Flexible online adaptation of learning strategy using EEG-based reinforcement signals in real-world robotic applications

Su Kyoung Kim[1], Elsa Andrea Kirchner[1,2], and Frank Kirchner[1,2]

Abstract— **Flexible adaptation of learning strategy depending on online changes of the user's current intents have a high relevance in human-robot collaboration. In our previous study, we proposed an intrinsic interactive reinforcement learning approach for human-robot interaction, in which a robot learns his/her action strategy based on intrinsic human feedback that is generated in the human's brain as neural signature of the human's implicit evaluation of the robot's actions. Our approach has an inherent property that allows robots to adapt their behavior depending on online changes of the human's current intents. Such flexible adaptation is possible, since robot learning is updated in real time by human's online feedback. In this paper, the adaptivity of robot learning is tested on eight subjects who change their current control strategy by adding a new gesture to the previous used gestures. This paper evaluates the learning progress by analyzing learning phases (before and after adding a new gesture for control). The results show that the robot can adapt the previously learned policy depending on online changes of the user's intents. Especially, learning progress is interrelated with the classification performance of electroencephalograms (EEGs), which are used to measure the human's implicit evaluation of the robot's actions.**

I. Introduction

Intuitive interaction and collaboration between human and robot has a high relevance in real-world robotic applications. Especially the use of human feedback is an interesting approach in robot learning. In most cases, explicit human feedback has been used for additional communication with a robot [1]–[3] or for online improvements of a reward function, in which the predefined reward function was optimized online based on the human's explicit feedback (e.g., ratings) on the robot's behavior [4], [5]. However, the pre-configuration of evaluation criteria on robots' behaviors before online application (reward shaping) is challenging for complex and dynamic task situations. Further, even for less complex task situations, humans' explicit evaluations on robots' behavior are straightforward only for clear situations, in which human can give a best or worst rating (e.g., 1 point or 10 points) on the robots' task performance. In cases of more or less average task performance (e.g., between 4 and 6 points), a human needs deeper consideration and more decision time (e.g., more time to give 6 points compared to 1 point or 10 points) on the robot's task performance. Hence, the use of explicit human feedback is limited in complex task situations and is not always the best option for specific applications. In contrast, the use of implicit human feedback is a promising approach in complex robotic tasks, since intrinsic human evaluation can be used without predefining a reward function beforehand. In recent studies, implicit human feedback, e.g., signals from the electroencephalogram (EEG) have been used for brain-computer interfaces and human-robot interaction [6], [7].

In our previous studies [7], we tested the feasibility to use EEG-based reinforcement signals in real robot applications. Our approach called ”intrinsic interactive reinforcement learning” enables a robot to learn human gestures by interacting with a human (Fig. 1-c). A subject performs a certain gesture and the robot perceives gesture features, which are provided by the recording device called Leap Motion [8]. In the first run (trial), the robot randomly chooses an action based on the observed current context (human gesture) and executes the selected action. The human observes the robot's action and gives feedback in form of EEG, which is intrinsically generated in the human's brain as neural signature of the human's implicit evaluation on the robot's action. The robot receives human feedback as reward and computes an action strategy (policy) for the second run based on the reward that is given to the chosen action in the first run. In this way, the robot updates the current policy for each run and tries to build an optimal policy by collecting experiences. In the end, the robot learns correct mappings between human gestures and robot's actions (i.e. correct gesture-action pairs).

In our human-robot interaction scenario, an implicit evaluation of each observed action of the robot is intrinsically generated in the human brain which is measured on the surface of the subject's head using EEG technique. The relevant EEG component in our application is the error-related potential (ErrP), which is evoked when the subject observes unusual or wrong actions of a robot [9]–[13]. However, ErrP detection is challenging for continuous actions, since the onset of human perception on wrong actions of the robot is unknown and can also vary between different actions within the same subject or between subjects within the same robot's action. Hence, the use of multiple sliding windows on a continuous action of the robot has shown to be a good option for ErrP detection as investigated in our previous study [7]. Here, we could show that the use of two time windows (Tab. I) improved the performance of ErrP detection compared to the use of a single time window. Although the use of multiple sliding windows is a good option, we assumed that the ErrP detection performance is crucial in the beginning of the learning phase. Hence, in our previous study, we pre-trained the learning algorithm by randomly using one, two, or

[1] Robotics Innovation Center (RIC), German Research Center for Artificial Intelligence (DFKI), Bremen, Germany.

[2] Robotics group, University of Bremen, Germany.

This work has been supported by the BMWi, DLR, and BMBF (grant number: FKZ 50RA1701, FKZ 50RA1703, and FKZ 01IW18003).

three gesture-actions pairs before online applications to avoid consecutive occurrence of errors in the initial learning phase [7]. In this paper, we tested our approach without pre-training the learning algorithm. For evaluation, we recorded both learning conditions (pre-training/*no* pre-training) from the same subjects and compared the pattern of learning progress between both learning conditions. Further, this work analyzes the correlation between ErrP-classification performance and learning performance, which can be differently expressed depending on the learning condition.

In the proposed human-robot interaction scenario, the subjects determine gesture meaning (human intent) and the robot learns to choose actions that are best assigned to the *current* human intents based on humans intrinsic feedback. Hence, when the subjects change previous gesture mapping or add a new gesture, the robot can relearn new assignments of gestures or integrate a newly added gesture meaning in the previous accumulated knowledge (policy). In this way, the robot can *adapt* its learning strategy according to online changes of the *current* human intent. In this study, this adaptivity of robot learning was tested on eight subjects who added a new gesture meaning (a new human intent) while online learning/application. For evaluation, we analyzed learning progress by comparing learning phases (before and after adding a new gesture) to determine how learning performance is affected by adding a new gesture. Learning progress was also analyzed depending on learning condition (pre-training/*no* pre-training) to find how differently warm-start learning (pre-training) and cold-start learning (no pre-training) affect learning progress.

II. Methods

A. Experiment setup

1) Observation scenario: Figure 1-a shows the observation scenario, which is used to record training data to build an ErrP decoder. In this scenario, the subjects do not interact with robot, since human gestures and robot's action selections are already preprogrammed. A hand gesture is displayed to the subjects as a word (*left*, *right*, *forward*, or *upward*) on the monitor, which is located on the left side of the robot. A feature vector of the displayed gesture is sent to the pseudo-learning algorithm, where action selections are preprogrammed (1). The selected action is sent to the robot (2) that executes the selected action (3). The subjects observe the robot's action (4) and evaluate the correctness of the robot's action in form of EEG (5). Two different kinds of markers were written in EEG data for offline post-hoc analysis: gesture markers and action markers. The comparison between gesture markers (e.g., gesture: *left*) and robot's action markers (e.g., action: *left*) enables to evaluate the correctness of robot's actions (e.g., correct action), which is served as *actual label* (Tab. I-a) for evaluation on *predicted label* of an ErrP decoder (e.g., ErrP is detected $\rightarrow$ wrong action, Tab. I-c, i.e., false positive, Tab. I-e). Note that positive class refers to *wrong* action. The reason for preprogramming is to reduce the recording time of EEG data.

2) ErrP decoder: A classifier is trained on data that is recorded in the observation scenario (Fig. 1-a). The trained classifier (Fig. 1-b) is used to detect ErrPs for each single action of the robot in the interaction scenario (Fig. 1-c).

3) Human-robot interaction scenario: As shown in Figure 1c, a subject performs one of four gesture types (*left*, *right*, *forward*, and *upward*), which is further sent to the learning algorithm as human intent (1). The learning algorithm selects an action based on the current context (human gesture), which is further sent to the robot (2). The robot executes the selected action (3). The human observes the robot's action (4) and gives his/her evaluation on the correctness of robot's action in form of EEG (i.e., ErrP is detected or not) to the robot (5). The robot updates an action strategy (policy) based on human feedback, which is used as reward in reinforcement learning (6). ErrPs are detected online for each single action of the robot. For a single action of the robot, two decisions are made from each sliding window (Tab. I-b1, b2), but one decision is used as reward for learning algorithm (Tab. I-c, e), which receives 1 for correct actions (NoErrP) and -0.25 for wrong actions (ErrP).

In contrast to the previous application [7], the subjects are instructed to add a new gesture after performing three gestures (*left*, *right*, *forward*) for a while. An acoustic signal is given to the subjects after the first 30 trials to avoid counting the number of trials while interacting with the robot. However, the subjects can freely determine the time when to add a new gesture after the acoustic signal. Hence, the onset of new gestures is varied between subjects. Further, two learning conditions are tested in the presented study: (a) In the warm-start learning condition, one or two gesture-action pairs is/are pretrained before online learning; (b) In the cold-start learning condition, gesture-action pairs are not pretrained before online learning. All subjects are instructed to start with the warm-start learning condition.

B. Dataset

Eight subjects (2 females, 6 males, age: 27.5 ± 6.61 years, right-handed, normal or corrected-to normal vision) participated in this study. The experiments were carried out in accordance with the approved guidelines. Experimental protocols were approved by the ethics committee of the University of Bremen. Written informed consent was obtained from all participants that volunteered to perform the experiments.

For each subject, we recorded six datasets to train an ErrP decoder and three datasets for online learning. Six datasets that were recorded from the observation scenario (Fig. 1a) were used for training a classifier to detect two distinct classes (ErrP/no ErrP), which are correlated with each correct and wrong action of the robot. Each dataset contained 80 correct and 10 wrong actions. The trained classifier (Fig 1b) was used to detect ErrPs for each single action of the robot during the online learning session (Fig 1c). In the online learning session, each dataset contained 90 trials for all subjects except for two subjects who performed 120 trials and 60 trials respectively. The subject-specific differences

TABLE I

USE OF ERRP DETECTION AS REWARDS IN LEARNING ALGORITHM.

(a) Actual label (comparison between gestures and actions)	correct	correct			wrong	wrong		
(b1) Prediction from the *first* time window (ErrP decoder)	NoErrP	NoErrP	ErrP	ErrP	NoErrP	NoErrP	ErrP	ErrP
(b2) Prediction from the *second* time window (ErrP decoder)	NoErrP	ErrP	NoErrP	ErrP	NoErrP	ErrP	NoErrP	ErrP
(c) Predicted label (one decision from two predictions (b))	NoErrP (correct)	ErrP (wrong)			NoErrP (correct)	ErrP (wrong)		
(d) Rewards used for learning algorithm (online learning)	1	-0.25			1	-0.25		
(e) Evaluation on ErrP detection (offline post-hoc analysis)	TN	FP			FN	TP		

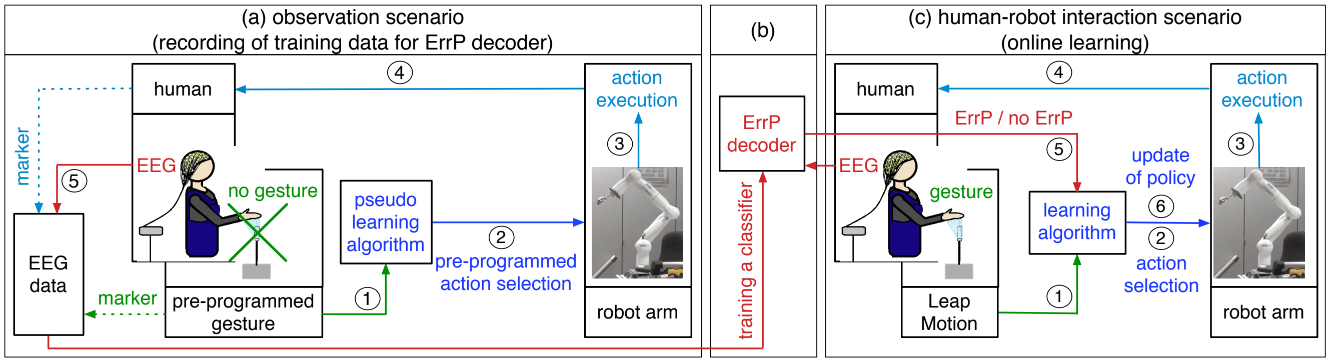

Fig. 1. Experiment setup: (a) observation scenario, (b) ErrP decoder, and (c) human-robot interaction scenario. Details, see text.

between the total numbers of trials were taken into account for evaluation and the results are reported in percentage. In online learning, the ratio between correct and wrong actions varied between subjects, since learning performance depends on the quality of reward, i.e., ErrP-detection performance between subjects.

C. *EEG recording, preprocessing and classification*

EEGs were continuously recorded using the actiCap system (Brain Products GmbH, Munich, Germany), in which 64 active electrodes were arranged in accordance to an extended 10-20 system with reference at electrode FCz. Impedance was kept below 5 kΩ. EEG signals were sampled at 5 kHz, amplified by two 32 channel Brain Amp DC amplifiers (Brain Products GmbH, Munich, Germany), and filtered with a low cut-off of 0.1 Hz and high cut-off of 1 kHz. The EEG data was analyzed using a Python-based framework for preprocessing and classification [14]. The continuous EEG signal was segmented into epochs from -0.1 s to 1 s after the start of the robot's action for each action type (correct/wrong trial). Two sliding windows were used for the same robot's action. All epochs were normalized to zero mean for each channel, decimated to 50 Hz, and band pass filtered (0.5 to 10 Hz). The xDAWN spatial filter [15] was used to enhance the signal-to-noise ratio and 8 pseudo channels were obtained after spatial filtering. Features were extracted from two windows (8 pseudo channels): [-0.1 s to 0.6 s, 0 s to 0.7 s]] and normalized over all trials. A total of 280 features (8 pseudo channels $\times$ 35 data points = 280 for each window) were used to train a classifier. A linear support vector machine (SVM) [16] was used to classify correct and erroneous trials. The cost parameter of the SVM (i.e., regularization constant [17]) was optimized using a gird-search with predetermined values [10^0, 10^{-1}, ... , 10^{-6}], which was evaluated by a stratified five-fold cross validation.

D. *Gesture recording and robot arm (COMPI)*

The Leap Motion [8] system was used to record hand gestures. A stereo image is generated by using two monochromatic infrared cameras. The positions of hand and finger bones can be detected in x, y, and z coordinates relative to the sensor. We used the x, y, z components of the palm normal vector and a value from 0 to 1, which describes how far the hand is opened or closed. Ten samples (length of 100 ms) were recorded per gesture and averaged. Gesture feature vectors were used as inputs (human intent) for the learning algorithm. The learning algorithm selects actions, which are sent to a six degree of freedom (6-DOF) robotic arm called COMPI [18], which was developed at our institute (RIC, DFKI). Six predefined actions (left, right, forward, upward, and back to start) were implemented in joint space, which were triggered from the learning algorithm.

E. *Learning algorithm*

We used a contextual bandit approach as a variant of reinforcement learning, in which only one action is selected per episode. The contextual bandit approach is well suited for our human-robot interaction scenario, in which a robot learns to choose the action which is best assigned to a gesture.

In LinUCB [19], which is given in algorithm 1, one context x_t is given for each discrete time point t. The agent observes the current context, chooses the action that is best assigned to the given context, and finally observes the reward that is given to the chosen action. In the first run, the robot randomly chooses an action after observing the current context and the payoff is calculated for the second run based on the reward that is given to the chosen action in the first run. From the second run on, the agent chooses the context-action pair ($x_{t,a}$) with the highest payoff ($P_{t,a}$) and observes the reward r_t that is given to the chosen context-action pair (line 11 in algorithm 1). Accordingly, action space ($A_{a,t}$) and context space ($b_{a,t}$) are updated (line 12, 13 in algorithm 1).

```
Algorithm 1. LinUCB [19]
0: Inputs: $\alpha \in \mathbb{R}_+$
1: for $t = 1,2,3,...,T$ do
2:    Observe features of all arms $a \in \mathscr{A}_t$: $x_t \in \mathbb{R}^d$
3:    for all $a \in \mathscr{A}_t$ do
4:       if $a$ is new then
5:          $A_a \leftarrow I_d$ (d-dimensional identity matrix)
6:          $b_a \leftarrow O_{d\times 1}$ (d-dimensional zero vector)
7:       end if
8:       $\hat{\theta} \leftarrow A_a^{-1}\ b_a$
9:       $P_{t,a} \leftarrow \hat{\theta}^T x_{t,a} + \alpha \sqrt{x_t^T A_a^{-1} x_{t,a}}$
10:   end for
11:   Choose arm $a_t = arg\ max_{a \in \mathscr{A}_t, P_{t,a}}$ with ties broken
   arbitrarily and observe a real valued payoff $r_t$
12:   $A_{a,t} \leftarrow A_{a,t} + x_t\ x_{t,a_t}^T$
13:   $b_{a,t} \leftarrow b_{a,t} + r_t\ x_{t,a_t}$
14: end for
```

For computation of payoffs, the agent counts how often each context-action pair ($x_{t,a}$) was chosen in the past and calculates its mean value and the corresponding upper confidence interval (UCI). Finally, the so called upper confidence bound (UCB), i.e., payoff is computed as the sum of the mean value and its UCI. Payoffs are computed for each context-action pair (line 9 in algorithm 1) and the reward is given only to the chosen context-action pair (line 11 in algorithm 1). The UCI values of all context-action pairs are high in the beginning of learning phase and becomes lower when they are chosen more often. Hence, the mean value is more crucial for achieving a high payoff than UCI. For this reason the context-action pair that was chosen more often in the past will further on be chosen in the future (exploitation). However, the context-action pair that was seldom chosen in the past is also chosen due to a high value of UCI (exploration). Such a context-action pair has a low mean value, but a high UCI value, which leads to a high payoff. The degree of exploration is determined by the parameter α (line 0 in algorithm 1). When choosing a high value of α, the UCI value increases, since α is multiplied with the UCI value (line 9 in algorithm 1), and the agent explores more frequently.

In our application, a human performs a certain gesture (context) and the robot tries to learn the action that is best assigned to the given gesture. Hence, the action selection of the robot depends on the value of payoff of gesture-action pairs. When the *right-right* pair was often selected in the past, the *right-right* pair has a high payoff. That is true when we use a predefined reward function. However, we used the outputs of ErrP decoder. Hence, the reward should be correctly given to the robot (i.e., ErrP classification should be correct) to learn correct gesture-action pairs. Otherwise, the robot learns wrong gesture-action pairs due to high payoffs. In this study, the exploration parameter α was empirically determined based on the previous data [7]. However, the determined α value is not perfect for individual applications of the presented study, since the pattern of gesture choices is biased between subjects. For example, some subjects mostly performed two types of gestures (e.g., *left*, *right*), since they were instructed to freely choose any gesture. In this case, the mean values of two gesture-action pairs (*left-left* pair and *right-right* pair) can be increased and their UCI values can be reduced with increased trials. Hence, the agent can learn the less often chosen gesture-action pairs (*forward-forward* pair, *upward-upward* pair) by exploration.

F. Data analysis

1) Robot's learning performance: As performance metric, we used the number of wrong actions of the robot (i.e., wrong mappings between human gestures and robot's actions): *mapping errors*. First, we calculated the number of mapping errors in the whole learning process and compared them between both learning conditions to find out the effect of the learning condition on the whole learning performance (Fig. 2-a1). Second, we analyzed how the learning process is affected by adding a new gesture. To this end, we grouped the whole learning process in three learning phases and compared them for each learning condition (Fig. 2-a2).

- First learning phase (before adding a new gesture) [start-1/3]
- Second learning phase (after adding a new gesture) [1/3-2/3]
- Final Learning phase [2/3-end]

We also compared the number of mapping errors between both learning conditions for each learning phase to analyze the effect of the learning condition on the learning performance for each learning phase (Fig. 2-a2). Finally, the learning progress was analyzed for individual datasets to visualize different patterns in the learning progress depending on the learning condition (Fig. 3).

2) ErrP-detection performance: The outputs of ErrP decoder were analyzed both in the whole learning process (e.g., Fig. 2-b1) and in each learning phase (e.g., Fig. 2-b2). As performance metric, we used the number of false positive (FP), false negative (FN), true negative (TP), and true positive (TP), and ErrP misclassifications (FP $\cup$ FN).

- ErrP is detected although the robot's action is correct (FP)
- ErrP is not detected although the robot's action is wrong (FN)
- ErrP is detected when the robot's action is wrong (TP)
- ErrP is not detected when the robot's action is correct (TN)

3) Statistical analysis: A one-way ANOVA was performed with *learning condition* (pre-training/*no* pre-training) as an independent variable to compare the learning performance between both learning conditions in the whole learning process (Fig. 2-a1). For the analysis of learning progress, a two-way repeated measures ANOVA was performed with *learning phase* (three levels of learning phase) as a within-subjects factor and *learning condition* (pre-training/*no* pre-trianing) as a between-subjects factor (Fig. 2-a2). Dependent variable of both analyses was the robot's performance (mapping errors). Two post-hoc analyses were performed for two independent variables (learning condition/learning phase): (1) comparison of learning performance between three learning phases for each learning condition and (2) comparison of learning performance between both learning conditions for each learning phase. For multiple comparisons, Bonferroni correction was performed. Note that the sample size was

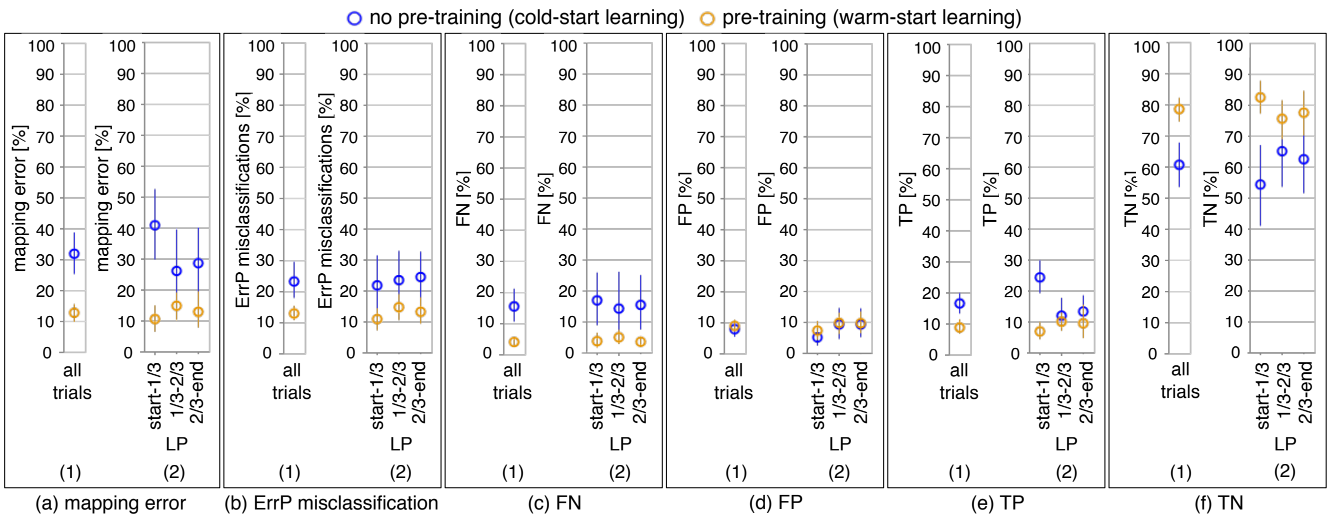

Fig. 2. Robot's learning performance (mapping errors) and ErrP-detection performance (FN, FP, TN, TP, ErrP misclassifications) in the whole learning process (all trials) and in each learning phase (start-1/3, 1/3-2/3, 2/3-end) for both learning conditions (no pre-training, pre-training). LP: learning phase.

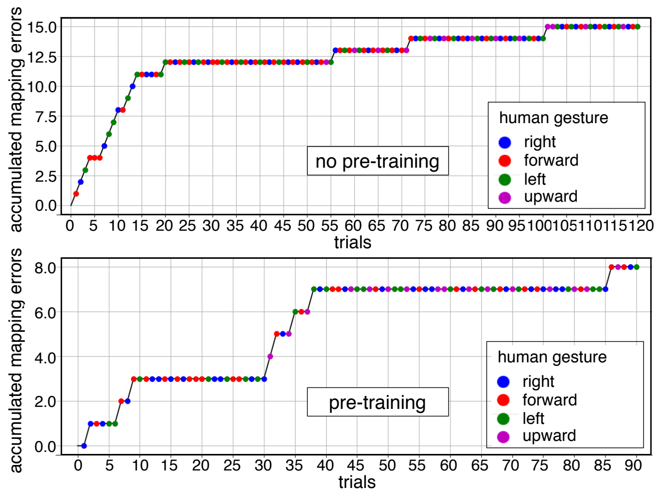

Fig. 3. Learning process in individual datasets in both learning conditions. Mapping errors that are accumulated with increased robot's actions (trials) are depicted in the y-axis and the number of trials is depicted in the x-axis.

unequal for *learning condition*, since one subject ended the experiment before recording data in the cold-start learning condition. For this reason, the independent variable *learning condition* was considered as between-subjects factor in the two-way repeated measures ANOVA. The same statistical designs were used for evaluation of ErrP misclassifications (FN $\cup$ FP), FN, FP, TN, and TP.

III. Results and Discussion

A. *Robot's learning performance (mapping errors)*

Figure 2-a1 shows the comparison between both learning conditions in the whole learning process. As expected, the learning performance was significantly improved in the warm-start learning condition [$F_{1,35} = 11.28$, warm-start learning vs. cold-start learning: $p < 0.003$]. Figure 2-a2 shows (1) the comparison between three learning phases for each learning condition and (2) the comparison between two learning conditions for each learning phase.

1) Comparison between three learning phases: In the warm-start learning condition (pretraining), the number of mapping errors was already small in the end of the first learning phase. However, the number of mapping errors was slightly (but not significantly) increased immediately after adding a new gesture [start-1/3 vs. 1/3-2/3: $p = n.s.$] and was again slightly (but not significantly) decreased in the final learning phase [2/3-end vs. start-1/3: $p = n.s.$]. That means, the robot achieved a high performance already in the end of the first learning phase and the learning performance was not significantly varied between three learning phases, although the learning performance was slightly decreased immediately after adding a new gesture and was later slightly increased with more trials after adding a new gesture.

In contrast, we observed a different pattern of the learning progress in the cold-start learning condition. The number of mapping errors was very high in the first learning phase. However, the number of mapping errors was significantly decreased even after adding a new gesture [start-1/3 vs. 1/3-2/3: $p < 0.001$, start-1/3 vs. 2/3-end: $p < 0.01$] and was later slightly (but not statistically significant) increased in the final learning phase [1/3-2/3 vs. 2/3-end: $p = n.s.$]. That means, the learning progress was very slow during the first learning phase. However, the learning performance was significantly improved with increased trials even after adding a new gesture, since the learning algorithm collected more trials for the previous learned three gesture-action pairs in the second learning phase compared to the first learning phase. This indicates that the amount of experience (trials) has a higher impact on learning progress in the cold-start compared to the warm-start learning condition. The slight (but not statistically significant) reduction of learning performance in the final learning phase indicates that more experiences are needed to achieve a high learning performance when a new gesture is added. Especially, the substantial improvement of learning performance between the first and final learning phase was observed only in the cold-start learning condition.

2) Comparison between both learning conditions: In the first learning phase, we found a significant reduction of mapping errors for the warm-start compared to cold-start learning condition [start-1/3: no pre-training vs. pre-training: $p < 0.001$]. In the second learning phase, the number of mapping errors in the cold-start learning condition was radically reduced so that there was no significant difference between both learning conditions [1/3-2/3: *no* pre-training vs. pre-training: $p = n.s.$]. In the final learning phase, we again observed a significant difference between both learning conditions: the number of mapping errors was slightly reduced for the warm-start learning condition, whereas the number of mapping errors was slightly increased for the cold-start learning condition [2/3-end: *no* pre-training vs. pre-training: $p < 0.021$]. In summary, the difference between learning conditions was reduced, when the learning algorithm of the cold-start learning condition collected more experiences for three gesture-action pairs (no difference between both learning conditions in the second learning phase). However, this difference occurred in the final learning phase, since the collected experiences on a new gesture-action pair were not yet sufficient for the cold-start learning condition.

3) Individual datasets as examples: Figure 3 shows the learning process in individual datasets as examples. In the cold-start learning condition (no pre-training), the number of mapping errors was very high in the beginning of the first learning phase, but reduced after the first learning phase. The learning performance was stable although a new gesture was added, which indicates that the collection of more experience reinforces the whole learning process. Further, the number of mapping errors was not rapidly increased after adding a new gesture. In the warm-start learning condition (pre-training), the number of mapping errors varied in a small range. The number of mapping errors was already small in the beginning of the first learning phase and increased immediately after adding a new gesture and again reduced with increased trials.

B. ErrP-detection performance

Figure 4 shows the correlation between mapping errors and ErrP misclassifications. In both learning conditions, ErrP misclassifications highly correlated with mapping errors. Especially, such correlation was higher for the cold-start compared to the warm-start learning condition [*no* pre-training: $r = 0.89$, pre-training: $r = 0.68$]. The results indicate that learning performance was influenced by ErrP-detection performance, in which the cold-start learning condition is more dependent on ErrP-classification performance compared to the warm-start learning condition.

Figure 2-b1 shows ErrP-detection performance in the whole learning process. The number of ErrP misclassifications was reduced in the warm-start compared to the cold-start learning condition [$F_{1,35} = 5.13, p < 0.031$]. This supports our assumption that pre-training can reduce ErrP misclassification. The same pattern was shown for FN, but not for FP [$F_{1,35} = 6.73, p < 0.015$, Fig. 2-c1; $F_{1,35} = 0.22, p = n.s.$, Fig. 2-d1], which indicates that FN has a more crucial impact on learning performance than FP.

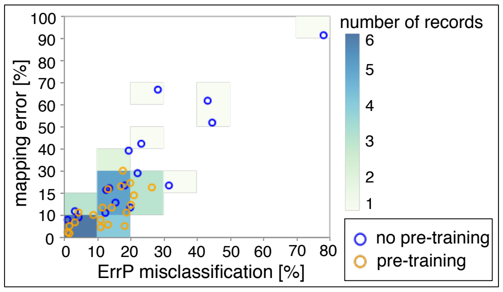

Fig. 4. Correlation between robot's learning performance (mapping errors) and ErrP classification for each learning condition

Figure 2-b2 shows ErrP-classification performance for each learning phase. In the warm-start learning condition, the pattern of ErrP-detection performance between three learning phases was coherent with the pattern of the learning progress (pre-training: Fig. 2-a2 vs. Fig. 2-b2). A similar pattern was also shown for FN, FP, and TP. The pattern of TN was reversed compared to the learning progress. This is expected, since TN deals with correct detections of NoErrP (TN), but not correct or wrong detections of ErrP (TP or FP) including missing of ErrP detection (FN). In the cold-start learning condition, the pattern of FN and TP is coherent with learning progress (no pre-training: Fig. 2-a2 vs. Fig. 2-c2 and Fig. 2-e2). Again, the pattern of TN was reversed compared to learning performance, but the reversed pattern of TN is consistent with the pattern of learning progress. The pattern of FP did not differ from the warm-start learning condition.

In summary, the ErrP-classification performance had an effect on learning performance under both learning conditions and this effect was stronger for the cold-start compared to the warm-start learning condition. Further, the pattern of ErrP-detection performance was coherent with learning progress.

IV. Conclusion

Our results show that the robot can learn his/her action strategy based on human's intrinsic feedback and also adapt the previous learned knowledge (policy) according to online changes of the subject's intents. In this paper, we focus on the effect of ErrP-classification performance on learning performance, since ErrP-classification outputs are used as rewards for the learning algorithm. However, the learning algorithm also receives gesture features as inputs in our human-robot interaction scenario. Sometimes, gestures are not correctly recorded (e.g., subjects's hands are out of range of sensors) and thus features of the recorded gestures (gestures that robot perceives) are not coherent with gestures that the subjects intent to perform. Such gesture incoherence between human and robot can affect ErrP classifications. In the future, effects of interactions between different erroneous inputs that were used for learning algorithm, i.e., interactions between misinterpretation of human intent (gesture) and human feedback (ErrP) should be investigated. Further, our results suggest that approaches that allow for robust learning without pre-training should be developed for online applications.

References

[1] A. L. Thomaz, G. Hoffman, and C. Breazeal, “Real-time interactive reinforcement learning for robots,” in *Proceedings of AAAI Workshop on Human Comprehensible Machine Learning*, 2005.

[2] C. Stahlhut, N. Navarro-Guerrero, C. Weber, and S. Wermter, “Interaction in reinforcement learning reduces the need for finely tuned hyperparameters in complex tasks,” *Kognitive Systeme*, vol. 2, 2015.

[3] S. A. Raza, B. Johnston, and M.-A. Williams, “Reward from demonstration in interactive reinforcement learning,” in *The Twenty-Ninth International Flairs Conference*. AAAI, 2016.

[4] C. Daniel, M. Viering, J. Metz, O. Kroemer, and J. Peters, “Active reward learning.” in *Proceedings of Robotics: Science and Systems*, 2014.

[5] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan, “Cooperative inverse reinforcement learning,” in *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 3909–3917.

[6] I. Iturrate, R. Chavarriaga, L. Montesano, J. Minguez, and J. d. R. Millán, “Teaching brain-machine interfaces as an alternative paradigm to neuroprosthetics control,” *Scientific reports*, vol. 5, p. 13893, 2015.

[7] S. K. Kim, E. A. Kirchner, A. Stefes, and F. Kirchner, “Intrinsic interactive reinforcement learning – using error-related potentials for real world human- robot interaction,” *Scientific Reports*, vol. 7, p. 17562, 2017.

[8] “Leap motion developer portal [online],” Available: https://developer.leapmotion.com/.

[9] P. W. Ferrez and J. d. R. Millán, “You are wrong! - automatic detection of interaction errors from brain waves,” in *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2005, pp. 1413–1418.

[10] I. Iturrate, L. Montesano, and J. Minguez, “Single trial recognition of error-related potentials during observation of robot operation,” in *Proceedings of the 32th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2010, pp. 4181–4184.

[11] S. K. Kim and E. A. Kirchner, “Classifier transferability in the detection of error related potentials from observation to interaction,” in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, (SMC)*, 2013, pp. 3360–3365.

[12] R. Chavarriaga, A. Sobolewski, and J. d. R. Millán, “Errare machinale est: the use of error-related potentials in brain-machine interfaces,” *Front. Neurosci.*, vol. 8, 2014.

[13] S. K. Kim and E. A. Kirchner, “Handling few training data: classifier transfer between different types of error-related potentials,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 24, no. 3, pp. 320–332, 2016.

[14] M. M. Krell, S. Straube, A. Seeland, H. Wöhrle, J. Teiwes, J. H. Metzen, E. A. Kirchner, and F. Kirchner, “pySPACE - a signal processing and classification environment in Python,” *Frontiers in Neuroinformatics*, vol. 7, no. 40, 2013.

[15] B. Rivet, A. Souloumiac, V. Attina, and G. Gibert, “xDAWN algorithm to enhance evoked potentials: Application to brain-computer interface,” *IEEE Transaction on Biomedical Engineering*, vol. 56, no. 8, pp. 2035–2043, 2009.

[16] C.-C. Chang and C.-J. Lin, “LIBSVM: A library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, pp. 27:1–27, 2011.

[17] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett, “New support vector algorithms,” *Neural computation*, vol. 12, no. 5, pp. 1207–1245, 2000.

[18] V. Bargsten and J. d. G. Ferandez, “Compi: Development of a 6-dof compliant robot arm for human-robot cooperation,” in *Proceedings of the 8th International Workshop on Human-Friendly Robotics (HFR)*, 2015.

[19] L. Li, W. Chu, J. Langford, and R. E. Schapire, “A contextual-bandit approach to personalized news article recommendation,” in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 661–670.