The effects on adaptive behaviour of negatively valenced signals in reinforcement learning

Nicolás Navarro-Guerrero
Universität Hamburg
Department of Informatics
Vogt-Koelln-Str. 30
22527 Hamburg, Germany
nicolas.navarro.guerrero@gmail.com

Robert J. Lowe
University of Gothenburg
Department of Applied IT
Forskningsgången 6
41296 Gothenburg, Sweden
robert.lowe@ait.gu.se

Stefan Wermter
Universität Hamburg
Department of Informatics
Vogt-Koelln-Str. 30
22527 Hamburg, Germany
wermter@informatik.uni-hamburg.de

***Abstract*—Reinforcement learning algorithms and particularly those based on temporal-difference learning are widely adopted and have been successfully applied to a number of problems as well as used to model animal learning. However, they are based on neural pathways involved in reward-seeking behaviour since little is known about punishment-driven learning and less still about the combined effects of both types of reinforcement on learning. This may not only be a shortcoming for computational models of human and animal learning but we have recently shown that it may also carry detrimental effects for machine learning applications, with respect to task performance and convergence speed. Here, we further explore our original results and compare the effects of different functions, i.e. binary, linear, exponential with different variance, for punishment on learning. Our experiments confirm the original finding of punishment signals reducing learning speed. It appears this result generalizes across a number of different functions of punishment reinforcement.**

I. Introduction

We have recently presented evidence that suggests that conventional reinforcement learning algorithms should not underestimate the importance of modelling the interplay between reward- and punishment-driven learning [1]. Failing to adequately account for this interplay may not only constitute an inadequacy for computational models of human and animal learning but our results suggest that it may also carry detrimental effects to machine learning applications, with respect to task performance and convergence speed [1].

Based on the neuroscientific and computational identification of the problem for conventional reinforcement learning regarding the loss of predictive power of reinforcement signals that conflate punishment and reward information into a scalar value [2], we proposed the use of nociceptive units – interoceptive inputs to the network predictive of damage – as a way to increase the agent-environment interactive state space with the information typically used as punishment signals. Punishment being a damage indicator used as external reinforcer signals (rather than interoceptive inputs) [1]. We showed that the use of nociceptive units can benefit task performance (object reaching precision) during and after learning while not negatively affecting convergence speed. This can be understood when compared to either our baseline (devoid of any negatively valenced signal) or agents trained with reward and punishment. Furthermore, when nociceptive units were combined with punishment signals, the nociceptive units seemed to counteract the detrimental effects on convergence speed inherent in punishment signals.

Although in our original work [1] we perform hyperparameter optimization on all tested conditions, we did not test for the effect of different activation functions in either the punishment or the nociceptive units. Thus, in this paper we present such analysis for punishment and nociception on three metrics, i.e. the task performance (here positioning error), the potential for damage and the positioning speed.

II. Relevance of negatively valenced signals in adaptive behaviour

Just like rewarding stimuli, negatively valenced signals are essential feedback mechanisms for informing self-preservative behaviours and can heavily shape agent behaviour [3]. For instance, pain is often used as a trigger of protective reflexes to protect the body from injury. Although protective reflexes are essential to any autonomous agent, they are not enough to cope with a highly dynamic environment, thus *nocifensive responses*, such as avoidance, motivated and affective behaviour, memory formation and learning, are also needed. Adaptive nocifensive responses are critical to cope with the changes in the organism's body due to ageing, to lesions, to environmental constraints or to extraordinary events. For instance, changes in the gait may be required to prevent further injury or sometimes it may be required to experience pain to escape from life-threatening situations; how and when these changes in behaviour happen cannot be predetermined and need to be designed according to the problem at hand.

It is typically assumed that reward and punishment are opponent systems and that their interaction can be modelled by simply additively combining the external reward and punishment signals [4], [5], [2], [6]. However, there is substantial evidence in favour of separate systems for reward-related (positivity) and punishment-related (negativity) processing [7], [6]. Presently, their functioning as standalone systems is only partially understood, and even less is known about their interactions, which are considerably more complex [8], [3]. The interaction between reward and punishment processing gives rise to a mix of highly non-linear dynamics including competitive and cooperative about which we still know little [4], [8]. Conversely, computational modeling of the relationship between such reward and punishment (negatively valenced) systems is relatively lacking.

Interoceptive processing – including sensations of pain, temperature, itch, sensual touch, muscle tension, air hunger, intestinal tension, etc. – provides the means to convey signals of affective significance (value) to areas of the brain that simultaneously encode exteroceptive (external) and proprioceptive information [9], [10]. Such signalling has been suggested to be indispensable to action selection (e.g. [11], [9], [8], [12]) by effectively increasing the state space with which to modulate actions and the sensitivity to reinforcers. Without such interoceptive processing, action selection is not just maladaptive but in some cases seemingly not possible at all [13].

III. Methodology

We use the exact same methodology as in our original work [1]. Here, however, we further explore the effect of different activation functions for punishment (Section III-C) and nociception (Section III-D) to understand better the generalizability of our results. Thus, the description of the methodology overlaps to a great extent with that of [1].

We decided to use an inverse kinematic learning scenario, because it is a task actively studied (e.g. [14], [15], [16]), due to the high number of applications for industrial and domestic robots. Nevertheless, it is still challenging and many aspects remain to be studied such as self-calibration, adaptation, learning of speed and force control.

Evidence from both child development research [17] and adult novel sensorimotor task learning [18] suggests that learning to reach does not require visual feedback, but seems to be useful for fine corrections at the end of a reaching movement. Moreover, in early infancy, motor programs for reaching are not explicitly planned ahead of a movement (trajectory planning), which points at a trial-and-error learning paradigm. Reinforcement learning methods are particularly suitable for this type of learning.

Actor-Critic architectures are powerful TD-learning methods that model phasic changes in dopamine neuron activity [19]. The Critic guides the learning of action sequences generated by the Actor in order to maximise the accumulated reward. The dual memory structure, one for the Critic and one for the Actor, allows storing the learned policy explicitly, which significantly reduces the computation of action selection of large state and action spaces when compared to other TD-learning methods [20, p. 153]. Moreover, Actor-Critic methods are thought to be consistent with biological evidence [19]. This is due to the fact that the reward prediction signal of TD-learning resembles the dopamine neuron activity in the striatum. Also, the Actor typically connects a high-dimensional sensory input to a smaller action space, which resembles its neural equivalent, i.e. projections from the striatum to the basal ganglia output nuclei [19].

A. Experimental set up

We focus on the problem of autonomous learning or inverse kinematics of a single robot arm. The robot's objective is to move the geometrical centre of its end effector towards a target as precisely and as quickly as possible. Arm movements are controlled using motor commands relative to the current joint position, but no inverse or direct model of the arm dynamics is provided to the agent.

Because our main interest lies in the effects of punishment and nociception on the learning of motor skills, a number of simplifications are made. A simplified 2-degrees-of-freedom model of a NAO robot arm is used, i.e. restricted to only one shoulder and one elbow joint. The link lengths are 105mm for the upper arm, and 113.7mm in total for the lower arm and hand[1]. The shoulder joint is limited to the range $[-18, 76]$ degrees and the elbow joint is limited to the range $[-88.5, -2]$ degrees[2]. The robot is able to precisely perceive the target's position in an egocentric reference frame, i.e. exteroception. It can also precisely perceive the absolute angular position of its joints, i.e. proprioception. The robot can perceive when the joints are at or close to their upper or lower limits, i.e. interoception (nociception) [21]. Nociceptive input is maximal when a joint is at the mechanical limit and decreases based on one of four activation functions, see Section III-D. Nociception is perceived only when the current joint position is within the upper or the lower 10% of its mechanical range. Reaching is considered successful when the robot's hand is at most 10mm away from the target.

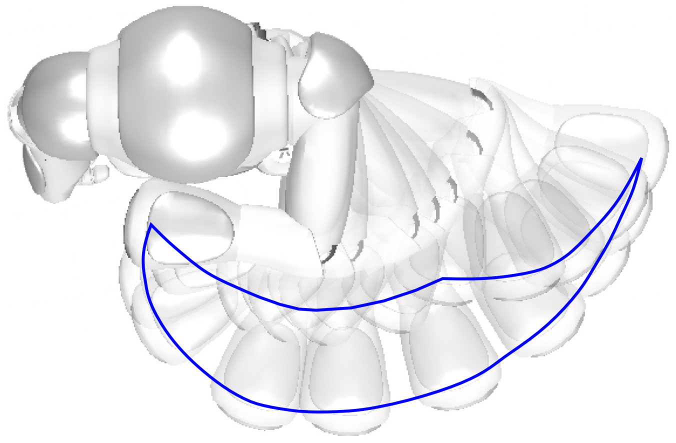
Fig. 1. Top view of the NAO robot facing right. The left arm is depicted in different positions and a blue line is superimposed to indicate the boundaries of the end-effector workspace.

To compare all different learning conditions a unique training, test and validation set for all conditions was used of sample size 1000, 100 and 1000 respectively. Each sample consists of a target in Cartesian coordinates and an initial joints configuration in degrees. Samples are randomly generated and the resulting end-effector positions are at least twice the reaching threshold

[1]http://doc.aldebaran.com/2-1/family/nao_h25/links_h25.html
[2]http://doc.aldebaran.com/2-1/family/nao_h25/joints_h25.html

of 10mm apart. Training, test and validation samples are always presented in the same order. A complete presentation of the training set is termed "epoch". Before any learning is performed, the agent is tested on the test set, and after each epoch afterwards. The test set is used to determine a winning set of hyperparameters for all conditions used, whereas the validation set is used for a detailed comparison between conditions.

Here, we compared three learning conditions. The first condition, *reward+punishment (R+P)*, used the target and the current joint position information as state space. It received a binary reward once the desired goal state was reached and a punishment term directly derived from the perceived nociception. We tested this condition with four different activation functions for punishment as described in Section III-C. The second condition, *reward+nociception (R+N)*, used only a binary reward once the desired goal state was reached, but extended the state space of the *R+P* condition by including one nociceptive unit per joint. We also tested this condition with four different activation functions as described in Section III-D. The last tested condition, *reward+punishment+nociception (R+P+N)*, used the same state space as the *R+N* condition and the reward and punishment of the *R+P* condition. We also tested this condition with four different activation functions for both nociception and punishment.

B. Continuous Actor-Critic Learning Automaton (CACLA)

CACLA [22] is a model-free reinforcement learning algorithm with Actor-Critic architecture. This algorithm was designed to work with large and continuous state and action space, thus an excellent alternative to learn the problem described in Section III-A. These characteristics are obtained through the use of function approximation techniques such as feed-forward multilayer perceptron neural networks (MLP) that allow generalisation, for instance, see [15].

Actor-Critic methods are on-policy temporal-difference (TD)-learning methods that have two memory structures, i.e. a dedicated memory for policies and another for value functions. The Actor represents the policy and this is denoted as $A(s)$. The Critic provides a state-value function $V(s)$. The Critic evaluates the outcome of the selected action against its existing value estimate (expectation) and generates a TD-error to the extent that it differs, see Eq. (1). The TD-error is then used to update both the Actor and the Critic. If the error is positive, the selected action should be strengthened, whereas a negative error suggests the opposite [20, p. 152]. The TD-error is defined as:

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{1}$$

where r_t is the reward received at time t, γ is the discount factor of future rewards, $V(s_{t+1})$ is the expected reward at the state s_{t+1} and $V(s_t)$ is the expected reward for state s_t.

Action selection is based on the current policy but in order to discover new and better policies, i.e. to learn, exploration is required. We use Gaussian exploration, where the performed action is sampled from a Gaussian distribution centred at the Actor's output $A(s_t)$. So the probability of selecting action a in time t is:

$$p_t(s_t, a) = \frac{1}{\sqrt{2\pi\sigma}} e^{-(a - A(s_t))^2/2\sigma^2} \tag{2}$$

where π denotes the mathematical constant and σ denotes the standard deviation and is here also called exploration rate. Finally, the performed action is determined by Eq. (2) and called a^*, see Figure 2 for a graphical representation of our implementation of CACLA.

CACLA differs from conventional Actor-Critic systems [20, p. 152] in the magnitude of the Actor's update being independent of the size of the TD-error. The Actor is instead updated towards the explored action only when the sign of the TD-error is positive. This idea originates from the fact that punishing or moving away from an action that does not lead to a higher reward does not guarantee a better solution [22]. Thus, the Actor is only updated towards actions that improve the agent performance instead of pulling the weights around without a destination. To control how strongly actions will be reinforced a derived algorithm called CACLA+var is used [22]. CACLA+var keeps a running average of the TD-error's variance, so actions leading to unusual higher rewards are reinforced more:

$$var_{t+1} = (1-\beta)var_t + \beta\delta_t^2 \tag{3a}$$

$$\text{number of updates} = \lceil \delta_t / \sqrt{var_t} \rceil \tag{3b}$$

CACLA+var requires two additional parameters to be tuned, i.e. var_0 which should be comparable to the typical value of δ. This is important to avoid high reinforcement rates early in learning when the agent behaviours are mostly random, and β.

Then the Actor's policy update can be expressed in pseudo-code as:

Algorithm 1 Actor's update

```
1: if $\delta_t > 0$ then
2:     for i := 1 to $\lceil \delta_t / \sqrt{var_t} \rceil$ step 1 do
3:         $\theta^A_{i,t+1} = \theta^A_{i,t} + \alpha\,(a^*_t - A(s_t))\,\frac{\partial A(s_t)}{\partial \theta^A_{i,t}}$
4:     end for
5: end if
```

where $\theta^A_{i,t}$ is the i^{th} item of the parameter vector of the Actor at time t, s_t is the state vector at time t and α is the learning rate for the Actor's function approximator. Unlike the Actor, the Critic is updated every time step as follows:

$$\theta^V_{i,t+1} = \theta^V_{i,t} + \eta\delta_t \frac{\partial V(s_t)}{\partial \theta^V_{i,t}} \tag{4}$$

where $\theta^V_{i,t}$ is the i^{th} item of the parameter vector of the Critic at time t, and η is the learning rate for the Critic's function approximator.

C. Reward function

The reward function consists of two parts, i.e. a rewarding component depending on the end-effector position and a punishing component used only in the *R+P* and *R+P+N* conditions depending on the joints' position. When both feedbacks are used there are additively combined into a single scalar value after every step. The rewarding component is computed as follows:

$$r_t^+ = \begin{cases} R & : d_t \leq 1.00\text{cm} \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

where R is the highest reward value, and d_t the distance from the end-effector to the target at time t.

Joint positions close to the lower or upper limit are considered harmful and a punishment signal is used to signal this. The amount each joint contributes to the total punishment per time step is determined by one of four functions, i.e. binary, linear, abrupt exponential (used in [1]) and smooth exponential. The difference between the abrupt and smooth exponential is the size of the variance which results in a discontinuity between the region with and without punishment for the abrupt exponential.

$$r_t^- = \begin{cases} -P & : \xi_i = \xi_i^{min} \vee \xi_i = \xi_i^{max} \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

$$r_t^- = -\frac{P}{dof} \times \begin{cases} \frac{-\xi_i^{lnb}}{\xi_i^{lnb}-\xi_i^{min}}\xi_i + \frac{1}{\xi_i^{lnb}-\xi_i^{min}} & : \xi_i \leq \xi_i^{lnb} \\ \frac{-\xi_i^{unb}}{\xi_i^{max}-\xi_i^{unb}}\xi_i + \frac{1}{\xi_i^{max}-\xi_i^{unb}} & : \xi_i \geq \xi_i^{unb} \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

$$r_t^- = -\frac{P}{dof} \times \begin{cases} e^{-0.5\left(\frac{\xi_i-\xi_i^{min}}{m_i}\right)^2} & : \xi_i \leq \xi_i^{lnb} \\ e^{-0.5\left(\frac{\xi_i-\xi_i^{max}}{m_i}\right)^2} & : \xi_i \geq \xi_i^{unb} \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

$$r_t^- = -\frac{P}{dof} \times \begin{cases} e^{-0.5\left(\frac{\xi_i-\xi_i^{min}}{3m_i}\right)^2} & : \xi_i \leq \xi_i^{lnb} \\ e^{-0.5\left(\frac{\xi_i-\xi_i^{max}}{3m_i}\right)^2} & : \xi_i \geq \xi_i^{unb} \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

where P is the maximum magnitude of punishment, dof the total number of degrees of freedom, ξ_i the absolute angular position of the i-th joint at time t. ξ_i^{min} and ξ_i^{max} are the minimum and the maximum possible angular position of the i-th joint, $\xi_i^{lnb} = \xi_i^{min} + m_i$ and $\xi_i^{unb} = \xi_i^{max} - m_i$ are the lower and upper nociceptive boundary of the i-th joint, and $m_i = 0.1 \times |\xi_i^{max} - \xi_i^{min}|$ is the margin of safety for a safety factor of 0.1 for the i-th joint. The punishment per joint is scaled by the number of degrees of freedom so that the total amount of punishment when all joints are at their boundary angular position is equal to the maximum magnitude of punishment P.

D. Neural architecture

We use two MLPs, one for the Actor and one for the Critic, see Figure 2. Both share the same input layer. The output layer for the Actor has two units that control the change in angular position, one per degree of freedom of the robot arm. The Critic has a single output unit to encode the expected reward. The rest of the layout is determined separately. The input layer is divided into three perceptual modalities. Firstly, there are two exteroceptive units which encode the Cartesian coordinates of the target in a 2-dimensional task space relative to the robot. Secondly, there are two proprioceptive units that encode the angular position of each of the joints of the robot's arm, i.e. the absolute joint value of the shoulder and elbow joint. Finally, there are two nociceptive units used only in the *R+N* and *R+P+N* conditions which are associated with each robot joint, with an activation almost identical to the function of punishment, see Equations (6)–(9).

All input values are scaled to the range $[-1, 1]$. The squashing function for the output units is linear, and for all other units, a custom hyperbolic tangent as defined by [23], [24] is used. Weight initialization is also performed as defined by [23], [24]. Bias units with value -1 are always used. Momentum and learning rate decay are not used. Both networks, the one for the Actor and the one for the Critic, are trained using back-propagation.

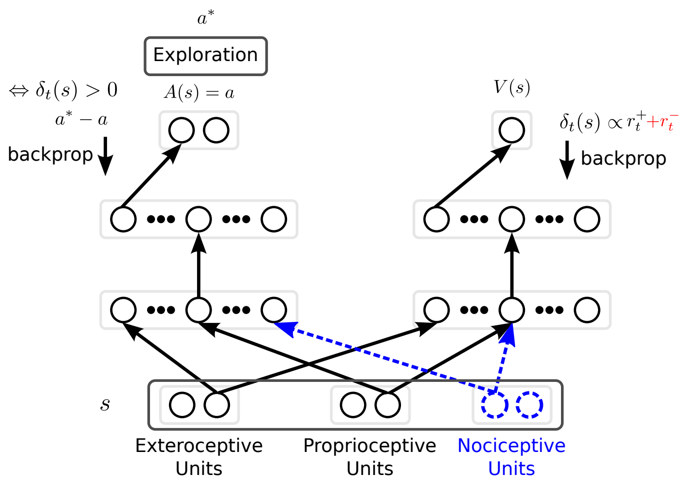

Fig. 2. The neural architecture used for inverse kinematics learning. For clarity, only one connection weight is shown (arrow between neuron layers). The hidden layers for both the Actor (left-hand side) and the Critic (right-hand side) are independently tuned. Solid units and connection weights in black correspond to the baseline, i.e. the *R* condition, and are extended by the 3 tested conditions. The punishment feedback given to the critic and depicted in red is only used for the *R+P* condition. Blue dashed units and connection weights are only considered under the *R+N* and *R+P+N* conditions. During training a^* is performed. a^* is determined based on the exploration of action a as described in Eq. (2). The Critic is trained every time step based on the TD-error δ, while the actor is trained only if the TD-error is positive.

Unlike punishment, the nociceptive units are able to discriminate between the upper and lower range of each joint, and the magnitude of the activation of each unit is not limited by the maximum punishment value and is not affected by the number of degrees of freedom of the arm. The activation of each nociceptive unit is determined by one of four functions, i.e. binary, linear, abrupt exponential (used in [1]) and smooth exponential.

$$n_t = \begin{cases} -1 & : \xi_i = \xi_i^{min} \\ 1 & : \xi_i = \xi_i^{max} \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

$$n_t = \begin{cases} \frac{\xi_i^{lnb}}{\xi_i^{lnb}-\xi_i^{min}}\xi_i - \frac{1}{\xi_i^{lnb}-\xi_i^{min}} & : \xi_i \leq \xi_i^{lnb} \\ \frac{-\xi_i^{unb}}{\xi_i^{max}-\xi_i^{unb}}\xi_i + \frac{1}{\xi_i^{max}-\xi_i^{unb}} & : \xi_i \geq \xi_i^{unb} \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

$$n_t = \begin{cases} -e^{-0.5\left(\frac{\xi_i-\xi_i^{min}}{m_i}\right)^2} & : \xi_i \leq \xi_i^{lnb} \\ e^{-0.5\left(\frac{\xi_i-\xi_i^{max}}{m_i}\right)^2} & : \xi_i \geq \xi_i^{unb} \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

$$n_t = \begin{cases} -e^{-0.5\left(\frac{\xi_i-\xi_i^{min}}{3m_i}\right)^2} & : \xi_i \leq \xi_i^{lnb} \\ e^{-0.5\left(\frac{\xi_i-\xi_i^{max}}{3m_i}\right)^2} & : \xi_i \geq \xi_i^{unb} \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

E. Hyperparameter optimization

Due to a large number of possible combinations of hyperparameters the systematic and exhaustive testing of them is impractical. Thus, we use an automated way to explore the hyperparameter space, in this case a genetic algorithm (GA), which helps to discover novel and better solutions than even an experienced human would be able to find [25], [26], [27]. We optimize the same hyperparameter and we use the same search space per hyperparameter as in [1]. The population per condition and per generation is 32 individuals. Evolution is carried out for 50 generations after which convergence was observed for all tested conditions.

We optimized the hyperparameters for all three conditions, i.e. *reward+punishment*, *reward+nociception* and *reward+punishment+nociception*, four times one per activation function with respect to the total distance between the robot's end-effector and target on the testing set after learning, i.e. after the last epoch. Thus, here, the lower the fitness values, the better. Eq. (14) shows the mathematical formulation of the fitness function:

$$D = \sum_{i=1}^{p} d(h_i, t_i) \tag{14}$$

where p represents the total number of testing pairs, h_i corresponds to the initial joint positions of the arm for testing pair i, t_i corresponds to the coordinates of the target for testing pair i and d is the final Euclidean distance between the arm's end-effector and the corresponding target.

IV. Results

Table I shows the summary of fitness values at the last generation. All conditions reach a comparable best fitness value. However, when comparing the results for a given activation function, the *R+N* condition reaches a considerably lower average and standard deviation than the *R+P* or the *R+P+N* conditions. The only exception is for the smooth exponential where the *R+P* reaches a slightly lower average and standard deviation.

TABLE I
Summary of the fitness values at generation number 50. The fitness is the total reaching distance, in meters, on the testing set, thus the smaller, the better.

		Binary	Linear	$e \propto \sigma$	$e \propto 3\sigma$
R+P	Best	1.563	1.533	1.551	1.512
	AVG	10.766	7.382	8.939	4.874
	SD	7.650	6.930	6.492	4.951
R+N	Best	1.497	1.453	1.425	1.418
	AVG	6.711	4.665	4.765	5.833
	SD	5.999	4.314	5.365	5.481
R+P+N	Best	1.595	1.489	1.378	1.552
	AVG	8.101	7.333	8.864	11.556
	SD	6.304	5.971	7.055	6.606

In the following sections the results for all conditions tested, i.e. *R+P*, *R+N*, and *R+P+N*, and the 4 different functions for punishment and nociception are presented. In all cases, the results presented are computed using multiple comparison tests using the 'Tukey-Kramer' procedure for a confidence interval of 95% on a two-way ANOVA, in Matlab version 2015a. The raw data and additional results are available as supplementary material [28]. The data was collected following the procedure used in [1], i.e. we used a genetic algorithm to search good hyperparameter sets for each condition [29, pp. 90–108]. Then the 4 best hyperparameter sets for each condition was tested 10 times with different random initializations on the validation set. For all presented metrics we provide the results for the performance after learning and the cumulative performance during learning. The value after learning provides a metric of the final behavioural performance of the agent, whereas the cumulative value provides an estimation of the convergence speed.

A. Positioning Error

Table II shows the estimated mean and the standard error of the positioning error after and during learning. The estimated mean of the positioning error of our baseline [1], the *reward only (R)* condition, is 0.0390m after learning and 1.1696m during learning or cumulative.

The best performance on the task is achieved when using smooth exponential activation of punishment, closely followed by the abrupt exponential function used as nociceptive input, see Table II. With all activation functions, punishment leads to a better performance than our baseline. Similarly, for nociception, all but the binary activation leads to better performance than our baseline. When looking at the cumulative values, however, it can be seen that no matter what activation of punishment is used, punishment always converges slower than our baseline. By contrary, both exponential functions used as nociceptive inputs converge faster than our baseline.

TABLE II
ESTIMATED VALUES OF THE MEAN AND STANDARD ERROR FOR THE POSITIONING ERROR AFTER AND DURING LEARNING.

		Punishment		Nociception	
		Mean	std	Mean	std
After Learning	Binary	0.0364	0.0036	0.0475	0.0043
	Linear	0.0334	0.0036	0.0330	0.0043
	$e^{-1/\sigma}$	0.0362	0.0036	0.0268	0.0043
	$e^{-1/3\sigma}$	0.0241	0.0036	0.0322	0.0043
Cumulative	Binary	1.4315	0.0600	1.4893	0.0422
	Linear	1.3510	0.0600	1.2332	0.0422
	$e^{-1/\sigma}$	1.6876	0.0600	1.1523	0.0422
	$e^{-1/3\sigma}$	1.9943	0.0600	1.1150	0.0422

Table III shows the p-values and effect size in percentage of all 4 activation functions of punishment and nociception in direct comparison to the other conditions. It terms of performance after learning all three conditions show comparable results. However, the combination of punishment and nociception with the smooth exponential is significantly detrimental. For the abrupt exponential, nociception seems to outperform punishment. The results for performance during learning are more categorically in favour of the use of nociceptive units instead of punishment. Here, both exponential functions used as nociceptive input are significantly better than when used as punishment. Also the linear function of nociception seems slightly better than punishment. When combined, nociception improves the agent's performance (*R+P* versus *R+P+N*) whereas punishment reduces performance (*R+N* versus *R+P+N*).

B. Perceived Nociception or Potential for Damage

The potential for damage was measured to understand the effects of the different activation functions on learning but this metric has no influence on the hyperparameter optimisation procedure. As defined in [1], the mean perceived nociception (potential for damage) is defined as the cumulative absolute value of nociception per joint, as defined in Equations 10–13, and per time step for all samples on the validation set after learning. The maximum number of steps allowed per action sequence is 10. Thus, the maximum possible value for the potential for damage per sample is 20. The estimated mean of the perceived nociception of our baseline is 3.9474 after learning and 95.4031 during learning or cumulative.

Table IV shows the estimated mean and the standard error of the perceived nociception (potential for damage) after and during learning. Both exponential functions of punishment also perform the best when observing the performance only after learning. Both exponential functions of nociception also

TABLE III
PAIR-WISE COMPARISON. P-VALUES AND EFFECT SIZE FOR THE DIFFERENT ACTIVATION FUNCTIONS FOR BOTH THE AVERAGE POSITIONING ERROR AFTER LEARNING AND THE AVERAGE CUMULATIVE POSITIONING ERROR DURING LEARNING. POSITIVE PERCENTAGE (SIGNIFICANT: BLUE) IN THE EFFECT SIZE INDICATES THAT GROUP 2 IS BETTER THAN GROUP 1 AND NEGATIVE PERCENTAGE (SIGNIFICANT: RED) INDICATES THAT GROUP 1 IS BETTER THAN GROUP 2.

G 1	G 2		Binary	Linear	$e \propto \sigma$	$e \propto 3\sigma$
R+P	R+N	After Learning	0.1401	0.9860	0.0742	0.4958
			−30.49 %	1.20 %	25.97 %	−33.61 %
R+P	R+P +N		0.9949	0.8151	0.9829	0.0011
			1.37 %	5.09 %	2.21 %	−108.30 %
R+N	R+P +N		0.1150	0.8945	0.1095	0.0345
			24.42 %	3.94 %	−32.09 %	−55.90 %
R+P	R+N	Cumulative	0.7141	0.0921	0.0000	0.0000
			−4.04 %	8.72 %	31.72 %	44.09 %
R+P	R+P +N		0.0065	0.0002	0.0016	0.5804
			−16.09 %	17.36 %	17.46 %	4.27 %
R+N	R+P +N		0.0550	0.0961	0.0124	0.0000
			−11.58 %	9.46 %	−20.89 %	−71.23 %

perform better than the baseline. Surprisingly, the binary function of punishment also performs better than the baseline.

Similar to the task performance metric, once we observe the performance of punishment during learning a different picture arises, i.e. punishment performs worse than nociception in all but the binary function. Consistent with the results on task performance in both after and during learning, both exponential functions of nociception perform better than the baseline.

TABLE IV
ESTIMATED VALUES OF THE MEAN AND STANDARD ERROR FOR THE POTENTIAL FOR DAMAGE AFTER AND DURING LEARNING.

		Punishment		Nociception	
		Mean	std	Mean	std
After Learning	Binary	3.1722	0.4073	4.0656	0.3894
	Linear	4.1635	0.4073	4.2963	0.3894
	$e^{-1/\sigma}$	4.8514	0.4073	3.0847	0.3894
	$e^{-1/3\sigma}$	1.7005	0.4073	3.6006	0.3894
Cumulative	Binary	90.5132	3.4787	96.5160	2.6107
	Linear	119.0987	3.4787	102.9289	2.6107
	$e^{-1/\sigma}$	142.9666	3.4787	92.4481	2.6107
	$e^{-1/3\sigma}$	135.9239	3.4787	88.1168	2.6107

Table V shows the p-values and effect size in percentage of all 4 activation functions of punishment and nociception in direct comparison to the other conditions.

Similar to the positioning error, the results after learning for all conditions are comparable for the binary and linear activation functions. The abrupt exponential function seems beneficial to nociceptive units and detrimental for punishment, while the smooth exponential leads to the opposite result.

During learning, however, the linear and both exponential activation functions of nociception are significantly better than punishment. Also, the combination of nociception and punishment leads to significant better results than when punishment is used alone. The binary function seems to be detrimental for nociception.

TABLE V
PAIR-WISE COMPARISON. P-VALUES AND EFFECT SIZE FOR THE DIFFERENT ACTIVATION FUNCTIONS FOR BOTH THE AVERAGE POTENTIAL FOR DAMAGE AFTER LEARNING AND THE AVERAGE CUMULATIVE POTENTIAL FOR DAMAGE DURING LEARNING. POSITIVE PERCENTAGE (SIGNIFICANT: BLUE) IN THE EFFECT SIZE INDICATES THAT GROUP 2 IS BETTER THAN GROUP 1 AND NEGATIVE PERCENTAGE (SIGNIFICANT: RED) INDICATES THAT GROUP 1 IS BETTER THAN GROUP 2.

G 1	G 2		Binary	Linear	$e \propto \sigma$	$e \propto 3\sigma$
R+P	R+N	After Learning	0.3259	0.9745	0.0113	0.0009
			−28.16 %	−3.19 %	36.42 %	−111.74 %
R+P	R+P +N		0.5912	0.6319	0.9461	0.0032
			−19.20 %	13.48 %	3.93 %	−100.83 %
R+N	R+P +N		0.8914	0.4968	0.0271	0.9301
			6.99 %	16.16 %	−51.08 %	5.15 %
R+P	R+N	Cumulative	0.3618	0.0001	0.0000	0.0000
			−6.63 %	13.58 %	35.34 %	35.17 %
R+P	R+P +N		0.0000	0.0000	0.0000	0.0665
			−27.42 %	22.36 %	15.37 %	8.28 %
R+N	R+P +N		0.0001	0.0201	0.0000	0.0000
			−19.49 %	10.16 %	−30.88 %	−41.49 %

C. Positioning Speed

The estimated mean of the positioning speed of our baseline, the *R* condition, measured as the length of the action sequence in the validation set is 8.3853 after learning and 175.4069 during learning or cumulative.

Table VI shows the estimated mean and the standard error of the positioning speed after and during learning. Similar to both previous metrics, the smooth exponential function for punishment is the only function of punishment performing better than the baseline after learning. Also, the abrupt exponential function for nociception performs better than the baseline and also better than all *reward+punishment* conditions. All cumulative results of punishment perform worse than the baseline and also each punishment function performs worse than its nociception equivalent. Both exponential functions of nociception perform better than the baseline.

TABLE VI
ESTIMATED VALUES OF THE MEAN AND STANDARD ERROR FOR THE POSITIONING SPEED AFTER AND DURING LEARNING.

		Punishment		Nociception	
		Mean	std	Mean	std
After Learning	Binary	8.5183	0.0865	8.5854	0.0899
	Linear	8.5174	0.0865	8.4904	0.0899
	$e^{-1/\sigma}$	8.6402	0.0865	8.0946	0.0899
	$e^{-1/3\sigma}$	8.2218	0.0865	8.4637	0.0899
Cumulative	Binary	179.0809	0.5992	178.6586	0.5364
	Linear	178.8567	0.5992	176.2517	0.5364
	$e^{-1/\sigma}$	181.4559	0.5992	172.8615	0.5364
	$e^{-1/3\sigma}$	184.0022	0.5992	174.5387	0.5364

For this metric, the effect size of all of the tested conditions and functions is rather modest, as shown is Table VII. The performance after learning for the binary and linear activation function is comparable for all conditions. However, there are some differences for both exponential activation functions. When the abrupt exponential is used nociception outperforms punishment, whereas when the smooth exponential is used punishment seems to be slightly better than nociception.

For the linear and both exponential functions, the performance during learning of the conditions using nociception is significantly better than the conditions using punishment. Results for the binary activation function are mixed.

TABLE VII
PAIR-WISE COMPARISON. P-VALUES AND EFFECT SIZE FOR THE DIFFERENT ACTIVATION FUNCTIONS FOR BOTH THE AVERAGE POSITIONING SPEED AFTER LEARNING AND THE AVERAGE CUMULATIVE POSITIONING SPEED DURING LEARNING. POSITIVE PERCENTAGE (SIGNIFICANT: BLUE) IN THE EFFECT SIZE INDICATES THAT GROUP 2 IS BETTER THAN GROUP 1 AND NEGATIVE PERCENTAGE (SIGNIFICANT: RED) INDICATES THAT GROUP 1 IS BETTER THAN GROUP 2.

G 1	G 2		Binary	Linear	$e \propto \sigma$	$e \propto 3\sigma$
R+P	R+N	After Learning	0.8685	0.9690	0.0001	0.1335
			−0.79 %	0.32 %	6.31 %	−2.94 %
R+P	R+P +N		0.5025	0.9988	0.2311	0.0002
			−1.75 %	0.06 %	2.34 %	−6.27 %
R+N	R+P +N		0.8117	0.9798	0.0163	0.0773
			−0.95 %	−0.26 %	−4.25 %	−3.23 %
R+P	R+N	Cumulative	0.8461	0.0016	0.0000	0.0000
			0.24 %	1.46 %	4.74 %	5.14 %
R+P	R+P +N		0.0046	0.0000	0.0000	0.1490
			−1.38 %	2.33 %	2.35 %	0.89 %
R+N	R+P +N		0.0007	0.0867	0.0000	0.0000
			−1.62 %	0.89 %	−2.50 %	−4.49 %

V. DISCUSSION

With respect to the three learning metrics under investigation: task performance, the potential for damage and the positioning speed, all conditions perform similarly when using the binary and the linear activation function. The abrupt exponential seems to be more beneficial for the conditions with nociception, whereas the smooth exponential seems to be more beneficial for punishment. The combination of both nociception and punishment lead to mixed results. Overall, the performance after learning for all conditions and activation functions is comparable (no statistically significant differences) to the baseline condition, see the supplementary material [28].

The performance during learning in all three metrics, on the other hand, emphatically supports the use of nociceptive units instead of punishment. The only exception seems to be for the binary activation function. However, the binary function is the lowest performing function in almost all metrics and conditions, see the supplementary material [28]. Paradoxically, it is the most common punishment function (e.g. [16], [15]).

During learning, punishment converges more slowly and leads to higher potential for damage than our baseline and than the *R*+*N* condition. The higher potential for damage may be a direct consequence of the slower convergence speed. This, however, contradicts the findings reported in [16] where punishment seems to improve convergence rate and speed. However, we hypothesise that the results reported in [16] represent a "lucky" case, because they did not perform any systematic hyperparameter optimisation procedure and they only examined a single target-starting position pair for training and testing at any given time.

Considering the reported results and in agreement with the concept that for real-world applications, the most important metrics are convergence and positioning speed [16], we suggest that punishment should be used prudently, because reinforcement learning algorithms in general cannot account for an adequate incorporation of both types of reinforcement signals. This is due to the inevitable loss of information when both reinforcement signals are conflated into a single scalar value [2]. Additionally, we suggest to further study the effect of nociceptive units [1], which may become a viable alternative to the use of punishment signals without comprising convergence ratio or speed as punishment does.

As future work, we would like to develop other ways of incorporating punishment feedback into reinforcement learning without being detrimental to the convergence rate, the convergence speed or the potential for damage. We would also like to further develop the embodied approach of using punishment signals as nociceptive units. Our results, albeit mixed, show that the concurrent use of nociception with punishment can counteract the detrimental effects of punishment during learning.

Acknowledgment

The authors gratefully acknowledge partial support from the German Research Foundation DFG under project CML (TRR 169), the European Union under project SECURE (No 642667), and the Hamburg Landesforschungsförderungsprojekt CROSS.

References

[1] N. Navarro-Guerrero, R. Lowe, and S. Wermter, "Improving Robot Motor Learning with Negatively Valenced Reinforcement Signals," *Frontiers in Neurorobotics*, vol. 11, no. 10, 2017.

[2] R. Lowe and T. Ziemke, "Exploring the Relationship of Reward and Punishment in Reinforcement Learning," in *IEEE Symposium on Adaptive Dynamic Programming And Reinforcement Learning (ADPRL)*. Singapore: IEEE, Apr. 2013, pp. 140–147.

[3] T. Wächter, O. V. Lungu, T. Liu, D. T. Willingham, and J. Ashe, "Differential Effect of Reward and Punishment on Procedural Learning," *The Journal of Neuroscience*, vol. 29, no. 2, pp. 436–443, Jan. 2009.

[4] Y.-L. Boureau and P. Dayan, "Opponency Revisited: Competition and Cooperation Between Dopamine and Serotonin," *Neuropsychopharmacology*, vol. 36, no. 1, pp. 74–97, Jan. 2011.

[5] B. Seymour, J. P. O'Doherty, M. Koltzenburg, K. Wiech, R. Frackowiak, K. Friston, and R. Dolan, "Opponent Appetitive-Aversive Neural Processes Underlie Predictive Learning of Pain Relief," *Nature Neuroscience*, vol. 8, no. 9, pp. 1234–1240, Sep. 2005.

[6] S. Palminteri and M. Pessiglione, "Opponent Brain Systems for Reward and Punishment Learning: Causal Evidence from Drug and Lesion Studies in Humans," in *Decision Neuroscience: An Integrative Approach*. San Diego: Academic Press, 2017, pp. 291–303, chapter: 23.

[7] J. M. Galea, E. Mallia, J. Rothwell, and J. Diedrichsen, "The Dissociable Effects of Punishment and Reward on Motor Learning," *Nature Neuroscience*, vol. 18, no. 4, pp. 597–602, Apr. 2015.

[8] D. Talmi, P. Dayan, S. J. Kiebel, C. D. Frith, and R. J. Dolan, "How Humans Integrate the Prospects of Pain and Reward During Choice," *The Journal of Neuroscience*, vol. 29, no. 46, pp. 14 617–14 626, Nov. 2009.

[9] A. B. Craig, "Interoception: The Sense of the Physiological Condition of the Body," *Current Opinion in Neurobiology*, vol. 13, no. 4, pp. 500–505, Aug. 2003.

[10] A. D. B. Craig, "Emotional Moments Across Time: A Possible Neural Basis for Time Perception in the Anterior Insula," *Philosophical Transactions: Biological Sciences*, vol. 364, no. 1525, pp. 1933–1942, Jul. 2009.

[11] A. Damasio, "Neuroscience and the Emergence of Neuroeconomics," in *Neuroeconomics*. London: Academic Press, 2009, pp. 207–213, chapter: 14.

[12] A. R. Damasio, "The Somatic Marker Hypothesis and the Possible Functions of the Prefrontal Cortex [and Discussion]," *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, vol. 351, no. 1346, pp. 1413–1420, Oct. 1996.

[13] M. P. Paulus, "Interoception and Decision Making," in *Decision Making, Affect, and Learning: Attention and Performance XXIII*. Oxford University Press, Mar. 2011, pp. 387–401, chapter: 18.

[14] C. Stahlhut, N. Navarro-Guerrero, C. Weber, and S. Wermter, "Interaction in Reinforcement Learning Reduces the Need for Finely Tuned Hyperparameters in Complex Tasks," *Kognitive Systeme*, vol. 3, no. 2, Dec. 2015.

[15] E. van der Wal, "Object Grasping with the NAO," MSc, University of Groningen, Groningen, The Netherlands, Apr. 2012.

[16] M. Tamosiunaite, T. Asfour, and F. Wörgötter, "Learning to Reach by Reinforcement Learning using a Receptive Field Based Function Approximation Approach with Continuous Actions," *Biological Cybernetics*, vol. 100, no. 3, pp. 249–260, Mar. 2009.

[17] E. Thelen, D. Corbetta, K. Kamm, J. P. Spencer, K. Schneider, and R. F. Zernicke, "The Transition to Reaching: Mapping Intention and Intrinsic Dynamics," *Child Development*, vol. 64, no. 4, pp. 1058–1098, 1993.

[18] D. W. Franklin, U. So, E. Burdet, and M. Kawato, "Visual Feedback is not Necessary for the Learning of Novel Dynamics," *PLoS ONE*, vol. 2, no. 12, p. e1336, Dec. 2007.

[19] R. E. Suri, "TD Models of Reward Predictive Responses in Dopamine Neurons," *Neural Networks*, vol. 15, no. 4-6, pp. 523–533, Jun. 2002.

[20] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 1st ed., ser. Adaptive computation and machine learning. Cambridge, MA, USA: A Bradford Book/The MIT Press, Feb. 1998.

[21] H.-G. Schaible, "Articular Nociceptors," in *Encyclopedia of Pain*. Springer Berlin Heidelberg, 2013, pp. 207–213.

[22] H. van Hasselt and M. A. Wiering, "Reinforcement Learning in Continuous Action Spaces," in *IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL)*. Honolulu, HI, USA: IEEE, Apr. 2007, pp. 272–279.

[23] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient BackProp," in *Neural Networks: Tricks of the Trade*, ser. LNCS. Springer Berlin Heidelberg, Jan. 1998, no. 1524, pp. 9–50.

[24] ——, "Efficient BackProp," in *Neural Networks: Tricks of the Trade*, 2nd ed., ser. LNCS. Springer Berlin Heidelberg, Jan. 2012, no. 7700, pp. 9–48.

[25] R. Bardenet, M. Brendel, B. Kégl, and M. Sebag, "Collaborative Hyperparameter Tuning," in *International Conference on Machine Learning (ICML)*, vol. 28. Atlanta, GA, USA: JMLR: W&CP, 2013, pp. 199–207.

[26] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for Hyper-Parameter Optimization," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 25. Granada, Spain: Curran Associates, Inc., Dec. 2011, pp. 2546–2554.

[27] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms," in *Advances in Neural Information Processing Systems (NIPS)*. Lake Tahoe, NV, USA: Curran Associates, Inc., Dec. 2012, pp. 2951–2959.

[28] N. Navarro-Guerrero, R. Lowe, and S. Wermter, "Supplementary Material for "The Effects on Adaptive Behaviour of Negatively Valenced Signals in Reinforcement Learning"," Sep. 2017. [Online]. Available: https://figshare.com/s/1eba690a643302c08e72

[29] N. Navarro-Guerrero, "Neurocomputational Mechanisms for Adaptive Self-Preservative Robot Behaviour," Dissertation, Universität Hamburg, Hamburg, Germany, May 2016. [Online]. Available: http://ediss.sub.uni-hamburg.de/volltexte/2016/7890/