is not enough to use the derivatives of the values. This is what the Hamilton–Jacobi–Bellman equation does in extending the Bellman equation to continuous state spaces. However, such derivation does not exploit fully the information contained in gradient-values. We cannot just consider the change in J over the steps *along* a trajectory. This is like forming the scalar product of $\frac{\partial J}{\partial \vec{x}}$ with $\Delta\vec{x}$, which will lose the *sideways* components of the derivative, those that are not parallel to $\Delta\vec{x}$. In VGL, *all* components are used in the calculation of the target. One needs to know the model functions in order to calculate a target value gradient, and one needs a target in order to do a weight update.

In addition, the model functions are relevant to the greedy policy. Using a first-order expansion of the greedy policy gives

$$\begin{aligned}
\vec{u} &= \arg\min_{\vec{u}\in A}\left(U(\vec{x},\vec{u}) + \gamma\tilde{J}(f(\vec{x},\vec{u}),\vec{w})\right) \\
&\approx \arg\min_{\vec{u}\in A}\left(U(\vec{x},\vec{u}) + \gamma\tilde{J}(\vec{x},\vec{w}) + \gamma\left(\frac{\partial\tilde{J}(\vec{x},\vec{w})}{\partial\vec{x}}\right)^T\left(f(\vec{x},\vec{u}) - \vec{x}\right)\right) \\
&\approx \arg\min_{\vec{u}\in A}\left(U(\vec{x},\vec{u}) + \gamma\left(\frac{\partial\tilde{J}(\vec{x},\vec{w})}{\partial\vec{x}}\right)^T f(\vec{x},\vec{u})\right)
\end{aligned} \tag{18}$$

We see that the greedy policy depends on the value-gradient but not on the values themselves.

Thus, the VGL algorithm is doubly model-based—once for the calculation of G', and once for the calculation of the greedy policy (or weight update of the actor). Changing $\frac{\partial J}{\partial \vec{x}}$ will immediately affect the greedy policy, and, by moving it towards its correct target, we will steer the trajectory in the correct (locally optimal) direction. Hence, TD's paradigm "exploration *vs.* exploitation" becomes "exploration and exploitation" or, in other words, exploration comes for free when we combine greedy and gradient.

AQ1