Training an 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 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 the most popularity is Electroencephalography (EEG), mainly because of its effectiveness, low cost and because it is noninvasive. 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 are 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 when 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. This work follow this path by detecting the ErrP potential from subjects witnessing an agent playing the game and improving the agent behavior using RL and obtaining rewards from the subjects.**

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

I. Introduction

THE effectiveness of today's human–machine interaction is limited by a communication bottleneck as operators are required to translate high-level concepts into a machine-mandated sequence of instructions [1]. This work tackles directly this problem by exploring the use of brain signals as an interface between human and computer, overcoming this limitation by enabling the control of the system without explicit communication from the user.

Using reinforcement learning with reward signals based on brain activity, recorded by an EEG-based BCI system during the task execution, in order to train an agent has been explored in previous research. The paper [2] has successfully demonstrated that a robot can be controlled by obtaining the reward from brain activity generated while observing the robot solving a task. The present project applies a similar process for making a virtual agent improve its performance using electroencephalography (EEG) signals as feedback for a reinforcement learning algorithm. The idea is that subjects observing a simple simulated system can train such system using only error-related potentials (ErrP) that can be identified in their brain signals. The detection of ErrP signals has been explored in the publication [3] which describes the methods that are used in order to identify this specific potential from EEG signals.

In recent years reinforcement learning has seen a come-back. This is mainly because of DeepBrain's AlphaGo who was the first to reach superhuman proficiency in challenging domains, when it won the complex game Go against several world champions. In 2015 AlphaGo won 5 matches against 3-times European Champion, Mr Fan Hui. AlphaGo then went on to compete against Mr Lee Sedol, winner of 18 world titles and considered to be the greatest player until then, and won 4 out of 5 matches in 2016 [4].

In Section I, in Section II and Results and conclusions are exposed at the end.

II. Materials and Methods

The project consists of collecting signals from a person's brain while they are watching a game where the agent has knowledge of the way it's played but not how to win it. Hence, the agent can learn using the person's feedback to get better at the game. This section describes the process of obtaining the person's brain signals and how they are used to make the agent improve it's performance.

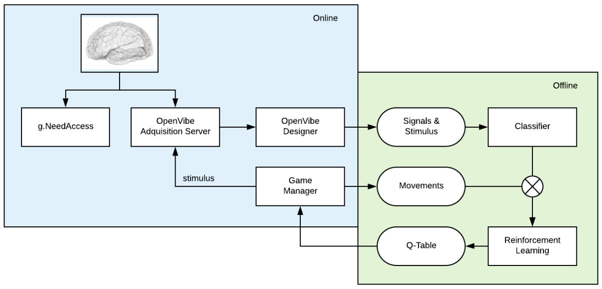


Fig. 1: Overview of the experimental procedure.

The experimental procedure is summarized in Figure 1. The core of the procedure is the retrieval of the subject's brain activity. This process is called brainwave session. In order to fetch this data, subjects use a wireless digital EEG device during the experiment. Once the subject has the headset on, a set up procedure is executed as described in Section **??** using a program called g.NeedAcceess to do an impedance check and to visually inspect the EEG signals in order to check if it's correctly applied. Subsequently, the OpenVibe Acquisition Server program is launched and configured. This program is part of the OpenVibe platform [5] and has the responsibility of receiving the signals data from the headset and stimulus information from the game and transfers it to the OpenVibe Designer application. After this step, the Game and the OpenVibe Designer programs are launched and configured to communicate with the previously mentioned acquisition server. Once the subject is ready the computer is positioned in front of them and the brainwave session starts. A brainwave

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.

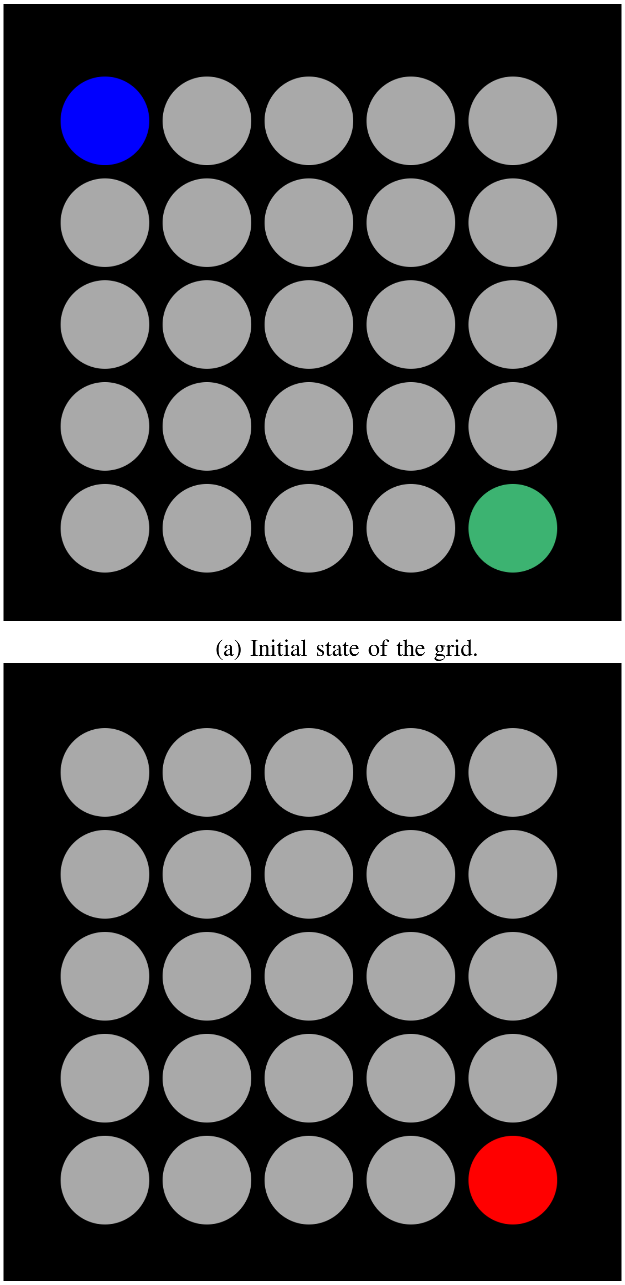
Manuscript received April 19, 2005; revised August 26, 2015.

session consists of several experiences, each of them being a game run, and it uses a pseudo-random function to determine the agent's movements. The system is described in more detail in Section II-A and the brainwave session in Section **??**. Once the experimental procedure is completed, the game state information and the signals data with the stimuli channel information of each run are stored. For each run, the signals and stimulus information are then passed to the classification module that uses them to train the classifier. The classification step is covered in detail in Section **??**. After this step is finished, the game movements and the trained classifier are used to update a Q-Table for each experience. Lastly, the calculated Q-Table is used to test if the agent has boosted its performance while playing the game. This module is explained in Section VI-A.

A. Cognitive Game experiment

This section details the game system characteristics, the brainwave session process and the retrieval and analysis of the generated data from the subjects interaction with the system.

The system consists of a $5x5$ grid where there are two lights, one representing the current position of the agent, and the other one representing the goal, shown in figure 2. The objective of the system is for the agent to reach the goal. The light 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. The grid is represented by circular spots with a black background. The agent's position has a blue light, the goal position has a green light, and the rest of cells are represented with grey spots. 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, as long as the move doesn't make the agent leave the grid. The movement direction is selected randomly and is executed once every 2 seconds. When the experience ends, there is a pause of 5 seconds before the next experience starts. The experience is designed for it to be evident whenever there is an error (when the agent moves away from the objective) so the subject can notice it immediately after the stimulus is presented, possibly triggering a cognitive response.



(a) Initial state of the grid.

(b) Final state of the grid.

Fig. 2: Grid system representation used in cognitive experience.

III. Calibration

The first step in order to be able to find ErrP signals is to choose the most efficient algorithm, and the proper calibration. In order to do this different parameters are tested for a set of algorithms and for each individual subject. Initially a sub-selection of experiments is used to define a subset of parameters to test with, and then all the data is tested with this subset of parameters.

IV. Signal Processing and Segmentation

The goal of this module is to build a classifier that is able to identify whether the action taken by the agent is an error or not. It is developed in Python using the MNE environment [**?**], which is a package designed specifically for processing EEG and MEG data, and the machine learning library Scikit-Learn.

Many classification methods are considered. The best classifier is selected based on a calibration and selection procedure that is further explained in sections **??** and **??**. After an optimal algorithm and a configuration are selected, a classifier is trained for each subject and then epochs from their experiences are classified. These classifications are then used by a reinforcement learning algorithm to train the agent. This last algorithm is explained in detail in section VI-A.

This step consists of 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 a band pass filter of 0.1-20.0Hz is applied to the signal, as seen in Figure **??**. 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.

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 entire brainwave session is processed into an array of experiences where each element is an array of epochs tagged with a number specifying 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 and a mne.decoding.vectorizer [1] function is applied to transform the data into a single array sample. Lastly, this data is used by the classification module as information to train and test a classifier.

V. Classification with Logistic Regression

For the logistic regression algorithm the parameters that are tested are C and the Solver. The C parameter is the inverse of the regularization strength. The values this parameter can take are 0.001, 0.01, 0.1, 1 and 10. The solver parameter refers to the algorithm that is used in the optimization problem, the ones that are tested are $lbfgs$ and $saga$. The different configurations that resulted as optimal for subjects can be seen in the apendix section **??**.

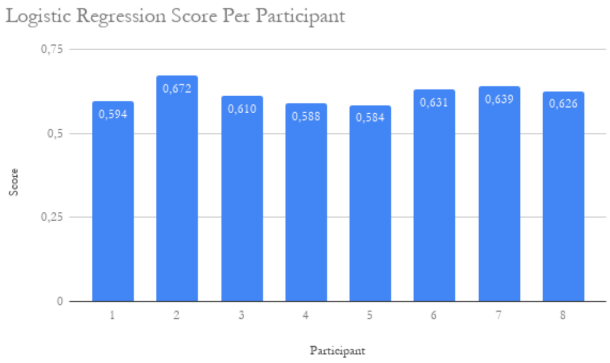


Fig. 3: Logistic Regression Algorithm Calibration

VI. 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. After training the classifier, the testing experiences are classified. Then, for each experience a reward file is generated. For this purpose, game states files, which are submitted by the game when an experience takes place, are used. These files contain an ordered list of the states of the game throughout an experience. With this file and the classification of the test data of an experience, a reward file is submitted. This file specifies a reward for each movement in the game, 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 Q-Learning algorithm to improve its performance, as detailed in section VI-A.

A. *Q-Learning*

A Q-Learning algorithm is used to train the agent. It is developed in Python and uses the OpenAI Gym toolkit. 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.

1) Algorithm implementation: 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. This is covered in further detail in **??**.

While using the previously mentioned step function, the Q-Table is updated in each iteration. This is done following the Equation 1.

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

After the algorithm finishes iterating through all the training episodes, the Q-Table is saved. Each experience is one $trainingEpisode$.

The reward function is calculated based on the current distance to the goal, if it has increased compared to the previous step then the rewards is negative, if not it returns zero. In order to test how the Q-Table is trained when the accuracy is not perfect the reward function doesn't always return the correct reward. It calculates a random number between 0 and

[1] Link to vectorizer documentation: https://mne.tools/dev/generated/mne.decoding.Vectorizer.html

1, and if it is larger than the accuracy then it will return 0 even if it should return -1 as the reward.

The reward function is described in algorithm **??**.

```
$reward \leftarrow 0$
if currentDistanceToGoal ¿ previousDistanceToGoal then
    $reward \leftarrow -1$
else if random.uniform(0, 1) ¡ accuracy then
    return reward
else
    return 0
end
```

Algorithm 1: Reward Calculation for $ChaseEnv$

VII. Results

A. Reinforcement Learning

Average steps to goal Figure **??** shows the average amount of steps it takes to reach the goal as the Q-Table is progressively trained using the reward information obtained for each subject. Each point corresponds to the average amount of steps it takes for the agent to reach the goal for a specific Q-Table in 200 iterations. 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 Q-Table trained with one experience, and so on.

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 4) show faster learning than those of subject 8 (fig 11).

In the case for subject 5 and 6, the reward information obtained from the brainwaves is not enough to train the agent effectively. Figures 8 and 9 show no apparent learning, as the amount of steps to reach the goal doesn't decrease when trained. Both subjects have less recorded data from the sessions in comparison to the rest of the subjects. In particular, subject 5 has less recorded experiences, so it is respective graphic shows the amount of steps for an empty Q-Table and for a Q-Table trained with only one experience. These results are consistent with fig. **??** and fig. **??**, which show that the signal classification for these subjects hasn't been particularly successful. The average steps result for these subject are similar to that of fig. **??** which is constructed from noise, so the generated Q-Table would be similar to one generated with noise.

VIII. Conclusion

The present research work set out to validate if ErrP signals could be used to train an agent using reinforcement learning. The collected data show that ErrP signals can in fact be classified and used to train an agent effectively.

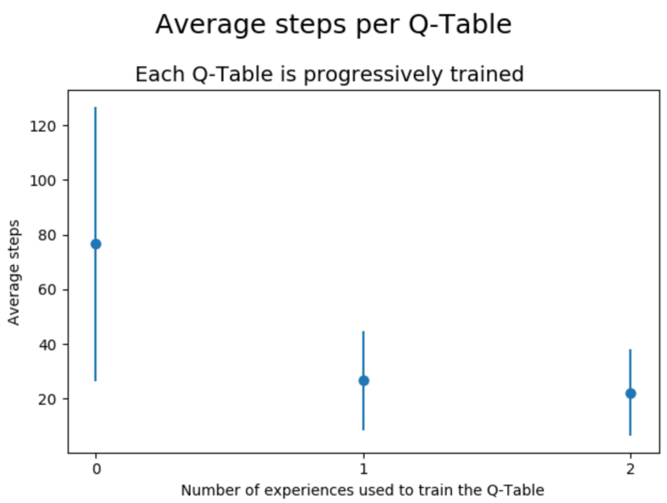


Fig. 4: Average steps using Q-table trained with the results of subject 1

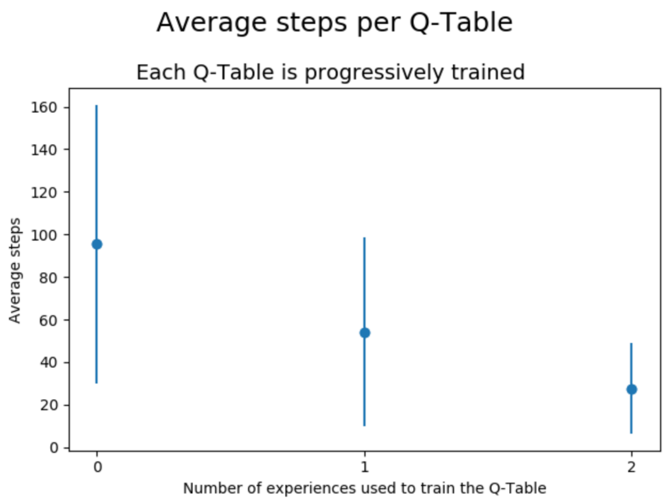


Fig. 5: Average steps using Q-table trained with the results of subject 2

When classifying, the better performing classifier is Logistic Regression. One important aspect of the classification results is the low percentage of false positives, meaning that 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. However, this can only be achieved by generating a specific classifier for each subject and using it for giving rewards for the corresponding subjects. 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, concluding that the experiment is not suited for

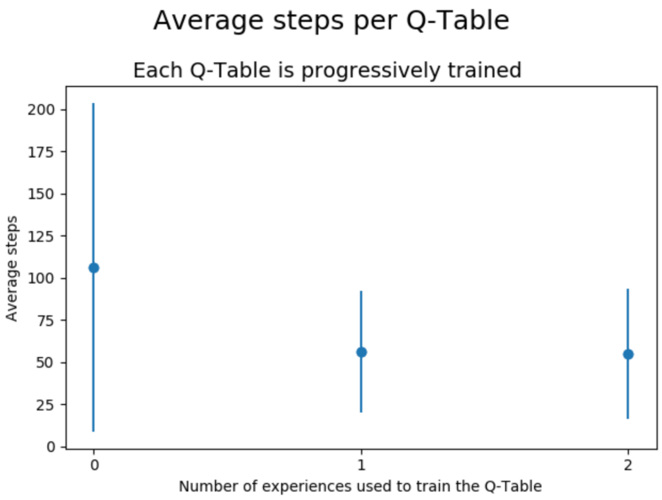


Fig. 6: Average steps using Q-table trained with the results of subject 3

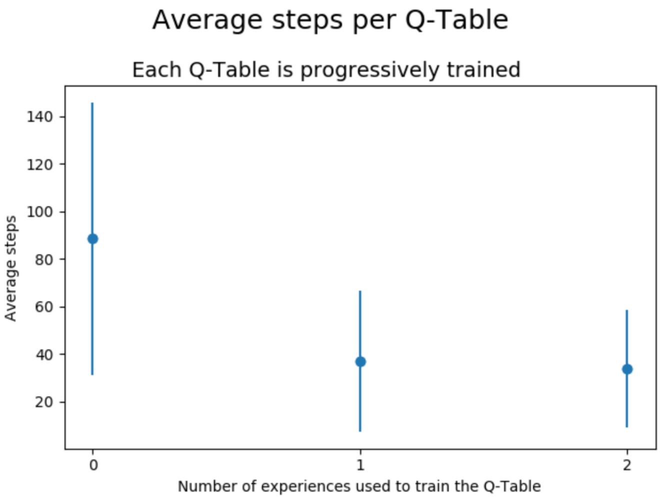


Fig. 7: Average steps using Q-table trained with the results of subject 4

transfer learning. Despite that, the rewards generated from different subjects can be used to train the same Q-Table to improve its performance.

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 bigger game system that generates more samples with every session.

Finally, this research verifies that brain signals can be used as an interface between human and computer enabling the control of the system without explicit input from the user.

Acknowledgment

The authors would like to thank...

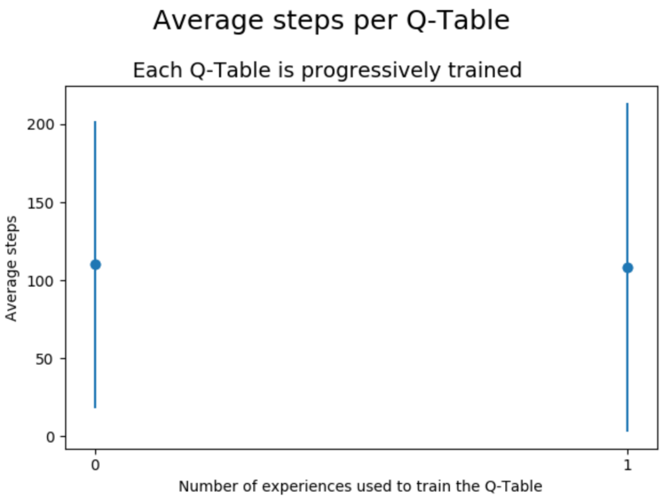


Fig. 8: Average steps using Q-table trained with the results of subject 5

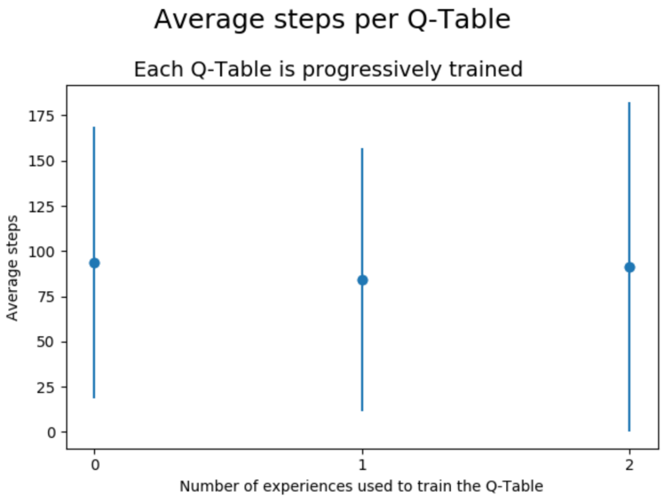


Fig. 9: Average steps using Q-table trained with the results of subject 6

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] J. M. I. Iturrate, L. Montesano, "Robot reinforcement learning using eeg-based reward signals," 2010.

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

[4] 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.

[5] 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.

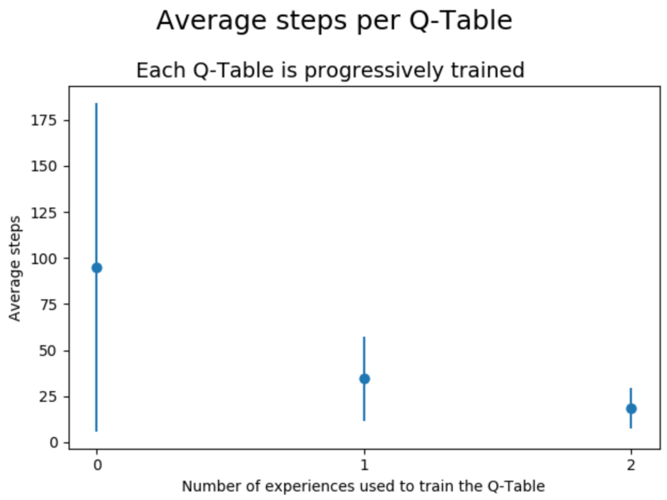


Fig. 10: Average steps using Q-table trained with the results of subject 7

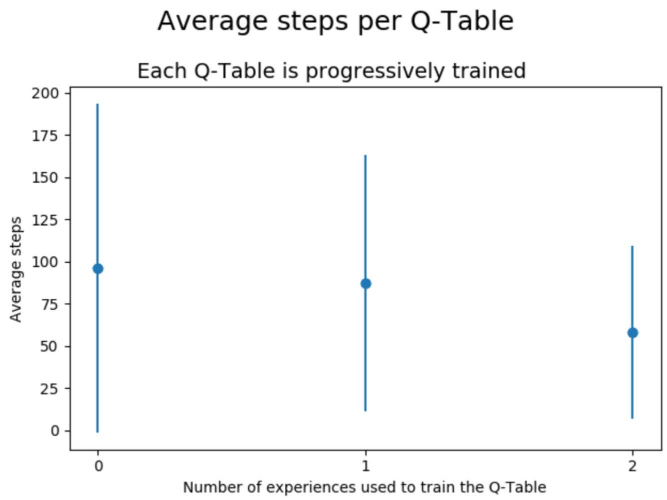


Fig. 11: Average steps using Q-table trained with the results of subject 8

PLACE PHOTO HERE

Michael Shell Biography text here.

John Doe Biography text here.

Jane Doe Biography text here.