Incremental Estimation of Natural Policy Gradient with Relative Importance Weighting

Ryo IWAKI\(^{(a)}\), Nonmember, Hiroki YOKOYAMA\(^{(b)}\), and Minoru ASADA\(^{(c)}\), Members

SUMMARY  The step size is a parameter of fundamental importance in learning algorithms, particularly for the natural policy gradient (NPG) methods. We derive an upper bound for the step size in an incremental NPG estimation, and propose an adaptive step size to implement the derived upper bound. The proposed adaptive step size guarantees that an updated parameter does not overshoot the target, which is achieved by weighting the learning samples according to their relative importances. We also provide tight upper and lower bounds for the step size, though they are not suitable for the incremental learning. We confirm the usefulness of the proposed step size using the classical benchmarks. To the best of our knowledge, this is the first adaptive step size method for NPG estimation.

key words: reinforcement learning, natural policy gradient, adaptive step size

1. Introduction

The natural policy gradient (NPG) method, originally proposed by Kakade [1], is one of the techniques used in reinforcement learning (RL) for finding the (locally) optimal policy by gradient ascent. By using the natural gradient, the plateau phenomenon can be avoided [2].

Several algorithms [3]–[7] have been proposed to estimate the NPG incrementally, whose computation and memory usage per time step scale linearly with the dimensionality of the policy parameter. However, the incremental estimation of the NPG itself requires a considerable sample size and is very sensitive to the values of meta-parameters such as step sizes. Slow convergence and instability are obstacles to the incremental NPG methods being applied to real problems.

Recently, RL agents, whose value functions and policies are parameterized using deep neural networks, were shown to give remarkable performance [8], [9]. The optimization of these networks is a highly non-linear problem, which makes adaptive step size methods, such as AdaGrad and ADAM [11], essential for the learning process. In previous RL studies, various adaptive step size methods were proposed for policy gradient algorithms [12], [13] and for conventional policy evaluation algorithms [15].

On the other hand, the NPG is estimated using a linear function approximation and a squared loss function. Though general adaptive step size strategies such as AdaGrad and ADAM are applicable to NPG estimation, performance and stability can be further improved by using an adaptive step size method that is specific to NPG estimation. To the best of our knowledge, despite its importance, such a special adaptive step size method for NPG estimation has not been proposed to date.

The purpose of this paper is to propose an adaptive step size strategy for NPG methods that can achieve both rapid convergence and stability in learning. We focus on the Natural policy gradient utilizes the Temporal Differences (NTD) algorithm [3], which uses the eligibility traces to solve a linear regression problem. First, we derive the upper bound of the step size for NPG estimation in the NTD algorithm. We also provide tight upper and the lower bounds for the step size, though they are not suitable for the incremental learning. Second, extending the approach of Karampatziakis and Langford [16], we propose an adaptive step size method for general linear regression using the trace of the feature vector, which guarantees that an updated parameter does not overshoot the target. This is achieved by weighting the learning samples according to their relative importances. The proposed adaptive step size determines the “aggressiveness” of the update from a given meta-parameter. The proposed adaptive step size and the derived bounds are generally applicable to the existing incremental NPG methods, because a small modification to the NTD algorithm yields other incremental NPG algorithms. We evaluate the validity of the proposed adaptive step size using classical benchmarks.

2. Preliminary

This section provides an overview of RL and the NPG method.

2.1 Notation of Markov Decision Process

We assume that the problem is a Markov decision process (MDP). A MDP is specified by a tuple \((S, A, P, R)\), where \(S\) represents the set of possible states of the environment and \(A\) is the set of possible actions that the agent can choose, both of which could be discrete or continuous. \(P\) and \(R\) denote the state transition probability and the bounded reward
function, respectively. In the case of a model-free RL, the agent has no knowledge of \( \mathcal{P} \) and \( \mathcal{R} \).

For each discrete time step \( t \in \mathbb{N} \), the agent observes a state \( s_t \in S \) and chooses an action \( a_t \in A \). The state of the environment changes to the next state \( s_{t+1} \) according to \( \mathcal{P}(s_{t+1} = s' | s_t = s, a_t = a) \), and the agent receives the reward \( r_t \in \mathbb{R} \) according to \( \mathcal{R}_t \). Decision making by the agent is characterized by a parameterized stochastic policy \( \pi(a | s, \theta) \). In the case of a model-free RL, the reward of the initial state: \( \mathbb{E}[r(s_0, a)] = \mathbb{E}[r_t | s_t = s, a_t = a] \).

Decision making by the agent is characterized by a parameterized stochastic policy \( \pi(a | s, \theta) \). We assume that \( \pi(a | s, \theta) \) is differentiable with respect to \( \theta \) for all \( s \) and \( a \). We adopt the shorthand notation: \( \pi_\theta \triangleq \pi(a | s, \theta) \). We assume that the underlying Markov chain is irreducible and aperiodic. Thus, the stationary distribution of the initial state: \( d^\pi(s) = \lim_{n \to \infty} \Pr(s_n = s | \theta_0) \), \( \forall s' \in S \).

For each policy \( \pi_\theta \), the state value function \( V_\pi(s) \) and the state-action value function \( Q^\pi(s, a) \) are given by

\[
V_\pi(s) = \mathbb{E}_{\pi_\theta} \left[ \sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s \right] ,
\]

and

\[
Q^\pi(s, a) = \mathbb{E}_{\pi_\theta} \left[ \sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a \right] ,
\]

respectively, where \( \gamma \in [0, 1) \) is the discount factor. The purpose of the agent is to find the (locally) optimal policy parameter \( \theta^* \) that maximizes the average reward:

\[
J(\theta) \triangleq \lim_{T \to \infty} \frac{1}{T} \mathbb{E}_{\pi_\theta} \left[ \sum_{t=0}^{T-1} r_t \right] = \sum_s d^\pi(s) \sum_a \pi(a | s; \theta) \mathcal{R}_t^a .
\]

### 2.2 Natural Policy Gradient

Let \( f_w(s, a) \) be a linear function approximator given by

\[
f_w(s, a) \triangleq w^\top \psi(s, a) , \tag{1}
\]

where \( |w| = |\theta| \), and \( \psi(s, a) = \nabla_\theta \ln \pi(a | s; \theta) \) is a basis function, called the characteristic eligibility. The approximator \( f_w(s, a) \) is compatible in the sense that

\[
\nabla_w f_w(s, a) = \nabla_\theta \ln \pi(a | s; \theta) \Rightarrow \frac{\nabla_\theta \pi(a | s; \theta)}{\pi(a | s; \theta)} .
\]

Consider the following equation:

\[
\mathbb{E}_\theta [(Q^\pi(s, a) - b(s) - f_w(s, a)) \nabla_w f_w(s, a)] = 0 , \tag{2}
\]

where \( \mathbb{E} [\cdot] \) denotes an expectation over the state-action pair under the current policy \( \pi_\theta \); namely, for an arbitrary variable \( x \), we have

\[
\mathbb{E}_\theta [x] \triangleq \sum_s d^\pi(s) \sum_a \pi(a | s; \theta) x .
\]

Here, \( b(s) \) is a state-dependent arbitrary function called the baseline. Then, when Eq. (2) holds, the policy gradient theorem with function approximation ensures that the following equation holds [3], [7], [17]:

\[
\nabla_\theta J(\theta) \approx \mathbb{E}_\theta [\nabla_\theta \ln \pi(a | s; \theta) f_w(s, a)] . \tag{3}
\]

Thus, the vanilla policy gradient (the left hand side of Eq. (3)) can be estimated by approximating \( Q^\pi(s, a) \), projected onto the subspace spanned by \( \nabla_\theta \ln \pi(a | s; \theta) \). The appropriate choice of baseline \( b(s) \) reduces the variance of Eq. (3). The variance is minimized if \( b(s) = V^\pi(s) \). In this case, the target of \( f_w(s, a) \) is the advantage function [18]:

\[
A^\pi(s, a) \triangleq Q^\pi(s, a) - V^\pi(s) .
\]

Furthermore, substituting Eq. (1) into Eq. (3) yields

\[
\nabla_\theta J(\theta) = G(\theta) w ,
\]

where \( G(\theta) \) is the Fisher information matrix of the policy weighted by the stationary state distribution:

\[
G(\theta) \triangleq \mathbb{E}_\theta [\nabla_\theta \ln \pi(a | s; \theta) \nabla_\theta \ln \pi(a | s; \theta)^\top] .
\]

Thus, the natural policy gradient (NPG) [1] is given by

\[
G^{-1}(\theta) \nabla_\theta J(\theta) = w . \tag{4}
\]

Intuitively, \( G(\theta) \) measures the distance on the probability manifold determined by \( \theta \); the NPG (the left hand side of (4)) indicates the direction of the steepest gradient on this manifold. Furthermore, Eq. (4) states that the parameter \( w \) of the compatible function approximator \( f_w(s, a) \) must encode the NPG directly when Eq. (2) holds.

### 2.3 Incremental NPG Estimation with Function Approximation

Several algorithms [3]–[7], [19], [20] have been proposed to estimate \( w \) which satisfies Eq. (2). Our study focuses on the Natural policy gradient utilizing the Temporal Differences (NTD) algorithm proposed by Morimura et al. [3], [4]. However, the adaptive step size and the bounds we propose are generally applicable to the incremental NPG estimation methods above, because small modifications to the NTD algorithm yield other incremental NPG algorithms.

The NTD algorithm approximates the advantage in terms of the regression of the temporal difference (TD) error, based on the following proposition [3], [19]:

**Proposition 1:** The expected TD error is equal to the advantage function:

\[
\mathbb{E}_{\pi, \theta} [\delta^\pi | s, a] = A^\pi(s, a) .
\]

**Proof:** The TD error \( \delta^\pi \) is defined by
\[ \delta_t^\pi = r_t + \gamma V(s_{t+1}) - V(s_t). \] (5)

Thus, the expectation of \( \delta_t^\pi \) in the state-action space yields
\[
\mathbb{E}_{\pi, s, \mathcal{A}} [\delta_t^\pi | s, a] = \mathbb{E}_{\pi, s, \mathcal{A}} [r_t + \gamma V(s_{t+1}) - V(s_t) | s, a] = R_t + \gamma \sum_{s', a'} \mathcal{P}_{s,a} V'(s') - V(s) = Q^\pi(s, a) - V^\pi(s) = A^\pi(s, a).
\]

In order to estimate \( w \) which satisfies Eq. (2) based on Proposition 1, the TD error must be computed from the true state-value function \( V^\pi(s) \) as in Eq. (5). However, \( V^\pi(s) \) cannot be utilized in general. In the forward TD(1) algorithms, the Monte Carlo return, \( R_t = \sum_{t=0}^{\infty} \gamma^t r_{t+\tau} \), can be used off-line, as an unbiased estimate of \( V^\pi(s) \). On the other hand, the TD(\( \lambda \)) algorithms use \( R_t \) approximately on-line\(^7\), in the form of the backward view using the eligibility trace [22].

In the framework of the backword TD(\( \lambda \)), the TD error \( \delta_t^\pi \) is approximated by \( \delta_t^w \), which is computed from the approximated value function \( V(s) \) characterized by the parameter \( v \in \mathbb{R}^m \). The estimation of \( w \) is also improved by the eligibility trace. The overall backward updates in the NTD algorithm are
\[
\delta_t^w = r_t + \gamma V(s_{t+1}) - V(s_t),
\]
\[
\delta_t^w = \delta_t^w - f_{w}(s_t, a_t),
\]
\[
e^w_t \leftarrow \gamma_{t} \lambda_{t} e^w_t + \nabla_v V(s_t),
\]
\[
e^w_t \leftarrow \gamma_{t} \lambda_{t} e^w_t + \psi(s_t, a_t),
\]
\[
v_t \leftarrow v_t + \alpha_{t} \delta_t^w e^w_t,
\]
\[
w_t \leftarrow w_t + \alpha_{t} \delta_t^w e^w_t,
\]
\[
\theta_t \leftarrow \theta_t + \alpha_{t} \psi_t e^w_t,
\]
where \( e^v_t \) and \( e^w_t \) are the eligibility traces for the value and advantage, respectively, and \( \lambda \in [0, 1] \) is the decay factor of the traces. Theoretically, the step sizes, \( \alpha_v \), \( \alpha_w \) and \( \alpha^\theta \), must be decreasing positive values that satisfy \( \sum_{t=0}^{\infty} \alpha_t = \infty \) and \( \sum_{t=0}^{\infty} \alpha_t^2 < \infty \) to guarantee convergence [23]. In practice, however, the step size is usually a fixed small value because the RL problem is not static. In other words, the change of policy shifts the distribution of the state-action pair. This study focuses on the method for determining \( \alpha_w \).

3. Incremental Estimation of the NPG with Relative Importance Weighting

In this section, we derive an adaptive step size method for the incremental NPG estimation. First, in Sect. 3.1, we derive an upper bound \( \alpha_{w}^\pi \) that avoids oscillation and divergence. This upper bound directly encodes a local optimum \( w_{local}^\pi \) for a given single sample. Next, in Sect. 3.2, in order to apply the derived upper bound, we propose an adaptive step size method, whereby the local optimum is approached gradually by weighting the learning sample according to its given relative importance. The proposed step size approach the derived upper bound as the relative importance approaches infinity. Figure 1 outlines these two approaches.

### 3.1 Upper bound of \( \alpha_w \)

First, the upper bound of the step size \( \alpha_w \) is derived. The underlying concept and derivations of the bounds are similar to [15], but the resulting bound is different. We simply compare the errors of the TD error regression before and after the update, and derive the upper bound of \( \alpha_w \) to avoid overshotting the target.

**Lemma 1:** The upper bound for non-zero step size \( \alpha_w \) is:
\[
\psi_t^\top e^w_t \leq 0 \implies \alpha_w = 0,
\]
\[
\psi_t^\top e^w_t > 0 \implies \alpha_w \leq \frac{1}{\psi_t^\top e^w_t} \approx \alpha_{w}^\pi.
\]

**Proof:** Let \( w_t \) and \( w_{t+1} \) be \( w \) at times \( t \) and \( t + 1 \), respectively. We adopt the shorthand notation \( \psi_t = \psi(s_t, a_t) \). Then, the error of the TD error regression at time \( t \) before the update is:
\[
\delta_t^w = \delta_t^w - \psi_t e^w_t.
\]
When \( \delta_t^w = 0, w \) is not updated. Assume that \( \delta_t^w \neq 0 \). Similarly, the error after the update is given by
\[
\delta_t^w = \delta_t^w - \psi_{t+1} e^w_t
\]
\[
= \delta_t^w - (w_t + \alpha_w^\pi \delta_t^w e^w_t)\psi_t
\]
\[
= \delta_t^w - \alpha_w^\pi \psi_t e^w_t.
\]
When \( \psi_t^\top e^w_t = 0 \), the update does not affect \( \delta_t^w \), and therefore
\[
\psi_t^\top e^w_t = 0 \iff \delta_t^w = \delta_t^w.
\]

We are interested in the case where \( \psi_t^\top e^w_t \neq 0 \). If both \( |\delta_t^w| < |\delta_t^w| \) and \( \text{sign}(\delta_t^w) = \text{sign}(\delta_t^w) \) are true, the error does
not increase and the update does not overshoot the target. The errors before and after the update are then compared:

\[ 0 \leq \frac{\delta w}{\delta r} \leq 1, \]

substituting (12) above yields

\[ 0 \leq 1 - \alpha^w \psi_t^w e^w \leq 1, \]
\[ 0 \leq \alpha^w \psi_t^w e^w \leq 1. \tag{13} \]

When \(\psi_t^w e^w < 0\), the inequality (13) yields

\[ \frac{1}{\psi_t^w e^w} \leq \alpha^w \leq 0. \tag{14} \]

Thus, the requirement \(\alpha^w \geq 0\) enforces \(\alpha^w = 0\). Finally, when \(\psi_t^w e^w > 0\), the inequality (13) yields

\[ 0 \leq \alpha^w \leq \frac{1}{\psi_t^w e^w} = \alpha_t^w. \]

\( \square \)

Using the bound (10) as a step size directly is too aggressive because the upper bound given by Lemma 1 reflects only the local effect of the update using a given single sample.

Theorem 1, given below, provides a tight upper bound that takes into account the global effect for the stationary policy. The lower bound is also given by Theorem 2. Proofs for both theorems are given in Appendix A.

**Theorem 1:** If the policy is stationary,

\[ \alpha^* \leq \min_{\tau \in [0,1]} \frac{\delta w_{r,j}}{\psi_t^w e^w} \tag{15} \]

is a tight upper bound for a non-zero step size \(\alpha^w\), where \(\delta w_{r,j}\) is the \(r\)th experienced error before the \(r\)th update:

\[ \delta w_{r,j} = \delta_t^w - w_t^w \psi_t. \]

**Theorem 2:** If the policy is stationary,

\[ \alpha^* \geq \frac{1}{\|\delta w^w\| \max_{\tau \in [0,1]} \|\psi_t^w\|} \tag{16} \]

is a lower bound for a non-zero step size \(\alpha^w\).

Note that the straightforward applications of both the upper (15) and lower bounds (16) require the computation and memory storage for all \(\tau \in [0,1]\). One of the possible heuristics for applying the upper bound (15) with the memory and the computation of \(O(n)\) per time step is to take the minimum upper bound:

\[ \alpha^w_{\tau} = \min \left[ \alpha^w_{r-1}, \frac{1}{\|\psi_t^w e^w\|} \right]. \tag{17} \]

where \(\alpha^w_0 = 1.0\), similar to the application of alpha bound [15]. However, we assume that the policy is updated at each iteration, and therefore our policy is not stationary. Thus, decreasing the step size simply may be too conservative.

### 3.2 Adaptive Step Size for the Linear Function Approximator with the Trace by Relative Importance Weighting

Next, we derive an adaptive step size for the general linear-function approximation using the trace of a given feature vector. The aim is to propose the adaptive step size that avoids an overshoot, and that can interpolate a very aggressive update and a less aggressive update according to given relative importance. The derivation is similar to [16], but the resulting step size is different because we consider the trace.

Let \(\mathbf{x} \in \mathbb{R}^n\) be a feature vector and \(y \in \mathbb{R}\) be a target signal. We focus on a linear function approximator, \(\hat{y} = \mathbf{w}^\top \mathbf{x}\), where \(\mathbf{w} \in \mathbb{R}^n\) is a parameter. Let \(\ell(\hat{y}, y)\) be a loss function. The goal is to find \(\mathbf{w}^\star\) such that \(\mathbf{w}^\star = \arg\min_\mathbf{w} \sum_t \ell(\mathbf{w}^\top \mathbf{x}_t, y_t)\), using stochastic gradient descent. Further, let \(\mathbf{z}_t \doteq \sum_{\tau=0}^t \beta^{t-\tau} \mathbf{x}_\tau\), be a trace of the feature vector,

\(\beta \in [0, 1]\) is a decay factor. The update of \(\mathbf{w}\) at time \(t\) using the standard stochastic gradient descent is

\[ \mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla_\mathbf{w} \ell(\mathbf{w}^\top \mathbf{x}_t, y_t) \]
\[ = \mathbf{w}_t - \alpha \left[ \frac{\partial \ell}{\partial \mathbf{w}} \right]_{\mathbf{w}=\mathbf{w}_t} \mathbf{x}_t, \]

where \(\alpha\) is a small positive step size. In this study, however, we focus on the update using the trace of the feature vector:

\[ \mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \left[ \frac{\partial \ell}{\partial \mathbf{w}} \right]_{\mathbf{w}=\mathbf{w}_t} \mathbf{z}_t. \]

The derivation of the adaptive step size begins with the following lemma where the relative importance is an integer.

**Lemma 2:** Let \(h \in \mathbb{N}\) be a relative importance. Updating \(h\) times using a sample \((x_t, y_t)\) is equivalent to the following update:

\[ \mathbf{w}_{t+1} = \mathbf{w}_t - s(h) \mathbf{z}_t, \]

where the scaling factor \(s(h)\) has the recursive form:

\[ s(h+1) = s(h) + \alpha \left[ \frac{\partial \ell}{\partial \mathbf{w}} \right]_{\mathbf{w}=\mathbf{w}_t-s(h)\mathbf{z}_t} \mathbf{x}_t, \tag{18} \]
\[ s(0) = 0. \]

**Proof:** We prove the lemma by induction with respect to \(h\). The initial case \(h = 0\) is self-evidently true. The intermediate parameter updated \(h\) times is

\[ \mathbf{w}_r = \mathbf{w}_t - s(h) \mathbf{z}_r. \]

Thus, after \(h + 1\) updates, the result can be written using \(\mathbf{w}_r\) as follows:

\(\text{Though the arguments in this section hold for any bounded feature vector } \mathbf{z}_r, \text{ this study concerns the trace only.}\)
\[ w_{t+1} = w_t - \frac{\partial \ell}{\partial y} \bigg|_{y=(w_t - s(h)z_t)^\top x_t} z_t \]
\[ = w_t - (s(h) + \alpha \frac{\partial \ell}{\partial y}) \bigg|_{y=(w_t - s(h)z_t)^\top x_t} z_t. \]

This study considers the squared loss, \( \ell(\hat{y}, y_t) = \frac{1}{2}(y_t - \hat{y})^2. \)

Substituting this value into (18) yields
\[
s(h + 1) = s(h) + \alpha (w_t - s(h)z_t)^\top x_t - y_t)
\[ = s(h) \left(1 - \alpha x_t^\top z_t\right) + \alpha (w_t^\top x_t - y_t).
\]

Thus, we have
\[
s(h) = \frac{w_t^\top x_t - y_t}{x_t^\top z_t} \left(1 - \left(1 - \frac{\alpha}{N} x_t^\top z_t\right)^b\right).
\]

Furthermore, we fix \( h \) and consider updating \( w \) \( hN \) times, each with a step size \( \alpha/N \), where \( N \in \mathbb{N}_{>0} \):
\[
s(hN) = \frac{w_t^\top x_t - y_t}{x_t^\top z_t} \left(1 - \left(1 - \frac{\alpha}{N} x_t^\top z_t\right)^{bN}\right).
\]

In the limit \( N \to \infty \),
\[
\lim_{N \to \infty} s(hN) = \lim_{N \to \infty} \frac{w_t^\top x_t - y_t}{x_t^\top z_t} \left(1 - \left(1 - \frac{\alpha}{N} x_t^\top z_t\right)^{bN}\right)
\[ = \frac{w_t^\top x_t - y_t}{x_t^\top z_t} \left(1 - \exp(-\alpha x_t^\top z_t)\right).
\]

Thus, the update of \( w_t \) can be expressed as
\[
w_{t+1} = w_t + \frac{1 - \exp(-\alpha x_t^\top z_t)}{x_t^\top z_t} (y_t - w_t^\top x_t) z_t.
\]

Next, we generalize Lemma 2 to the case \( h \in \mathbb{R}_{\geq 0} \), as in [16]. By analogy with Eq. (19), the key idea is to perform \( N \) times more updates, each with a step size smaller by a factor of \( N \).

**Theorem 3:** The limit of the gradient descent process with the trace for one sample using an infinitesimal step size and a relative importance \( h \in \mathbb{R}_{\geq 0} \) equals the update
\[
w_{t+1} = w_t - s(h)z_t,
\]
where \( s(h) \) satisfies the differential equation
\[
s'(h) = \alpha \frac{\partial \ell}{\partial y} \bigg|_{y=(w_t - s(h)z_t)^\top x_t},
\]
\[ s(0) = 0.
\]

**Proof:** First, Lemma 2 holds for a step size \( \alpha/N \). The dependence on \( N \) is indicated explicitly by writing \( s_{a/N}(h) \) instead of \( s(h) \):
\[
s_{a/N}(h + 1) = s_{a/N}(h) + \frac{\alpha}{N} \Delta(s_{a/N}(h)),
\]
\[ s_{a/N}(0) = 0,
\]
where
\[
\Delta(s_{a/N}(h)) = \frac{\partial \ell}{\partial y} \bigg|_{y=(w_t - s_{a/N}(h)z_t)^\top x_t}.\]

Second, let \( s_r(i) = s_{a/N}(Ni) \) be a function whose argument is a non-negative rational number \( i = h/N \). Then, it follows that
\[
s_r(i + 1/N) = s_{a/N}(Ni + 1)
\[ = s_{a/N}(Ni) + \frac{\alpha}{N} \Delta(s_{a/N}(Ni))
\]
\[ = s_r(i) + \frac{\alpha}{N} \Delta(s_r(i)).\]

Note that the recurrence (22) corresponds to an \( N \)-fold multiple of updates, each with a step size smaller by a factor of \( N \). By rearranging (22), we have
\[
s_r(i + 1/N) - s_r(i) = \alpha \Delta(s_r(i)).\]

Therefore, taking the limit \( N \to \infty \) of Eq. (23) and changing the notation yields (21).

Again, in this study, we focus on the squared loss \( \ell(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2 \). The differential equation (21) reduces to
\[
s'(h) = \alpha \left( (w_t - s(h)z_t)^\top x_t - y_t \right),
\]
and is solved by
\[
s(h) = \frac{w_t^\top x_t - y_t}{x_t^\top z_t} \left(1 - \exp(-\alpha x_t^\top z_t)\right).
\]

Thus, the update of \( w \) can be written
\[
w_{t+1} = w_t + \frac{1 - \exp(-\alpha x_t^\top z_t)}{x_t^\top z_t} (y_t - w_t^\top x_t) z_t,
\]
which is equivalent to (20), except that \( h \) is a non-negative real number in Eq. (24).

Finally, we compare the errors before and after the update using Eq. (24). Assume that \( y_t - w_t^\top x_t \neq 0 \). The error after the update is:
\[
y_t - w_{t+1}^\top x_t
\[ = y_t - w_t^\top x_t - \frac{1 - \exp(-\alpha x_t^\top z_t)}{x_t^\top z_t} (y_t - w_t^\top x_t) z_t
\]
\[ = \left(y_t - w_t^\top x_t\right) \exp(-\alpha x_t^\top z_t).
\]

Thus, the update (24) does not overshoot the given target:
\[
\frac{y_t - w_{t+1}^\top x_t}{y_t - w_t^\top x_t} = \exp(-\alpha x_t^\top z_t) > 0.
\]

If \( x_t^\top z_t > 0 \), the update decreases the error. The meta-step-size \( \alpha \) and the relative importance \( h \) determine the aggressiveness of the update. On the other hand, if \( x_t^\top z_t < 0 \), then the error after the update can become very large and can even diverge. Therefore, rejecting the sample becomes an option.
3.3 Incremental NPG Estimation with Relative Importance Weighting

We apply the derived update (24) to the incremental estimation of the NPG (6–8). Substituting \(x_t, z_t, y_t\) and \(\alpha\) for \(\psi_t\), \(e^w\), \(\delta_t^w\) and \(\alpha^w\), respectively, Eq. (24) yields

\[
\begin{align*}
\omega & \leftarrow \omega + \alpha^w_t \left( \delta_t^w - \omega^\top \psi_t^\omega \right) e^w, \\
\alpha^w_t & \doteq \left\{ \begin{array}{ll}
1 - \exp(-h \alpha^w \psi_t^\omega e^w) \\
\frac{\psi_t^\omega e^w}{\exp(\psi_t^\omega e^w)} & (\psi_t^\omega e^w > 0) \\
0 & (\psi_t^\omega e^w \leq 0)
\end{array} \right. \quad (25)
\end{align*}
\]

Remark: The derived step size (25) lies within the upper bound (9, 10). Furthermore, in the limit \(h \to \infty\), the adaptive step size (25) reduces to the upper bound (9, 10). Note that \(h\) appears only as a multiplier of \(\alpha^w\). Thus, hereafter, we omit \(h\) from the derived step size (25) and let \(\alpha^w\) be the relative importance: \(\alpha^w \in (0, \infty)\).

As explained in the previous subsection, if \(\psi_t^\omega e^w < 0\), the update can cause a divergence of the estimated NPG. However, rejecting all these samples becomes very inefficient. In order to use all the samples, we use the conservative step-size strategy (17) if \(\psi_t^\omega e^w \leq 0\). Algorithm 1 shows the resulting overall procedures. Lines 16–19 in Algorithm 1 correspond to (17), where \(\alpha^w_t = \frac{1 - \exp(-\alpha^w \psi_t^\omega e^w)}{\psi_t^w e^w}\) is used instead of \(\frac{1}{\psi_t^\omega e^w}\) in this implementation.

Algorithm 1 Incremental NPG Estimation with Relative Importance Weighting.

1. **Input:** Parameterized policy \(\pi(\cdot|\cdot; \theta)\), basis function \(\phi(s)\). Initial parameters \(\theta = \theta_0, w = w_0, v = v_0\). Discount rate \(\gamma\), eligibility decay rate \(\lambda\). Step sizes \(\alpha^\theta, \alpha^w, \alpha^v, \beta = 1\).
2. Draw initial state and action
3. \(s_0 \sim d(\cdot), a_0 \sim \pi(s_0; \theta_0)\)
4. \(e^w = 0, e^v = 0\)
5. for \(t = 0, 1, 2, \ldots\) do
6. **Execution:**
7. \(s_{t+1} \sim T(s_t, a_t), a_{t+1} \sim \pi(s_{t+1}; \theta)\)
8. \(r = R(s_t, a_t)\)
9. **Update Critic:**
10. \(\delta_t^v = r_t + \gamma V(s_{t+1}) - V(s_t)\)
11. \(\delta_t^w = \delta_t^v - f_{\phi}(s_t, a_t)\)
12. \(e^v \leftarrow \gamma e^v + \nabla V(s_t)\)
13. \(e^w \leftarrow \gamma e^w + \phi(s_t, a_t)\)
14. if \(\psi_t^\omega e^w > 0\) then
15. \(\alpha_t^w = \frac{1 - \exp(-\alpha^w \psi_t^\omega e^w)}{\psi_t^\omega e^w} \quad // \text{aggressive}\)
16. else if \(\alpha_t^w < \beta\) then
17. \(\beta = \alpha_t^w\)
18. else
19. \(\alpha_t^w = \beta \quad // \text{conservative}\)
20. \(v \leftarrow v + \alpha^v \delta_t^v e^v\)
21. \(w \leftarrow w + \alpha^w \delta_t^w e^w\)
22. **Update Actor:**
23. \(\theta \leftarrow \theta + \alpha^\theta w\)
24. end for

4. Numerical Experiment

4.1 MDP with Two States

First, we evaluate how fast the proposed method can estimate the NPG using a classical benchmark [1] [2] [24]. The environment has two states and the agent has two possible actions in each state. The state transition law and reward function is shown in Fig. 2. The policy is characterized by \(\theta \in \mathbb{R}^2\), using the sigmoidal function:

\[
\begin{align*}
\pi(a_1|s; \theta) = \frac{1}{1 + \exp(-\theta^\top s)} \\
\pi(a_2|s; \theta) = 1 - \pi(a_1|s; \theta)
\end{align*}
\]

where \(i \in \{1, 2\}\). Clearly, the optimal decision making in this MDP involves choosing \(a_2\) in \(s_1\) and \(a_1\) in \(s_2\). Thus, the optimal policy parameter is \(\theta^* = (-\infty, \infty)^\top\). However, even though this MDP is very simple, it is well known that the learning using the vanilla policy gradient from the initial parameter \(\theta_0 = (1.4, -2.2)^\top\) will be trapped into a plateau. By using this environment, we can evaluate how fast the agent can estimate the NPG. In this experiment, we compare the proposed step size with the fixed step size. Table 1 lists the best meta-parameters (i.e., yielding the fastest learning without divergence) and Fig. 3 shows the corresponding learning results. Performance was evaluated by averaging over 100 runs. As shown in Table 1, the proposed step size allowed the setting of a large value of the (meta) step sizes without

![Fig. 2 MDP with two states.](image)

![Fig. 3 Learning curves in the MDP with two states. Mean and standard deviation of the average reward over 100 runs. The horizontal axis indicates the time steps and the vertical axis indicates the mean of the average reward.](image)

Table 1 Meta-parameters for the MDP with two states.

| Step size | \(\gamma\) | \(\lambda\) | \(\alpha^\theta\) | \(\alpha^w\) | \(\alpha^v\) |
|-----------|----------|----------|----------------|----------------|----------------|
| Fixed     | 0.9      | 0.7      | 0.0003         | 0.1            | 1.0            |
| Proposed  | 0.9      | 1.0      | 0.0005         | 0.3            | 1.0            |
divergence. This therefore constitutes an improvement over the use of a fixed step size.

4.2 Pendulum Swing Up and Stabilizing with Limited Torque

In the next experiment, we evaluated the robustness with regard to step-size tuning. The pendulum swinging up and the stabilization problem with limited torque (see Fig. 4) is a well known benchmark in continuous state-action space RL [3], [25]. The environmental state comprises the angle $q \in [-\pi, \pi]$ and angular velocity $\dot{q} \in [-15, 15]$ of the pendulum, i.e., $s = (q, \dot{q})^T$. The action of the agent is applied in the form of a torque to the pendulum after scaling, that is, $5a = \tau \in [-5, 5]$. The pendulum dynamics is given by $ml^2\ddot{q} = -\mu \dot{q} + mgl\sin(q) + \tau$, where $m = l = 1$, $g = 9.8$ and $\mu = 0.01$, and numerically integrated with $\Delta t = 0.02$. One episode lasts for 1000 steps and the initial state in each episode is $s_0 = (q_0, 0)^T$, where $q_0$ is determined randomly. The policy parameter is not updated in the first 100 episodes, in order to avoid using the incomplete NPG estimates. The reward function is $R(s) = \cos(q) - (\dot{q}/15\pi)^2$, and there is no penalty for over-rotation. The policy is specified as a Gaussian distribution

$$\pi(a|s; \theta) = \frac{1}{\sigma_\theta(s) \sqrt{2\pi}} \exp\left(-\frac{(a - \mu_\theta(s))^2}{2\sigma_\theta(s)^2}\right),$$

where the mean $\mu_\theta(s)$ and the standard deviation $\sigma_\theta(s)$ are determined by the output of a three-layer fully connected neural network. The input vector is $(\cos(q), \sin(q), \dot{q})^T$, and the hidden layer has 10 sigmoidal units. The output layer consists of two units: the activation of the mean unit is the
hyperbolic tangent function and the activation of the standard deviation unit is the sigmoid function. A small constant value $\sigma_0 = 0.01$ is added to the output of the standard deviation unit to avoid the divergence of $\psi_i$. The state value function is implemented with seventh order Fourier features [26]. For comparison, a fixed step size, AdaGrad [10], and ADAM [11] were applied. The discount factor was set to $\gamma = 0.98$. We performed a grid search such that $\lambda \in \{0.0, 0.7, 0.9, 1.0\}$, $\alpha^\theta \in \{10^{-3}, 5 \cdot 10^{-4}, 10^{-4}, \ldots, 10^{-7}\}$ and $\alpha^w, \alpha^\pi \in \{1.5 \cdot 10^{-1}, 10^{-1}, \ldots, 10^{-4}\}$. Figure 5 shows the learning results for all the combinations of the meta-parameters. Each learning curve corresponds to a different combination of meta-parameters. For each combination, the result was averaged over 10 runs. If the estimate of even one run diverged, the learning curve was truncated after the divergence for the combination. Thus, if the learning succeeds without divergence in many settings of the meta-parameters, the plot area becomes dense at its upper side. On the other hand, if the estimate divergences in many settings of the meta-parameters, the plot area becomes sparse. Figure 5 indicates that the proposed method is more robust with regard to the value of meta-parameters than NTD, AdaGrad, and ADAM. We also conducted additional experiment to show how aggressive (9, 10) and conservative (17) step size strategies work. The result is provided in Appendix B.

5. Conclusion and Future Work

In this study, we derived the upper bound of the step size used for the NTD algorithm, and proposed an adaptive step size method that weights the learning samples according to their relative importance, in order to implement the derived upper bound. The proposed adaptive step size determines its aggressiveness from the given meta-parameter. Numerical experiments validated the proposed method. We also provided the tight upper and lower bound for the step size, though they are not suitable for the incremental learning. To the best of our knowledge, this is the first adaptive step size method used for NPG estimation.

An interesting extension to this work would be to determine $h$ in Eq. (25) adaptively. As discussed in Remark in Sect. 3.3, we used a constant $h = 1$ in the experiment. However, $h$ can be changed adaptively during the learning. One possible criterion for choosing $h$ is the derivative of the logarithmic stationary distribution, $\nabla_\theta \ln d^\theta (s)$, which can be estimated as in [27]. By regulating $h$ adaptively depending on the state distribution even indirectly, the approximation of the expectation $E_\theta [\cdot]$ would be accelerated.

Acknowledgments

We thank the anonymous reviewers for their constructive comments to improve this paper. This study was supported by JSPS KAKENHI Grant Number 24000012 and JST CREST Grant Number JPMJCR17A4, Japan.

References

[1] S. Kakade, “A natural policy gradient,” Advances in Neural Information Processing Systems 14, pp.227–242, 2001.
[2] S.-I. Amari, “Natural gradient works efficiently in learning,” Neural Computation, vol.10, no.2, pp.251–276.
[3] T. Morimura, E. Uchibe, and K. Doya, “Utilizing natural gradient in temporal difference reinforcement learning with eligibility traces,” Proc. 2nd International Symposium on Information Geometry and Its Applications, pp.256–263, 2005.
[4] T. Morimura, “Efficient task-independent reinforcement learning methods based on policy gradient,” Ph.D. thesis, Nara Institute of Science and Technology, 2008.
[5] S. Bhatnagar, R.S. Sutton, M. Ghavamzadeh, and M. Lee, “Natural actor-critic algorithms,” Automatica, vol.45, no.11, pp.2471–2482, 2009.
[6] T. Degriss, P.M. Pilarski, and R.S. Sutton, “Model-free reinforcement learning with continuous action in practice,” Proc. 2012 American Control Conference, pp.2177–2182, 2012.
[7] P.S. Thomas, “Bias in natural actor-critic algorithms,” Proc. 31st International Conference on Machine Learning, pp.441–448, 2014.
[8] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” Nature, vol.518, pp.529–533, 2015.
[9] 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, 2016.
[10] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” The Journal of Machine Learning Research, vol.12, pp.2121–2159, 2011.
[11] D.P. Kingma and J.L. Ba, “Adam: A method for stochastic optimization,” International Conference for Learning Representations, 2015.
[12] T. Matsubara, T. Morimura, and T. Nomoto, “Adaptive step-size policy gradients with average reward metric,” Proc. 2nd Asian Conference on Machine Learning, pp.285–298, 2010.
[13] M. Pirotta, M. Restelli, and L. Bascetta, “Adaptive step-size for policy gradient methods,” Advances in Neural Information Processing Systems 26, pp.1394–1402, 2013.
[14] M. Hutter and S. Legg, “Temporal difference updating without a learning rate,” Advances in Neural Information Processing Systems 20, pp.705–712, 2007.
[15] W. Dabney and A.G. Barto, “Adaptive step-size for online temporal difference learning,” Proc. 26th Conference on Artificial Intelligence (AAAI-12), pp.872–878, 2012.
[16] N. Karampatziakis and J. Langford, “Online importance weight aware updates,” Proc. 27th Conference on Uncertainty in Artificial Intelligence, pp.392–399, 2011.
[17] R.S. Sutton, D. McAllester, S. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” Advances in Neural Information Processing Systems 12, pp.1057–1063, 1999.
[18] L.C. Baird III, “Advantage updating,” Tech. Rep. WL-TR-93-1146, Wright-Patterson Air Force Base Ohio: Wright Laboratory, 1993.
[19] J. Peters, S. Vijayakumar, and S. Schaal, “Reinforcement learning for humanoid robotics,” Proc. Third IEEE-RAS International Conference on Humanoid Robots, pp.103–123, 2003.
[20] J. Peters and S. Schaal, “Natural actor-critic,” Neurocomputing, vol.71, no.7-9, pp.1180–1190, 2008.
[21] H. van Seijen and R.S. Sutton, “True online TD(λ),” Proc. 31st In-
international Conference on Machine Learning, pp.692–700, 2014.
[22] R.S. Sutton and A.G. Barto, Reinforcement Learning: An introduction, MIT Press, 1998.
[23] D.P. Bertsekas and J.N. Tsitsiklis, “Gradient convergence in gradient methods with errors,” SIAM Journal on Optimization, vol.10, no.3, pp.627–642, 2000.
[24] J.A. Bagnell and J. Schneider, “Covariant policy search,” Proc. 18th International Joint Conference on Artificial Intelligence, pp.1019–1024, 2003.
[25] K. Doya, “Reinforcement learning in continuous time and space,” Neural Computation, vol.12, no.1, pp.219–245, 2000.
[26] G. Konidaris, S. Osentoski, and P. Thomas, “Value function approximations in policy gradient methods with errors,” SIAM Journal on Optimization, vol.10, no.3, pp.342–376, 2010.

Appendix A: Proofs of Theorems 1 and 2

A.1 Theorem 1: The tight upper bound

Proof: We prove the lemma by contradiction. Assume the contrapositive proposition that there exists some \( \alpha \in (0, \alpha^*_s) \) such that \( \alpha \) is an upper bound for \( \alpha^w \). From this assumption, it follows that (i) using \( \alpha^*_s \) at time \( t \) causes an error in the past, \( \tau \in [0, t] \), to increase the absolute value or change the sign, (ii) while \( \alpha \) does not cause such a effect. The \( \tau \)th experienced error before the \( \tau \)th update is:

\[
\delta_{\tau}^w = \delta_{\tau}^e - w_{\tau+1}^\top \psi_{\tau}.
\]

Assume that \( \delta_{\tau}^w \neq 0 \). Similarly, the \( \tau \)th experienced error after the \( \tau \)th update using \( \alpha^w \) is given by

\[
\delta_{\tau}^w = \delta_{\tau}^e - w_{\tau+1}^\top \psi_{\tau} = \delta_{\tau}^e - (w_{\tau} + \alpha^w \delta_{\tau}^w e_{\tau}^w)^\top \psi_{\tau} = \delta_{\tau}^w - \alpha^w \delta_{\tau}^w \psi_{\tau} e_{\tau}^w.
\]

We compare the \( \tau \) th errors before and after the \( \tau \) th update:

\[
\frac{\delta_{\tau}^w}{\delta_{\tau}^w} = 1 - \alpha^w \frac{\delta_{\tau}^w}{\delta_{\tau}^w} \psi_{\tau}^w e_{\tau}^w = 1 - \alpha^w \frac{\delta_{\tau}^w}{\delta_{\tau}^w},
\]

(A.1)

where \( \eta = \frac{\delta_{\tau}^w}{\delta_{\tau}^w} \). Then, the contrary assumption implies that there exists some \( \tau \in [0, t] \) such that \( \delta_{\tau}^w / \delta_{\tau}^w > 1 \) or \( \delta_{\tau}^w / \delta_{\tau}^w < 0 \) for \( \alpha^w = \alpha^*_s \) and \( 0 \leq \delta_{\tau}^w / \delta_{\tau}^w \leq 1 \) for \( \alpha^w = \alpha^*_s \).

Let \( \alpha^w = \alpha^*_s \), then if \( \delta_{\tau}^w / \delta_{\tau}^w < 0 \), Eq. (A.1) yields \( \alpha^*_s / \eta > 1 \). This implies that there exists some \( \eta \) such that \( \eta \in [0, \alpha^*_s] \), which is a contradiction because \( \eta \) is the element of the set in which \( \alpha^*_s \) is the smallest value. If \( \delta_{\tau}^w / \delta_{\tau}^w > 1 \), Eq. (A.1) yields \( \alpha^*_s / \eta < 0 \). In this case \( \eta < 0 \) because \( \alpha^*_s > 0 \). Now we consider using \( \alpha^w = \alpha \) in this context:

\[
0 \leq \frac{\delta_{\tau}^w}{\delta_{\tau}^w}, \quad \leq 1,
\]

\[
0 \leq 1 - \frac{\alpha}{\eta} \leq 1,
\]

\[
\eta \leq \alpha \leq 0.
\]

This is again a contradiction because it was assumed that \( \alpha > 0 \). Therefore, there exists no lower upper bound such that \( \alpha \in (0, \alpha^*_s) \). \( \square \)

A.2 Theorem 2: The tight lower bound

Proof: The least upper bound given by Theorem 1 yields

\[
\alpha^*_t = \min_{\tau \in [0, t]} \frac{\delta_{\tau}^w}{\delta_{\tau}^w} e_{\tau}^w = \left( \max_{\tau \in [0, t]} \frac{\delta_{\tau}^w}{\delta_{\tau}^w} \right)^{-1} = \left( \max_{\tau \in [0, t]} \left( \frac{\delta_{\tau}^w}{\delta_{\tau}^w} \right) \right)^{-1},
\]

where \( \langle \cdot, \cdot \rangle \) denotes the inner product. Then, the Cauchy-Schwarz inequality yields

\[
\frac{1}{\alpha^*_t} = \max_{\tau \in [0, t]} \left| \left| \frac{\delta_{\tau}^w}{\delta_{\tau}^w}, \frac{\psi_{\tau}}{\delta_{\tau}^w} \right| \right| \leq \max_{\tau \in [0, t]} \left| \left| \delta_{\tau}^w \right| \right| \left| \left| \psi_{\tau} \right| \right| \leq \left| \left| \delta_{\tau}^w \right| \right| \max_{\tau \in [0, t]} \left| \left| \psi_{\tau} \right| \right|.
\]

Fig. A.1 Learning curves in the inverted pendulum for all the combinations of the meta-parameters. Top: aggressive step size strategy. Bottom: conservative step size strategy. Each learning curve corresponds to a different combination of meta-parameters. The result is averaged over 10 runs. The horizontal axis indicates the episodes and the vertical axis indicates the mean of average reward.
Appendix B: Learning Results for the Aggressive and Conservative Step Size Strategies

We here provide additional experimental results for the inverted pendulum domain, which show how the aggressive (9, 10) and conservative (17) step size strategies work. The experimental set up is the same as in Sect. 4.2, except that we did not conduct a grid search for $\alpha^w$ because aggressive and conservative step sizes do not require any meta-parameters. Figure A.1 shows the learning results. When using the aggressive step size, the learning resulted in the divergence of the estimates for many meta-parameter combinations, especially in the early stage of learning. For the conservative step size, the learnings were trapped in the plateaux, where the average rewards are approximately $-0.1$. These plateaux correspond to the over-rotation of the pendulum.

Ryo Iwaki received his B.E. (2014), M.E. (2016) degrees from Department of Adaptive Machine Systems at Osaka University, Japan. He is currently a Ph.D. student in the same institute. His research interests include the theory of reinforcement learning.

Hiroki Yokoyama received his Ph.D. degree in Engineering from Muroran Institute of Technology in 2013. He worked as a JSPS research fellow (DC2) at the same institute and (PD) at the University of Electro-Communication. He is currently an adjunct research fellow at Tamagawa University Brain Science Institute. His research interests include computational neuroscience and machine learning.

Minoru Asada received his B.E. (1977), M.E. (1979), and Ph.D. (1982) degrees in Control Engineering from Osaka University, Osaka, Japan. Since 1997, he has been a Professor at the Department of Adaptive Machine Systems at Osaka University. He is one of founders of RoboCup, and an advocate of cognitive developmental robotics focusing in constructive approaches to human cognitive development.