Exploring Multi-Action Relationship in Reinforcement Learning

Han Wang and Yang Yu*

National Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing 210023, China
{wangh,yuy}@lamda.nju.edu.cn

Abstract. In many real-world *reinforcement learning* problems, an agent needs to control multiple actions simultaneously. To learn under this circumstance, previously, each action was commonly treated independently with other. However, these multiple actions are rarely independent in applications, and it could be helpful to accelerate the learning if the underlying relationship among the actions is utilized. This paper explores multi-action relationship in reinforcement learning. We propose to learn the multi-action relationship by enforcing a regularization term capturing the relationship. We incorporate the regularization term into the *least-square policy-iteration* and the *temporal-difference* methods, which result efficiently solvable convex learning objectives. The proposed methods are validated empirically in several domains. Experiment results show that incorporating multi-action relationship can effectively improve the learning performance.

1 Introduction

Reinforcement learning techniques intend to enable an agent to learn how to behave through trial-and-error interactions with its environment. Reinforcement learning has been an attractive field experiencing progress from theory to practice, with wide applications, including robotics [2], computer Go [10], and combinatorial optimization problems [3].

In many cases, an agent needs to control multiple actions simultaneously. For example, an automated driving agent needs to control multiple components in parallel: when approaching a turning, the agent is supposed to turn the steering wheel, meanwhile release the accelerator and hit the brake. However, most of previous reinforcement learning methods treated each action independently with each other, and thus learned each action separately. The only previous study that explicitly considered multi-action setting is [9], where a concurrent action model was proposed to solve multi-action problems based on SMDP Q-learning.

There are also several studies that are apparently related to this work, but are actually different, including multi-task, multi-agent, and multidimensional reinforcement learning. Multi-task reinforcement learning assumes that the agent faces multiple learning tasks within its lifetime [4]. These approaches often solve different tasks sequentially, rather than the same take concurrently as in the multi-action setting. The need

* This research was supported by the NSFC (61375061, 61223003), Foundation for the Author of National Excellent Doctoral Dissertation of China (201451)

for adaptive multi-agent systems has led to the development of a multi-agent reinforcement learning field [13]. While multi-agent methods can be used to learn multi-actions, however, they were developed for solving more general problems [14], but not particularly for the multi-action problem. Reinforcement learning in multidimensional continuous action spaces [8] considered an MDP with an N-dimensional action space, thus provided an effective approach for learning in domains with multidimensional control variables. To the best of our knowledge, few studies tried to utilize the relationship among multiple actions.

In this paper, noticing that actions in a task usually have a significant relationship, we explore utilizing multi-action relationship to improve the learning performance. Inspired by the supervised multi-task learning [16, 15] in *supervised learning*, where the relationship among labels can be learned through a regularization term, we propose to capture the relationship among multiple actions by enforcing a regularization term. Specifically, we model the relationships between actions in a nonparametric manner as a covariance matrix. By utilizing a matrix-variate normal distribution [5] as a prior on learning parameters, we incorporate the regularization term into the *least-square policy-iteration* and the *temporal-difference methods*, which result efficiently solvable convex learning objectives. We then conduct experiments on a 2-action Grid world task, 3-action unmanned aerial vehicle task, and a 3-action helicopter hovering task. Experiment results show that incorporating multi-action relationship can effectively improve the learning performance.

The rest of this paper is organized into 5 sections: Section 2 introduces the background; Section 3 presents the proposed methods; The experiment results are reported in Section 4, and Section 5 concludes this paper.

2 Background

This section introduces the notation in the paper and provides further background on the standard framework for solving reinforcement learning problems and then formalizes the problem setting considered in the paper.

2.1 Reinforcement learning

Based on *Markov Decision Process* (MDP) (S, A, T, R, γ), where S and A are finite sets of states and actions, T a transition function, R a reward function and γ the discount factor, the reinforcement learning (RL) agent's goal is to construct an optimal policy, a mapping from states to actions, that maximizes the expected cumulative reward. Most learning algorithms compute optimal policies by learning value functions, which represent an estimate of how good it is for the agent to be in a certain state (or how good it is to perform a certain action in that state) [13]. In this paper, we consider the state value function and state-action value function defined over all states and all possible combinations of states and actions. The state value function indicates the expected, discounted, total reward thereafter policy π^* while the state-action value function indicates

the same reward when taking action a in state s. The value functions can be expressed by recursion according to *Bellman equations*:

$$Q^*(s,a) = \sum_{s'} T(s,a,s')(R(s,a,s') + \gamma \max_{a'} Q^*(s',a')), \tag{1}$$

$$V^*(s) = \sum_{s'} T(s,\pi^*(s),s')(R(s,a,s') + \gamma V^*(s')). \tag{2}$$

Since RL is primarily concerned with learning an optimal policy for an MDP assuming that a (perfect) model is not available, RL can be regarded as model free solution techniques. The transition and reward models are iteratively learned from interaction with the environment when model-based RL methods are used. In contrast, model-free RL methods, which we employ in this paper, step right into estimating values for actions, without even estimating the model of the MDP.

However, in the problem of control, more specifically, MDPs with continuous states and continuous or discrete actions, our aim is to learn an approximation of the optimal policy. Consequently, we can estimate π^* directly, or estimate Q^* to approximate π^* indirectly, or even estimate R and T to construct Q^* and π^* when needed. In this paper, we solve the continuous MDP by applying value approximation which uses samples to approximate Q^* directly. When updating the approximations of value functions, we apply the temporal-difference learning (TD-learning) [12] algorithms to bring some value in accordance to the immediate reward and the estimated value of the next state or the state-action pair.

Modeling concurrent decision making, [9] cast insight in a very general setting, inherited from the ability of *Semi-Markov Decision Process* (SMDP) to model a large class of decision making problems. This is a more realistic view of the agent environment interaction where the agent may not have access to the complete model of the environment. In order to realize this general setting in MDPs, we consider the thought of recursive decomposition of the action space proposed by [8], and decompose the high-dimensional action spaces into a set of sub-action spaces according to the agent's inner structure or physical laws (e.g., degree of freedom) in advance. Thus, the agent can learn the relationships between sub-actions that help the agent improve the future performance.

In the following subsections, we introduce two typical reinforcement learning algorithms. Our work will adapt these algorithms for the multi-action setting.

2.2 LSPI with value function approximation

Given the Bellman equations for state-action values, the reward can be expressed as:

$$r(\boldsymbol{s},a) = Q^\pi(\boldsymbol{s},a) - \gamma \mathbb{E}_{\pi(a'|\boldsymbol{s}')p(\boldsymbol{s}'|\boldsymbol{s},a))} \left[Q^\pi(\boldsymbol{s}',a') \right].$$

As illustrated in [11], if we approximate the state-action value Q by a linear function $\boldsymbol{\theta}^\top \boldsymbol{\phi}$, the immediate reward can be approximate as:

$$r(\boldsymbol{s},a) \approx \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{s},a) - \gamma \mathbb{E}_{\pi(a'|\boldsymbol{s}')p(\boldsymbol{s}'|\boldsymbol{s},a)} \left[\boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{s}',a') \right],$$

where $\boldsymbol{\theta}$ denotes the adaptable parameter vector and $\boldsymbol{\phi}(\boldsymbol{s}, a)$ is the feature vector in state $\boldsymbol{s}$ when taking action a. Thus, a new basis function vector is defined as:

$$\boldsymbol{\psi}(\boldsymbol{s}, a) \approx \boldsymbol{\phi}(\boldsymbol{s}, a) - \gamma \mathbb{E}_{\pi(a'|\boldsymbol{s}')p(\boldsymbol{s}'|\boldsymbol{s},a)} \left[\boldsymbol{\phi}(\boldsymbol{s}', a')\right],$$

and the expected immediate reward $r(\boldsymbol{s}, a)$ is be approximated as:

$$r(\boldsymbol{s}, a) \approx \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\psi}(\boldsymbol{s}, a). \tag{3}$$

The linear approximation problem of state-action value function $Q^{\pi}(\boldsymbol{s}, a_i)$ can be reformed into immediate-reward regression problem according to Eq.(3), and further can be solved by learning $\boldsymbol{\theta}$ in the least-squares framework:

$$\min_{\boldsymbol{\theta}} \frac{1}{|\boldsymbol{D}|} \sum_{t=1}^{|\boldsymbol{D}|} \left(r_t - \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\psi}^t\right)^2 + C\|\boldsymbol{\theta}\|_2, \tag{4}$$

where $\boldsymbol{D}$ is the source of samples. Overall, the *LSPI* algorithm is summarized in Algorithm 1 [7].

Algorithm 1 LSPI

Initialize Initialize a policy π, given as $\boldsymbol{\Theta} = \mathbf{0}$
Feature vectors are denoted as $\boldsymbol{\Phi}$
repeat
  Set training set $D = \emptyset$
  Initialize state $\boldsymbol{s}_0$
  for $t = 1$ to T **do**
    Choose action $\boldsymbol{a}$ using policy π, observe $\boldsymbol{s}'_t$ and $\boldsymbol{r}_t = (r_t^1, ..., r_t^N)$
    Add the training tuple $(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{r}_t, \boldsymbol{s}'_t)$ to training set D
  end for
  Given training set D, update $\boldsymbol{\Theta}$ by solving objective function Eq.(4)
  Update policy π by $\boldsymbol{\Theta}$
until $\boldsymbol{\Theta}$ converges.

2.3 Gradient temporal-difference learning

Concluded from standard *temporal-difference learning* (*TD-learning*), the tabular TD-learning update is [6],

$$V(s_t) = V(s_t) + \alpha_t \cdot \delta_t,$$

where $\delta_t = r_{t+1} - (V(s_t) - \gamma V(s_{t+1}))$ is the one-step temporal-difference error, and $\alpha_t \in [0, 1]$ is a step-size parameter.

When V_t is expressed as $V(s_t) = \boldsymbol{\theta}^{\top} \boldsymbol{\phi}(s_t)$, the parameter of V_t can be solved by minimizing the following equation,

$$\mathrm{E}(s_t) = \frac{1}{2}(\delta_t)^2 = \frac{1}{2}(r_{t+1} - (V(s_t) - \gamma V(s_{t+1})))^2$$

and the gradient with respect to the parameters is

$$-\nabla_{\boldsymbol{\theta}}\mathrm{E}(s_t,\boldsymbol{\theta}) = -(r_{t+1}+\gamma V(s_{t+1})-V(s_t))\cdot\nabla_{\boldsymbol{\theta}}(r_{t+1}+\gamma V(s_{t+1})-V(s_t)).$$

Treating $r_{t+1}+\gamma V(s_{t+1})$ to be irrelevant with $\boldsymbol{\theta}$ [6], so that the final gradient is:

$$\begin{aligned} -\nabla_{\boldsymbol{\theta}}\mathrm{E}(s_t,\boldsymbol{\theta}) &= -(r_{t+1}+\gamma V(s_{t+1})-V(s_t))\cdot\nabla_{\boldsymbol{\theta}}V(s_t) \\ &= -(r_{t+1}+\gamma\boldsymbol{\theta}^\top\boldsymbol{\phi}(s_{t+1})-\boldsymbol{\theta}^\top\boldsymbol{\phi}(s_t))\cdot\boldsymbol{\phi}(s_t) \end{aligned} \tag{5}$$

Then the parameters $\boldsymbol{\theta}$ is updated by adding this negative gradient with a step size coefficient.

Algorithm 2 Gradient TD Learning

Initialize: Feature function: $\boldsymbol{\phi}$
Step size: α
Parameters: $\boldsymbol{\theta}=\mathbf{0}, t=0$
for each episode **do**
  set $\boldsymbol{\delta}=\mathbf{0}$
  Initialize $\boldsymbol{s}_0$
  repeat
    Take action from $\boldsymbol{\theta}$ with exploration, observe $\boldsymbol{s}'$ and r_t
    set $\boldsymbol{\delta}=\boldsymbol{\phi}(r_t+(\boldsymbol{\phi}'-\boldsymbol{\phi})^{\mathrm{T}}\boldsymbol{\theta})$ by Eq.(5)
    $\boldsymbol{\theta}=\boldsymbol{\theta}+\alpha\boldsymbol{\delta}$
    $\boldsymbol{s}\leftarrow\boldsymbol{s}'$
  until $\boldsymbol{s}$ is terminal
end for

3 Multi-action Relationship Learning

3.1 Problem setting

A multi-action reinforcement learning problem domain $\mathcal{D}$ is defined as an MDP $M(S,A,T,R,\gamma)$. The MDP is as usual, only that the action set $\mathcal{A}=\mathcal{A}_1\times\mathcal{A}_2\times\ldots\times\mathcal{A}_N$ is multi-dimensional, where each $\mathcal{A}_i$ is a sub-action set. To solve this MDP, an agent can treat each dimensional action as a separated MDP, and learn multiple MDPs simultaneously. Suppose the agent's action space $\mathcal{A}$ is decomposed into N sub-action spaces, $\mathcal{A}_1,\mathcal{A}_2,\ldots,\mathcal{A}_N$. Each sub-action space has a value function to be approximated.

We consider the value function approximation by approximating the state-action value function $Q^\pi(\boldsymbol{s},a)$ in linear model:

$$Q^\pi(\boldsymbol{s},a)=\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{s},a), \tag{6}$$

where $\boldsymbol{\theta}$ denotes the adaptable parameter vector and $\boldsymbol{\phi}$ is the feature vector in state s when taking action a. Thus, since there are N sub-action spaces, we formalize N state-action value functions according to Eq.(6). For the i-th sub-action, the state-action value

function $Q_i^\pi(\boldsymbol{s}, a_i)$ is approximated by the following vector representation:

$$\boldsymbol{\theta}_i^{\mathrm{T}} \boldsymbol{\phi}_i(\boldsymbol{s}, a_i).$$

However in many applications, these actions usually have a significant relationship, learning separated MDPs ignores the relationship. Inspired by the success in multi-task supervised learning [16], where the relationship among tasks is essential and helpful, we hypothesize that explore the relationship among actions in multi-action reinforcement learning might also be helpful.

However, unlike in multi-task supervised learning, where the relationship among tasks is often assumed, the relationship among actions can be very different in problems, and thus must be learned. In this work, we adopt the regularization idea from [16], and propose to learn and utilize the action relationship as follows:

1. assume that all sub-actions are unrelated initially;
2. receive state-action pairs according to current policy π;
3. for each sub-action space, learn the value function using the multi-action relationship, and update the relationship;
4. loop from step 2.

In the following two subsections, we implement the idea into two reinforcement learning algorithms.

3.2 LSPI with multi-action relationship regularization

Based on Section 2.2, the immediate reward $r_i(\boldsymbol{s}, a_i)$ can be approximated as:

$$r_i(\boldsymbol{s}, a_i) \approx \boldsymbol{\theta}_i^{\mathrm{T}} \boldsymbol{\psi}_i(\boldsymbol{s}, a_i).$$

The linear approximation problem of state-action value function $Q_i^\pi(\boldsymbol{s}, a_i)$ can be reformed into immediate-reward regression problem and it can be solved by learning $\boldsymbol{\theta}_i$ in the least-squares framework with multi-action regularization.

We assume that the likelihood for r_t^i at time t, given $\boldsymbol{\psi}_i(\boldsymbol{s}_t, a_t^i)$, $\boldsymbol{\theta}_i$ and ϵ_i can be modeled as:

$$r_t^i | \boldsymbol{\psi}_i(\boldsymbol{s}_t, a_t^i), \boldsymbol{\theta}_i, \epsilon_i \sim \mathcal{N}(\boldsymbol{\theta}_i^{\mathrm{T}} \boldsymbol{\psi}_i(\boldsymbol{s}_t, a_t^i), \epsilon_i^2),$$

where $\mathcal{N}(\boldsymbol{m}, \Sigma)$ denotes the multivariate normal distribution with mean $\boldsymbol{m}$ and covariance matrix Σ.

Inspried by [16], the prior on $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N)$ can be defined as:

$$\boldsymbol{\Theta} | \epsilon_i \sim \left(\prod_{i=1}^{N} \mathcal{N}(\boldsymbol{\theta}_i | \mathbf{0}_d, {\epsilon_i}^2 \boldsymbol{I}_d) \right) q(\boldsymbol{\Theta}),$$

where $\boldsymbol{I}_d$ is the $d \times d$ identity matrix, $\mathbf{0}_d$ is the $d \times 1$ zero vector. While the first term is to penalize the complexity of each column of $\boldsymbol{\Theta}$ separately, the second term is to model the structure of $\boldsymbol{\Theta}$ and is characterized by a matrix-variate distribution:

$$q(\boldsymbol{\Theta}) = \mathcal{MN}_{d \times N} \left(\boldsymbol{\Theta} | \mathbf{0}_{d \times N}, \boldsymbol{I}_d \otimes \boldsymbol{\Omega} \right).$$

$\mathcal{MN}(\boldsymbol{M}, \boldsymbol{A} \otimes \boldsymbol{B})$ represents the matrix-variate normal distribution, where row covariance matrix $\boldsymbol{A}$ models the relationships between features and column covariance matrix $\boldsymbol{B}$ models the relationships between each $\boldsymbol{\theta}_i$. Thus, $\boldsymbol{\Omega}$ models the relationships between multi-action.

As a result, the posterior distribution for $\boldsymbol{\Theta}$ is proportional to the product of the prior and the likelihood function:

$$p\left(\boldsymbol{\Theta}|\boldsymbol{\Psi}(\boldsymbol{s},\boldsymbol{a}),\boldsymbol{r},\boldsymbol{\epsilon},\boldsymbol{\Omega}\right) \propto p\left(\boldsymbol{r}|\boldsymbol{\Psi}(\boldsymbol{s},\boldsymbol{a}),\boldsymbol{\Omega},\boldsymbol{\epsilon}\right) \cdot p\left(\boldsymbol{\Theta}|\boldsymbol{\epsilon},\boldsymbol{\Omega}\right) \tag{7}$$

where $\boldsymbol{r} = \left(r_1^1, \ldots, r_T^1, \ldots, r_1^N, \ldots, r_T^N\right)^{\mathrm{T}}$, $\boldsymbol{\Psi}(\boldsymbol{s},\boldsymbol{a})$ denotes the basis function of all sub-actions. Taking the negative logarithm of Equation 7, the agent obtain the maximum a posterior (MAP) estimation of $\boldsymbol{\Theta}$, and the maximum likelihood estimation (MLE) of $\boldsymbol{\Omega}$ by solving the following problem:

$$\min_{\boldsymbol{\Theta},\boldsymbol{\Omega}\succeq\mathbf{0}} \sum_{i=1}^{N} \frac{1}{\epsilon_i^2} \sum_{t=1}^{T} \left(r_t^i - \boldsymbol{\theta}_i^{\mathrm{T}} \boldsymbol{\psi}_i^t\right)^2 + \sum_{i=1}^{N} \frac{1}{\epsilon_i^2} \boldsymbol{\theta}_i^{\mathrm{T}} \boldsymbol{\theta}_i + \mathrm{tr}(\boldsymbol{\Theta}\boldsymbol{\Omega}^{-1}\boldsymbol{\Theta}^{\mathrm{T}}) + d\ln(|\boldsymbol{\Omega}|)$$

where $\mathrm{tr}(\cdot)$ denotes the trace of a square matrix, $|\cdot|$ denotes the determinant of a square matrix, and $\boldsymbol{\Omega} \succeq \mathbf{0}$ means that the matrix $\boldsymbol{\Omega}$ is positive semidefinite due to the fact that $\boldsymbol{\Omega}$ is defined as a covariance matrix.

The squared loss and the revised regularization are expressed as:

$$\begin{aligned} \min_{\boldsymbol{\Theta},\boldsymbol{\Omega}} & \sum_{i=1}^{N} \frac{1}{T^2} \sum_{t=1}^{T} \left(r_t^i - \boldsymbol{\theta}_i^{\mathrm{T}} \boldsymbol{\psi}_i^t\right)^2 + \frac{\lambda_1}{2} \mathrm{tr}(\boldsymbol{\Theta}\boldsymbol{\Theta}^{\mathrm{T}}) + \frac{\lambda_2}{2} \mathrm{tr}(\boldsymbol{\Theta}\boldsymbol{\Omega}^{-1}\boldsymbol{\Theta}^{\mathrm{T}}) \\ \text{s.t.} \quad & \boldsymbol{\Omega} \succeq \mathbf{0} \\ & \mathrm{tr}(\boldsymbol{\Omega}) \leq 1, \end{aligned} \tag{8}$$

where $\lambda 1$ and $\lambda 2$ are regularization parameters.

Since Equation 8 is jointly convex with respect to $\boldsymbol{\Theta}$ and $\boldsymbol{\Omega}$, we can optimize the objective function with respect to $\boldsymbol{\Theta}$ when $\boldsymbol{\Omega}$ is fixed, and then optimize the objective function with respect to $\boldsymbol{\Omega}$ when $\boldsymbol{\Theta}$ is fixed alternatively. Then, the parameter $\boldsymbol{\Theta}$ and multi-action relationship matrix $\boldsymbol{\Omega}$ are updated simultaneously to improve the agent's performance eventually. And the LSPI with multi-action relationship regularization term is described in Algorithm 3.

3.3 Gradient TD learning with multi-action relationship regularization

Generalized from Section 2.3, in standard *temporal-difference learning* (*TD-learning*), negative gradient can be expressed as:

$$-\nabla_{\boldsymbol{\theta}} \mathrm{E}(s_t, \boldsymbol{\theta}) = -(r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)) \cdot \nabla_{\boldsymbol{\theta}} V_t(s_t).$$

When incorporating the multi-action relationship regularization into gradient temporal-difference learning, we assume that the ith sub-action space maintains the state-action

Algorithm 3 LSPI with Multi-action Relationship Regularization

Initialize Feature functions $\boldsymbol{\Psi}$
Initialize a policy π, given as $\boldsymbol{\Theta} = \mathbf{0}$
Initialize the relationship matrix $\boldsymbol{\Omega} = \frac{1}{N} I_N$
repeat
Set training set $D = \emptyset$
Initialize state $\boldsymbol{s}_0$
for $t = 1$ to T **do**
Choose action $\boldsymbol{a}$ using policy π, observe $\boldsymbol{s}'_t$ and $\boldsymbol{r}_t = (r_t^1, ..., r_t^N)$
Add the training tuple $(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{r}_t, \boldsymbol{s}'_t)$ to training set D
end for
Given training set D, update $\boldsymbol{\Theta}$ by solving objective function Eq.(8)
Update $\boldsymbol{\Omega}$ by Eq.(11)
Update policy π by $\boldsymbol{\Theta}$
until $\boldsymbol{\theta}$ converges.

value function $\mathrm{Q}_t^i(\boldsymbol{s}_t, a_t^i) = \boldsymbol{\theta}_i^{\mathrm{T}} \boldsymbol{\phi}_i(\boldsymbol{s}_t, a_t^i)$ at time t. The TD error can be rewritten into

$$\mathrm{E}(\boldsymbol{s}_t) = \sum_{i=1}^{N} \left(r_t^i + \gamma \boldsymbol{\theta}_i^{\mathrm{T}} \boldsymbol{\phi}_t^i(\boldsymbol{s}_{t+1}, a_{t+1}^i) - \boldsymbol{\theta}_i^{\mathrm{T}} \boldsymbol{\phi}_t^i(\boldsymbol{s}_t, a_t^i)\right)^2 + \frac{\lambda_1}{2} \mathrm{tr}(\boldsymbol{\Theta}\boldsymbol{\Theta}^{\mathrm{T}}) + \frac{\lambda_2}{2} \mathrm{tr}(\boldsymbol{\Theta}\boldsymbol{\Omega}^{-1}\boldsymbol{\Theta}^{\mathrm{T}}).$$

Following the method in Section 2.3, the agent can also obtain a *maximum a posterior* (MAP) estimation of $\boldsymbol{\Theta}$, and a *maximum likelihood estimation* (MLE) of $\boldsymbol{\Omega}$ by solving the following problem:

$$\begin{aligned} \min_{\boldsymbol{\Theta}, \boldsymbol{\Omega}} & \sum_{i=1}^{N} \left(r_t^i + \gamma \boldsymbol{\theta}_i^{\mathrm{T}} \boldsymbol{\phi}_t^i(\boldsymbol{s}_{t+1}, a_{t+1}^i) - \boldsymbol{\theta}_i^{\mathrm{T}} \boldsymbol{\phi}_t^i(\boldsymbol{s}_t, a_t^i)\right)^2 + \\ & \frac{\lambda_1}{2} \mathrm{tr}(\boldsymbol{\Theta}\boldsymbol{\Theta}^{\mathrm{T}}) + \frac{\lambda_2}{2} \mathrm{tr}(\boldsymbol{\Theta}\boldsymbol{\Omega}^{-1}\boldsymbol{\Theta}^{\mathrm{T}}) \\ \text{s.t.} \quad & \boldsymbol{\Omega} \succeq \mathbf{0} \\ & \mathrm{tr}(\boldsymbol{\Omega}) \leq 1. \end{aligned} \tag{9}$$

This updating process is implemented in every learning step to estimate $\boldsymbol{\Theta}$ and $\boldsymbol{\Omega}$ alternatively.

When optimizing $\boldsymbol{\Theta}$ when $\boldsymbol{\Omega}$ is fixed, we can formulate the optimization problem as

$$G = \sum_{i=1}^{N} \left(r_t^i + \gamma \boldsymbol{\theta}_i^{\mathrm{T}} \boldsymbol{\phi}_t^i(\boldsymbol{s}_{t+1}, a_{t+1}^i) - \boldsymbol{\theta}_i^{\mathrm{T}} \boldsymbol{\phi}_t^i(\boldsymbol{s}_t, a_t^i)\right)^2 + \lambda_1 \mathrm{tr}(\boldsymbol{\Theta}\boldsymbol{\Theta}^{\mathrm{T}}) + \lambda_2 \mathrm{tr}(\boldsymbol{\Theta}\boldsymbol{\Omega}^{-1}\boldsymbol{\Theta}^{\mathrm{T}}).$$

Thus the gradient of G with respect to $\boldsymbol{\Theta}$ is

$$\begin{aligned}\frac{\partial \mathrm{G}}{\partial \boldsymbol{\Theta}} = -2\sum_{i=1}^{N}\left(r_t^i + \gamma\boldsymbol{\theta}_i^{\mathrm{T}}\boldsymbol{\phi}_t^i(\boldsymbol{s}_{t+1}, a_{t+1}^i) - \boldsymbol{\theta}_i^{\mathrm{T}}\boldsymbol{\phi}_t^i(\boldsymbol{s}_t, a_t^i)\right)\cdot \\ \boldsymbol{\phi}_t^i\boldsymbol{e}_i^{\mathrm{T}} + 2\boldsymbol{\Theta}(\lambda_1\boldsymbol{I}_N + \lambda_2\boldsymbol{\Omega}^{-1}),\end{aligned} \tag{10}$$

where each $\boldsymbol{e}_i$ is the ith column vector of $\boldsymbol{I}_N$.

Then we can optimize the $\boldsymbol{\Omega}$ when $\boldsymbol{\Theta}$ is fixed according to [16]:

$$\boldsymbol{\Omega} = \frac{(\boldsymbol{\Theta}^{\mathrm{T}}\boldsymbol{\Theta})^{\frac{1}{2}}}{\mathrm{tr}((\boldsymbol{\Theta}^{\mathrm{T}}\boldsymbol{\Theta})^{\frac{1}{2}})}. \tag{11}$$

Turning this into a control method by always updating the policy to be greedy with respect to the current estimate can be concluded as Algorithm 4.

Algorithm 4 Gradient TD Learning with Multi-action Relationship Regularization

Initialize Feature function $\boldsymbol{\Phi}$
Step sizes $\boldsymbol{\alpha}$
Parameters $\boldsymbol{\Theta}, t = 0$
for each episode **do**
  $\boldsymbol{\delta} = \mathbf{0}$
  Initialize $\boldsymbol{s}$
  repeat
    Take action $\boldsymbol{a}$ from $\boldsymbol{\theta}$ with exploration, observe $\boldsymbol{s}'$ and $\boldsymbol{r}_t$
    set $\boldsymbol{\delta} = (\boldsymbol{r_t} + (\boldsymbol{\Phi}'-\boldsymbol{\Phi})^{\mathrm{T}}\boldsymbol{\Theta})\frac{\partial \mathrm{G}}{\partial \boldsymbol{\Theta}}$ (Equation 10)
    $\boldsymbol{\Theta} = \boldsymbol{\Theta} + \boldsymbol{\alpha}\boldsymbol{\delta}$
    Update $\boldsymbol{\Omega}$ using Equation 11
    $\boldsymbol{s} \leftarrow \boldsymbol{s}'$
  until $\boldsymbol{s}$ is terminal
end for

3.4 Discussions

In general, we set the initial value of $\boldsymbol{\Omega}$ to $\frac{1}{N}\boldsymbol{I}_N$, which is corresponding to the assumption that all sub-actions are unrelated initially. However, in some specific domains, there is some prior knowledge about the relationships between some sub-actions. When this happens, the corresponding $\boldsymbol{\Omega}$ can be represented as equality relations between elements in $\boldsymbol{\Omega}$.

4 Experiments

In this section, we study the multi-action relationship learning in several typical reinforcement learning domains and compare it with other multi-action algorithms which can also be used in multi-action reinforcement learning.

4.1 2-Action GridWorld

The GridWorld domain simulates a path-planning problem for a mobile robot in an environment with obstacles. The goal of the agent is to navigate from the starting point to the goal state. We decompose the action orthogonally into its horizontal and vertical components. In this way, each component represents a sub-action space consisting of three sub-actions: forward, backward, statically.

Thus, there are 2 sub-action spaces, each of them maintains 3 legal sub-actions. We compare the multi-action relationship learning algorithm with normal *LSPI*.

To evaluate the agent's performance, we report the total reward per episode averaged over the test set and the multi-action correlation matrix. The multi-action correlation matrix shows that the two sub-action spaces are correlated closely. On maps of 6-by-9, based on *LSPI*, we observe that learning with the multi-action relationship can converges faster than ignoring the relationship, as showed in Figure 1(a).

4.2 PST

This domain concerns Persistent Search and Track mission with multiple unmanned aerial vehicle (UAV) agents. The goal is to perform surveillance and communicate it back to base in the presence of stochastic communication and health (overall system functionality) constraints, without losing any UAVs because of running out of fuel. Each UAV has 4 state dimensions: position of a UAV; integer fuel qty remaining; actuator status and sensor status. Each UAV can take one of 3 actions: retreat, loiter, advance.

Namely, this domain can be regarded as a centralized multi-agent system with 3 sub-action spaces (3 UAV agents). Based on *Gradient TD learning*, we evaluate the performance of the PST system by the total rewards that the agents can obtain.

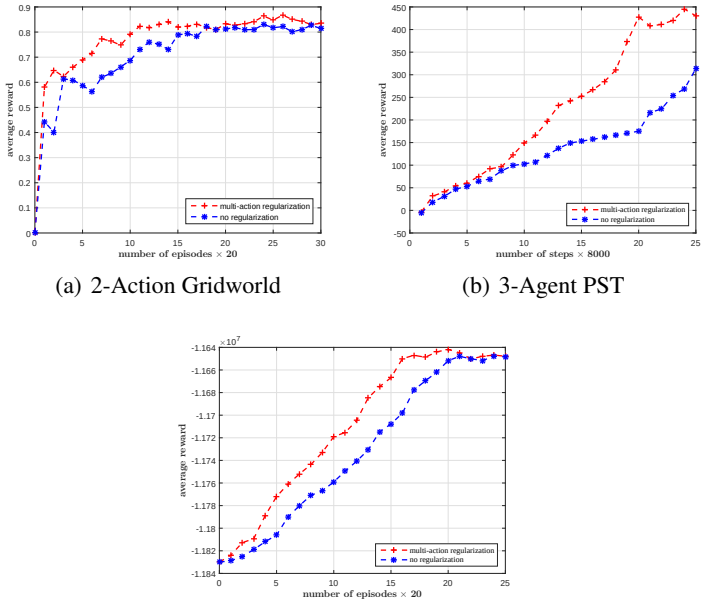
To demonstrate the importance of our idea, the average reward in the first 200 thousand steps are showed in Figure 1(b). The result shows that multi-action relationship learning is significantly better than ignoring the relationship, as the steps increases.

4.3 Helicopter

An implementation of a simulator that models one of the Stanford autonomous helicopters (an XCell Tempest helicopter) in the flight regime close to hover is represented in this domain. Some pilots consider hovering the most challenging aspect of helicopter flight. Generally, the pilot's use of control inputs in a hover is as follows: the cyclic is used to eliminate drift in the horizontal plane; the collective is used to maintain desired altitude; and the tail rotor (or anti-torque system) pedals are used to control nose direction or heading [1]. It is the interaction of these controls that can make learning to hover difficult, since often an adjustment in any one control requires the adjustment of the other two. We decompose the action orthogonally into its forward, sideways and downward components.

Analogically, this domain can be regarded as a centralized control system with 3 sub-action spaces when conducting *Gradient TD learning*. It can be observed that tracking the relationships between helicopter's 3 DOFs outperforms ignoring the relationship, as it converges fasters shown in Figure 1(c).

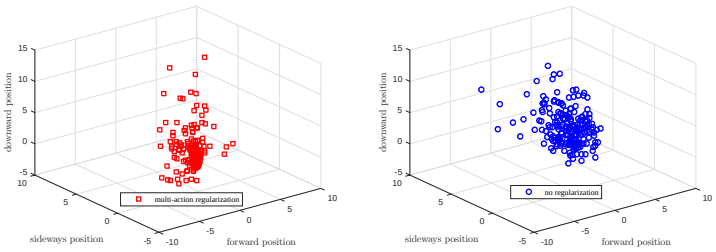
To demonstrate the learning effect intuitively, the helicopter's attitude is recorded throughout the two learning processes. As Figure 2 shows, the helicopter reached to a relatively stable position $(0, 0, 0)$ more quickly. Each point in the Figure 2 denotes the helicopter's position during the experiment. Apparently, learning the multi-action relationship can improve the performance in the long run and with less fluctuation.



(a) 2-Action Gridworld (b) 3-Agent PST

(c) 3-Action Helicopter

Fig. 1. Average rewards on 2-Action Gridworld, 3-Agent PST and 3-Action Helicopter.

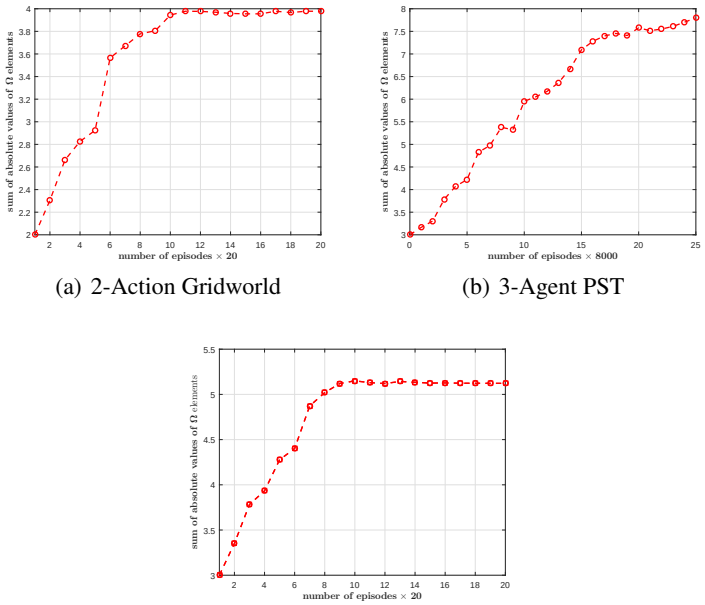


(a) multi-action regularization (b) no multi-action regularization

Fig. 2. Flight positions on 3-Action Helicopter Hover problem.

4.4 On the multi-action relationship

During the experiments described above, we track the relationships matrix's changing by using the sum of absolute values of $\boldsymbol{\Omega}$ elements. It can be observed from Figure 3 that the multi-action correlations are learned along with the reinforcement learning. We can also note that, as long as the correlation is growing, the performance of multi-action methods are superior, by cross-comparison with Figure 3(a), 3(b) and 3(c).



(a) 2-Action Gridworld (b) 3-Agent PST

(c) 3-Action Helicopter

Fig. 3. Sum of Absolute Values of $\boldsymbol{\Omega}$ Elements.

5 Conclusion

In this paper, we explore multi-action relationship in learning multi-action reinforcement learning, where multiple actions are required to be controlled simultaneously. By employing the matrix-variate distribution, we enforce a regularization term capturing the multi-action relationship and obtain efficiently solvable convex learning objectives for value function approximation reinforcement learning algorithms. We incorporate the regularization term into the *LSPI* and the *Gradient TD methods*, which are empirically demonstrated to improve the learning performance. In the future, we will explore the multi-action relationship in more reinforcement learning approaches.

References

[1] Abbeel, P., Ganapathi, V., Ng, A.Y.: Learning vehicular dynamics, with application to modeling helicopters. In: Weiss, Y., Schölkopf, B., Platt, J. (eds.) Advances in Neural Information Processing Systems 18, pp. 1–8. MIT Press, Cambridge, MA (2005)

[2] Cheng, G., Hyon, S.H., Morimoto, J., Ude, A., Hale, J.G., Colvin, G., Scroggin, W., Jacobsen, S.C.: Cb: A humanoid research platform for exploring neuroscience. In: Proceedings of the 6th IEEE-RAS International Conference on Humanoid Robots. pp. 182–187. Genova, Italy (2006)

[3] Dorigo, M., Gambardella, L.M.: Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1(1), 53 – 66 (1997)

[4] Fumihide, T., Masayuki, Y.: Multitask reinforcement learning on the distribution of mdps. In: Proceedings of the 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation. pp. 1108–1113. Kobe, Japan (2003)

[5] Gupta, A.K., Nagar, D.K.: Matrix Variate Distributions. Chapman and Hall/CRC, Florida (1999)

[6] van Hasselt, H.: Reinforcement learning in continuous state and action spaces (2012)

[7] Lagoudakis, M.G., Parr, R.: Least-squares policy iteration. Journal of Machine Learning Research 4, 1107–1149 (2003)

[8] Pazis, J., Lagoudakis, M.G.: Reinforcement learning in multidimensional continuous action spaces. In: Proceedings of the 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning. Paris, France (2011)

[9] Rohanimanesh, K.: Concurrent Decision Making in Markov Decision Processes. Ph.D. thesis, University of Massachusetts Amherst (2006)

[10] Silver, D., Sutton, R.S., Müller, M.: Temporal-difference search in computer go. Machine Learning 87(2), 183–219 (2012)

[11] Sugiyama, M.: Statistical Reinforcement Learning: Modern Machine Learning Approaches. Chapman and Hall/CRC, Florida (2015)

[12] Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press, Cambridge, Massachusetts (1998)

[13] Wiering, M., van Otterlo, M. (eds.): Reinforcement Learning: State-of-the-Art. Springer, Berlin (2012)

[14] Wunder, M., Littman, M.L., Babes, M.: Classes of multiagent q-learning dynamics with ε-greedy exploration. In: Proceedings of the 27th International Conference on Machine Learning. pp. 1167–1174. Haifa, Israel (2010)

[15] Zhang, M.L., Zhou, Z.H.: A review on multi-label learning algorithms. IEEE Transactions on Knowledge and Data Engineering 26(8), 1819–1837 (2014)

[16] Zhang, Y., Yeung, D.Y.: A regularization approach to learning task relationships in multitask learning. ACM Transactions on Knowledge Discovery from Data 8(3), 1–31 (2014)