Momentum and mood in policy-gradient reinforcement learning

Daniel Bennett
Princeton Neuroscience Institute
Princeton University
Princeton, NJ 08544
daniel.bennett@princeton.edu

Guy Davidson
College of Computational Sciences
Minerva Schools at KGI
San Francisco, CA 94103
guy@minerva.kgi.edu

Yael Niv
Princeton Neuroscience Institute and Department of Psychology
Princeton University
Princeton, NJ 08544
yael@princeton.edu

Abstract

Policy-gradient reinforcement learning (RL) algorithms have recently been successfully applied in a number of domains. In spite of this success, however, relatively little work has explored the implications of policy-gradient RL as a model of human learning and decision making. In this project, we derive two new policy-gradient algorithms that have implications as models of human behaviour: TD(λ) Actor-Critic with Momentum, and TD(λ) Actor-Critic with Mood. For the first algorithm, we review the concept of momentum in stochastic optimization theory, and show that it can be readily implemented in a policy-gradient RL setting. This is useful because momentum can accelerate policy gradient RL by filtering out high-frequency noise in parameter updates, and may also confer a degree of robustness against convergence to local maxima in reward. For the second algorithm, we show that a policy-gradient RL agent can implement an approximation to momentum in part by maintaining a representation of its own mood. As a proof of concept, we show that both of these new algorithms outperform a simpler algorithm that has neither momentum nor mood in a standard RL testbed, the 10-armed bandit problem. We discuss the implications of the mood algorithm as a model of the feedback between mood and learning in human decision making.

Keywords: actor-critic, policy gradient, momentum, mood

1 Introduction

RL algorithms can be divided into three broad classes: value-based, policy-based, and actor-critic. Algorithms in these classes are distinguished by whether they learn just a value function, and then use this value function to generate a policy (value-based algorithms), learn only a policy function and no value function (policy-based algorithms), or learn both a value function and an independent policy function (actor-critic algorithms). In psychology and neuroscience, recent studies of learning and decision making have tended to focus on the processes by which humans and other animals learn a value function. By contrast, comparatively little work has explored the implications for human learning and decision making of the different methods of policy improvement employed by policy-based and actor-critic algorithms, despite some evidence that human learning is consistent with policy updating rather than value updating [1].

Policy-gradient RL is a well-studied family of policy improvement methods that uses feedback from the environment to estimate the gradient of reinforcement with respect to the parameters of a differentiable policy function [2, 3]. This gradient is then used to adjust the parameters of the policy in the direction of increasing reinforcement. In recent years, RL algorithms that incorporate a policy-gradient component have been successfully applied to real-time robot control in a continuous action space [4], to Atari computer games [5], and to the board game Go [6].

Because it uses the gradient of reinforcement to update a policy, policy-gradient RL can be thought of as a form of stochastic gradient descent (SGD). As a result, theoretical advances in stochastic optimization theory concerning SGD methods can be applied directly to policy-gradient RL algorithms. For instance, it is known that SGD convergence can be greatly accelerated by a *momentum* term [7, 8, 9]. Here, we show that adding a momentum term to policy-gradient RL improves performance in a standard testbed, the 10-armed bandit problem, and we explore the implications of momentum as a model of phenomena in human learning and decision making. Specifically, we show that a mood variable—defined as an exponential moving average of reward prediction errors [10]—can be used to help approximate a momentum update without ever computing a momentum term explicitly.

2 Theoretical framework and background

2.1 Gradient descent and momentum

Gradient descent tries to find the parameters θ that minimize an objective function $J(\theta)$ by incrementally updating θ in the direction of $\nabla_\theta J(\theta)$, the gradient of J with respect to θ. The step size is controlled by a learning rate η:

$$\begin{aligned} u_t &= \eta \nabla_\theta J(\theta) \\ \theta_{t+1} &= \theta_t - u_t \end{aligned} \tag{1}$$

In many cases it may be computationally infeasible to calculate $\nabla_\theta J(\theta)$. *Stochastic* gradient descent therefore uses an estimated gradient $\nabla_\theta \widetilde{J}(\theta)$, typically calculated as the mean of a mini-batch of training samples. In practice, a *momentum* term [7, 8] is often also used to help overcome two distinct but related limitations of SGD. These limitations are, first, that SGD considers only the the slope of the objective function (first derivative) and not its curvature (second derivative). This can cause difficulty in the presence of 'ravines' in the objective function where the curvature of J differs with respect to different dimensions of θ [11]. Second, SGD's use of an estimated rather than an exact gradient increases the variance of parameter updates: although $\nabla_\theta \widetilde{J}(\theta)$ is equal to $\nabla_\theta J(\theta)$ in expectation, at any single time-step unsystematic error in gradient estimation can lead to suboptimal parameter updates.

Momentum addresses both of these limitations by updating parameters according to both the estimated gradient of J at the current θ and a proportion m of the parameter update at the previous timestep: $u_t = \eta \nabla_\theta \widetilde{J}(\theta) + m u_{t-1}$. In this way, momentum effectively implements a low-pass filter on parameter updates. This filtering resolves the limitations of SGD described above, because both oscillations resulting from differential curvature of J and unsystematic error in gradient estimation slow SGD by introducing high-frequency noise to parameter updates. For this reason, momentum has long been used in machine learning, especially in training neural networks by backpropagation [9].

2.2 Policy-gradient reinforcement learning

A policy-gradient RL algorithm performs gradient *ascent* on an objective function that evaluates its policy under current parameters (e.g., the expected future reward under the current policy). This is achieved by updating the parameters θ of a differentiable policy π_θ [2, 3] in the direction of the gradient of the objective function with respect to θ.

In this project we sought to explore the consequences of adding a momentum term to a policy-gradient RL algorithm. Our starting point was *TD(λ) Actor-Critic*, which implements policy-gradient RL via a form of advantage updating [4, 5]:

TD(λ) Actor-Critic. The critic learns a state-value function V using a learning rate α, and provides a scalar reward prediction error δ to the actor at each timestep. In turn, the actor uses this reward prediction error to improve its policy by updating θ in the direction of the product of δ and the accumulating eligibility trace e_t.

$\delta_t = r + V_{t-1}(s') - V_{t-1}(s)$	Calculate reward prediction error
$V_t(s) = V_{t-1}(s) + \alpha\delta_t$	Update state value
$e_t = \lambda e_{t-1} + \nabla_\theta \log_{\pi_\theta}(s, a)$	Increment eligibility trace
$u_t = \eta e_t \delta_t$	Calculate parameter update
$\theta_{t+1} = \theta_t + u_t$	Update parameters

TD(λ) Actor-Critic makes use of a variable called the score function: $\nabla_\theta \log \pi_\theta(s, a)$. This quantity is a vector defined as the gradient of the log policy with respect to θ. It quantifies how the policy would change with changes in the different entries in θ. Then, given a positive or a negative reward prediction error, this vector can be used to adjust the policy appropriately [2, 5]. The score function is aggregated over time into the eligibility vector e, subject to an eligibility decay parameter λ. When λ is greater than 0, this permits the algorithm to assign credit for rewards received at the current timestep to actions taken at previous timesteps [4].

3 Momentum in policy-gradient reinforcement learning

3.1 TD(λ) Actor-Critic with Momentum

Since the algorithm described in Section 2.2 implements a form of SGD, it is amenable to improvement using the momentum principle described in Section 2.1. Specifically, we can add momentum to policy-gradient RL by augmenting the update u_t from TD(λ) Actor-Critic with a proportion m of the update from the previous timestep u_{t-1}: $u_t = \eta e_t \delta_t + m u_{t-1}$. This produces a new algorithm, *TD(λ) Actor-Critic with Momentum*.

In addition to helping stabilise learning by filtering out high-frequency noise in parameter updates, another potential advantage of momentum in policy gradient RL is that it may help the algorithm to find global rather than local maxima of reinforcement, or at least to find better local maxima. A limitation of SGD in general is that it is guaranteed only to converge to local optima; this can be especially problematic in RL environments, which are often characterised by non-convex objective functions. In such settings, adding momentum to a policy-gradient RL algorithm might serve to propel the algorithm *past* poor local maxima of reward, and thereby help to produce better overall policies at convergence.

3.2 TD(λ) Actor-Critic with Mood

In animal learning and decision making, one potential impediment to the use of a momentum term is that momentum requires the algorithm to have access to u_{t-1}, the vector of parameter updates at the previous timestep. We aim to show here that a reasonable approximation to u_{t-1} can be constructed using a moving average of reward prediction errors. We are interested in this moving average because of its psychological interpretation as a *mood* variable [10]. Specifically, we follow [10] in defining a mood variable h_t that is recursively updated via a simple error-correcting rule (delta rule) with learning rate η_h and the current prediction error δ_t as a target:

$$h_t = h_{t-1} + \eta_h (\delta_t - h_{t-1}) = \eta_h \sum_{\tau=0}^{t-1} [(1 - \eta_h)^\tau \delta_{t-\tau}] \tag{2}$$

Next, we can unroll the definition of the momentum term and rewrite it as a sum of the products of eligibility traces and prediction errors at previous timesteps:

$$m u_{t-1} = m\eta e_{t-1}\delta_{t-1} + m^2 \eta e_{t-2}\delta_{t-2} + m^3 \eta e_{t-3}\delta_{t-3} + \ldots + m^{t-1}\eta e_1 \delta_1 = \eta \sum_{\tau=1}^{t-1} [m^\tau e_{t-\tau}\delta_{t-\tau}] \tag{3}$$

We now seek to show that this sum of products can be approximated by a moving average of reward prediction errors (that is, by the mood variable h). To this end, we first approximate the past eligibility traces $e_{t-\tau}$ from Equation 3 with the most recent eligibility trace e_{t-1} and move this term outside the sum. Then, by setting the learning rate η_h from Equation 2 to $1 - m$, the mood variable h becomes proportional to the approximated sum in Equation 3. Consequently, mood from trial $t - 1$ can be used to approximate momentum at trial t:

$$m u_{t-1} \approx \eta e_{t-1} \sum_{\tau=1}^{t-1} [m^\tau \delta_{t-\tau}] \approx \eta e_{t-1} \frac{m}{1-m} h_{t-1} \tag{4}$$

Finally, since the approximate momentum update in Equation 4 depends only on quantities available at trial $t-1$, it can be applied in advance at the end of trial $t-1$, rather than waiting until the end of trial t (cf. Nesterov momentum [8]). This produces the *TD(λ) Actor-Critic with Mood* algorithm, which uses a mood variable to help approximate momentum:

TD(λ) Actor-Critic with Mood. The general structure of TD(λ) Actor-Critic is retained, except that a mood variable is calculated on each trial and used to bias parameter updates according to the recent history of reward prediction errors.

$$\delta_t = r + V_{t-1}(s') - V_{t-1}(s) \quad \text{Calculate reward prediction error}$$
$$V_t(s) = V_{t-1}(s) + \alpha\delta_t \quad \text{Update state value}$$
$$h_t = h_{t-1} + (1-m)(\delta_t - h_{t-1}) \quad \text{Update mood}$$
$$e_t = \lambda e_{t-1} + \nabla_\theta \log_{\pi_\theta}(s,a) \quad \text{Increment eligibility trace}$$
$$u_t = \eta e_t \left[\delta_t + \frac{m}{1-m} h_t\right] \quad \text{Calculate parameter update}$$
$$\theta_{t+1} = \theta_t + u_t \quad \text{Update parameters}$$

4 Simulation results

Above, we described one extant policy-gradient RL algorithm (TD(λ) Actor-Critic), and two novel algorithms (TD(λ) Actor-Critic with Momentum, and TD(λ) Actor-Critic with Mood). Here, we assess the simulated performance of these three algorithms in a standard reinforcment learning testbed: the 10-armed bandit problem. Our goal is to determine whether either momentum or mood-approximated momentum help to accelerate learning in this setting.

In the 10-armed bandit problem, the agent can choose from among 10 choice options ('arms'), each of which is characterised by a different mean payout drawn from a unit normal. An agent's task in this environment is to choose arms that maximise the amount of reward that it receives.

We implemented the three algorithms with a softmax policy parameterised by the vector θ, which has length equal to the number of choice options, where the i-th entry of θ denotes strength of preference for the i-th choice option. These preferences can be thought of as analogous to Q-values, in that they denote some measure of the subjective utility of choosing different options; unlike Q-values, however, preferences for different options are not interpretable in terms of expected future return. Each choice option is represented by the feature vector $\phi(a)$, which is a one-hot vector (all 0 except for the entry corresponding to a, which is 1). The exact form of the policy is as below, where A is the set of bandit arms:

$$\pi_\theta(a) = \frac{e^{\phi(a)^T\theta}}{\sum_{\hat{a}\in A} e^{\phi(\hat{a})^T\theta}} \tag{5}$$

With this policy parameterisation, the score function can be expressed in terms of ϕ:

$$\nabla_\theta \log \pi_\theta(a) = \phi(a) - \mathbb{E}_{\pi_\theta}[\phi(\cdot)] \tag{6}$$

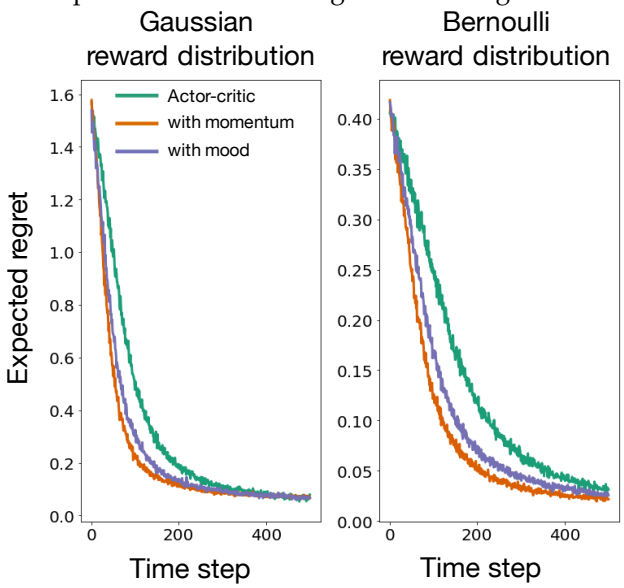

Figure 1: Learning curves (quantified by expected regret averaged across 2000 simulations of 500 timesteps each) on two variants of the 10-armed bandit testbed for three algorithms: TD(λ) Actor-Critic (green), TD(λ) Actor-Critic with Momentum (orange), and TD(λ) Actor-Critic with Mood (purple). Left: testbed with Gaussian payout distribution (payout standard deviation = 1). Right: testbed with Bernoulli payout distribution. In both settings, an algorithm with momentum performs best, followed by an algorithm with mood, followed by an algorithm with neither.

Figure 1 displays the performance of the three algorithms for both Gaussian and Bernoulli payout distributions. Parameters for simulation are: $\lambda = 0.1$, $\gamma = 1$, $\alpha = 0.01$, $\eta = 0.1$, $m = 0.5$. From these results, we can make three primary observations. First, it is clear that a moderate degree of momentum ($m = 0.5$, orange line) accelerates learning performance relative to an equivalent algorithm without momentum (green). Second, TD(λ) Actor-Critic with Mood (purple line), which uses a mood term to approximate momentum, captures much of the benefit of momentum without requiring an explicit representation of previous parameter updates. Third, the relative benefits of momentum are stronger for a Bernoulli payout distribution than a Gaussian. This is because Bernoulli payouts have greater variance than Gaussian payouts, such that individual pieces of feedback are less informative regarding the true gradient of reinforcement. As in SGD, in this context a momentum term allows the algorithm to reduce the variance of updates to the parameters of its policy, and therefore to learn more quickly.

5 Conclusions

It is common practice in optimization by SGD to use a momentum term to accelerate convergence [7, 8, 9]. Here, we provide a proof-of-concept result showing that momentum can also be used to accelerate policy-gradient RL. The underlying reason for this is that momentum helps overcome two limitations of steepest-ascent policy-gradient RL methods: first, momentum can help prevent parameter oscillation in settings where performance is more sensitive to small changes in some parameters than in others (e.g., driving in an urban environment, performance is much more sensitive to small differences in the angle of the wheels than small differences in speed). Second, for on-line RL algorithms in stochastic environments, individual rewards or state transitions may provide very noisy estimates of the true gradient of reinforcment with respect to the parameters of the policy. For instance, a single reward from a bandit that pays out probabilistically gives only a high-variance sample of its underlying reward probability. By averaging parameter updates across multiple points in time, momentum acts as a low-pass filter on this high-variance quantity, and therefore allows parameters to be updated using a more stable (and more accurate) gradient estimate.

We also show that a momentum term can be reasonably well approximated in the policy-gradient RL setting by a *mood* variable, where mood is defined as an exponential moving average of reward prediction errors [10]. This derivation may shed light upon the role of mood in human and animal learning and decision making. For instance, [10] found evidence for a mood-congruent interaction between mood and RL in human participants, such that participants who self-reported high levels of mood instability tended over-value stimuli that they had encountered in a positive mood, and under-value stimuli encountered in a negative mood. A mood model such as TD(λ) Actor-Critic with Mood provides one explanation for this finding, because this algorithm approximates momentum by updating its policy parameters at each timestep in the direction of the *sum* of the current reward prediction error *and* current mood. The effect of this is to boost preferences for stimuli encountered when mood is positive, as [10] observed in participants high in mood instability.

More broadly, the fact that RL algorithms are improved by the addition of either momentum or mood is a reflection of an interesting general property of learning: if an agent consistently receives positive reward prediction errors, this can often be taken as a sign that the policy the agent has lately been following ought to be reinforced. By contrast, the reverse is true for consistent negative reward prediction errors, which might indicate the necessity of altering the current policy. The two policy-gradient RL algorithms that we have derived in this project take advantage of this general property of learning. In the case of TD(λ) Actor-Critic with Momentum, this representation is made explicit in the form of a momentum term; for TD(λ) Actor-Critic with Mood, it is accomplished implicitly by the use of mood to help approximate momentum.

References

[1] J. Li and N. D. Daw, "Signals in human striatum are appropriate for policy update rather than value prediction," *The Journal of Neuroscience*, vol. 31, no. 14, pp. 5504–5511, 2011.

[2] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, 1992.

[3] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*, pp. 1057–1063, 2000.

[4] T. Degris, P. M. Pilarski, and R. S. Sutton, "Model-free reinforcement learning with continuous action in practice," in *American Control Conference (ACC), 2012*, pp. 2177–2182, IEEE, 2012.

[5] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, pp. 1928–1937, 2016.

[6] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, Jan. 2016.

[7] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[8] Y. E. Nesterov, "A method for solving the convex programming problem with convergence rate $O(\frac{1}{K^2})$," *Dokl. Akad. Nauk SSSR*, vol. 269, pp. 543–547, 1984.

[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, p. 533, 1986.

[10] E. Eldar and Y. Niv, "Interaction between emotional state and learning underlies mood instability," *Nature Communications*, vol. 6, p. 6149, 2015.

[11] R. S. Sutton, "Two problems with back propagation and other steepest descent learning procedures for networks," in *Proceedings of the 8th Annual Conference of the Cognitive Science Society, 1986*, pp. 823–832, 1986.