Error Potentials for a Reinforcement Learning-Based Brain-Computer Interface

Gabriel Ibagon
gibagon@ucsd.edu
Department of Computer Science
University of California, San Diego

I. Introduction

Advances in computational intelligence have allowed us to reconsider the balance of autonomy in human-computer interactions. Artificially intelligent agents are able to learn powerful representations of environments, allowing computers to make increasingly sophisticated decisions based on limited inputs. Various fields of research are exploring how to best leverage these computer agents to act in cooperation with human cognition to accomplish a range of tasks.

Brain-computer interfaces introduce an intimate paradigm of human-computer interaction, where computers access a direct line of information from a user's neural signals. Computer systems can utilize this information to perform tasks based on the user's state, behavior, or intention. Researchers aim to increase the effectiveness of this information transfer, as well as create BCIs robust to new tasks and environments.

The shared-control approach to BCIs allows certain aspects of decision making to be divided between the user and an artificially intelligent agent. The intention of this cooperation is to transfer certain responsibilities of decision-making and control from the human to the computer. If an artificially intelligent agent can act in accordance with user intention, using minimal information from the user, we can remove the burden and limitations inherent in operating computer systems to accomplish certain tasks.

This report will explore the use of non-invasive electroencephalography (EEG) to detect event-related potentials (ERPs) in order to teach an artificially intelligent agent to perform a task. Error potentials (ErrP) emitted during user observation of the task provides relevant feedback necessary to inform the agent how to behave according to the user's intentions. Through this, I aim to propose a model of a reinforcement learning-based BCI (RL-BCI) that is robust to a variety of tasks and environments.

II. The Error Potential

The presence of a neurological potential in response to the perception of error was first detected by several independent groups around 1990 [1] [2]. Researchers have since explored various properties, detection techniques, and applications of this signal for BCIs.

A. Characteristics

A distinct ERP is unconsciously evoked in response to the performance or observation of an error. This potential is referred to as the Error Potential (ErrP), and the shape, latency, and intensity of this potential can vary depending on the form of the stimulus [3]. Although ErrPs vary across different individuals and forms of stimuli, they can still be characterized by a positive peak at about 50-100ms after the stimulus, followed by a strong negative peak around 200-400ms [4]. This waveform is thought to originate from areas of the medial-frontal cortex, particularly the anterior cingulate cortex.

B. Error Potentials in BCI

Error potentials have been used as input to BCIs as both a corrective signal and a control signal.

As a corrective signal, ErrPs have been used to send feedback to an EEG decoder in order to improve classification accuracy [5].

Other researchers have implemented ErrP as a control signal in order to achieve external control through a BCI. The task of external control involves using neural signals to send control signals to a system in order to dictate its actions. External control has been achieved using a number of BCI paradigms, such as the P300, SSVEP, and motor imagery [6] [7] [8]. In the case of motor imagery-based *active brain-computer interfaces*, the user consciously alters their brain signal in order to control the object. Although a great degree of control can be exercised by using an active approach, the approach is limited due to user fatigue associated with continuous modulation of neural signals. Thus, active BCIs for external control become less effective throughout a session, as fatigue begins to distort the neural signals of interest, thus leading to less accurate decoding. Furthermore, extensive training is necessary in order for a user to successful induce neural changes associated with motor imagery, limiting the accessibility of this paradigm.

Using an implicitly evoked signal such as ErrPs avoids the issue of user fatigue. The unconscious nature of ErrPs make error responses prime candidates for use within a *passive brain-computer interface* (pBCI), a BCI that reacts to neural signals without the user's conscious control [9]. The user observes the actions of an agent in an environment, and, based on whether or not these actions are congruent with the user's goal, an ErrP is implicitly evoked. Furthermore, the binary nature ErrP classification allows the user to control decisions over a wide variety of actions, environments, and tasks, simply by evaluating whether or not an action is erroneous.

III. Reinforcement Learning

A. Introduction

Reinforcement learning is a machine learning technique that determines an optimal sequence of actions based on the evaluation of previous agent-environment interactions [10]. In this paradigm, we are presented with an agent, $\boldsymbol{A}$, and an environment, represented by a collection of states $\boldsymbol{S}$. The agent performs actions within the environment, where the action performed at each state is determined by the agent's *policy*. These actions are evaluated through feedback from the environment, in the form of a numerical reward or punishment. The environment uses this feedback to discover the *optimal policy*, which is the sequence of state-action pairs that will lead to the highest long-term reward. There are various algorithms to determine how and when this reward is assessed, such as SARSA, policy gradients, and Q-learning.

B. Q-Learning

This report focuses on the Q-learning variant of reinforcement learning. In Q-learning, the agent attempts to derive a policy that attains the maximum reward at each state. Each state-action pair has a Q-value associated with it, and this Q-value represents the current value attributed with taking that action in that state. During learning, the environment returns immediate feedback after each state transition, which then updates the Q-value associated with the state-action pair. The Q-value update formula associated with each state transition is as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{1}$$

where $Q(s_t, a_t)$ is the Q-value of the state-action pair, r_{t+1} is the reward associated with the transition, and $\max_a Q(s_{t+1}, a)$ is the maximum Q-value attainable in the next state. There are two hyperparameters: α, the learning rate and γ, the discount rate.

The agent-environment interaction is continued until the optimal policy is established – in the case of Q-learning, this is that state sequence of maximum Q-value.

C. Reinforcement Learning-Based BCIs

This report focuses on the use of reinforcement learning to achieve a form of external control with a BCI. These reinforcement learning-based brain-computer interfaces (RL-BCI) contain the components of reinforcement learning systems: an environment, an intelligent agent, and an environmental feedback signal. The RL-BCI is used to achieve external control over this agent by teaching the agent to make goal-oriented decisions based on learned behavior. The user of the RL-BCI observes the agent interacting with it's an environment, and ErrPs are implicitly evoked if the user implicitly evaluates the agent's behavior as erroneous to the goal of the task. Thus, the presence of an ErrP is used as environmental feedback to the agent, providing a teaching signal so that it can learn behavior that is optimal to performing the task.

There are several ways to implement this teaching algorithm, each with their strengths and weaknesses towards accomplishing certain types of goals. One algorithm implements the agent as a *goal-seeking* agent, where the learning paradigm focuses on teaching the agent to identify the location of the goal. For example, in cursor navigation task, the cursor (which is the agent) learns the optimal directional vector in which to move, and thus all future actions are in accordance with this locational estimate. The strengths of this approach are that it can quickly learn an optimal policy for simple environments where every action can be approximated by one policy. An alternative approach implements the agent as a *path-seeking* agent, where the learning paradigm focuses on teaching the agent to consider the optimal action for each state individual. This path-seeking model is explored throughout the next few sections for the proposed model of a robust RL-BCI.

IV. Proposed Model for a Reinforcement Learning-based BCI

The model proposed in this paper extends the successes of past RL-BCIs by attempting to address the limitations in task scalability and information transfer rate. The following sections will explain the details and algorithms associated with this model.

A. Goal

The purpose of this interface is to achieve external control via non-invasive EEG signals. By solely relying on error potentials as user input, this model is designed to generalize to a variety of external objects and performance tasks, without any additional modification to the basic control scheme.

B. Training

In an RL-BCI for external control, the controlled object acts as an *autonomous agent*, since it alone determines the sequence of actions to take. The agent develops this policy through a reinforcement learning paradigm, based on feedback provided by the human observer.

By using a Q-learning training algorithm for the reinforcement learning model, this model emphasizes control over each action performed during the undertaking of a task. In other words, the learning scheme relies on the agent learning an optimal state-action pair for every state, rather than trying to learn a general strategy based on the end-goal of the task. The subtle difference in its objective allows for the RL-BCI to be used in much more complex environments and tasks, which will be further explored in the **Environments** and **Tasks** section.

The training sequence is described in the following paragraph. The agent is initialized using an arbitrary policy; the Q-values of each state-action pair are initialized with random values between 0 and 1. The agent begins in an initial state and chooses its first decision based on the action with the maximum Q-value in that state. After the agent performs this action, the user's EEG signal is analyzed for the presence of an error potential. If an error potential is detected, the reward

for that action is -1. If no error signal is detected, the reward is +1. Then, using the update formula in (1), the Q-value for that state-action pair is updated accordingly. This sequence is performed continuously until the agent learns the optimal policy for that environment – that is, the agent attributes the maximum Q-values to the state-action pairs that lead to the least frequent occurrences of the error signal. This optimal policy allows the agent to act in accordance with the user's intention of accomplishing a task.

C. Environments

The use of this Q-learning-based RL-BCI emphasizes user control of the sequence of actions during the accomplishment of a task, rather than solely on training the agent to reach the target of a task. This detail allows this model to be used for a far wider range of environments than the target-seeking agents, as well as for a new subset of tasks that emphasize paths in addition to the target.

These models were evaluated using simulated control signals, provided by button presses corresponding to user evaluation of the state-action pair. Although this does not accurately reflect the performance of a BCI in the system, the following experiments were used to demonstrate the potential for various environments to be used with the proposed learning scheme.

Figure 1 demonstrates the variety of environments in which this model can perform well, using the context of a navigation task. In all of the following environments, the user intends to move the cursor (green) towards the target (red) using a series of actions to transfer between the states. At each state, there are 8 actions that the agent can perform: North, Northeast, East, Southeast, South, Southwest, West, and Northwest.

Figure 1a shows a simple 6x6 square open environment, where user intends to reach the target without any particular constraints on the path taken. The initial policy is shown in **Figure 1c**, and its final policy after convergence is shown in **Figure 1g**. This straightforward environment shows a basic case of how Q-learning converges to an optimal policy. This is further illustrated in **Figure 2**, which demonstrates the convergence to a minimum number of steps as more targets are reached.

Figure 1b shows a more complex environment, where an environmental obstacle obscures a linear path between the initial state and the target. The target-seeking model would not be able to converge in this environment, since the state-action pair sequence could not be accurately described by a simple directional vector (although a model with more complex, non-linear directional representations could achieve this). However, the path-seeking RL-BCI could easily learn the policy that accomplishes this task, as the path can be accurately described in a series of state-action pairs (see **Figure 1h**.

Furthermore, the environment in Figure 1c demonstrates the flexibility of choice between multiple paths when reaching a target. In this environment, an obstacle creates two paths towards the target. This creates a conundrum for target-seeking reinforcement learning systems, since no path is more optimal than the other in terms of distance. However, in the case that one path is preferred over the other, the path-seeking RL-BCI model allows the user to teach the agent to learn that path. This preference may stem from features of the environment that might not be obvious to an autonomous agent, such as emotional preference or other forms of prior knowledge that cannot easily be transferred to the agent. By allowing the user to select between various paths, the user can teach the agent policies driven by more complex rational than metrics like distance. The optimal policies for two different paths are shown in **Figure 1i** and **1j**.

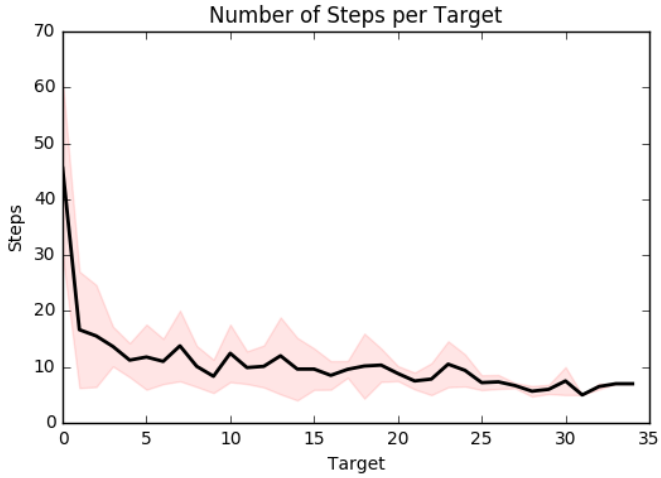

Fig. 2. The number of steps per target for the open 6x6 environment shown in **Figure 1a**. The number of steps taken converges to a minimum of 11 steps after just 4 targets, demonstrating the efficacy of Q-learning for the performance of a task.

D. Tasks

Although the examples in the previous section explored the RL-BCI in a navigation task, the use of error potentials can also be used to extend a BCI to a variety of different tasks with somewhat different actions. As discussed earlier, error potentials solely perform binary evaluation on each state-action pair. For that reason, there is no theoretical limit on the number of actions that can be made available to the agent in a system. The previous section used the example of 8 available actions at each state, corresponding to 8 different directional movements. However, this number can be extended to any number of actions, since the agent will eventually converge to an optimal policy after interacting with the environment. In practice, there may be a cognitive limitation on the number of actions that user can consider and evaluate during one session. Additionally, in order to converge to an optimal policy from a large set of actions, the agent may have to iterate through many epochs before establishing the correct action at each state. Long training times may not be feasible in an RL-BCI setting, as user fatigue may limit the number of effective iterations.

Figure 3 demonstrates the flexibility of tasks in this RL-BCI model, by using a theoretical speller task. In this task, the goal

of the user is to write a particular sequence (in this case, the sequence *abcabcabc*). The autonomous agent always moves in a fixed pattern, from left to right over the spaces. At each space, the agent chooses to write a symbol from a collection of 3 symbols (the letters *a*, *b*, and *c*), according to its current policy. After each action, the environment returns feedback regarding the symbol chosen at each state, and this feedback is used as a teaching signal to inform the agent which symbols belong in which states. If negative feedback is continuously evoked when the agent selects the wrong letter at each state, the agent will eventually converge to write the correct sequence.

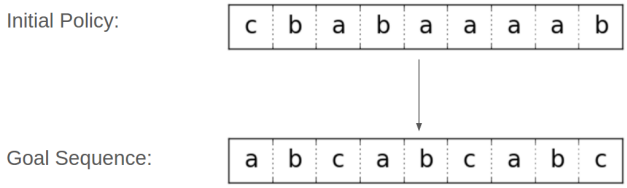

Fig. 3. This figure shows the policies involved in an RL-BCI speller. The top figure is the initial policy of the speller, while the bottom figure shows the final policy after convergence. This hypothetical example is used to demonstrate the efficacy of the RL-BCI system beyond navigation tasks

E. Collaborative Filtering

As noted, one of the issues with using the RL-BCI model is the low information transfer rate (ITR) between the user and the agent. Training an agent through Q-learning strategies involves visiting every state, and evaluating one or more actions from that state. In cases with high spatial or action dimensionality, the agent will have to iterate through a substantial amount of runs before it converges to an optimal policy. Although error potentials are less subject to cognitive fatigue than other neural signals used for control, such as motor imagery, long training time before effective performance hinders the ease of use of a BCI.

One possible solution to this slow training procedure is to employ collaborative filtering on policies. Collaborative filtering is a statistical technique that predicts the future actions of an agent based on similarity measures with other actions of other agents. In this case, the RL-BCI will search through policies learned in previous sessions of the same environment, and estimate whether or not the current user is following a similar policy. If the current policy is sufficiently similar to a previous policy, the agent can use the previous policy's state-action pairs in order to make informed guesses on what optimal policy might be for this session. This collaboration can happen between different sessions of the same user or different users of the same environment.

There are various strategies that can be employed to implement collaborative filtering. The simplest approach might be to use a modification of the *Jaccard Similarity* metric [11], taking the ratio of the shared states between the two agents and which of those states have the same action attributed to it. The equation is as follows:

$$J(A,B) = \frac{|\{s \in S_A \cap S_B | \operatorname{argmax}_i s_A = \operatorname{argmax}_i s_B\}|}{|S_A \cap S_B|} \tag{2}$$

where $J(A,B)$ is the similarity measure between two different policies A and B, $|S_A \cap S_B|$ is the size of the set of visited states both agents, and the numerator is the size of the set of states from the set $\{S_A \cap S_B\}$ that have the same current policy, the action of maximum value. In other words, this measure quantifies how many states so far have the same state-action pair evaluation as a previous policy. This previous policy can then be used by the current agent in order to inform all future actions. If the actions determined by this previous policy begin to evoke negative feedback, then the similarity measure will begin to fall. This may cause the agent to adopt a different past policy, or, if no previous policy is similar enough, to continue establishing its own policy.

Collaborative filtering can greatly reduce the training time required to teach a reinforcement learning agent. In the most favorable theoretical case, where a past policy is correctly adopted at the start of training, the optimal policy for that agent will have converged in just one step. In the worst case, collaborative filtering will lead to the same training time as if no collaboration was employed. In practice, careful tuning of similarity thresholds and probabilities will be needed in order to discover the best balance of exploration and exploitation of policy learning.

V. Discussion

The proposed RL-BCI model in this report shows great promise in establishing a BCI that is robust to a variety of state and action spaces. By using ErrPs as a universal classification scheme for any task or environment, RL-BCIs can extend to any sort of state-action sequence that can be observed and evaluated by a user. Furthermore, ErrPs provide an easier paradigm than certain active BCI techniques, and thus, ErrP-based systems can increase the accessibility of BCIs.

The practical implementation of this model relies on a thorough understanding of the nature of ErrPs. As we begin extending the model's action- and environment-spaces, we must be aware of the impact that these new contexts may have on the characteristics of the neural signals. For example, large environment spaces may reduce the intensity of error responses due to a user's inability to evaluate actions that lay far from the goal. Varying visual stimuli, such as different environmental geometries, may have a disruptive effect on our classifier. A deeper understanding of the variability of ErrPs across environments, tasks, and individuals is a crucial step towards implementing this model.

There are also a few open problems in developing the learning paradigm of RL-BCIs. Findings from ErrP research should inform the design of an RL agent. The hyperparameters in the Q-learning update rule should be chosen with consideration to aspects of ErrP decoding accuracy. For example, mapping the Q-Learning learning rate (α) to the confidence level of the

classifier could adapt the sensitivity of the update rule to the strength of the classifications, thus mitigating the effects of bad predictions. Furthermore, the intensity of the error signal can be mapped to the feedback of the update rule, making greater errors correspond to more negative punishment values. Adaptive learning rates may be used to reduce the learning rate as the agent begins to converge to it's optimal policy. A similar approach would be to use a probabilistic instead of a deterministic policy, by using a softmax to determine which action to take at each state:

$$P(s,a) = \frac{e^{Q_s,a}}{\sum_{a'} e^{Q_{s,a'}}} \tag{3}$$

A temperature value τ may also be used in this softmax formula to adaptively alter the strength of these probabilities.

As mentioned, low ITR diminishes the training speed of the agent, and thus creates a substantial delay before the agent can accurately perform a task. The inefficiency of training creates a large obstacle towards the practical use of this system. A crucial line of research would be to further develop intelligent strategies to be used by the agent, in the aim of allowing the computer to fully utilize its sparse inputs. Techniques such as collaborative filtering allow the agent to shorten this training time by evaluating similar policies instead of manually training an original policy. Learning algorithms beyond Q-Learning should also be explored, in order to find the best framework in which these error potentials can maximally reveal information about user intent of the task.

VI. Conclusion

Overall, this proposed model emphasizes the potential of RL-BCI systems. Through shared-control BCIs, users can take advantage of artificial intelligence techniques, by allowing external agents to make informed decisions based on minimal user input. By employing a Q-learning scheme, RL-BCIs can translate to a variety of different models and performance tasks. It is with optimism that I demonstrate the extensibility of this model; however, the practical implementation of this approach is dependent on further exploration of both neural signal processing and computational intelligence. By combining advances in neuroscience and in artificial intelligence, BCIs show great promise for creating a stronger paradigm of human-computer interaction.

References

[1] Falkenstein, M., et al. "Effects of Crossmodal Divided Attention on Late ERP Components. II. Error Processing in Choice Reaction Tasks" in *Electroencephalography and Clinical Neurophysiology*, issue 78: 447-455, 1991.

[2] Gehring, W. J., Coles, M. G. H., Meyer, D. E., Donchin, E. "The error-related negativity: An event-related brain potential accompanying errors" in *Psychophysiology*, 1990.

[3] Spler, M. and Niethammer, C. "Error-related potentials during continuous feedback: using EEG to detect errors of different type and severity" in *Frontiers in Human Neuroscience.* 2015;9:155. doi:10.3389/fnhum.2015.00155.

[4] Falkenstein, M., Hoormann, J., Christ, S., and Hohnsbein, J. "ERP Components on Reaction Errors and Their Functional Significance: A Tutorial" in *Biological Psychology*, 2000.

[5] Ferrez, Pierre W. and Millan, Jose del R. "Simultaneous Real-Time Detection of Motor Imagery and Error-Related Potentials for Improved BCI Acurracy" in *Proceedings of the 4th International Brain-Computer Interface Workshop and Training Course*, 2008.

[6] Bell, CJ, Shenoy, P, Chalodhorn, R Rao, RPN 2008, 'Control of a humanoid robot by a noninvasive brain-computer interface in humans' Journal of Neural Engineering, vol 5, no. 2, pp. 214-220. DOI: 10.1088/1741-2560/5/2/012

[7] Muller-Putz, Gernot R., and Gert Pfurtscheller. "Control of an electrical prosthesis with an SSVEP-based BCI." IEEE Transactions on Biomedical Engineering 55.1 (2008): 361-364.

[8] Wolpaw JR, McFarland DJ. Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans. Proceedings of the

[9] Zander, T. O., Kothe, C. "Towards passive braincomputer interfaces: applying braincomputer interface technology to humanmachine systems in general" in *Journal of Neural Engineering*, 1991.

[10] Sutton, Richard S. and Barto, Andrew G. "Reinforcement Learning: An Introduction." *MIT Press*, Cambridge, Massachussetts, 2017.

[11] Tan, P., Steinbach, M., and Kumar, V. "Introduction to Data Mining". *Pearson*, 2006.

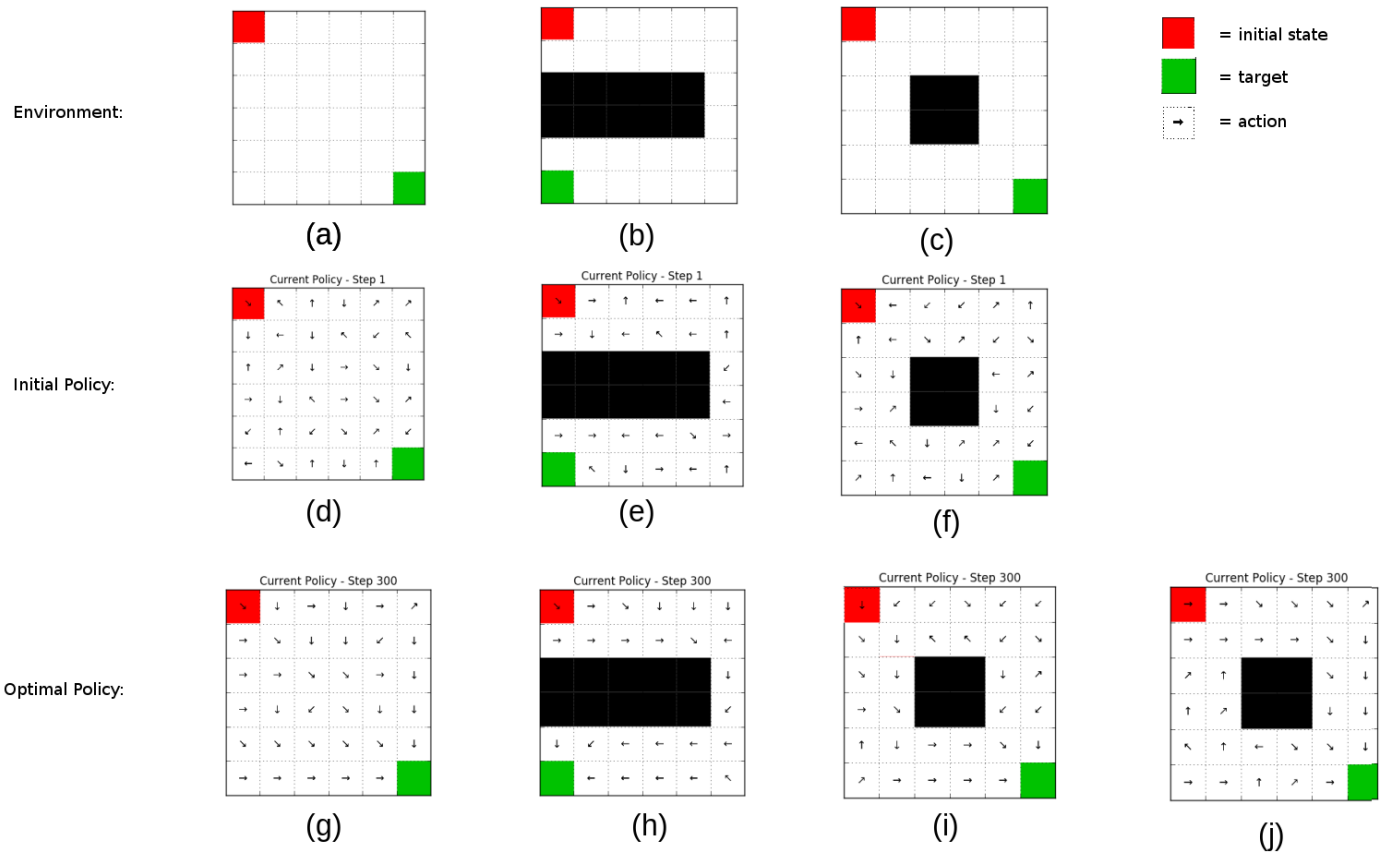

Fig. 1. This figure shows various proposed environments that the RL-BCI paradigm would work in. The first row, **Figures 1a-c** show the test environments. The second row, **Figures 1d-f** show the initial policy, determined by randomly initializing the Q-values of each state-action pair. The third row **Figures 1g-j** show the final policy of the agent.