Human-Machine Coadaptation Based on Reinforcement Learning with Policy Gradients

Karim A. Tahboub

Abstract— **The problem of adaptive human-machine interaction is investigated. It is sought that not only the human learns how to perform a task with a novel machine, but the machine itself co-adapts to the human style in the interaction. This requires solving the problem of two agents co-adapting or co-learning at the same time. Due to the lack of human learning and performance models, it is hypothesized that reinforcement learning with policy gradient algorithms are good candidates for addressing this problem with robustness and fast convergence.**

I. INTRODUCTION

Classically in human-machine interaction (HMC), the burden of learning the interface lies completely at the side of the human. The human operator of a novel machine is instructed how to operate/interact with the machine and then starts, through practice, to learn the interface and to adapt his/her sensorimotor controls to optimize the interaction. In some cases, the interfacing modality is redundant at the side of the human. For example, one can use two 3-degree-of-freedom joysticks to manipulate the three-dimensional position of a robot end effector. This leaves the human operator with more degrees of freedom than the task requires. This redundancy could cause confusion and impose difficulty to the operator. However, if the interface, or more specifically the mapping of the human commands to the by-the-machine interpreted human desired commands is designed to be adaptive, then the concept of human-machine co-adaptation and co-learning can be realized. This makes the machine to appear more intelligent and friendlier to the user.

This approach of adaptive HMC – in contrast to static HMC interaction where the function allocation and the interface design does not vary in run-time (see e.g. [1]) – tries to shift the burden from operating a machine from the user to the machine itself by an online allocation of the functions in questions (see e.g., [2]). Parasuraman [1] identified five main categories of techniques for implementing adaptive automation logic (1) on the basis critical events, (2) on operator/user performance measurements, (3) on operator/user physiological assessments, (4) on operator/user modeling, and (5) on hybrid methods combining one or more of these techniques. While these classical approaches to adaptive automation mainly use these techniques to allocate functions to either the human user or the machine at hand, the research at hand dynamically adapts the interfacing devices or modalities. There has been a preliminary research work on this [3]. It has been shown that humans are capable of internally representing different spaces (Euclidian and non-Euclidean) and perform efficient transformations on these spaces. For this application, it is necessary to consider the user redundant command space (non-Euclidean) and the robot Euclidian physical space. The map between these two spaces (where classically it is the responsibility of the user to learn it statically) can be adapted to the user's style in such a way to minimize efforts in the null space. In a later study, Danziger et al. [4] experimented with two machine-learning algorithms to adaptively change a transformation between finger motion recorded by a glove and a simulated robot arm. The two algorithms tested are LMS gradient descent and Moore-Penrose (MP) pseudo-inverse transformation. The goal was to change the transformation in such a way to reduce the endpoint errors measured in past performance. It is reported in that work [4] that the MP group performed worse than the control group while the LMS group outperformed the control subjects; however even this LMS group failed to achieve better generalization than the control subjects.

Traditional adaptive systems have been investigated along three directions [5]: (1) Adaptive interfaces or user-adaptive systems: a system adapts itself to changes in user-related or environmental characteristics. Such automatic adaptation is based on the evaluation of user behavior, assumes user needs, and takes explicit user inputs into account. The major concern is between an individual user and the machine. (2) Adaptive workflows: a system provides workflows that are appropriate to the working environment in order to assist users to work properly to accomplish their tasks. The major consideration is on functions. (3) Adaptive computing or adaptive services: a system adjusts its components, configurations, and structures based on dynamic situations to provide highly-available and highly-efficient computing services. The systems should be robust and efficient in the presence of changes [5].

However, the topic of human-machine coadaptation has been the subject of only a few studies. For example, in [6], the authors proposed a mechatronic adaptive architecture that corresponds to the human performance but relies on [6]:

1. Modeling human and machine dynamics. Especially the variable constraints should be considered.
2. Modeling the operation base on skill, rule, knowledge, decision combining event and dynamical systems.
3. Modeling of the psycho-physical characteristics of human operator
4. Development of Mechatronic systems supporting human operation not only assisting control action but also

K. A. Tahboub is with the Mechanical Engineering Department of the Palestine Polytechnic University, Hebron, Palestine (mobile: +972 599 65 66 65; email: tahboub@ppu.edu).

978-1-7281-1938-0/19/$31.00 ©2019 IEEE

providing proper data and knowledge for understanding the situation and giving better decision.

It is clear from that work that the human operator and his/her performance needs to be modeled and identified online. Such an approach for dealing with coadaptation is thought to entail implementation difficulties especially regarding the stability and convergence of the adaptation scheme.

Recently, human-machine coadaptation has become an important issue when designing brain-machine interfaces for the sake of controlling prosthesis or external devices [7]. Ideally, interfaces with such machines should be as active and bidirectional as the interactions with other human beings or animals where the connection between the user and tool is such that both can experience the unique abilities of the counterpart [7]. Several research groups suggested different solutions to the machine adaptation problem. However, it is reported in [7] that these approaches are primarily data-driven techniques that seek out correlation and structure between the spatial-temporal neural activation and behavior. Further, it has also been shown that the time that it takes to achieve a certain level of ``mastery'' of the prosthetic device can be extremely slow especially when the details of the dynamics of control are unknown to the user. It is argued in [7] that the symbioses will be easier to define and implement if both the user and neuro-prosthetic share similar learning architectures. The authors [7] concluded that reinforcement learning (RL) became the natural choice since there is evidence that parts of the limbic system implement a reinforcement learning type of architecture [8].

II. Reinforcement Learning with Policy Gradients

Reinforcement learning is a general description of the learning problem where the aim of the learning agent is to maximize a long-term objective. The learning system consists of an agent which interacts with the environment via its actions at discrete timesteps and receives each time a reward. As a result of its action, the agent finds itself in a new state and accordingly acts in a feedback manner. The sequence of actions, *A*, states, *S*, and rewards, *R*, forms a trajectory. The reward is an abstraction of the agent's purpose or goals it tries to achieve and thus the objective of learning is to maximize the sum of rewards. So, it is imperative to formulate an appropriate award (a scalar) for a given problem.

A discounted Markov Decision Process can be thought of as a describing framework:

$$p(s', r|s, a) = \Pr\left[S_{t+1} = s', R_{t+1} = r|S_t = s,\ A_t = a\right]$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

where $S_t,\ S_{t+1} \in S$ (state space), $A_t,\ A_{t+1} \in A$ (action space), $R_t, R_{t+1} \in R$ (reward space), p defines the dynamics of the process which is not known to (outside the control of) the agent and G_t is the discounted return. This MDP defines the probability of transitioning into a new state, getting some reward given the current state and the execution of an action. Although the agent cannot control the dynamics of the process, it follows however a policy in choosing its actions to maximize the reward:

$$\pi_{\theta_t}(A_t = a|S_t = s)\ \forall A_t \in A(s), S_t \in S$$

where the policy π_{θ_t} is a probability distribution of actions given state and depending on a specific policy parameter vector θ_t. Accordingly, the reinforcement learning problem reduces here to how to update the parameter vector given a trajectory, τ. In other words, the reinforcement learning objective is to maximize the expected reward following the policy:

$$J(\theta) = E_\pi[r(\tau)]$$

A standard approach to solving this maximization problem is to employ gradient descent (ascent):

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t)$$

This imposes a challenge in finding the gradient of the objective which contains the expectation. The policy gradient theorem offers a computational solution [8]:

$$\nabla E_\pi[r(\tau)] = E_\pi[r(\tau)\nabla \log \pi_\theta(\tau)]$$

which applies the gradient to the known policy rather than to the objective. This can be rewritten as:

$$\nabla E_\pi[r(\tau)] = E_\pi\left[r(\tau)\left(\sum_{t=1}^{T} \nabla \log \pi_\theta(a_t|s_t)\right)\right]$$

This is an important applicable result as it is free of the process dynamics, *p,* which makes this approach model free.

III. Reinforce With Baseline Algorithm

The reinforce with baseline algorithm [9] is adopted here. The baseline, *b*, reduces the variance of the sampled trajectories while not biasing the gradient:

$$\nabla E_\pi[r(\tau)] = E_\pi\left[\left(\sum_{t=1}^{T} (G_t - \mathrm{b}) \nabla \log \pi_\theta(a_t|s_t)\right)\right]$$

The corresponding algorithm is given as:

Table 1: General likelihood ratio policy gradient estimator "Episodic REINFORCE" with an optimal baseline [9]

	input: policy parameterization $\boldsymbol{\theta}$.
1	**repeat**
2	perform a trial and obtain $\boldsymbol{x}_{0:T}$, $\boldsymbol{A}_{0:T}$, $\boldsymbol{r}_{0:T}$
3	**for each** gradient element G_t
4	estimate optimal baseline $b^t = \frac{\langle (\Sigma_{k=0}^{T} \nabla_{\theta_t} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{A}_k|\boldsymbol{x}_k))^2 \Sigma_{l=0}^{T} \gamma_1 r_l \rangle}{\langle (\Sigma_{k=0}^{T} \boldsymbol{\nabla}_{\theta_t} \log \pi_\theta(\boldsymbol{A}_k|\boldsymbol{x}_k))^2 \rangle}$
5	estimate the gradient element $g_t = \langle \left(\Sigma_{k=0}^{T} \boldsymbol{\nabla}_{\theta_t} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{A}_k|\boldsymbol{x}_k)\right) (\Sigma_{l=0}^{T} \gamma_l r_l - b^t) \rangle$.
6	**end for**.
7	**Until** gradient estimate g converged.
	Return: gradient estimate g $(or\ \nabla J(\theta_t))$

IV. Application to HM Coadaptation

In human-machine interaction, both the human and the machine contribute to the fulfilment of a specific task. Classically, the properties and behavior of the machine are static. For example, human commands are mapped into actions by the machine in a predetermined fashion that does not change with respect to human skill level nor to human style. To make human-machine interaction approaches human-human interaction, some intelligent features should be added. Among these are intention recognition [10] and co-adaptation.

Naturally, a human interacting with a machine adapts his/her behavior to the dynamics of the machine and the environment. So, what is left is to make the machine adapts its behavior to the performance of the human partner/user. The following are assumed:

1. The human is rational in dealing with the machine (i.e., his/her actions are chosen to achieve the goal)
2. The human co-adapts while interacting with the machine
3. The machine follows a policy to achieve the goal. This policy includes some parameters that are to be updated (while learning/co-adapting)
4. The human policy and dynamics (model) as well as the environment dynamics are not known to the machine
5. The goal to be achieved is known to both the machine and the human and thus rewards can be computed relative to the goal
6. The policy of the machine and the reward function are task specific and have to be determined a priori

V. Example Application

A simple example is detailed here to show the applicability of the proposed method. A human interacts with the computer through a joystick to move an object in the two-dimensional space of the computer screen. For some reason, the orientation of the coordinate systems of the joystick and the screen are not identical but rather rotated with a certain angle as shown in Fig. 1. However, the user does not know that and believes that the two coordinate systems are identical. So, according to the position of the object and the target, the user plans the motion in the screen coordinate system and executes it in the joystick space as shown in Fig. 2. It is assumed that the user applies full speed in reaching the target to obtain the maximum reward.

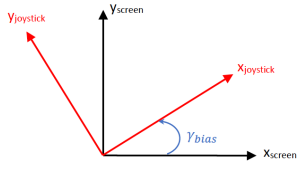

Fig. 1: Joystick - screen coordinates

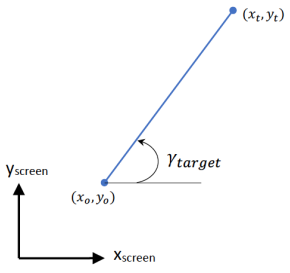

Fig. 2: Task in screen coordinates

Ideally, the machine should rotate the "joystick" coordinate system with an angle, θ, counter clockwise about the z-axis to obtain the mapped movement in the screen coordinate system:

$$\begin{bmatrix} v_{xs} \\ v_{ys} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} v_{xj} \\ v_{yj} \end{bmatrix},$$

Where the subscript "*s*" denotes screen coordinate system and the subscript "*j*" denoted the joystick coordinate system, "*v*" denotes the speed of motion towards the target. In the optimal learning case, θ is equivalent to γ_{bias}.

Let the state, *x*, be:

$$x_t = \begin{bmatrix} v_{xj} \\ v_{yj} \end{bmatrix},$$

And the action, *a*, be:

$$a_t = \begin{bmatrix} v_{xs} \\ v_{ys} \end{bmatrix},$$

then the machine policy, π_θ, is modelled as a probability distribution:

$$a_t \sim \pi_\theta(a_t|x_t).$$

Assuming a normal distribution with the average be given by a function $\mu(x,\theta) \in R^2$ and the covariance be given by the diagonal square matrix $\Sigma \in R^{2x2}$ with the variance elements on the diagonal. Then:

$$\pi_\theta = \frac{1}{\sqrt{(2\pi)^2|\Sigma|}} e^{-\frac{1}{2}(a-\mu)^T\Sigma^{-1}(a-\mu)},$$

accordingly,

$$log\pi_\theta = \log\left(\frac{1}{\sqrt{(2\pi)^2|\Sigma|}}\right) - \log\left(-\frac{1}{2}(a-\mu)^T\Sigma^{-1}(a-\mu)\right)$$

then

$$\nabla log\pi_\theta = -(\Sigma^{-1}(a-\mu))^T\nabla_\theta\mu$$

where

$$\mu = \begin{bmatrix} v_{xs} \\ v_{ys} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} v_{xj} \\ v_{yj} \end{bmatrix}$$

leading to

$$\nabla_\theta\mu = \begin{bmatrix} -\sin(\theta)\, v_{xj} + \cos(\theta) v_{yj} \\ -\cos(\theta)\, v_{xj} - \sin(\theta) v_{yj} \end{bmatrix}$$

and finally

$$\nabla log\pi_\theta = \frac{(a_1 - \cos(\theta)\, v_{xj} - \sin(\theta) v_{yj})(-\sin(\theta)\, v_{xj} + \cos(\theta) v_{yj})}{\sigma_1^2} + \frac{(a_2 + \sin(\theta)\, v_{xj} - \cos(\theta) v_{yj})(-\cos(\theta)\, v_{xj} - \sin(\theta) v_{yj})}{\sigma_2^2}$$

where σ_1^2 and σ_2^2 are the variances.

Procedure:

1. Choose σ_1^2 and σ_2^2 leading to $\Sigma = \text{diag}(\sigma_1^2, \sigma_2^2)$
2. Based on state $x = \begin{bmatrix} v_{xj} \\ v_{yj} \end{bmatrix}$ and current policy parameter θ, find
$$\mu = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} v_{xj} \\ v_{yj} \end{bmatrix}$$
3. Select action $a = \begin{bmatrix} v_{xs} \\ v_{ys} \end{bmatrix}$ according to the normal distribution $a = N(\mu, \Sigma)$
4. Calculate reward r as (see Fig 3.):
$$r = \frac{|d_{t-1}| - |d_t|}{|p_t - p_{t-1}|}$$
which leads to $r = 1$ for a perfect motion in the direction of the target and $r = -1$ for a motion completely away from the target
5. Update the parameter, θ, according to the algorithm in Table 1.

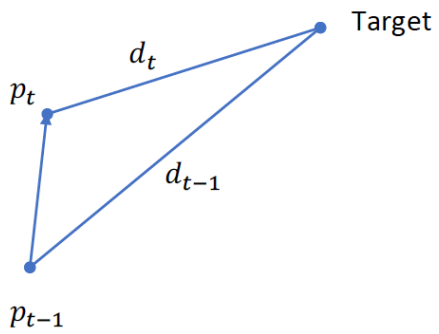

Fig. 3: Reward Calculation

Implementation:

For testing the proposed machine learning procedure, the following scenario is considered. An object of radius of 0.5 unit exists somewhere on a screen of 70*70 units. A target of radius 1 unit pops up and the user is supposed to move the object to the target as fast as possible by applying a velocity vector through his/her twisted joystick. The user assumes that the both coordinate systems are identical but in reality, they are rotated (Fig. 1). For automation purposes an ideal human agent is programmed. This agent calculates the motion angle (of its action) as (see Fig. 2):

$$\gamma_{target} = tan^{-1}\left(\frac{y_t - y_0}{x_t - x_0}\right),$$

and applies a full speed towards the target. However, and since the joystick coordinate system is twisted, the action will be:

$$\begin{bmatrix} v_{xj} \\ v_{yj} \end{bmatrix} = \begin{bmatrix} \cos(\gamma_{target} - \gamma_{bias}) \\ \sin(\gamma_{target} - \gamma_{bias}) \end{bmatrix} v_{max}$$

where $v_{max} = 1\frac{unit}{sec}$ and $\gamma_{bias} = 10$ degrees.

This forms the state that is translated to an action through the machine policy. Accordingly, the object moves from the initial position p_{t-1} to the new position p_t.

Results:

The scenario is implemented in MATLAB environment. Different experiments are performed to test the performance of the algorithm and its dependency on its parameters. Figure 4 shows the effect of the reward factor, γ, on the performance of computing the gradient, whereas the effect of the variance of the distribution, σ_1^2 and σ_2^2 is shown in Fig 5. It is clear that the performance of the algorithm is sensitive regarding the selection of these parameters.

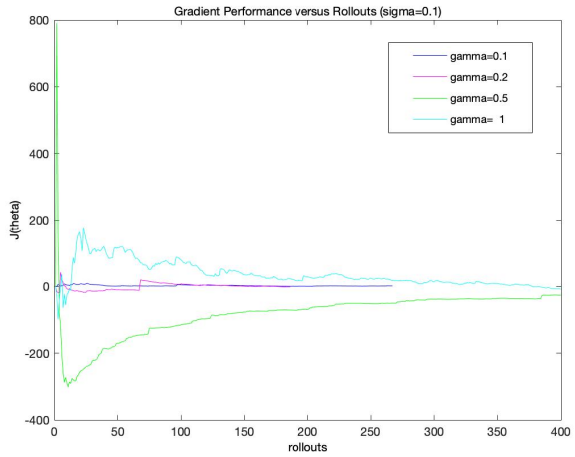

Fig. 4: Effect of reward factor on learning

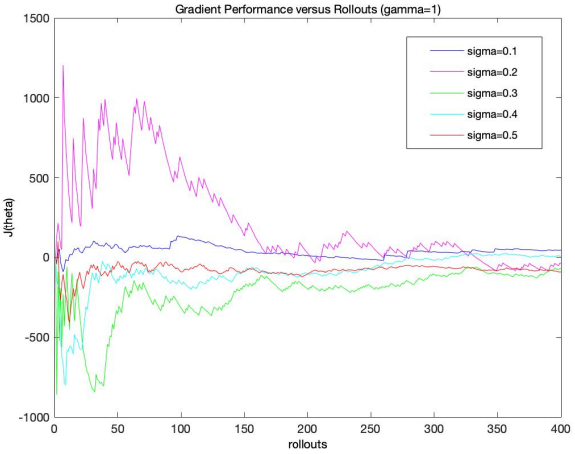

Fig. 5: Effect of variance on learning

Accordingly, the following parameters are chosen: $\gamma = 0.5, \alpha = 0.1,$ and $\sigma_1 = \sigma_2 = 0.1$. For these parameters, the policy parameter, θ, as shown in Fig. 6 converges to its correct value of -10 degrees (to counter γ_{bias}). The computation of the gradient as well converges as shown in Fig. 7.

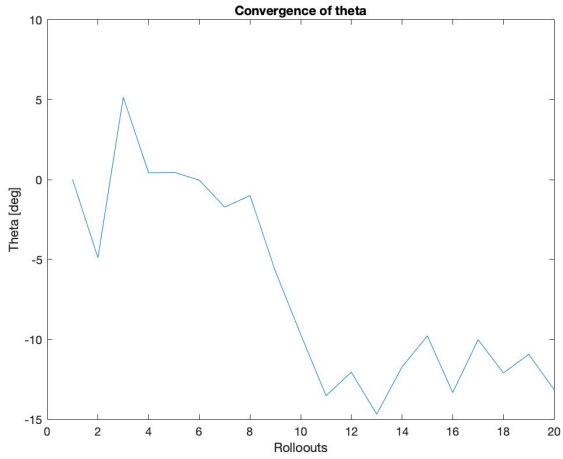

Fig. 6: Convergence of the policy parameter θ

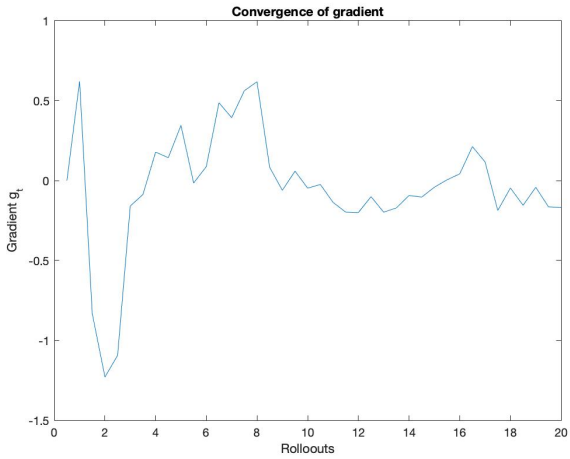

Fig. 7: Convergence of the gradient g_t

VI. Conclusion

The applicability of reinforcement learning with policy gradients to human-machine coadaptation is demonstrated. This group of algorithms do not require the knowledge of the process dynamics (model free). This attribute is crucial here as the interacting machine is freed from having a model for the interacting human. The results shown are collected from a simulation environment where the action of the interacting human is automated where rationality is assumed. For future works, learning will be added to human action to make the scenario resemble human-machine coadaptation. Further, experiments with real persons will be performed.

References

[1] R. Parasuraman, T. Bahri, J. E. Deaton, J. G. Morrison, and M. Barnes, "Theory and design of adaptive automation in adaptive systems," Progress Report No. NAWCADWAR-92033-60. Warminster, PA: Naval Air Warfare Center, Aircraft Division, 1990.

[2] M. W. Scerbo, "Theoretical perspectives on adaptive automation, " In R. Parasuraman and M. Mouloua (Eds.), Automation and human performance: Theory and application. Mahwah, NJ: Lawrence Erlbaum Associates. 1996, pp. 37-63.

[3] F. Musa-Ivaldi and Z. Danziger, "The remapping of space in motor learning and human-machine interfaces," Journal of Physiology – Paris, vol. 103, No. 3-5, pp. 263-75, 2009.

[4] Z. Danziger, A. Fishbach, and F. Mussa-Ivaldi, "Learning Algorithms for Human–Machine Interfaces," IEEE Transaction on Biomedical Engineering, Vol. 56, No. 5, pp. 1502-1511, May 2009

[5] H. Zhu, M. Zhou, "Issues in Adaptive Collaboration," IEEE International Conference on Systems, Man and Cyberntetics, pp. 2828-2833, 2010.

[6] S. Suzuki, K. Furuta, and F. Harashima, "Overview of Human Adaptive Mechatronics and Assist-control to Enhance Human's Proficiency," ICCAS, 2005.

[7] J. C. Sanchez, B. Mahmoudi, J. DiGiovanna, and J. C. Principe, "Exploiting co-adaptation for the design of symbiotic neuroprosthetic assistants," Neural Networks, Vol. 22, pp. 305-315, 2009.

[8] R. Sutton and A. Barto, "Reinforcement Learning: An Introduction," Second Edition, MIT Press, 2018.

[9] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," Neural Networks, Vol. 21, pp. 682-697, 2008.

[10] K. A. Tahboub, "Intelligent Human-Machine Interaction Based on Dynamic Bayesian Networks Probabilistic Intention Recognition," Journal of Intelligent and Robotic Systems, Vol. 45, pp 31-52, 2006