Training a Gaming Agent on Brainwaves

Bartolomé Francisco, Moreno Juan, Navas Natalia, Vitali José,
Rodrigo Ramele, *Member, IEEE,* Juan Miguel Santos

***Abstract*—The field of Brain Computer Interfaces (BCI) has seen a bloom in the past few years. One of its applications is the training of systems using signals originated from the human brain. Out of all the methods used to acquire information from the central nervous system, the one that has gained most popularity is Electroencephalography (EEG), mainly because of its effectiveness, low cost and its noninvasive characteristic. Event-related potential (ERP) are brain signals that are time-locked to a particular event. Error-related potential (ErrP) are a particular type of ERP that can be elicited by a person who attends a recognizable error. When subjects observe an agent who makes a mistake this potential can be discerned by analyzing the signals which are directly related with the cognitive interpretation of an erroneous outcome or decision. Reinforcement Learning (RL) is an approach to machine learning where an agent tries to maximize the reward obtained while interacting with an unknown environment. In this work, a gaming agent is trained using Reinforcement Learning, and using the feedback obtained from human critic observers. Results show that there is an effective transfer of information and that the agent learns successfully to solve the game efficiently.**

***Index Terms*—ErrP, BCI, EEG, RL, Agent, AI**

I. Introduction

THE effectiveness of today's human–machine interaction and artificial intelligence is limited by a communication bottleneck as humans are required to translate high-level concepts into a machine-mandated sequence of instructions [1]. This work tackles this problem by exploring the use of brain signals as an interface between human and computer, trying to overcome this limitation by providing information to the system without explicit communication from the user. This information is used to make a gaming agent improve its operational performance using electroencephalography (EEG) signals as feedback of the performed task, obtained from an observational human critic. The idea is that subjects observing a computer game play can train the gaming agent using only signals components called Error-related Potentials (ErrP) that can be identified from their brain signals. These can be found on EEG traces and are elicited when a subject is aware of the presence of un unexpected outcome, which she identifies as an error. It is currently an extensive area of research in the neuroscience community [2]. They can be detected by signal processing and machine learning techniques [3] and they are also used in Brain-Computer Interfaces (BCI) to implement or enhance artificial communication channels [4].

In order to train the agent, Reinforcement Learning (RL) [5] comes as natural solution for this scenario from the field of Artificial Intelligence. RL is an algorithmic learning strategy that can be established by mimicking how biological agents learn from the environment by exploring it and getting feedback rewards, either negative or positive. This strategy aims to maximize the amount of positive rewards while keeping low the number of negative feedback. Thus, the learning problem is posed as a mere stochastic optimization strategy [6]. Recently, this technique has seen a come-back. Nonneglected is the influence of DeepBrain's AlphaGo project, which was the first to reach superhuman proficiency when it won the complex game Go against several world champions [7].

Previous research has explored the usage of RL with reward signals based on brain activity, recorded by an EEG-based BCI system during task execution. The papers [8], [9] have successfully demonstrated that a robot can be controlled by obtaining a reward signal from brain activity of a person which is observing the robot solving a task. Other approaches have used the signal as an important feedback to identify targets for robots when implementing shared-control strategies [10]. Additionally, ErrPs have been used as well in the context of games as an additional feedback channel that can be explored to enhance gaming experience [11], [12]. In this same line, the objective of this work is to use the information extracted from brainwaves to enhance the performance of a gaming agent, replicating and extending those results.

In Section II the general layout of the cognitive game experiment is described. Section III outlines the processing pipeline used to detect ErrP components. The following Section IV describes the agent learning phase. Results and Conclusions are exposed in Sections V and VI.

II. Materials and Methods

This experiment consists of collecting signals from a person's brain while they are watching a game where the agent knows the game rules but not how to win it. Hence, the agent can learn the winning strategy using the person's feedback to improve its own performance. This section describes the process of obtaining the person's brain signals that are used to make the agent more efficient.

The experimental procedure is summarized in Figure 1. The central part is the retrieval of the subject's brain activity. This process is called brainwave session. Subjects are recruited voluntarily. They are given a consent form with questions regarding their health (previous health issues and particular visual sensitivity), habits (sleeping hours, caffeine and alcohol consumption), as well as an approval petition to collect the required data. The brainwave sessions are performed with 8 subjects, 5 males and 3 females, average age 25.125 years,

R. Ramele and J.M.Santos are with the Department of Computer Engineering, Instituto Tecnológico de Buenos Aires(ITBA), Argentina, e-mail: rramele@itba.edu.ar.

Manuscript received April 19, 2005; revised August 26, 2015.

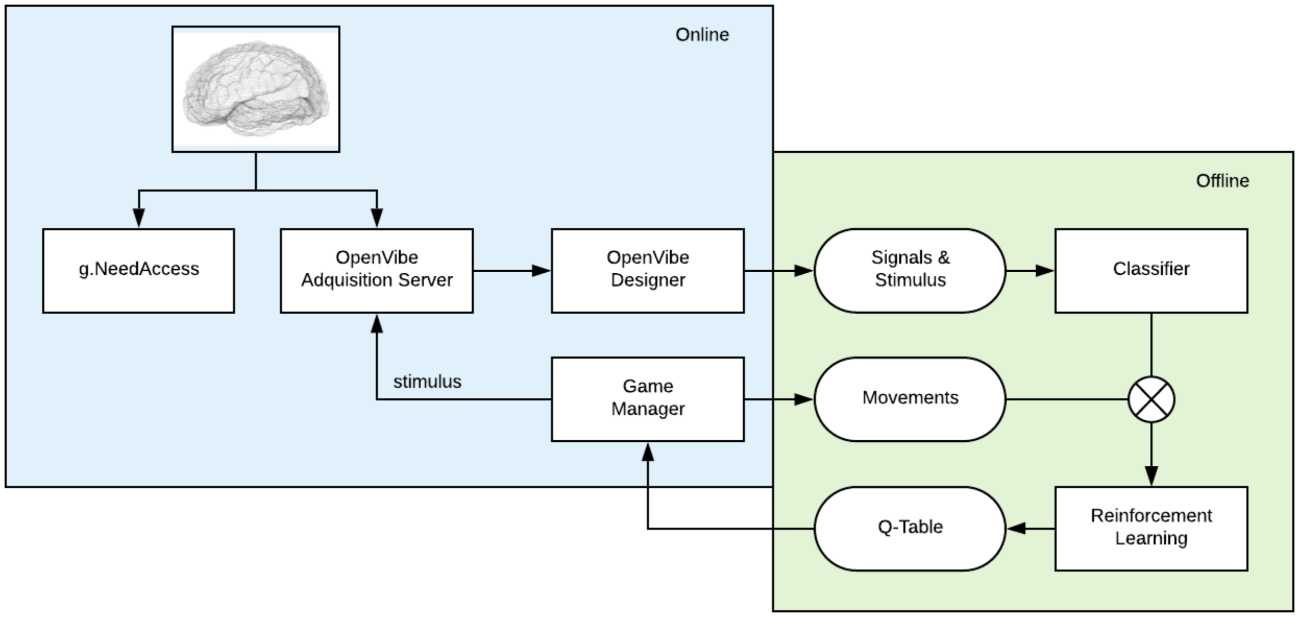

Fig. 1: Overview of the experimental procedure.

standard deviation 1.54 years, range of 22-28 years. All subjects have normal vision, are right-handed and have no history of neurological disorders.

After the form is filled out, a short description of the procedure is given to each subject. They are only told that the objective of the agent is to reach the goal and the four movements that the agent can make. When this concludes, the subject is introduced to the wireless digital EEG device (g.Nautilus, g.Tec, Austria) that they have to wear during the brainwave session. It has eight electrodes (g.LADYbird, g.Tec, Austria) on the positions Fz, Cz, Pz, Oz, P3, P4, PO7 and PO8, identified according to the 10-20 International System, with a reference set to the right ear lobe and ground set as the AFz position. The electrodes contact points are adjusted applying conductive gel until the impedance values displayed by the program g.NeedAccess (g.Tec, Austria) are within the desired range. This process takes between 10 and 15 minutes. After this step, the subject is instructed to close their eyes and the same program is used to check the live channel values so that there are no dead ones and the expected values are displayed for eye movements or muscle chewing.

Once the headset is correctly applied, the OpenVibe Acquisition Server program, from the OpenVibe platform [13], is launched and configured with a sampling rate of 250 Hz. A 50 Hz notch filter is applied to filter out power line. An additional bandpass filter between 0.5 Hz and 60 Hz is applied as well. Data is handled and processed with the OpenVibe Designer, from the same platform, using 8 channels for the brain data (one channel per electrode) and an additional channel for the stimuli. After everything is connected, the subject seats in a comfortable chair in front of a computer screen. The brightness of the screen is set to the maximum setting to avoid any visual inconvenience in which the subject can not distinguish the components of the game that appear on the screen.

The Acquisition Server has the responsibility of receiving and synchronizing the signal data from the headset and any event information from the game, and transfers it to the OpenVibe Designer application. When the subject is ready, the Game Manager and the OpenVibe Designer programs are launched and configured to communicate with the previously mentioned acquisition server. A brainwave session consists of several experiences, each one being a game run. At the end, the game state information and the signal data with the game event information of each run are saved for offline processing.

A. *Cognitive Game experiment*

The game parsimoniously consists of a $5x5$ grid of grey circular spots with a black background. A blue spot indicates current position of the agent whereas a green spot represents the goal, as shown in Figure 2. The objective of the system is for the agent to reach the goal. The circular spot representing the goal remains static at the bottom-right position of the grid, while the one representing the position of the agent always starts at the upper-left position of the grid. When the agent reaches the goal, the position where the agent and the goal are located turns red, showing that the experience has ended. There are four possible movements that the agent can perform: it can go upwards, downwards, towards the left and towards the right, and those moves are bounded to avoid the agent from leaving the grid. The movement direction is selected randomly and is executed once every 2 seconds. At the end, there is a pause of 5 seconds before the next experience starts. Each time an agent moves, the Game Manager program sends an event marker to the Acquisition Server. This is considered as a stimulus to the observational human critic. The experience is designed for it to be evident whenever there is an error (i.e. the agent moves away from the objective) so the subject can notice it immediately after the stimulus is presented, possibly triggering the expected cognitive response, which can be imprinted as an ErrP component within the EEG stream.

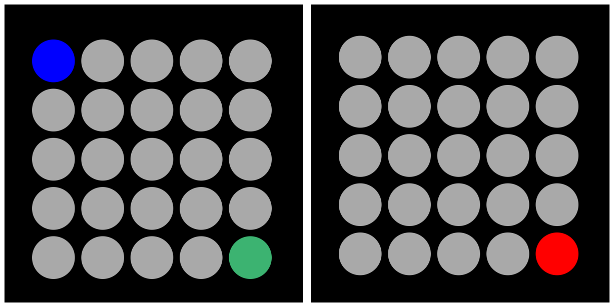
Fig. 2: Grid system representation used in the Cognitive Game Experience. The blue spot represents the initial location while the green spot represents the target location.

III. Signal Processing, Segmentation and Classification

To aid in detecting the ErrP response, an offline processing pipeline and classifier is constructed to identify whether the action taken by the agent is an error or not. It is developed in Python using the MNE platform [14], which is a package designed specifically for processing EEG and Magnetoencephalography (MEG) data, and based on top of the machine learning library Scikit-Learn [15].

This step consists of offline processing the collected signals in order to train a classifier that can decide whether an error potential is triggered or not. Firstly, the output of a brainwave session is read and an additional band pass filter of 0.1-20.0Hz is applied to the signal. Samples that correspond to the start of an event are tagged using the data from the stimuli channel.

After the data is loaded and tagged, epochs are extracted from the raw data. Epochs consist of all the sample points that take place between the start of the event and 2 seconds later (time for each action to take place), resulting in 500 samples per channel, as the sample frequency is 250 Hz. Thus, each epoch is composed of a matrix 500 x 8.

Samples that do not correspond to an epoch are not used. Also, epochs referring to the start or finish of the experience are excluded. This is done because the start and the end of the experience doesn't involve the agent taking an action, thus giving signals that should not be considered in the classification process.

In this way, the raw data of an brainwave session is processed into an array of experiences where each element is an array of epochs tagged with a number specifying the prediction of the classifier, i.e. if the epoch corresponds to an action that made the agent move further from the goal (hit) or an action that made the agent move closer to the goal (no-hit). The ErrP is expected to be found in hits. To get the data ready for classification, the stimuli channel is removed in order to classify the signals using only the EEG data. Each epoch is regularized using a MinMaxScaler, i.e. substracting the minimun value in the feature and dividing by the signal peak-to-peak amplitude. The eight channels are concatenated using the MNE Vectorizer function to transform the data matrix into a single array sample. Lastly, this data is used by the classification module as information to train and test a classifier.

Vectorized epochs are used in the final classification step. Four different classification algorithms are used. Logistic Regression, Multi-layer Perceptron with a hidden layer of 100 neurons (i.e. default values for the Scikit-Learn MLPClassifier), Random Forest and finally a Support Vector Classifier (i.e. SVM) [16].

IV. Reinforcement Learning

The set of experiences of each subject is split into training and testing, so that the results of the classification can then be used to improve the performance of the agent. Each experience includes an ordered list of extracted epochs for each movement that the agent triggered plus the game state information, i.e. the movement specification. After a classifier is trained, the epochs of the experience are classified as hit or no-hit. With the game state information and the classification of the test data of an experience, a reward file is composed. This file specifies a reward for each movement in the game run, based on the classification of the event that corresponds to that movement. The reward can either be -1 when the event is classified as a hit or 0 when it is classified as a no-hit. The accuracy of this rewards depends on the performance of the classifier. This reward file is used by the a variant of a Reinforcement Learning algorithm called Q-Learning algorithm.

A. *Q-Learning*

Q-Learning [17] is a form of model-free reinforcement learning algorithm where an agent tries an action at a particular state and evaluate its consequences in terms of the reward or penalty it receives. In order to represent rewards, a matrix is used, where rows correspond to all the possible states, whereas columns represent all possible actions. The reward for an action given a state, is the value that can be found on the corresponding row and column. This matrix is known as a Q-Table. The algorithm proceeds by randomly choosing what action to do and updating iteratively the Q-Table based on the received reward r by the following equation

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma * \max Q(s,a) - Q(s,a)] \quad (1)$$

where s is the current state, a is the action, α represents the learning rate between 0 exploration and 1 explotation and γ represents the discount rate, a value between 0 and 1 that determines the importance of long term results versus immediate rewards. Once the environment has been extensively explored and the Q-Table has been obtained, the action chosen is the one that maximizes the expected reward according to the Q-Table matrix.

For this experiment, the algorithm is developed in Python and uses the OpenAI Gym toolkit [18]. Gym is a toolkit for developing and comparing reinforcement learning algorithms. It makes no assumptions about the structure of an agent, and is compatible with any numerical computation library, such as TensorFlow or Theano [19].

B. Agent Learning

The Q-Table is initialized with zeros, unless a preexisting Q-Table is passed as a parameter. In order for the agent to learn from the feedback generated by the subject, the Q-Table is not used to determine the action to take at a given state. Instead, the policy which determines which action to take in a particular state is given by the steps taken from the brainwave session results. This allows to train the Q-Table based on the subject's feedback from the movements the agent took, which are chosen pseudo-randomly, while executing the brainwave session. The previously mentioned feedback is not explicit as it comes from the interpreted brain signal data, which is collected while the agent is executing the brainwave session and then each action is classified as an error or not. This implies that the reward is determined by the subject's brain activity.

While using the previously mentioned step function, the Q-Table is updated in each iteration. This is done following the Equation 1. After the algorithm finishes iterating through all the training episodes, the Q-Table is stored to test the performance of the agent.

V. Results

Figures 3 and 4 show the binary classification accuracy obtained for the eight subjects for the four different classification algorithms using a 10-fold cross validation procedure. The best overall performance is obtained using Logistic Regression.

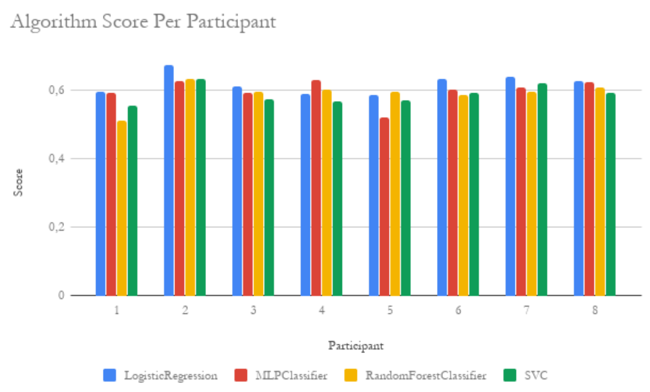

Fig. 3: Binary Classification Accuracy using different classifiers while recognizing ErrP potentials for the eight subjects.

On the other hand, Figure 5 shows for each subject the average amount of steps the agent takes to reach the goal as the Q-Table is progressively trained using the reward information obtained from their tested experiences. Each point corresponds to the average number of steps in 200 iterations that it takes for the agent to reach the goal for a specific Q-Table. The first point represents the amount of steps it takes to reach the goal for a Q-Table that hasn't been trained at all, where movements are decided randomly. The next point corresponds to the amount of steps it takes to reach the goal using a policy derived from a Q-Table trained with one experience, and so on.

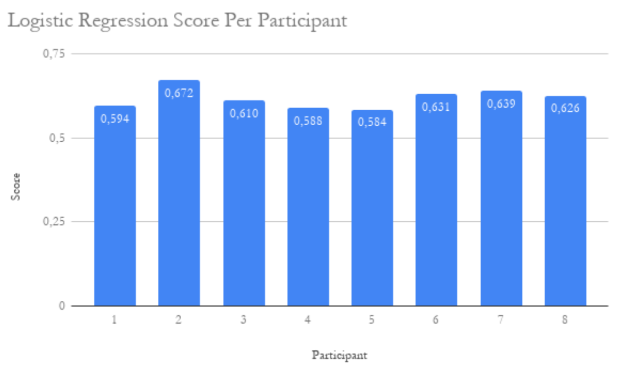

Fig. 4: Binary Classification Accuracy using Logistic Regression for the eight subjects.

The results show that as the Q-Table is progressively trained the average amount of steps decreases, meaning that the agent learns. However, the rate at which it learns varies per subject, certain subjects have more effective experiments than others. For example results for subject 1 (fig 5 A) show faster learning than those of subject 8 (fig 5 H).

In the case for subject 5 and 6, the reward information obtained from the brainwaves is not enough to train the agent effectively. Figures 5 E and 5 F show no apparent learning, as the amount of steps to reach the goal doesn't decrease when trained. These results are also consistent with their ROC curves, shown on Figures 6 obtained for both Subjecs, where the area under the curve are close to chance level. Both subjects have less recorded data from the sessions in comparison to the rest of the subjects.

Figure 7 shows the result of an agent successively trained with experiences obtained from a brainwave session, generated with sham signals. In this case, random EEG signals were generated using OpenVibe Acquisition Server signal generator for all channels, as if they were generated from a human observer. As it can be seen, the agent learns nothing, and regardless of the amount of experiences that are used to train the Q-Table, the number of steps required to reach the goal does not decrease. This pattern is also obtained when the experiences from Subjects 5 and 6 are used, showing that the reward labeling predicted by the trained classifier for those subjects worked like a random classifier.

Electroencephalographic signals have high inter-subject variability [4]. This is evidenced in Figure 8 where the agent training is performed by using a classifier trained with experiences from one subject, but tagging the experiences from a different subject using the predicted classification. Not performance gain is evidenced, the agents learn nothing which implies that the reward information is useless.

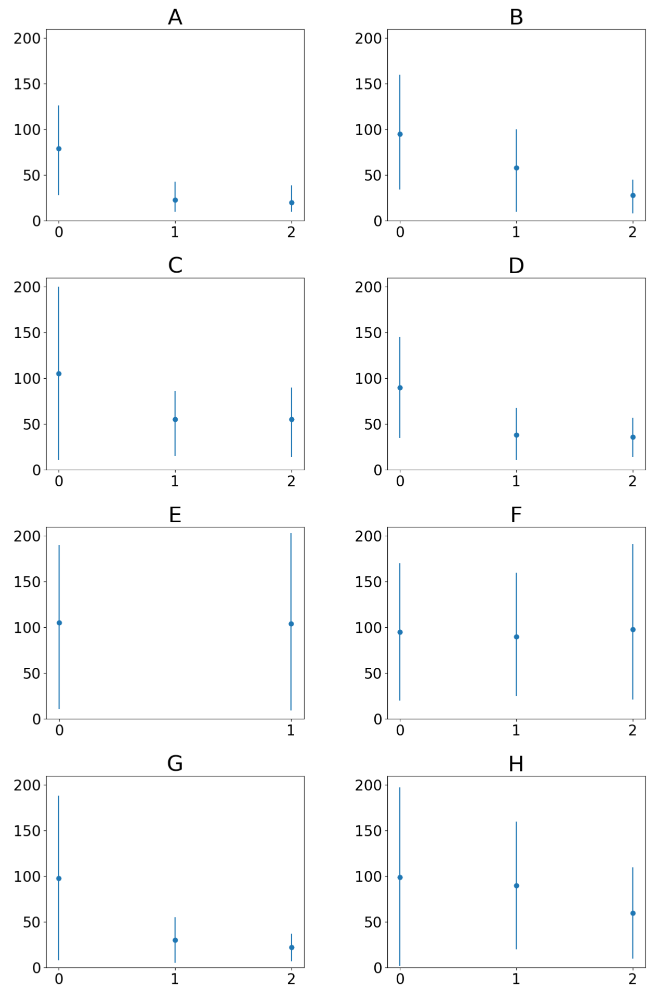

Fig. 5: Average number of steps for the agent when trained with experiences from subjects 1(A)-8(H). Y axis show the averaged number of steps, while x axis show the number of experiences used to cumulative train the Q-Table.

Finally, Figure 9 shows the result of training an agent with accumulative experiences from subjects 1,2,3,4,7, and 8.

VI. Conclusion

The present research work aims to reproduce similar results that states whether ErrP signals could be used to train a gaming agent using reinforcement learning. The collected data show that ErrP signals can in fact be classified and used to train an agent effectively.

This work aims to keep the system as simple as possible, emphasizing the flow of information from the subjective error perception of the human critic, through the reward generation using the signal processing and classification pipeline, and finally Q-Table updating to enhance the performance of the gaming agent.

While classifying, the obtained better performing classifier is Logistic Regression. One important aspect of the classification results is the low percentage of false positives (Figure 10), showing a high specificity. It is not common that the agent learns that an action is wrong when in fact it is an action that takes it closer to the goal. On the other hand, the percentage of false negatives is generally higher, but this is not a serious issue since missing out on learning that an action is wrong does not lower the performance of the agent, but only means it will take more experiences to learn a correct path.

Once ErrP signals are identified they can be used to train a reinforcement learning algorithm. Brainwave sessions have a low amount of experiences in order to reduce fatigue from the subjects. However data suggests that longer sessions are required in order to reach better classification scores, since more data is available in order to train the classifier. It can be seen that subjects with the largest amounts of data have the best classification. This can also be achieved designing a

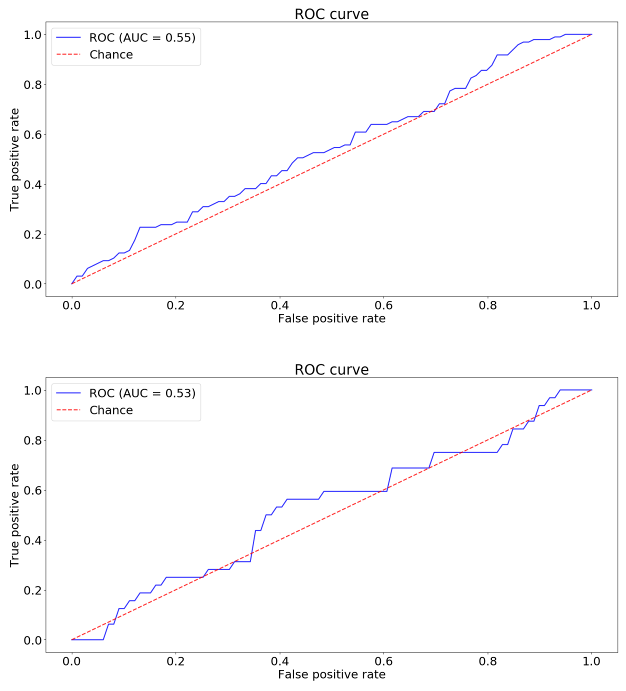

Fig. 6: ROC Curves for Subjects 5 (up) and 6 (down)

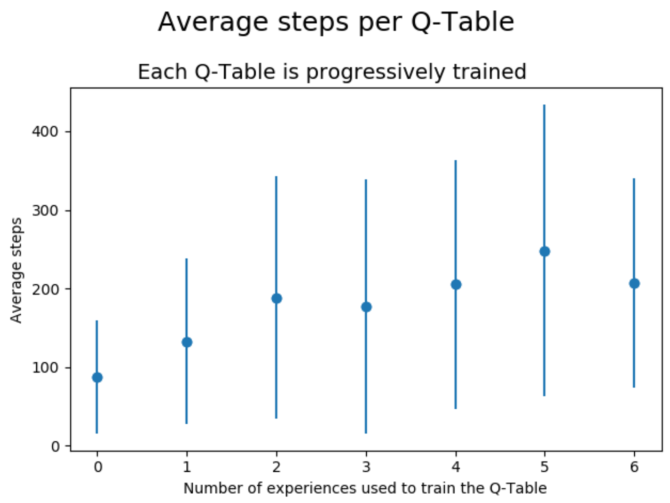

Fig. 7: Average steps using Q-Table trained with a noisy signal.

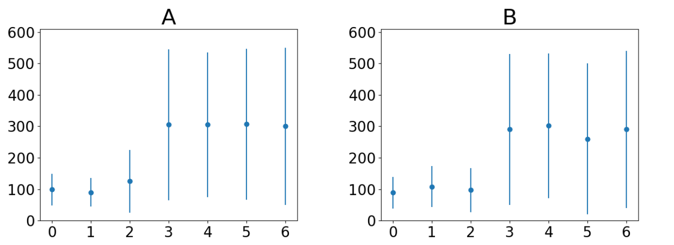

Fig. 8: (A) Average steps using Q-table trained with experiences from subject 8 classified with a classifier trained with data from subject 6. (B) Average steps using Q-table trained with experiences from subject 1 classified with a classifier trained with data from subject 3.

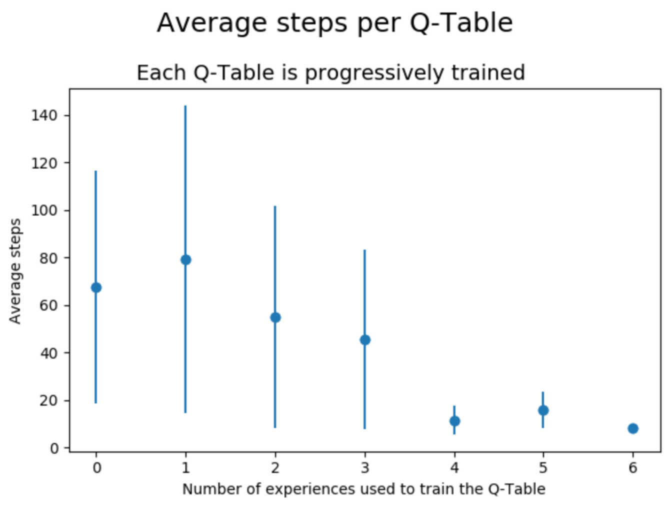

Fig. 9: Average steps using Q-Table trained with 6 experiences from different subjects.

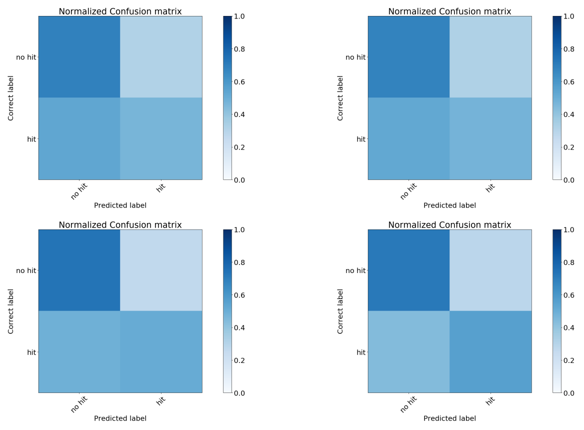

Fig. 10: Confusion Matrix for subjects 1,3,4 and 8. Darker colors show higher values. It can be seen the lower percentage of false positives (upper right corner of each chart).

bigger game system that generates more samples with every session.

At the same time, the effective agent training depends on the subject's data that was used to train it. Results show that training a classifier with data of one subject, but using it to classify the events of experiences of another subject does not lead to an improvement on the performance of the agent. Despite that, the rewards generated from different subjects can be used to train the same Q-Table to improve its performance, which may lead to strategies where the overall performance is improved based on the information from different human critics at the same time.

One additional aspect to remark is the robustness of the learning strategy based on Q-Learning [20], [21]. The obtained accuracy to discriminate ErrPs is low. However, even with such a lower accuracy values, the RL algorithm was able to extract meaningful information from rewards files that were helpful to improve the agent performance.

Further work could be conducted in order to increase the complexity of the game to allow at least the possibility that the target position be dynamically changed. In that case the agent would start to learn to follow the target, instead of learning to go to a specific static destination spot. Additionally, the classifier could be enhanced to recognize more effectively the Error Potential.

Finally, in line with other results, this research verifies that brain signals can be used as an interface between human and a gaming computer enabling an alternative communication with the system without explicit input from the user.

Acknowledgment

The authors would like to thank...

References

[1] N. P. B. Thorsten O. Zander, Laurens R. Krol and K. Gramann, *Neuroadaptive technology enables implicit cursor control based on medial prefrontal cortex activity*, 2016.

[2] C. B. Holroyd, O. E. Krigolson, R. Baker, S. Lee, and J. Gibson, “When is an error not a prediction error? An electrophysiological investigation,” *Cognitive, Affective and Behavioral Neuroscience*, vol. 9, no. 1, pp. 59–70, mar 2009. [Online]. Available: http://www.springerlink.com/index/10.3758/CABN.9.1.59

[3] P. Ferrez, “Error-related eeg potentials in brain-computer interfaces,” 2007.

[4] R. Chavarriaga, A. Sobolewski, and J. d. R. Millán, “Errare machinale est: The use of error-related potentials in brain-machine interfaces,” *Frontiers in Neuroscience*, vol. 8, no. 8 JUL, p. 208, jul 2014. [Online]. Available: http://journal.frontiersin.org/article/10.3389/fnins.2014.00208/abstract

[5] J. M. Santos and C. Touzet, “Exploration tuned reinforcement function,” *Neurocomputing*, vol. 28, no. 1-3, pp. 93–105, oct 1999. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231298001179

[6] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[7] K. S. I. A. A. H. A. G. T. H. L. B. M. L. A. B. Y. C. T. L. F. H. L. S. G. v. d. D. T. G. D. H. David Silver, Julian Schrittwieser, “Mastering the game of go without human knowledge,” 2017.

[8] J. M. I. Iturrate, L. Montesano, “Robot reinforcement learning using eeg-based reward signals,” 2010.

[9] J. Omedes, I. Iturrate, L. Montesano, and J. Minguez, “Using frequency-domain features for the generalization of EEG error-related potentials among different tasks,” in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*. IEEE, jul 2013, pp. 5263–5266. [Online]. Available: http://ieeexplore.ieee.org/document/6610736/

[10] L. Schiatti, J. Tessadori, N. Deshpande, G. Barresi, L. C. King, and L. S. Mattos, “Human in the Loop of Robot Learning: EEG-Based Reward Signal for Target Identification and Reaching Task,” in *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, may 2018, pp. 4473–4480. [Online]. Available: https://ieeexplore.ieee.org/document/8460551/

[11] D. Plass-Oude Bos, B. Reuderink, B. van de Laar, H. Gürkök, C. Mühl, M. Poel, A. Nijholt, and D. Heylen, "Brain-Computer Interfacing and Games." Springer, London, 2010, pp. 149–178. [Online]. Available: http://link.springer.com/10.1007/978-1-84996-272-8{_}10

[12] S. E. Kober, M. Ninaus, E. V. Friedrich, and R. Scherer, "Bci and games: Playful, experience-oriented learning by vivid feedback?" in *Brain–Computer Interfaces Handbook*. CRC Press, 2018, pp. 209–234.

[13] G. G. M. C. E. M. V. D. O. B. Yann Renard, Fabien Lotte and A. Lecuyer, "Openvibe: An open-source software platform to design, test and use brain-computer interfaces in real and virtual environments," 2010.

[14] E. L. D. E. D. S. C. B. R. G. M. J. T. B. L. P. M. H. A. Gramfort, M. Luessi, *MEG and EEG data analysis with MNE-Python, Frontiers in Neuroscience, Volume 7*, 2013. [Online]. Available: https://mne-tools.github.io/0.13/index.html

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[16] F. Lotte, L. Bougrain, A. Cichocki, M. Clerc, M. Congedo, A. Rakotomamonjy, and F. Yger, "A review of classification algorithms for EEG-based brain-computer interfaces: A 10 year update," *Journal of Neural Engineering*, vol. 15, no. 3, p. 031005, jun 2018. [Online]. Available: http://stacks.iop.org/1741-2552/15/i=3/a=031005?key=crossref.9cd2b15ab65c8ad34b475584b43dc509

[17] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, may 1992. [Online]. Available: http://link.springer.com/10.1007/BF00992698

[18] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[19] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[20] R. Bauer and A. Gharabaghi, "Reinforcement learning for adaptive threshold control of restorative brain-computer interfaces: A Bayesian simulation," *Frontiers in Neuroscience*, vol. 9, no. FEB, p. 36, feb 2015. [Online]. Available: http://journal.frontiersin.org/Article/10.3389/fnins.2015.00036/abstract

[21] J. Rubin, O. Shamir, and N. Tishby, "Trading value and information in MDPs," in *Intelligent Systems Reference Library*. Springer, Berlin, Heidelberg, 2012, vol. 28, pp. 57–74. [Online]. Available: http://link.springer.com/10.1007/978-3-642-24647-0{_}3