Strategies for Affect-Controlled Action-Selection in Soar-RL

Eric Hogewoning[1], Joost Broekens[1], Jeroen Eggermont[2], and Ernst G.P. Bovenkamp[2]

[1] Leiden Institute of Advanced Computer Science, Leiden University, P.O. Box 9500, 2300 RA Leiden, The Netherlands
[2] Leiden University Medical Center, Department of Radiology, Division of Image Processing, P.O. Box 9600, 2300 RC Leiden, The Netherlands

Abstract. Reinforcement learning (RL) agents can benefit from adaptive exploration/exploitation behavior, especially in dynamic environments. We focus on regulating this exploration/exploitation behavior by controlling the action-selection mechanism of RL. Inspired by psychological studies which show that affect influences human decision making, we use artificial affect to influence an agent's action-selection. Two existing affective strategies are implemented and, in addition, a new hybrid method that combines both. These strategies are tested on 'maze tasks' in which a RL agent has to find food (rewarded location) in a maze. We use Soar-RL, the new RL-enabled version of Soar, as a model environment. One task tests the ability to quickly adapt to an environmental change, while the other tests the ability to escape a local optimum in order to find the global optimum. We show that artificial affect-controlled action-selection in some cases helps agents to faster adapt to changes in the environment.

1 Introduction

At the core of emotion and mood are states that have certain levels of valence and arousal, i.e, affective states. Valence represents the goodness versus badness of that state, while arousal represents the activity of the organism associated with that state. Affect plays an important role in thinking. Normal affective functioning seems to be necessary for normal cognition [1]. In fact, many cognitive processes (attention, memory) are to some level influenced by affective states [2,3]. Acknowledging the need for affect in human decision making, we investigate how artificial affect can be used to control an artificial agent's equivalent for decision making, i.e., its action-selection mechanism.

Different learning tasks and often even different phases in a single task require different learning-parameters. Tuning these parameters manually is laborious and the algorithms currently available for automatic tuning are often task-specific or need several meta-parameters themselves. Better methods for regulating meta-parameters are needed. In this paper we focus on regulating exploration/exploitation behavior, an important outstanding problem.

More specifc, we compare diferent methods that use artificial affect to control the greedyness versus randomness of action-selection, thereby influencing exploitation versus exploration respectively. Two existing affect-controlled action-selection strategies are implemented and their performance is compared with that of static action-selection strategies and a new affect-controlled strategy. These strategies are tested on two tasks in which an artificial agent has to find the optimal path to food in a maze. One task tests the agent's capacity to adapt to a change in its environment. The second task tests the agent's ability to escape a local optimum in order to find the global optimum. Affect is operationalized as a measure that keeps track of "how well the agent is doing compared to what it is used to". As such, in this paper we only model the valence part of affect. For a psychological grounding of artifical affect as used in this paper the reader is referred to [4].

In the next sections we first introduce the core components of our approach, after which we discuss our experimental results.

2 Learning, Action-Selection and Soar-RL

The agents in this project are Soar-RL agents. Soar (States, Operators and Results) is a production-rule based architecture [5] that enables the design of cognitive agents, and is used extensively in Cognitive Modelling. Soar agents solve problems based on a cyle of state-perception, proposal of actions, action-selection, and action execution.

Soar has recently been augmented with reinforcement learning. This new version, Soar-RL [6], uses Q-learning [7], which works by estimating the values of state-action pairs. The value $Q(s, a)$ is defined to be the expected discounted sum of future rewards obtained by taking action a from state s and following an optimal policy thereafter. Values learned for actions are called Q-values, and are learned by experience. From the current state s, an action a is selected. This yields an immediate reward r, and arrival at a next state s'. $Q(s, a)$ is updated by a value propagation function [7].

Important to our research is that in each cycle, Soar-RL uses a Boltzmann equation (see Eq. 1) to select an action from a set of possible actions.

$$P(a) = \frac{e^{Q_t(s,a)\cdot\beta}}{\sum_{b=1}^{n} e^{Q_t(s,b)\cdot\beta}} \tag{1}$$

$P(a)$ is the probability of action a being chosen. $Q_t(s, a)$ is the estimated value for performing action a in state s at time t. This equation returns the probability of action a being chosen out of the set of possible actions, based upon an agent's Q-values of those actions and upon the variable β, called *inverse temperature*. The β parameter controls the greediness of the Boltzmann probability distribution. If $\beta \Rightarrow \infty$, the system becomes deterministic and will select the action with the highest estimated value. If $\beta \Rightarrow 0$, each action has the same probability ($\frac{1}{n}$) of being selected. In other words, a low β corresponds to random behavior and a high β to greedy behavior. This β can be adjusted during simulations and is used by our adaptive strategies for regulating the agent's behavior.

Soar-RL creates new actions and updates the estimated values of existing actions as they are explored. Therefore, exploration is needed to construct a good model of the environment. When the agent's model of the environment (its state $\Rightarrow$ action mapping) is accurate enough, the agent should exploit its knowledge by performing greedy behavior in order to maximize the received rewards. Thus, a strategy balancing exploration/exploitation is needed in order to perform tasks efficiently. However, it is hard to decide when an agent should stop exploring and start exploiting, especially in dynamic environments.

3 Affect-Controlled Action-Selection

To address the aforementioned exploration/exploitation tradeoff problem, we investigate whether artificial affect can be used to control the β parameter. For more detail on the relation between artificial affect and natural affect see [4][8].

3.1 Schweighofer and Doya's Method

Schweighofer and Doya [9] proposed that emotion should not just be considered 'emergency behavioral routines', but a highly important component of learning: emotion can be considered a metalearning system. Doya argued that a mapping exists between RL parameters and neuromodulators [10]. The β parameter is regulated by a search strategy. A random amount of noise is added to the β and the newly obtained β is tested. If the resulting behavior proves to perform better, then the β is adjusted in the direction of the noise. If, for example, positive noise yields higher rewards, then the β is increased.

In Schweighofer and Doya's model (referred to as SD), β is governed by the following set of equations:

$$\beta(t) = e^{\kappa} + \sigma(t) \tag{2}$$

β is used in the Boltzmann distribution to determine the amount of exploration, σ is a Guassian noise source term with mean 0 and variance ν. A new noise value is drawn every N steps, with N $\gg 1$.

$$\Delta\kappa = \mu \cdot (\bar{r}(t) - \bar{\bar{r}}(t)) \cdot \sigma(t-1) \tag{3}$$

$\bar{r}(t)$ is the short-term average reward and $\bar{\bar{r}}(t)$ is the long-term average reward. μ is a learning rate.

$$\Delta\bar{r}(t) = \frac{1}{\tau_1} \cdot (r(t) - \bar{r}(t-1)) \tag{4}$$

$$\Delta\bar{\bar{r}}(t) = \frac{1}{\tau_2} \cdot (\bar{r}(t) - \bar{\bar{r}}(t-1)) \tag{5}$$

τ_1 and τ_2 are time constants for respectively the short-term and the long-term. The term $(\bar{r}(t) - \bar{\bar{r}}(t))$ in equation 4 represents valence: positive when doing better than expected, negative when doing worse than expected.

3.2 Broekens and Verbeek's Method

Where Schweighofer and Doya use a search-based method, Broekens and Verbeek [8] use a direct relation between affect and exploration. High valence results in exploitation, while low valence leads to exploration. As a measure for valence, the difference between short and long term average rewards is used. This method (referred to as BV) is not a search algorithm and does not attempt to find an optimal value for β, but tries to balance exploration/exploitation by responding to changes in the environment.

The following equations are used to govern the agent's exploration behavior:

$$e_p = (\bar{r}(t) - (\bar{\bar{r}}\,(t) - f \cdot \sigma_{ltar}))/2 \cdot f \cdot \sigma_{ltar} \tag{6}$$

e_p is a measure for valence and σ_{ltar} is the long-term variance of $\bar{r}(t)$. $\bar{r}(t)$ and $\bar{\bar{r}}\,(t)$ are again short-term and long-term running averages and are computed in the same way as in SD.

$$\beta = e_p \cdot (\beta_{max} - \beta_{min}) + \beta_{min} \tag{7}$$

Thus, the agent's valence (on a scale from 0 to 1) directly translates to the amount of exploration/exploitation (on a scale from β_{min} to β_{max}).

3.3 Hybrid-χ^2 Method

Both methods described above have their own strengths and drawbacks. BV has the ability to respond quickly to sudden changes in the environment. However, it is bound to a fixed range of values, and β will always converge to the center of that range when the environment stabilizes. SD, on the other hand, is able to cope with a broader value range, and β converges to the optimal[1] value in this range. The downside of this method is that it has trouble responding to sudden changes (see results section). To overcome these drawbacks, we propose a new method, The Hybrid-χ^2 method. This method combines both methods and uses the environmental stability to balance the contribution of SD and BV to the actual β used. The more substantial the changes in the environment, the more influence BV has. In stable environments, SD gets most influence. A heuristic for detecting environmental change is the reward distribution. If we assume two equally long, consecutive reward histories, the difference in reward distribution between these histories is a measure for the stability of the environment. A large difference indicates substantial changes[2], and vice versa.

This difference is computed with the aid of the statistical χ^2 test. It measures whether two sets of numbers are significantly different, and returns a value between 0 (different) and 1 (equal). In our case, we compare the long term rewards

[1] Possibly a local optimum.

[2] Or a situation in which the agent is heavily exploring, in which case BV should also have the most influence.

with the short term rewards and compute the significance of the differences. Using χ^2, β is computed as follows:

$$\beta(t) = \chi^2 \cdot \beta_{SD} + (1 - \chi^2) \cdot \beta_{BV} \tag{8}$$

The value for β is restricted to the interval $[\beta_{min} \ldots \beta_{max}]$ because BV directly couples valence to the amount of exploration. Some flexibility was added to this interval by using the, β found by SD to influence β_{max}. If SD's proposed β is different from the current value, the maximum value of the range will be adjusted according to:

$$\beta_{max} = \beta_{max} + \chi^2 \cdot (\beta_{SD} - \beta(t-1)) \tag{9}$$

4 Experiments

We tested the methods described in the previous section on two different maze tasks. We focused on the resulting exploration/exploitation behavior of these methods, not exclusively on their learning performance. In addition to the affective methods, we tested Soar-RL's own performance for several fixed β's without the addition of affect-controlled action-selection. We refer to 'unaffected' Soar-RL as the *static method*.

In both tasks a Soar-RL agent needs to find food in a maze. The vision of the agent is limited to one tile in all directions. For both mazes, this vision is large enough to satisfy the Markov property[3]. The agent has to learn this behavior purely from the rewards it receives.

4.1 Cue-Inversion and Candy Task

In the maze of Figure 1a, the agent has to learn to go to A if the light is *on* and to B if the the light is *off*, using the shortest path. A reward of +1 is given if the agent reaches the correct goal state. -1 is the reward for going to the wrong goal state or for walking into a wall. The agent is set back to the starting position when an end-state has been reached.

After 3000 steps, after the agent has learned this maze quite well, we switch the rewards of the two goal states; so now the agent has to go to A when the light is *off* to get the +1 reward and to B when the light is *on*. The agent has to unlearn its previous behavior and learn a new behavior. This task is constructed to be similar to psychological cue inversion tasks (e.g., [11]) and tests the agent's capacity to adapt to a sudden change in the environment. One run equals 8000 steps in this maze.

Figure 1b shows the maze used for the Candy task (see also [4]). Close to the starting position of the agent is some candy (end state B). The agent receives a reward of 0.5 for grabbing this candy. A few steps further is some real food (A), rewarded with +5. The reward per step is higher for going after this food

[3] No knowledge of the history of states is required to determine the future behavior of the environment.

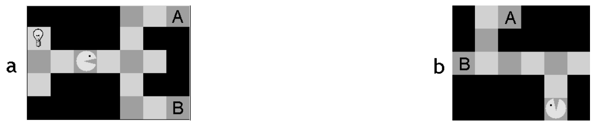

Fig. 1. a: Maze used in Cue-inversion task. b: Maze used in Candy task.

and it is the agent's goal to find this real food, instead of the candy. Whenever the agent finds the candy or the real food, it is set back to the starting position. This task tests the agent's ability to escape a local optimum. Contrary to the other task, this task does not have any switches, but it does however require flexible behavior: the agent needs to switch from initial exploration to exploiting the local optimum (the candy) and then revert to exploration to find the global optimum (the real food) and exploit that afterwards. In this task a run is aborted at t=25000. At this point it was clear how different methods perform.

4.2 Simulation Settings

Simulations are performed using a short term moving average of 50 rewards and a long term moving average of 400 short term averages. In initial experiments, these values proved to be useful for determining whether 'we are doing better than we used to do'. The two other parameters used by Q-learning (the learning rate (α) and a discount factor (γ)), are set at the default values of Soar-RL: $\alpha = 0.4$ and $\gamma = 0.9$. Results are aggregated over 50 runs. This proved to be enough to reproduce the same results when repeating the experiments. Every 30 steps, the average values of β and the received rewards are computed and printed.

5 Results

BV performs best on the cue inversion task (Figure 2). It is faster than the three other methods both for initial learning and for relearning after the cue inversion. The impact of the cue-inversion on β is visible in Figure 2B. The agent appears to benefit from the exploration phase after the inversion. SD performs poorly, even worse than the control method. SD does not seem to respond to the switch, but slowly increases β over time instead. This is an average β and the actual situation is somewhat more complex. We measured a high diversity in results: the standard deviation near the switch is 8.1, but even at step=6000 it is still 5.6. Data of individual runs seem to suggest that there are runs that increase rapidly to the maximum value of β=30 and there are some that cannot escape low β values. These high β runs do not compensate for the lower rewards obtained by low β runs.

The characteristics of the β curves of the Hybrid-χ^2 method are quite similar to those of BV: β quickly decreases after the inversion and it slowly converges to a center value. SD's influence in determining the β_{max} is not noticeable. All

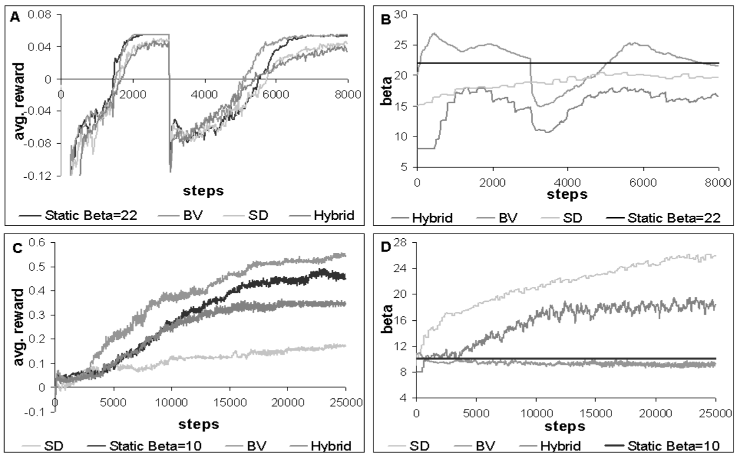

Fig. 2. Graphs A and C display the average reward per time step on the Cue-inversion task (A) and the Candy task (C). Graphs B and D show the corresponding value for β. The inversion occurs at t=3000 in the cue-inversion task.

runs seem to stay well within their initial ranges and no trend can be detected besides the one that pulls the β to the center value.

It is clear from Figure 2C and 2D that SD performs poorly on the Candy task. The β increases when the agent encounters the candy and, as a result, the agent exploits the candy afterwards.

BV again clearly shows that it cannot search for an optimal β, it merely responds to the rewards received: after an initial exploitation burst, β converges to the center of the β range. Although the optimal center value for BV is close to the optimal value (β=10) for the static method, BV's 'noisy' β value performs slightly better than the static β value.

Again, BV's component in the hybrid method has the upper hand. The β's are drawn to the center value (β=15). It is not fair to compare the rewards of the Hybrid-χ^2 method directly with those of BV, as the center values for the hybrid runs were set higher than those of BV's simulations. As mentioned above, SD cannot compensate by decreasing β, as SD mainly attempts to exploit the candy by increasing the β even further.

6 Discussion and Future Work

For an elaborate discussion of related work (e.g., the work on Soar and emotion by Marinier [12], and the work on affect and problem solving by Belavkin [13]) we would like to refer the reader to [14]. In this paper, we concentrate on comparing the three methods discussed above. For more detail on BV, we refer to [4,8].

SD performed poorly on the given tasks. In the Candy task this is probably due to the agent's quick exploitation. To exploit the local optimum, a very high β is beneficial. The random noise used as seed for the search process is not enough to trigger exploration. A similar problem can occur with a very low β: an agent can get trapped in explorative behavior when changes to an already low β do not change the agent's behavior enough to lead to significantly different rewards. Further increasing the noise, to counter these problems, would change SD into a random search method, which is not the intention of SD.

On the switch task, SD performed worse than expected, given that successful use was reported on another task [10]. In that task, an agent repeatedly reacts to a stimulus by pressing one of two buttons, one leading to a small or large positive reward, the other to a small or large loss. The problem is such that the optimal behavior is to take a number of small losses but receive a large reward later. In contrast taking a number of small rewards in the beginning results in a large loss in the end. After a number of trials, the rewards are changed and the agent has to switch from a 2-step planning situation to one where it must loose 7 times to get the large reward at step 8. On this task, they found a strong response to the switch and noticed the agent relearning the new scenario quickly, whereas our results do not show a clear response to a change in the enviroment. A possible explanation for this is that in Schweighofer and Doya's scenario, there are no walls or other none-goal states that give big negative rewards. The behavior the agent showed before the switch is the worst possible thing to do after the switch and thus exploration will be beneficial to the agent in terms of reward, i.e., exploration is encouraged. In contrast, in our task higher exploration also results in more bumping into a wall and can therefore be discouraged by SD.

SD seems to be more suitable for fine-tuning parameters in a stable environment, but its use as an adaptive control method is limited.

In the switch task, the β guided by BV improves learning performance, and reacts to the switch. It qualifies for a useful affect-based method for steering the β parameter. The opposite is true for the β in the Candy task. The method guides β only when the agent is learning to find the candy. After that, the β gets very noisy. The β fluctuates around its center value, only limited by the value range imposed on β. Granted, learning performance is better than the best static method, but not due to meaningfully balancing exploration and exploitation, as is the method's goal. As such, it does not qualify for a good adaptive method in Candy-like tasks. The main cause for the instability is the standard deviation of the long term average reward (σ_{ltar}), which is meant to normalize valence, but when the environment stabilizes, this term approaches 0. As a result, all minor differences between $\bar{\bar{r}}$ and $\bar{r}$ are magnified to extreme values. A possible solution to this problem is the introduction of some noise in the reward history, such that σ_{ltar} never approaches 0.

It is clear that the hybrid method does not provide the desired behavior. The influence of BV on SD's measurements is probably one of the main reasons for this. SD measures the influence of the added noise over a long period and then checks whether the added noise was beneficial to the agent's performance. With BV influencing the β and thus the performance within this period, SD's method cannot accurately measure the influence of the noise. The result is that the two methods are in each others' way instead of cooperating. Updating BV's β only when SD's β is updated (i.e., *not* influence β during SD's evaluation of its effect on reward) is not an option; it dramatically reduces BV's capacity to quickly react to changes in the environment.

Another important reason for the failing of the hybrid method is the incorrect assumption that BV stabilizes in a static environment. Because of large fluctuations of BV, the two reward histories keep having different reward distributions, and thus SD will never have much influence. It would be interesting to test the performance of the hybrid method with the addition of noise as describe above.

As Soar-RL agents attempt to maximize reward over time, our performance measure is the average reward the agent received per step: the agent's *Quality of Life (QoL)*. Nonetheless, it would be interesting to also measure the 'time to goal' to obtain more information on the agent's actual behavior.

We only focussed on regulating exploration/exploitation through the β parameter. The methods in this paper might also be used to control other RL parameters ([10] used their method to control α and γ as well). The χ^2 test in the hybrid method could be used to control, e.g., α.

Better hybrid methods might be constructed by merging a search strategy and a directional approach by another balancing algorithm (instead of χ^2). An alternative would be to let one strategy search the window in which parameters can vary, while the second determines the exact position within this window.

7 Conclusions

We conclude that Soar-RL agents can use artificial affect to directly control the amount of exploration [4] by coupling valence to the β in the Boltzmann distribution used in action-selection. Experiments show that such agents adapt better to a sudden change in the environment, as compared to agents that use an amount of exploration that is either static or determined by an affect-based search strategy (e.g., as used in [10]). These results are compatible with those presented in [4], indicating that this specific benefit of affective control of exploration is not tied to a particular learning architecture.

On the other hand, affect-based search enables convergence of this β to a (local) optimal value, while the directed affect-based control method converges to a "middle" β value in a range of values. This requires careful setting of the value range. The downside of the affect-based search method is that it cannot guarantee to escape a local optimum (as in the Candy task), and that it cannot react to sudden changes in the environment.

We conclude that the affect-based search method examined in this paper is suitable for fine-tuning parameters in static environments, but that its use as an adaptive control method is limited. We further conclude that directed exploration-control is suitable for fast reaction to changes in the environment, but has little use in static environments. Currently, our hybrid method that attempts to merge positive elements of both does not behave as intended.

Artificial affect can make a useful contribution to controlling an agent's exploration/exploitation, but there is much work to be done and a better understanding of the interplay between human learning and affect is needed.

Acknowledgments

We wish to thank S. Nason and J. Laird for providing us with the experimental Soar-RL software. This research is supported by the Netherlands Organization for Scientific Research (NWO, grant 612.066.408).

References

1. Damasio, A.R.: Descartes' error: Emotion, reason, and the human brain. Gosset/Putnam Press, New York (1994)
2. Craig, S.D., Graesser, A.C., Sullins, J., Gholson, B.: Affect and learning: An exploratory look into the role of affect in learning with autotutor. Journal of Educational Media **29** (2004) 241–250
3. Ashby, F.G., Isen, A.M., Turken, U.: A neuropsychological theory of positive affect and its influence on cognition. Psychological Review **106** (1999) 529–550
4. Broekens, J., Kosters, W.A., Verbeek, F.J.: On affect and self-adaptation: Potential benefits of valence-controlled action-selection, submitted to IWINAC'2007. (2007)
5. Newell, A.: Unified Theories of Cognition. Harvard University Press, Cambridge, Massachusetts (1990)
6. Nason, S., Laird, J.: Soar-RL, integrating reinforcement learning with Soar. Cognitive Systems Research **6** (2005) 51–59
7. Sutton, R., Barto, A.: Reinforcement learning, an introduction. MIT Press, Cambridge, Massachusetts (1998)
8. Broekens, J., Verbeek, F.J.: Simulation, emotion and information processing: Computational investigations of regulative role of pleasure in adaptive behaviour. In: Proceedings of the Workshop on Modeling Natural Action Selection, Edinburgh, UK (2005) 166–173
9. Doya, K.: Metalearning, neuromodulation and emotion. Affective Minds **1** (2000) 101–104
10. Schweighofer, N., Doya, K.: Meta-learning in reinforcement learning. Neural Networks **16** (2003) 5–9
11. Dreisbach, G., Goschke, T.: How positive affect modulates cognitive control: Reduced perseveration at the cost of increased distractibility. Journal of Experimental Psychology **30** (2004) 343–353
12. Marinier, R., Laird, J.: Towards a comprehensive computational model of emotion and feelings. In: Proceedings of the Sixth International Conference on Cognitive Modeling. (2004) 172–177
13. Belavkin, R.V.: On relation between emotion and entropy. In: Proceedings of the AISB'04 Symposium on Emotion, Cognition and Affective Computing, Leeds, UK (2004) 1–8
14. Hogewoning, E.: Strategies for affect-controlled action-selection in soar-rl. Technical Report 07-01, Leiden Institute of Advanced Computer Science, Leiden University (2007)