COOPERATIVE OFF-POLICY PREDICTION OF MARKOV DECISION PROCESSES IN ADAPTIVE NETWORKS

Sergio Valcarcel Macua  Jianshu Chen  Santiago Zazo  Ali H. Sayed

ABSTRACT

We apply diffusion strategies to propose a cooperative reinforcement learning algorithm, in which agents in a network communicate with their neighbors to improve predictions about their environment. The algorithm is suitable to learn off-policy even in large state spaces. We provide a mean-square-error performance analysis under constant step-sizes. The gain of cooperation in the form of more stability and less bias and variance in the prediction error, is illustrated in the context of a classical model. We show that the improvement in performance is especially significant when the behavior policy of the agents is different from the target policy under evaluation.

Index Terms—adaptive networks, dynamic programming, diffusion strategies, gradient temporal difference, mean-square-error, reinforcement learning.

1. INTRODUCTION

Consider the problem in which a network of autonomous agents collaborate to predict the response of their environment to their actions. The network forms a connected graph (i.e., there is at least one path between every pair of nodes) and communication is only allowed within each neighborhood. The agents operate in an environment that is modeled as a Markov Decision Process (MDP) [1]. The MDP is characterized by a finite and countable set of states, $S$, a finite and countable set of actions, $A$, and the kernel of transition probabilities from one state to another given an action, $P$. Although the agents can communicate with their neighbors, each agent is assumed to operate a similar but independent MDP. Therefore, the state transition probability at each agent $k = 1 \ldots N$ is only determined by its own action and the previous state of its environment, i.e., $s_k(i + 1) \sim P(s_k(i), a_k(i))$, where $s_k(i) \in S$ denotes the state of the environment seen by agent $k$ at time $i$, and $a_k(i) \in A$ stands for its action.

We assume agents follow some stationary policy. A policy $\pi$ is a mapping $\pi : S \rightarrow \Theta(A)$, where $\Theta(A)$ is the set of all probability distributions over $A$. We denote by $\pi(a, s) = P(a|s)$ the probability of an agent choosing action $a$ when it follows policy $\pi$ and the environment is at state $s$.

At every time step, there is a reward function, $r : S \times A \times S \rightarrow \mathbb{R}$, which agents receive and which they want to predict. Let $r(i) = r(s(i-1), a(i-1), s(i))$ denote the reward received by a generic agent, for the transition from $s(i-1)$ to $s(i)$, after taking action $a(i-1)$. In order to make predictions of the reward signal, we use state general-value-functions (GVF), $v : S \rightarrow \mathbb{R}$, which provide the expected cumulative sum of the reward until the agent reaches a state that marks the end of the prediction period, $\Omega$. Introduce the termination-indicator function $\gamma : S \rightarrow \{0, 1\}$, which sets the time-interval of the prediction period so that $\gamma(s(i)) \equiv 0$ indicates that the agent has reached $\Omega$ at time $i$. We denote $\phi$ the random stopping time at which the agent reaches the termination state $\Omega$. Then, the GVF for an arbitrary initial state $\xi \in S$ is defined by:

$$v(\xi) = \mathbb{E} \left[ \sum_{i=0}^{\phi} \gamma(s(i)) \cdot r(s(i), a(i), s(i+1)) \mid s(0) = \xi, a(i) \sim \pi \right]$$

where we are using the boldface notation to denote random variables, and where the notation $a(i) \sim \pi$ means that we are interested in the reward received by following the policy $\pi$, which we name the target policy. The policy $\pi$ may be different from the actual behavior policy followed by the agents, and denoted by $\pi_v$. The prediction problem that corresponds to the situation $\pi \neq \pi_v$ is called off-policy learning. We can see the functions $r$, $\pi$, and $\gamma$ are a grammar that the agents use to ask questions about the expected future response of their environment, and the GVF given by (1) is the answer to these questions (see [2, 3, 4] and Section 4).

In many real applications, agents do not have access to the state, but to a feature vector of the state. Also, in problems with very large state-space dimension, even if the state of the system can be known precisely, it is computationally more efficient to work with carefully chosen features [2, 5] with much smaller dimension $M$ than $|S|$. Let $x : S \rightarrow \mathbb{R}^M$ be some mapping from states to features. Then, we denote $x_{k,i} = x(s_k(i))$, the $M \times 1$ feature vector that represents (maybe ‘roughly’) the state of the agent $k$ at time $i$. Furthermore, note that $v(\xi)$ in (1) could be a nontrivial function of $\xi$. Therefore, it would be efficient to approximate $v(\xi)$ as a linear function of $x(\xi)$, say, $x(\xi)^T \omega$, where $\omega \in \mathbb{R}^M$ denotes a parameter vector. In this way, we would only need to learn the parameter $\omega$ in order to approximate (or learn) $v(\xi)$. Such an approximation would provide good results if we carefully choose the mapping of features. Stacking the output of (1), for every possible state $\xi \in S$, into the vector $v = \text{col}\{v(\xi)\} \in \mathbb{R}^{sM}$, we express the linear approximation as $v \approx X \omega$, where $X \in \mathbb{R}^{sM \times sM}$ is a non-singular matrix that is formed by stacking the feature row vectors $x^T(\xi)$, for every state $\xi \in S$, on top of each other.

Now, the problem becomes that of seeking a parameter vector $\omega^*$ that is optimal in a certain sense using the available data that arrive sequentially at each agent $k$ (i.e., the tuples $(x_k(i), r_k(i), y_k(i))$ where $r_k(i) = r(s_k(i-1), a_k(i-1), s_k(i))$ and $\gamma_k(i) = \gamma(s_k(i)))$.

In the context of a single agent scenario (i.e., without co-
operation, reference [6] proposed the mean-square-projected-Bellman-error (MSPE) as a performance criterion for finding \( \omega \) (see also [7, 8, 9]). However, the MSPE depends on both \( \pi \) and \( \pi_k \). Nevertheless, using the importance sampling weights, \( \tau_k(i) \equiv \pi(a_{k}(i), s_{k}(i))/\pi_k(a_{k}(i), s_{k}(i))\), reference [10] showed that the MSPE can be expressed as a product of expectations, which are taken with respect to the same distribution induced by \( \pi_k \). Specifically, let \( e_{k,i} = \tau_k(i)x_{k,i}, \) then the MSPE can be expressed as

\[
J_k(\omega) = \mathbb{E}[\delta_k^* (i) \cdot e_{k,i}]^\top \mathbb{E}[x_{k,i}x_{k,i}^\top]^{-1} \mathbb{E}[\delta_k^* (i) \cdot e_{k,i}] \tag{2}
\]

where \( \delta_k^* (i) \) is a scalar that estimates the error of the prediction:

\[
\delta_k^* (i) \equiv r_k(i + 1) + \gamma_k(i + 1)x_{k,i+1}^\top \omega - x_{k,i}^\top \omega \tag{3}
\]

In order to apply stochastic gradient descent to (2), we cannot sample the three expected values at every iteration, since the random quantities at each expectation would be correlated and their product would be biased. One solution from [11] is to sample only one of the expectations while tracking a long-term, quasi-stationary estimate of the others. In particular, note that the gradient of (2) is given by

\[
\frac{1}{2} \nabla_\omega J_k(\omega) = \mathbb{E} \left[ (\gamma_k(i + 1)x_{k,i+1} - x_{k,i})e_{k,i}^\top \right] \cdot \hat{\theta}_k^\top \tag{4}
\]

where

\[
\hat{\theta}_k = \mathbb{E} \left[ x_{k,i}x_{k,i}^\top \right]^{-1} \mathbb{E}[\delta_k^* (i) \cdot e_{k,i}] \tag{5}
\]

It was noticed in [6] that \( \hat{\theta}_k \) in (5) is similar to the solution of the normal equations for linear least-mean-squares error estimation, so it can be approached using the least-mean-squares (LMS) algorithm.

### 2. Diffusion Adaptation Policy

In the context of networked agents, we propose to find the optimal parameter vector \( \omega^o \in \mathbb{R}^k \) that minimizes the global cost:

\[
J^{glob}(\omega) = \sum_{k=1}^N J_k(\omega) \tag{6}
\]

where \( J_k(\omega), k = 1, \ldots, N \), are the individual MSPE given in (2). In order to minimize (6) in a cooperative and fully distributed manner, we apply diffusion strategies [12, 13] on (4) and (5), obtaining the diffusion-off-policy-gradient-temporal-difference (D-OPTGD) algorithm given in (7), with step-size parameters \( \mu \) and \( \eta \):

\[
\theta_{k,i} = \theta_{k,i-1} - \eta \mu (x_{k,i-1} \omega_{k,i-1} - \delta_{k,i-1}^*(i-1)x_{k,i-1})
\]

\[
\omega_{k,i} = \omega_{k,i-1} - \eta (\gamma_k(i)x_{k,i} - x_{k,i-1})^\top e_{k,i-1}^\top \theta_{k,i-1}
\]

\[
\theta_{k,i} = \sum_{i \in N_k} b_{k,i} \tilde{\theta}_{k,i}
\]

\[
\omega_{k,i} = \sum_{i \in N_k} b_{k,i} \tilde{\omega}_{k,i}
\]

where \( \omega_{k,i} \) denotes the local estimate at node \( k \) of \( \omega^o \) at time \( i \). In order to consider \( \theta_{k,i} \) constant during the update of \( \omega_{k,i} \), the latter must be updated at a lower speed than the former. In other words, the step-size ratio, \( \eta \), should satisfy \( \eta \ll 1 \). Note that (7) is the cooperative distributed extension of the single-agent GTD2 algorithm introduced in [6].

The matrix with the combination coefficients, \( B = [b_{k,i}] \), is constrained by the network topology, in the sense that non-zero elements can appear only at the locations corresponding to the active-links. These elements can be freely chosen by the designer, as long as \( B \) is a left stochastic matrix (i.e., the entries on each column of \( B \) add up to one).

### 3. Performance Analysis

#### 3.1. Data Model

To analyze the performance of the distributed solution (7), we extend the energy conservation arguments of [12, 14] to carry out a mean-square-error (MSE) analysis; this is in contrast to the ordinary-differential equations method used in [6, 10, 11, 15] for the variations of the single-agent GTD algorithm. In particular, our analysis relies on studying the evolution of the following pair of stochastic equations, which are of the same nature as the updates appearing in (7):

\[
\psi_{k,i} = \alpha_{k,i-1} - \mu (G_{k,i}\alpha_{k,i-1} + g_{k,i})
\]

\[
\alpha_{k,i} = \sum_{i \in N_k} b_{k,i} \psi_{k,i}
\]

Let us introduce the matrices:

\[
C_{k,i} = x_{k,i}x_{k,i}^\top, \quad A_{k,i} = e_{k,i}(\gamma_k(i + 1)x_{k,i+1} - x_{k,i})^\top
\]

and the variables:

\[
\alpha_{k,i} = \begin{bmatrix} \theta_{k,i} \\ \omega_{k,i} \end{bmatrix}, \quad \psi_{k,i} = \begin{bmatrix} \tilde{\theta}_{k,i} \\ \tilde{\omega}_{k,i} \end{bmatrix}
\]

\[
g_{k,i} = \begin{bmatrix} -\eta e_{k,i-1}r_k(i) \\ 0 \end{bmatrix}
\]

\[
G_{k,i} = \begin{bmatrix} \eta C_{k,i-1} - \eta A_{k,i-1} \\ A_{k,i-1} \end{bmatrix}
\]

such that \( \alpha_{k,i}, \psi_{k,i}, \) and \( g_{k,i} \) are vectors of length \( 2M \) and \( G_{k,i} \) is a matrix of size \( 2M \times 2M \). Then, the model (8) is equivalent to (7).

We introduce \( C = C_k = \mathbb{E}[C_{k,i}], A = A_k = \mathbb{E}[A_{k,i}], G = G_k = \mathbb{E}[G_{k,i}], \) and \( g = g_k = \mathbb{E}[g_{k,i}] \), which are the same for every node \( k = 1, \ldots, N \). Then, for our data model we assume the following conditions.

**Assumption 1.** Samples \( \{x_{k,i-1}, x_{k,i}, r_k(i), \gamma_k(i)\} \) are assumed to be drawn i.i.d. from the steady-state visitation probability distribution of the underlying MDP (induced by \( \pi_n \)).

**Assumption 2.** Matrices \( C \) and \( A \) are non-singular.

Assumption 1 is customary and it is reasonable if the Markov chain of the state-process is mixing fast enough [16]; it renders \( \psi_{k,i}, \alpha_{k,i} \) independent of \( \alpha_{k,i-1} \). Assumption 2 guarantees the existence and uniqueness of the fixed point of (8), \( \alpha^o \), such that \( G^o + g = 0 \); it should be satisfied when the features capture the structure of the state-space and the sample set is rich enough. Reference [6] proposed a slightly different set of aggregated variables for studying the performance of the single-agent algorithm; it showed that, under Assumption 2, the real part of the complex eigenvalues of the coefficient matrix is always positive. It turns out that \( G \) is a similarity transformation of the coefficient matrix used in [6], so we conclude that the real part of the eigenvalues of \( G \) is also always positive.

#### 3.2. Convergence in the Mean

Introduce the following error quantities:

\[
\tilde{\psi}_{k,i} \triangleq \alpha^o - \psi_{k,i}, \quad \tilde{\alpha}_{k,i} \triangleq \alpha^o - \alpha_{k,i}
\]

In order to describe these relations more compactly we introduce the following network error vectors of length \( 2MN \):

\[
\tilde{\psi}_i \triangleq \text{col}(\tilde{\psi}_{i1}, \ldots, \tilde{\psi}_{iN_i}), \quad \tilde{\alpha}_i \triangleq \text{col}(\tilde{\alpha}_{i1}, \ldots, \tilde{\alpha}_{iN_i})
\]

where \( \text{col}(\cdot) \) denotes the column vector and \( i \) denotes the node. Refer to [6] for details on how the network error vectors can be used to study the performance of the distributed algorithm.
Let $B \triangleq B \otimes I_{2M}$ and $R_i \triangleq \text{diag}(G_{i,1}, \ldots, G_{i,N})$ be of size $2MN \times 2MN$ each, and $G_i \triangleq \text{col}(G_{i,1}, \ldots, G_{i,N})$ of size $2MN \times 2M$. Finally, we aggregate the network signal $g_i = \text{col}(g_{i,1}, \ldots, g_{i,N})$ into another vector of length $2MN$, and introduce the network noise term $n_i \triangleq G_i o + g_i$. Then, the individual error recursions (13) lead to the following network recursion:

$$\bar{\alpha}_i = B^T (I_{2MN} - \mu R_i) \bar{\alpha}_{i-1} + \mu B^T n_i \quad (15)$$

Since $E n_i = \mathbb{E} G_i o + \mathbb{E} g_i = 0$, taking expectation of both sides of (15), we obtain, under the assumption that $\bar{\alpha}_i$ and $R_i$ are independent of each other,

$$E \| \bar{\alpha}_i \|^2 = E \| \bar{\alpha}_{i-1} \|^2 + 2\mu \cdot \mathbb{E} \| \bar{\alpha}_{i-1} \| \mathbb{E} n_i$$

and introduce the network noise term $n_i = G_i o + g_i$.

Let $\sigma = \text{vec}(\Sigma)$ denote the vectorization operation that stacks the columns of a matrix $\Sigma$ on top of each other. We can vectorize $\Sigma$ in (21), leading to $\sigma' \triangleq \text{vec}(\Sigma') = \mathcal{F} \sigma$, where

$$\mathcal{F} \triangleq \left( \begin{array}{c} (I_{2MN} - \mu R) B \\ (I_{2MN} - \mu R) B' \end{array} \right)$$

$$+ \mu^2 \mathbb{E} \left[ \begin{array}{c} (R_i - R \otimes) B \\ (R_i - R \otimes) B' \end{array} \right]$$

(24)

Therefore, we rewrite (20) in the following compact form, where we are replacing the weighting matrices by their vector representations:

$$\| \bar{\alpha}_i \|^2 = E \| \bar{\alpha}_{i-1} \|^2 + 2\mu \cdot \mathbb{E} \| \bar{\alpha}_{i-1} \| \mathbb{E} n_i + \mu^2 \mathbb{E} n_i^T n_i$$

(25)

where

$$\mathcal{U} \triangleq \left[ \begin{array}{c} B^T G_i o + g_i \\ B^T (I_{2MN} - \mu R_i) \end{array} \right]$$

(26)

$$h \triangleq \text{vec}(B^T R_i B)$$

(27)

Note that the mean-square-error recursion in (25) is not a true recursion because the norms are different. Moreover, it is coupled with the mean-error recursion in (16). We can expand (25) into a state-space model [12, 14, 17] that can be aggregated with (16). Let $L \triangleq 2MN$ and let $p(x)$ be the characteristic polynomial of the $L^2 \otimes L^2$ matrix $\mathcal{F}$. By the Cayley-Hamilton Theorem [17], we know that every matrix satisfies its characteristic equation (i.e. $p(\mathcal{F}) = 0$), so we have

$$\mathcal{F} L^2 = -p_0 I_{L^2} - p_1 \mathcal{F} - \ldots - p_{L^2-1} \mathcal{F}^{L^2-1}$$

(28)

Replacing $\sigma$ with $\mathcal{F} \sigma, j = 0, \ldots, L^2 - 1, we obtain the following state-space model:

$$\begin{bmatrix} \mathbb{E} \| \bar{\alpha}_i \|^2 \\ \mathbb{E} \| \bar{\alpha}_{i-1} \|^2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 \end{bmatrix} \begin{bmatrix} \mathbb{E} \| \bar{\alpha}_i \|^2 \\ \mathbb{E} \| \bar{\alpha}_{i-1} \|^2 \end{bmatrix}$$

$$+ 2\mu \begin{bmatrix} \sigma^T \mathcal{U} \\ \sigma^T \mathcal{U} \mathcal{F} \end{bmatrix}$$

(29)

Finally, aggregating the mean-square-error recursion (29) with the mean-error recursion (16), we obtain

$$\begin{bmatrix} \mathbb{E} \| \bar{\alpha}_i \|^2 \\ \mathbb{E} \bar{\alpha}_i \end{bmatrix} = \begin{bmatrix} 0 & 2\mu Q \\ \mathcal{F} \end{bmatrix} \begin{bmatrix} \mathbb{E} \| \bar{\alpha}_{i-1} \|^2 \\ \mathbb{E} \bar{\alpha}_{i-1} \end{bmatrix}$$

(30)

Observe that the stability of the joint recursion (30) is equivalent to the stability of the matrices $T$ and $B^T (I_{2MN} - \mu R)$. We note from (29) that $T$ is in companion form, and it is known that its eigenvalues are also eigenvalues of $\mathcal{F}$. When the step-sizes are small enough, we have the following approximation for $\mathcal{F}$:

$$\mathcal{F} \approx \left( \begin{array}{c} (I_{2MN} - \mu R) B \\ (I_{2MN} - \mu R) B' \end{array} \right)$$

$$= \left( B^T (I_{2MN} - \mu R) \right)$$

(31)

which is stable if, and only if, $B^T (I_{2MN} - \mu R)$ is stable. Therefore, sufficiently small step-sizes guarantee stability in the mean and mean-square-error senses.
3.4. Mean-Square Performance

If we take the limit of both sides of (25), and use the facts that $\lim_{t \to \infty} \mathbb{E} \mathbf{a}_i = 0$ and $\lim_{t \to \infty} \mathbb{E} \| \mathbf{a}_i \|^2 = \lim_{t \to \infty} \mathbb{E} \| \mathbf{a}_i \|^2$, we obtain:

$$\lim_{t \to \infty} \mathbb{E} \| \mathbf{a}_i \|^2 = \mu^2 h^{-1} (I - F)^{-1} \sigma$$

Expression (32) is useful because it allows us to derive several performance metrics through the proper selection of the free weighting parameter $\sigma$ (or, equivalently, the parameter matrix $\Sigma$). For example, the network mean-square-deviation (MSD) is defined as the average of the error of the nodes across the network:

$$\text{MSD}_{\text{network}} \triangleq \lim_{t \to \infty} \frac{1}{N} \sum_{k=1}^{N} \mathbb{E} \| \mathbf{a}_{k,i} \|^2 = \mathbb{E} \| \mathbf{a}_i \|^2 \frac{I_{2M N}}{N}$$

Choosing the weighting matrix as $\Sigma = I_{2MN}/N$, we get:

$$\text{MSD}_{\text{network}} = \frac{\mu^2}{N} h^{-1} (I - F)^{-1} \text{vec}(I_{2MN})$$

We can also obtain the MSD of any particular node $k$, as

$$\text{MSD}_k \triangleq \lim_{t \to \infty} \mathbb{E} \| \mathbf{a}_{k,i} \|^2$$

Introduce a block diagonal matrix $J_k$, of $N \times N$ blocks of size $2M \times 2M$ each, such that all its blocks are zero except for block $k$ which is the identity matrix. Then, we obtain:

$$\text{MSD}_k = \mu^2 h^{-1} (I - F)^{-1} \text{vec}(J_k)$$

4. SIMULATIONS

Consider the following example. An automated taxi is driving on a motorway that is 14 miles long, from the suburbs to downtown. During the journey, the taxi can exchange information with other cars within its communication range. Every mile along the road, there is an exit that the autopilot can choose to take or not. If the car takes the exit, it can move towards the goal at a slower speed than in the motorway, but at a slower speed. In either case, the taxi can get stuck in the same state. When an exit is taken, the probability of moving towards the destination is high and constant, $P_{\text{exit}} = 0.5$. The combination coefficients of the estimates, $B$, are computed by 10 cooperative versus 10 non-cooperative agents. In (b), we see some improvement in the cooperative agents, non-cooperative agents are still good at finding the estimate. In (c), when the behavior policy is very biased, there is a clear benefit of diffusion: while non-cooperative agents diverge, cooperative agents achieve a good, stable estimate of the value.

![Taxi example](image)

Fig. 1. Taxi example. (a) State diagram, and (b)-(c) mean-square projected-Bellman error (MPSBE) for a target policy $\pi = [0.2, 0.8]$, estimated by 10 cooperative versus 10 non-cooperative agents. In (b), though we see some improvement in the cooperative agents, non-cooperative agents are still good at finding the estimate. In (c), when the behavior policy is very biased, there is a clear benefit of diffusion: while non-cooperative agents diverge, cooperative agents achieve a good, stable estimate. The feature vectors for states $13, 9, 5$ and $1$ are $x(13) = [1, 0, 0, 0]^T$, $x(9) = [0, 1, 0, 0]^T$, $x(5) = [0, 0, 1, 0]^T$ and $x(1) = [0, 0, 0, 1]^T$ respectively, for the rest of states, the features are obtained by interpolating linearly between these (i.e., $x(2) = [0, 0, 1/4, 3/4]^T$, $x(3) = [0, 0, 1/2, 1/2]^T$, $x_4 = [0, 0, 3/4, 1/4]^T$, and so on). Step-sizes are constant $\mu = 0.1$ and $\eta = 10$. The combination coefficients of the estimates, $B$, are obtained using the Metropolis method [13]. Finally, $P_{\text{exit}}(\text{exit}) = 0.8, \forall s \in S$, and $P_{\text{exit}}(s, \text{motorway}) = e^{-\frac{13}{13}}, s \in S$.

5. CONCLUSIONS

We proposed a distributed diffusion strategy for off-policy learning and provided a mean-square projected-Bellman error analysis showing that sufficiently small step-sizes guarantee convergence in the mean-square error sense. This result complements and extends the ODE analysis of the original single-agent GTD algorithm, which requires diminishing step-sizes [10, 11, 15].
6. REFERENCES


