INVERSE REINFORCEMENT LEARNING FOR AGENT CONTROL USING ERROR RELATED POTENTIALS

27 Juin 2017

Tutors: Iñaki Iturrate, Ricardo Chavarriaga
Student: Alvaro Serra,

ÉCOLE POLYTECHNIQUE
UNIVERSITÉ PARIS-SACLAY

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

TABLE DES MATIÈRES

1 INTRODUCTION AND OBJECTIVES

New materials and recent advances in computation power have opened the door to new research fields in computer science and its application in the enhancement of other disciplines. One of these fields is the field of Brain-Computer Interface. A technology based in the acquisition of signals emitted by the brain which are later on analyzed and decoded by a machine or computer. This field not only opens the door to another way for us humans to interact with the world enabling alternatives for teleoperation and manipulation research, but may also one day allow disabled people to regain their physical capabilities by controlling an exoskeleton with these very brain waves. Following this train of though, the problem we are facing in this project is the control of upper-limb movements using brain signals obtained by non-invasive methods, in particular Electroencephalography (EEG) signals.

Processing and analyzing EEG signals represent a great challenge. EEG signals are mainly non-stationary, non-linearly distributed, are temporally highly correlated and can present a lot of artifacts. On top of that, subjects are difficult to obtain, the signal obtained is different for each one of them, the experiments give a limited amount of data and these are rarely standardized. In summary, we have a highly complex signal to process and a narrow database which makes it difficult to obtain generalized accurate supervised learning models.

A lot of work has been conducted during the past decades in order to surpass the difficulties EEG presents. Most of this work focuses in obtaining better and more stable features either by applying feature engineering (temporal and spatial filtering, temporal and spectral analysis,...) or feature selection [4] [6]. These methods arise from a deep knowledge on the field of neuroscience and allow the study of relatively stable signal patterns which enable the use of machine learning for event-related signal classification, intention decoding, ...opening the door to several applications in the field of Brain-Machine Interface. One of these stable signal patters are the Error Potentials (ErrPs) and they mainly manifest whenever the subject regards something in their environment as incorrect. They will be crucial for the rest of the project and will be explained in more detail later.

Our approach to this problem has in mind recent advances in Machine Learning (ML), particularly Reinforcement Learning (RL) and Inverse Reinforcement Learning (IRL). Briefly speaking, Reinforcement Learning is the area concerned with training and designing intelligent agents, capable of learning and improving its decision-making from interactions with an environment. Inverse Reinforcement Learning, on the other hand, is another algorithm built upon RL which allows the automatic construction of the reward function (often manually tweaked) in order to have the desired agent behaviour just by observing few demonstrations previously provided by a human (or expert). Techniques from these areas have been successfully applied in many different situations including problems which were previously considered to be intractable to computers, such as playing Go and others such as car driving simulation, parking lot navigation, [1] quadruped locomotion and robotic arm control achieving very interesting results in all of them [**?**].

In face of these results in RL and EEG analysis, our expectations are to make application of Inverse Reinforcement Learning techniques in the upper-limb movement control problem using Error

Potential Signals. In particular we have studied classic and state of the art Reinforcement and Inverse Reinforcement Algorithms in trajectory generation and the stability of ErrPs and their application in EEG based spellers (ErrPs and RL) and robotic arm inverse optimal control problems which stimulated us to combine the best of both worlds.

Our main objective in our research project is to be able to control an agent using ErrP signals and make it follow predefined trajectories as a simplification of the brain controlled upper-limb movement problem. In order to achieve this main goal we have set, at least for the first part of this project, the following objectives :

— Develop a linear IRL algorithm and consider the relevance of ErrP against other non-EEG derived features
— Study the results of the linear IRL algorithm compared to the non-existence of such a feature in the same algorithm

— Validate our algorithm using real ErrP signals.

At the moment we are working on the 2nd objective. However, we do not seek to stay at that point but to further progress on this path. All details concerning future objectives will be specified later.

We start our exposition on the topic by first exploring EEG signals, their extraction and analysis. After, we expose the basics behind Reinforcement Learning and Inverse Reinforcement Learning, the motivation behind using those methods, and finally how this knowledge can be applied to develop brain-controlled agents (analog to upper-limb movement control). Finally we will show the work and results obtained during these first steps of our research project as well as discuss the steps to take from now on.

2
EEG

There are two methods of acquisition of brain waves which depend on whether the device of acquisition is invasive or non-invasive.

Invasive devices take measures directly from the subject's brain. Though this kind of acquisition results in obtaining a good signal-to-noise ratio in the extracted signals, it requires a surgical intervention. This method is often applied on animals because of lack of human volunteers and ethical reasons. Despite this, great results have been obtained concerning vision and movement regaining.

On the other hand, non-invasive devices do not require any kind of surgery to be applied. Compared to invasive devices this kind has notably a worse signal-to-noise ratio. However, due to the fact that it does not require permanent changes to be made, there is much more subjects willing to volunteer for experiments which accelerates the obtention of data. Among popular non-invasive BCIs we find : Pupil-size oscillation, Electroencephalography (EEG)- based brain-computer interfaces, Electromyography (EMG), Magnetoencephalography (MEG) and functional magnetic resonance imaging (fMRI)... The CNBI (Chair in Brain-Machine Interface) laboratory specializes in EEG signals decoding and treatment. This is why we will focus on EEG signals from now on. Ahead we briefly explain

Figure 1 – Example of Invasive BCI applied on a subject

EEG signals main characteristics and how Error Potentials can be identified.

2.1 Signal Characteristics

Electroencephalography (EEG) is an electrophysiological monitoring method to record electrical activity of the brain. As we can observe in figure 2, the electrodes are placed along the scalp. The locations of the electrodes are normalized and different configurations exist depending on the number of electrodes used.

In that case, the received signal from the scalp has 64 channels or raw features. Each one of these signals capture the local electrical activity around the electrode, which gives us spatial resolution over the scalp activity. However, due to the fact that brain signals propagate along the scalp, EEG signals are not only temporally but also spatially correlated. EEG signals are also very sensitive to artifacts. These range from artifacts external to our body such as cable movement, bad contact between electrode and scalp,...to artifacts which come from within the body such as muscular activation, heart beat, blinking,... These ErrPs will affect more or less the electrode depending on the proximity to its source. Among classical methods to deal with this noise on our signal and improve our chances of extracting information we count : laplacian spacial filtering (local reference method), common average referencing (global reference method), frequency filtering between 0.1-1Hz and 0.1-25Hz where most of Event-Related Potential signals can be extracted, applying Independent Component Analysis in order isolate and extract artifacts,... In addition, channel selection might be applied, not only to remove

artifacts but also to select the channels that are more likely to provide us with the information we are searching for. As an example, it is common to reject frontal electrodes as they often contain artifacts produced from blinking. In the same way, when trying to find Motor Related Cortical Potentials it is common to select central scalp channels.

Even if there is still a lot to do, there is a lot of literature regarding EEG-based systems and research searching for possible applications deriving from information we can obtain from these signals. As a matter of fact, successful research has been conducted regarding movement intention detection [4], correction intention [6], controlling a P300 BCI speller [3] and performing 2D reaching tasks [5]. Among the different possible signals we can extract from EEG, the ones we are interested in are Error Potentials (ErrPs) related to error monitoring.

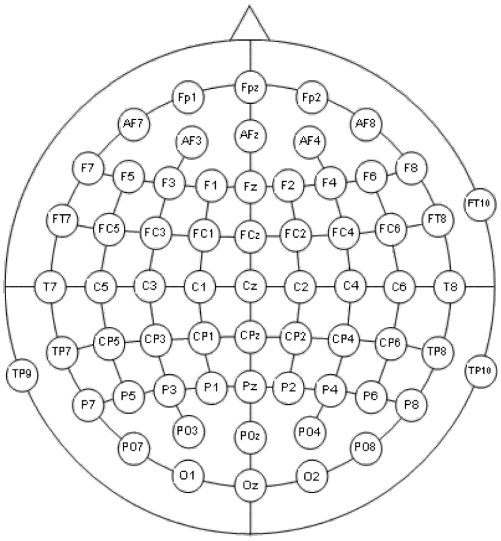

Figure 2 – Placement of the electrodes in the EEG cap

2.2 Error Potential Signals (ErrPs)

When a person makes of perceives an error, an error-related potential can be detected in the EEG due to the person recognizing that error. This detected signals were first studied in choice reaction tasks and consist in the combination of a negative error related potential (appearing 50-100 ms after erroneous response) and a positive error related potential appearing directly after the negative one (200-400ms after the error) [8]. In figure 3 we show an example of what an error potential signal looks like.

As explained in [6], several works have uncovered EEG correlates of errors of different nature. Among these errors we count : errors committed by oneself, error related to the perceived feedback as well as errors in the interaction with external devices, which, for instance, are particularly interesting in the field of Brain Machine Interfaces. However, this is not the only reason for which we have gained an interest in these signals. According to [6], it is possible to obtain features from ErrP that remain stable even across different conditions, different recording days and different feedback characteristics. In addition, single-trial performance has been proved to be feasible in different situations such as

Motor-imagery BCI, human-robot interaction and car driving. In other words, it should be feasible to reliable features based on Error Potential detection in order to train Semi-Supervised Machine Learning models for real time applications.

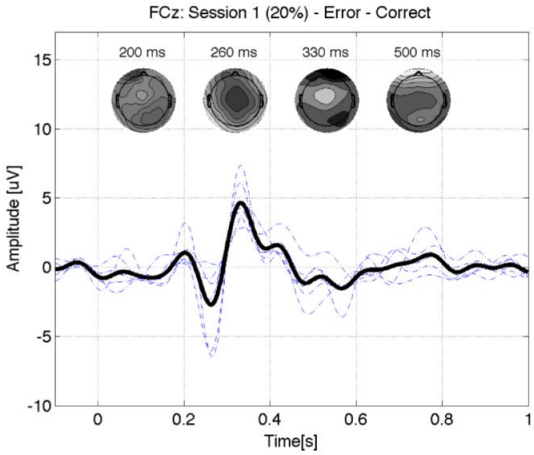

Figure 3 – Example of Error Potential Signal extracted from the results obtained at [6]

3 REINFORCEMENT AND INVERSE REINFORCEMENT LEARNING

For the purpose of training an intelligent agent capable of learning trajectories and movement style from previously generated demonstrations, we turned ourselves to the Machine Learning subarea known as Reinforcement Learning. In this section we show which are the motivations behind this area and which solutions it offers for training intelligent adaptable agents.

The Machine Learning domain is usually divided in three categories : Supervised Learning, Unsupervised Learning and Semi-Supervised Learning. This categorization depends on the approach we take on how to learn to perform a determined task. Supervised Learning manages this activity by learning from known solutions to the prediction we want to make. For example, given a pair (x_i, y_i) and a task in which we want to predict y_i given x_i, we build a model using already known pairs of data (x_i,y_i) and extrapolate the predictions to generate y_i values for other unseen x_i. However, this approach is not capable of learning everything a human is able to. In fact, it can only feed on data we already know the solution to. We can relate this with how we humans usually learn through a professor or a book : we receive correct solutions to certain problems and try to apply them to problems that we deem similar. Nonetheless, for the kind of problem approached in this problem we don't have a database large enough so as to be able to build a good model. On top of that, there are not good examples that clearly specify which is the good decision to be made at every situation, trying to predict how it will affect future situations at the same time. Considering this, we might feel a *trial*

and error based learning might fit better this kind of situations.

In other words, what we are really looking for is a way to learn through experience. Let's consider a chess game. It is through iterations that we can start to infer which is the good decision to take at every moment by predicting the potential *reward* every state and movement can have in the long run, what we call learning strategy. This can apply to learning trajectories in a system or agent with many degrees of freedom as a robotic arm[9].

It is this kind of problems that Reinforcement Learning tries to tackle. It has had astonishing achievements in the past few years : "computers" have been successfully trained to play Atari at human level [7], play Go at professional level and even achieved to train robots to play soccer. On top of that, when it comes to very complex high degree of freedom problems where a reward function or a model is difficult to hand build or specify Inverse Reinforcement Learning has managed to increase the state of the art on that kind of problems. We are talking about cars which are capable of reproducing optimal trajectories the same driving style of the expert giving the demonstrations or quadruped robots capable of walking through a previously unseen irregular terrains [1][2].

3.1 Reinforcement Learning

In order to describe the Inverse Reinforcement Learning Algorithm we must first explain the basics on Reinforcement Learning Algorithms, which starts by explaining Markov Decision Processes.

3.1.1 • Markov Decision Processes (MDP)

A Markov Decision Process is a general mathematical model for the interaction of an agent with a random environment. In this model, the agent interacts with the underlying environment taking decisions on which action to perform depending on the predicted total reward it can get by taking it.

As described in [1], a finite state Markov decision process can be considered as a tuple (S, A, T, γ, D, R), where S denotes a set of states ; A is a set of actions ; T $= P_{sa}$ is a set of state transition probabilities (which in this case represent the transition distribution when in the state s we take the action a) ; γ is a discount factor in [0,1) roughly representing how much ahead the agent can predict the value of the state he is into ; D is the initial-state distribution, from which the start state s_0 is drawn ; and $R : S \to \mathbb{R}$ is the reward function, which we assume to be bounded in absolute value by 1.

Given an MDP, the Reinforcement Learning problem tries to learn a policy or course of action for the agent to take in order to maximize its final reward. Sometimes some of the information presented above is not present or might have special characteristics, which may determine the way we approach the situation and the RL algorithm we might want to apply. For example, when T is missing this means we no longer have knowledge over our system and, thus, we can apply model-free algorithms. This is useful for the cases when we want our solution to give general results, not relying on a model of our environment [7]. The application of IRL over an MDP is also a good example, as R is missing from the considered MDP.

3.2 Linear Inverse Reinforcement Learning

When it comes to linear IRL, we must consider first some hypothesis. We assume that there is some vector of features $\phi : S \rightarrow [0,1]^k$ over states, and that there is some "true" reward function $R^*(s) = w*^T \phi(s)$ (or $R^*(s,a) = w^*T\phi(s,a)$), where $w* \in \mathbb{R}$. In order to ensure that the rewards are bounded by 1, we also assume $\|w*\|_1 \leq 1$. The state features represented by the ϕ application do not necessarily need to be actual state vector, it can be a vector of features indicating the different main factors in our problem that we would like to trade off. For example, in a driving problem these factors could be such as whether we have just collided with another car or a wall, we're driving in the middle or side lanes, going forward, backwards,... Therefore, the unknown vector of weights w* specifies the relative weighting between the accounted factors.

A stationary policy π being a mapping from states to probability distributions over actions, the value of a policy π can be expressed as :

$$E_{s\ D}[V^\pi(s_0)] = E[\sum_{t=0}^{infty} \gamma^t R(s_t)|\pi] = E[\sum_{t=0}^{infty} \gamma^t w^t \phi(s_t)|\pi] = w^t E[\sum_{t=0}^{infty} \gamma^t \phi(s_t)|\pi]$$

where the expectation is taken with respect to the random sequence of states drawn from an initial state s_0 depending on D and picking actions according to π. We define the expected discounted accumulated feature value vector $\mu(\pi)$ or feature expectations to be :

$$\mu(\pi) = E[\gamma^t \phi(s_t)|\pi] \in \mathbb{R}^k$$

Using this notation, the value of a policy may be written as $E_{s\ D}[V^\pi(s_0)] = w^T \mu(\pi)$ Considering that the reward R is expressible as a linear combination of the features ϕ, the feature expectations for a given policy π completely determine the expected sum of discounted rewards when acting according to that policy. Knowing these basics, we can briefly describe the goal and algorithm of the Inverse Reinforcement Learning algorithm to provide a general sense of the algorithm we will make use of later. For more details on the algorithm and theoretical results the reader may refer to [1]

Being previously given a set of demonstrations, the main goal of Inverse Reinforcement Learning algorithm is to find a set of weights which result in a reward function and thus a policy that results in a trajectory the feature expectations of which are as close as possible to the mean of the demonstrations' feature expectations. In order to implement Inverse Reinforcement Learning there exist two possible algorithms : max-margin and projection. Unlike the max-margin algorithm which has need of a quadratic problem solver, the projection algorithm does not need one. Instead, the latter computes the orthogonal projection of the expert's feature expectations onto the line through a reference feature expectations vector and the obtained from the most recent policy. Still achieves the same results. This is why, for the sake of computing simplicity, when applying IRL we will use the latter.

4 METHODS

4.1 Testing Environment

4.1.1 • Gridworlds

Due to being our first contact with RL algorithms and the uncertainty when it comes to the effects of the corrective feature (the ErrP feature) when applied to them, we decided to keep the first testing environment as simple and as possible. This means that we will consider a discrete state space as well as a discrete action space. In order to do that we took the decision to start our experiments on gridworlds which we programmed from scratch in order to have more control over the events being registered when analyzing. These gridworlds are 10x10 grids with axes defined between $x \in [-4, 5], x \in \mathbb{R}$ and $y \in [0, 9], y \in \mathbb{R}$ where the agent can perform 5 actions : move left, right, up, down and guess whether it has arrived at the final point (from now on we will call this the *final* action). For the sake of simplicity we have defined the start and final points (x,y) equal to (0,0) and (0,9) respectively. In RL our agent has the only objective of attaining the final point, while in IRL not only it has the objective of attaining the final point but also to follow a certain pre-demonstrated trajectory. It must be noted that, due to the low number of states and discrete action-state space of our environment we can afford to use RL algorithms based on table lookup, which means that we store the value for each state-action pair in a matrix $Q(s, a)$. Other than being simple, this environment is still useful in state of the art applications when it comes to EEG-based 2D reaching tasks [5] and EEG-based spellers [3] which decode the letter that the human intends to input using a combination of RL and Bayesian filtering.

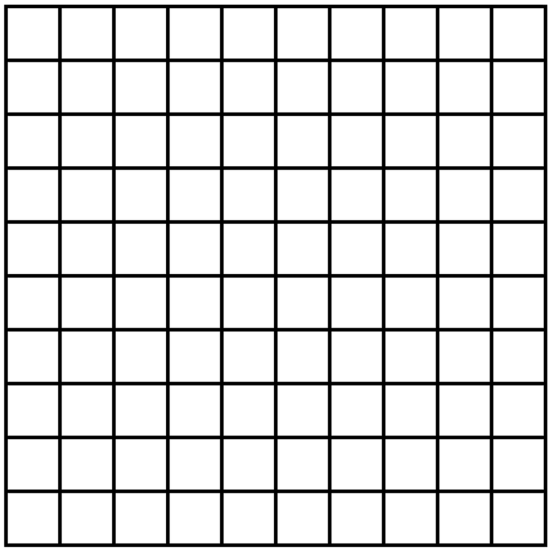

Figure 4 – Example of our 10x10 gridworlds

4.1.2 • State-Action Representation

How we decide to represent the state-action in the feature space is very important in our problem. Depending on it, our algorithm may converge faster, slower or not converge at all. This is why we must put extreme care into how we represent it. Let's us remember the main characteristics we need to take into account and how we could represent them in the case of a 10x10 gridworld.

— Physical state : it defines the position of the agent in the gridworld. In the feature space we represent this characteristic as a 20-dimensional vector where the 10 first positions represent the x-axis and the following 10 the y-axis (all 0 except for the x and y coordinates of the position of the agent).
— Action : a correction signal loses a lot of its worth if we cannot relate it to a certain action taken in a certain state. This is why the RL learning actions we will use must take into account the state-action space instead of just the state space. This means we need a feature representation of this information. We represent it as a 5-dimensional vector where each position represents the action taken ordered in the following order : left, right, up, down, final.
— ErrP : this is the feature emulating the detection of Error Potentials in the final user. It is important so as to mark whether an action in a certain state has been deemed correct or incorrect. We represent this characteristic by a 1-dimensional vector containing 1 if an ErrP has been registered (signal of incorrectness) and 0 otherwise. This is the only characteristic we will not include in the state-action used for table lookup.

As we can observe, the final feature vector consists in a binary 26-dimensional vector representing the position in the state-action space incorrect or not. Even if this representation in the feature space is not needed for the Q-Learning algorithm we will use (as we are applying the table lookup method). It is important to define it for the IRL problem. Remember that the IRL algorithm is built upon the hypothesis that optimal rewards can be considered as a dot product between a set of weights and a feature vector.

Furthermore, representation in feature space might make scaling from gridworld to continuous state space much easier either for RL and IRL. Due to an infinite number of states, we shall not be able to perform table lookup anymore in which case we will need predictive models of the value of every state-action. These predictive models feed on the feature space representation of state-actions.

4.1.3 • ErrP Model

Without a previous study confirming our hypothesis regarding the usefulness of the *ErrP feature*, we cannot start recording EEG from subjects. However, we still need a way to ascertain the utility of this feature. This is why we have developed an ErrP model which tries to emulate this kind of human behaviour deciding which situations might be registered as incorrect in real life. There are two ErrPs models we will apply depending on the situation :

— When no trajectory is given, like when we compute a simple RL on our gridworld and the optimal trajectory is a straight line, this error signal is given whenever we don't reduce the distance between our agent and the final state.
— In the IRL case where trajectories are given, the ErrP signal is given whenever the state-action the agent is not present in any of the given trajectories. If it is present, we could still receive

an error potential signal if the agent is following one of the trajectories backwards.

Of course, these are not perfect models Still, this should not be a problem. RL, and thus IRL, algorithms usually take a lot of iterations through the state-action space in order to converge. Every iteration, specially the first ones, has a huge number of steps. For a subject to have to keep concentrated through all those steps could be too tiring. Therefore, even if they are imperfect, they might as well serve as a pretraining for the model to get as near as possible to the real solution, attainable only by real ErrPs.

Although in this report we just give results using a 100% accurate ErrP model, it is our intention to give results and comparisons for IRL using 90%, 80% and 70%.

5
RESULTS AND DISCUSSION

Before proceeding with the first steps taken in our project, it is important for us to remember the main statements that have driven us to choose our first environment and RL model. First, we are trying to solve the gridworld problem as a first and simple approach to the robot arm trajectory generation using the *ErrP feature*. In order to do this, we have set our first goal to study the usefulness of the *ErrP feature* in a RL context. Also, as we have said before, considering the characteristics of the main problem and for the sake of finding more general results we have decided our RL model to be model-free. In addition, inspiring ourselves by solutions obtained in a similar environment [speller] we have decided to use, among the family of model-free RL algorithms, the same value-based algorithm : Q-Learning. We seek, in this way, to find a common ground so that our results can be of profit to related research fields.

5.1 Implementing Inverse Reinforcement Learning

Having obtained the previous results, we go on to apply the IRL algorithm using the same feature space representation. In order to evaluate the usefulness of the *ErrP feature* in this context, will compare two models, one including ErrP feature in the feature representation space and the other not including it, in the following increasingly difficult demonstrated trajectories :

In order to assess the importance of the feature we will compare the 5 most influencial features in the reward function in both situations by analyzing the mean of the weights obtained through 40 computations of the IRL algorithm. In addition, we will compare other relevant data such as mean and std of number of IRL iterations and total number of steps taken through the computation of a IRL cycle.

5.1.1 • Trajectory 1

For the first trajectory, both algorithms converge obtaining the following results :

No ErrP	action_up	action_down	$x = 0$	action_right	action_left
	0.240	-0.151	0.088	-0.043	-0.042

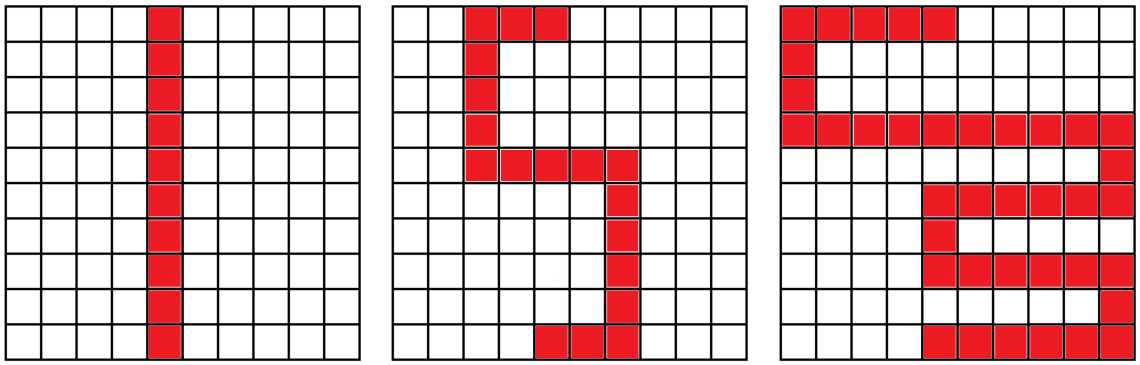

Figure 5 – From left to right : trajectory 1, trajectory 2, trajectory 3 demonstrated by the expert

ErrP - 100% accuracy	*ErrP feature*	action_up	action_down	$x = -2$	$y = 9$
	-0.212	0.070	-0.067	0.067	0.044

Looking at the weight values for both models, we note that the values generally make intuitive sense. The trajectory to follow being a straight line, it is not surprising that both the action of moving up and staying at coordinate $x = 0$ register in both cases the highest weights. It is also logical for $y = 0$ to have a high negative weight as we seek to get out from $y = 0$ as much as possible. What is interesting is the appearance of the *ErrP feature* among the top 3 most relevant features with a high negative value. This makes sense with the other comparative results obtained for this first trajectory. While there is no significant changes regarding the mean number of IRL iterations necessary for convergence, we see a significant decrease in standard deviation and an even larger decrease in the mean and standard deviation values of the total timesteps per IRL computation.

	IRL iterations (1)		Total timesteps(2)	
	mean	std	mean	std
No ErrP	302.50	454.78	421704.90	533346.01
ErrP-100% accuracy	4.53	4.01	121446.83	173697.52

T-test No ErrP vs 100% accuracy : p-value = 2e-04 (1), 1e-03 (2)

Finally, in figure 6 we can see how the model integrating the ErrP feature converges much faster than the classic IRL with no EEG-based features included even in the simplest demonstrated trajectory.

In summary, we see a clear difference between the two models where the one including the *ErrP feature* performs much better (tables and 6). We can assume this is a consequence of adding this new feature as the high negative value of the weight attributed to this one shows it has a clear influence on the behaviour of the final policy.

5.1.2 • Trajectory 2

When trying to compute the same algorithms for the second trajectory demonstrations, we observe that the IRL model not including the *ErrP feature* does not converge in a certain trajectory. Unlike the case studied in the literature [1], the features we use describe our state more ambiguously. If it was

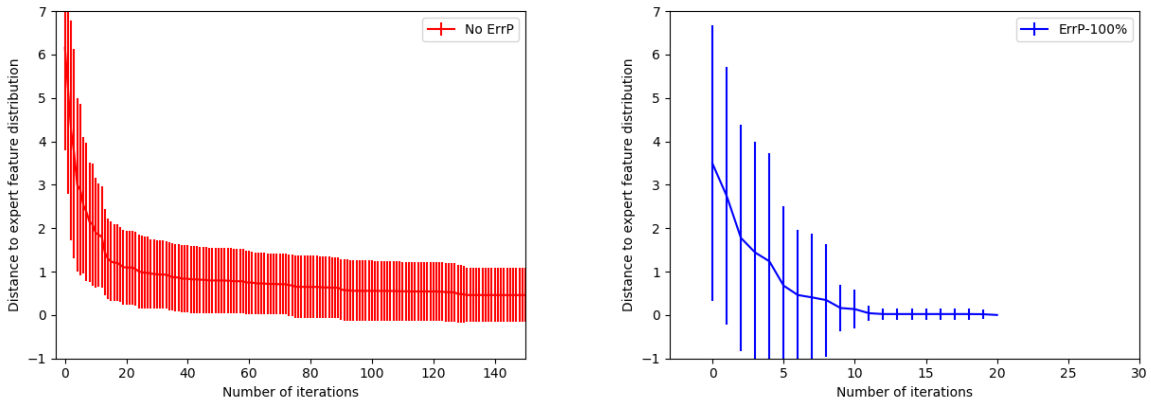

Figure 6 – A comparison of the convergence speed between the No-ErrP and ErrP-100% accuracy versions of the IRL algorithm. Euclidean distance to the expert's feature expectations. The plot shows averages over 40 runs with the standard deviations.

possible to cope with the previous trajectory it was because of its simplicity ; it was fairly easy as the only two things we needed to do was going up and stay in the middle of the horizontal axis. However, this trajectory requires the agent to explore both axis and 3 different actions. It is a good example of the limitations of linear IRL where the model quality depends on the quality of the features and how well these can distinguish the agent's state from the others.

When looking at the ErrP-IR modelL, we observe convergence and the following results :

ErrP - IRL	*ErrP feature*	action_up	action_down	$x = -2$	$y = 9$
	-0.256	0.070	-0.067	0.067	0.044

First, regarding the feature weights, we observe that this time the *ErrP feature* is much more important compared to others than before. As we have mentioned before, this trajectory is more complex and thus it is more difficult for the algorithm to infer which physical features and actions should be weighted more. As a result, it relies much more on the only feature not based on the physical plane as it carries much clearer information. Still, it is possible to observe a certain logic in the rest of the shown features. These verticals and horizontals coincide precisely with the ones composing the demonstrated trajectory as well as the general motion direction of the agent. Therefore, even if weakly in comparison to the weight attributed correction feature, the algorithm still manages to capture a sense of the trajectory.

	IRL iterations		Total steps	
	mean	std	mean	std
ErrP - 100% accuracy	12.03	15.28	457715.93	258940.31

Regarding the rest of the results obtained, we see effectively how Figure 7 validates the results obtained in the table above. It is important to remark that, even if the IRL iterations is not high, the number of total time-steps necessary for convergence is excessive for a human to be able to perform without getting tired or losing concentration, even more considering there exist more complex

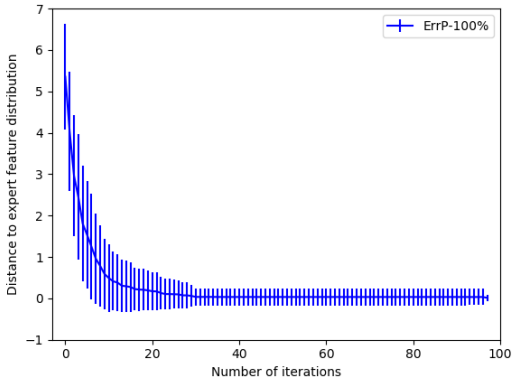

Figure 7 – Illustration of the convergence speed of the ErrP-100% accuracy IRL algorithm into the second trajectory. Euclidean distance to the expert's feature expectations. The plot shows averages over 40 runs with the standard deviations.

trajectories than this one. This implies that, in order to apply this algorithm to real time situations, a pretraining should be performed beforehand using ErrP models as we have done during this project.

5.1.3 • Trajectory 3

In this case neither of the models did manage to converge in the demonstrated trajectory. The trajectory being long and complex, we attained the limit of what our features could give us. This is a general problem in linear IRL and linear Inverse Optimal Control (IOC) problems as the results depend highly on how well the features have been designed. In order to be able to tackle this kind of trajectories, possible solutions range from designing a better feature space representation to applying non-linear IOC models.

6
CONCLUSION AND FUTURE CHALLENGES

6.1 Conclusion

In conclusion, we have confirmed our hypothesis concerning the performance improving effects of adding a correction signal in the form of the *ErrP feature* when included in Q-Learning and IRL algorithms. On top of that, when applied in IRL algorithms combined with other physical features, this feature maintains a high degree of relevance standing among the top 3 most influential features. In fact, the more complex a trajectory is, the more the algorithm will rely on the corrective feature, giving it more influence on the reward function and allowing it to learn more complex demonstrated trajectories than it would have been able with just physical state features and actions. Despite this boost in performance attributed by the new feature, linear IRL algorithms are still dependent on good quality physical features when dealing with complex trajectories. These conclusions pushes us forward and poses new paths to further develop this project.

6.2 Future Challenges

In order to build upon the results and conclusions explained above there are two main paths for us to follow. The first one is that of *model refining*. The main points of further study in this domain are the following :

— Until now we have considered ideal conditions in our environment. For example, we considered our ErrP model as an oracle which would tell us with all certainty whether a situation should be considered as correct or incorrect. In real life ErrPs are not detectable with 100% accuracy. This is why, we shall start by checking the effects of adding noise to the ErrP signal on the results previously observed.

— Something similar happens with the provided demonstrations. During the previous steps we have provided perfect demonstrations. However in real life perfect demonstrations can't always be provided. This is why, in the same way as the ErrPs, we shall check the effects of providing suboptimal demonstrations on our previous results.

— We have observed the limitations of linear IRL algorithms and their dependence on good features. In order to be able to reproduce trajectories we will need to find a solution either by designing more reliable features or recurring to non-linear IOC solutions.

On the other hand, further work could be conducted on the possible applications of ErrP-using IRL algorithms. For example, we could change the table lookup method for a value approximation method. This should make it easier for us to make our algorithm scalable and consider environments with a huge number of state-action pairs. With this we would take our analog environment a step closer to the real problem we are trying to solve. In addition, projects such as the speller in [] could benefit from the results obtained above and, thus, possible applications could be considered in that project as well.

RÉFÉRENCES

[1] Pieter Abbeel. *Apprenticeship Learning and Reinforcement Learning with Application to Robotic Control*. PhD thesis, Stanford, CA, USA, 2008. AAI3332983.

[2] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 1–, New York, NY, USA, 2004. ACM.

[3] R. Chavarriaga. Robust, accurate spelling based on error-related potentials. 2016.

[4] S. Silvoni J.d.R. Millán E. Lew, R. Chavarriaga. Detection of self-paced reaching movement intention from eeg signals. *Frontiers in Neuroengineering*, 5 :13, 2012.

[5] Inaki Iturrate, Luis Montesano, and Javier Minguez. Shared-control brain-computer interface for a two dimensional reaching task using eeg error-related potentials. *Annual International Conference of the IEEE Engineering in Medicine and Biology Society.*, 2013 :5258–62, 2013.

[6] F. Iwane, R. Chavarriaga, I. Iturrate, and J. del R. Millán. Spatial filters yield stable features for error-related potentials across conditions. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 000661–000666, Oct 2016.

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540) :529–533, February 2015.

[8] Martin Spüler and Christian Niethammer. Error-related potentials during continuous feedback : using eeg to detect errors of different type and severity. *Frontiers in human neuroscience*, 9, 2015.

[9] H. Yin, A. Paiva, and A. Billard. Learning cost function and trajectory for robotic writing motion. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 608–615, Nov 2014.