CHALMERS

Modulating Reinforcement-Learning Parameters Using Agent Emotions

Master's Thesis for the Intelligent Systems Design programme

Rickard von Haugwitz

Department of Applied Information Technology
Chalmers University of Technology
Gothenburg, Sweden 2012
Report No. 2012:091
ISSN: 1651-4769

Abstract

When faced with the problem of learning a strategy for social interaction in a multiagent environment, it is often difficult to satisfactorily define clear goals, and it might not be clear what would constitute a "good" course of action in most situations. In this case, by using a computational model of emotion to provide an intrinsic reward function, the task can be shifted to optimisation of emotional feedback, allowing more high-level goals to be defined. While of most interest in a general, not necessarily competitive, social setting on a continuing task, such a model can be better compared with more conventional reward functions on an episodic competitive task, where its benefit is not as readily visible.

A reinforcement-learning system based on the actor-critic model of temporal-difference learning was implemented using a fuzzy inference system functioning as a normalised radial-basis-function network capable of dynamically allocating computational units as needed and to adapt its features to the actual observed input. While adding some computational overhead, such a system requires less manual tuning by the programmer and is able to make better use of existing resources.

Tests were carried out on a small-scale multi-agent system with an initially hostile environment, with fixed learning parameters and separately with modulated parameters that were allowed to deviate from their base values depending on the emotional state of the agent. The latter approach was shown to give marginally better performance once the hostile elements were removed from the environment, indicating that emotion-modulated learning may lead to somewhat closer approximation of the optimal policy in a difficult environment by focusing learning on more useful input and increasing exploration when needed.

Acknowledgements

First of all I would like to extend my deepest thanks to Professor Kitamura Yoshifumi of the Interactive Contents Design lab at Tohoku University, who supervised the project, and to Claes Strannegård of Chalmers University of Technology and the University of Gothenburg, who examined this thesis and my presentation. Rong Rong, previously of Osaka University deserves special thanks for help on reimplementing the VAE system. Furthermore, I would like to thank Assistant Professor Takashima Kazuki and all other members - students and staff - of the ICD lab for assistance in various matters and for giving me a good time in Japan, and not least my opponents Mikael Bung and Mattis Jeppsson for useful feedback on the thesis.

Contents

Chapter 1

Introduction

An autonomous agent that has to accomplish some task in an environment is faced with the challenge of learning how to select, at each state, among the actions it has at its disposal. One form of machine learning concerned with this type of problem is *reinforcement learning*, which uses a trial-and-error approach to learn an optimal policy for action selection. The reinforcement learning problem, however, is complicated when extended to a multiagent environment, in which several agents individually are trying to optimise. In this case, better performance might be achieved by introducing a model of emotion and shifting the goal to optimisation of emotional feedback. Another use of emotion might be in a social environment, where it is difficult to define clear goals or what constitutes a "good" course of action in most cases. Here, emotion can provide an intrinsic reward function and allow for the definition of more high-level goals desirable in the case of social interaction.

1.1 Background

In reinforcement learning, the agent explores different courses of action and receives at each state transition a *reward*, a single scalar which is used to reinforce the choices leading to that state. It is the geometrically discounted estimated cumulative reward, given by the *value function*, that the agent is trying to maximise by adjusting its *policy* for action selection. Temporal-difference (TD) learning is the most popular class of reinforcement learning algorithms used today, with Q-learning being the most widely used. Other TD-learning variants include SARSA, TD(λ) and the actor-critic model.

Although the value function can be stored exactly as a table with one value per possible state, this leads to the *curse of dimensionality* as complexity increases exponentially with the dimensionality of the state set. To remedy this problem, some form of function approximation is required. Function approximation is usually carried out by performing gradient descent over a parameterisation of the value function, but one then faces the problem of

convergence to a local optimum, which might be quite far from the ideal global optimum. The most commonly used function approximation schemes include linear models such as coarse coding, CMAC or radial basis functions, and non-linear models like multilayer-perceptron artificial neural networks, decision trees and Bayesian networks.

Social interaction between agents can be particularly difficult to model, as there is no clear definition of what constitutes a desirable outcome, making it difficult to specify a reward function. In this case, it might be useful to employ some model of emotion to indirectly provide an intuitive measure of reward.

Several of the main signal substances responsible for modulating learning and controlling the reward system in the brain are also known to give rise to emotions, which more immediately modify behaviour. It is therefore clear that there exists a deeply rooted connection between emotion and learning in biological agents. In a reinforcement learning system aiming to model biological agents having emotions, emotion may be utilised to provide an intrinsic reward function, and having agents thus optimise emotional feedback one might hope to achieve realistic behaviour. However, the role of neuromodulators is not limited to reward, but includes controlling the rate of learning and forgetting, the time scale of state evaluation and the degree of exploration. Dynamic modulation of these parameters is of particular interest in non-stationary environments such as multi-agent systems, where state transition probabilities may change during the course of learning and the agent needs to continually adapt to changing conditions.

1.2 Objective

This thesis aims to evaluate the utility of modulating the learning parameters of a reinforcement learning algorithm using the agent's simulated emotions, in a small-scale multi-agent environment. The performance of such a system should be evaluated through fair comparison with the same algorithm using fixed parameters. Learning should be driven by emotional feedback and function in the presence of other agents and, preferably, human users.

1.3 Limitations

Despite being of considerable interest in multiagent reinforcement learning, due to time constraints and a wish to keep the model simple, game-theoretic approaches to multiagent systems have not been included in this report. Readers who wish to acquire a foundation on the subject are directed to Shoham and Leyton-Brown (2009) and Vidal (2010).

Artificial neural networks other than radial-basis-function networks are not treated in this work; the interested reader is recommended Rojas (1996).

While Wang et al. (2007) suggest using evolutionary algorithms for parameter tuning of the fuzzy radial-basis-function network, and such an algorithm was implemented and tried for this purpose (albeit with limited success), the subject was not deemed relevant enough for inclusion in the thesis. The theory of evolutionary algorithms is described in Yu and Gen (2010) and Weise (2009).

Although appraisal theory were not used in the final implementation of the system, models based on such theories were at an early stage considered for use, and the corresponding section included in the thesis to provide some insight into alternative models of emotion.

1.4 Previous Work

Reinforcement learning as a concept in psychology has a long history, but the computational theory in machine learning stems from early work in dynamic programming by Richard Bellman (Bellman, 1957b,a) on the one hand and Monte-Carlo methods, developed by a team of scientists including Stanisław Ulam, John von Neumann and Nicholas Metropolis during the work on nuclear weapons at Los Alamos in the 1940s, on the other.

Temporal-difference learning likewise has old roots, but should be attributed to Sutton (1988). Eligibility traces (TD(λ)) was introduced in the same work. Q-learning, probably the most popular version of TD-learning and reinforcement learning in general in use today, is due to Watkins (1989). The actor-critic architecture was introduced by Sutton (1984).

Function approximation with artificial neural networks of the multilayer perceptron type has been applied successfully to the reinforcement learning problem by several authors, notably by Tesauro (1992, 1994, 1995) for the backgammon-playing program *TD-Gammon* using Sutton's TD learning. Radial-basis-function networks seem to have been first introduced in Broomhead and Lowe (1988), although Moody and Darken (1989), wherein the normalised architecture was also first suggested, is more often cited as the source of this type of neural networks (to be fair it appears they were published independently of each other). In the following years, authors such as Poggio and Girosi (1989, 1990) advanced the theory of RBFNs. Using ideas from pioneering authors such as Platt (1991), Cheng et al. (2004) created an actor-critic normalised RBFN capable of allocating computational units as they are needed.

Fuzzy logic was developed by Zadeh (1973) and generalised to allow functions as conclusions by Takagi and Sugeno (1985). Takagi-Sugeno fuzzy inference systems were proven by Jang and Sun (1993) to be functionally equivalent, under some restrictions, to radial-basis-function networks. Wang et al. (2007) applied these results to the work of Cheng et al. (2004) to create a resource-allocating fuzzy inference system working as an NRBFN.

An algorithm for parameter and structure learning of neural-network fuzzy inference systems had previously been described by Lin and Lee (1994).

Emotion-derived reward functions have been used in reinforcement learning by a number of researchers. Malfaz and Salichs (2006) is a noteworthy example of the use of emotion in multi-agent reinforcement learning. Appraisal theory, in which much important work has been done by Scherer (see *e.g.* Scherer (2001)), was related to reinforcement learning by authors such as Marinier and Laird (2007); Marinier et al. (2008). Gratch and Marsella (2004); Marsella and Gratch (2009), although not working in reinforcement learning, developed and implemented computational models of emotion based on appraisal theory. Gadanho (1999); Gadanho and Hallam (2001); Gadanho (2003) explore the utility of emotion in a reinforcement-learning setting for robotic control.

Doya (2000, 2002), expanding on previous work by, among others, Montague et al. (1996) and Schultz et al. (1997), investigates the modulating effect of emotion on learning in the human brain, and attempts to relate the observed effects to reinforcement learning in the sphere of machine learning. He considers the TD error δ and what he terms *metaparameters*, referring to the learning rate α, the discount factor γ and the inverse Gibbs/Boltzmann temperature τ^{-1}, and how they are controlled in the brain by neuromodulators.

The exact relationships between the various neurotransmitters and emotions are not entirely understood, but a number of models exist; recently, Lövheim (2012) proposed a model where the three main monoamine neurotransmitters serotonin, dopamine and noradrenaline form the axes of a three-dimensional coordinate system, in which the eight basic emotions (following Tomkins and McCarter (1964)) fit into the corners of the resulting cube. While the validity of the model remains unverified as of yet, if valid it would allow direct extraction of emotions from given combinations of the monoamines (and possibly vice versa).

An attempt to implement the ideas expounded upon in Doya (2002), by mapping neuromodulators to single specific emotions and thus modulating the learning parameters in a reinforcement learning system using Q-learning, is described in Akiguchi and Maeda (2006).

The *Virtual Alter Egos/Video Agents* system, and its predecessors, is described in Asai et al. (2005, 2007); Kitamura et al. (2008); Rong et al. (2010) and Rong (2011), and the project web site can be found at `http://www.icd.riec.tohoku.ac.jp/project/vae/index.html`.

Chapter 2

Reinforcement Learning

The class of machine learning algorithms concerned with maximising some form of reward for an agent in an environment by learning how to best select among a set of actions is called *reinforcement learning*.

In the reinforcement learning problem, the agent acts upon an *enviroment* described by a set of discrete *states* S. The agent has at its disposal a set of *actions* A, which could be anything from voltages to apply to a motor to chess moves. As the agent selects an action at state s_t, the environment makes a transition to the next state s_{t+1} according to some, possibly stochastic, transition rules $S \to S$ depending in part on the agent's action. During this transition, the agent receives a *reward* $r \in \Re$, which it uses to update the action selection *policy* before the next state. Thus, the interface between the agent and the environment consists solely of a state and a reward provided by the environment and an action provided by the agent.

Methods for solving the RL problem are based on ideas from dynamic programming (see for example Bellman (1957a) or Barto et al. (1995)) and Monte-Carlo methods which, while directly applicable on reinforcement learning, are impractical for use on large problems. While the methods and algorithms presented herein should be understandable without in-depth knowledge of dynamic programming or Monte-Carlo methods, the interested reader is directed to Sutton and Barto (1998).

2.1 Markov Decision Processes

The environment in classical reinforcement learning is defined as a *Markov Decision Process* (MDP). An MDP is a reinforcement learning task that satisfies the *Markov property*; that the transition from any state to the next depends only on the current state and the agent's action, and not on any previous states or actions. Then, the state transition probability $\mathcal{P}^a_{ss'}$ from state s to state s' at time step t can be completely specified as

$$\mathcal{P}^a_{ss'} = \Pr\left\{ s_{t+1} = s' \,|\, s_t = s, a_t = a \right\}. \tag{2.1}$$

If $\mathcal{P}^a_{ss'}$ is given by the transition function $T(s, a, s')$, the MDP can be defined as consisting of an initial state s_1 drawn from the set of all states S, $T(s, a, s')$ and a reward function $r : S \to \Re$ (Vidal, 2010).

An extension of Markov chains, MDPs model processes in which the transition probabilities are in part stochastic (or at least appear stochastic to the system, which may or may not have knowledge about transition mechanics) and in part depend on the selected action. The action selection probabilities, in turn, depend on the rewards received on state transitions, as the system tries to maximise long-term reward[1]. In reinforcement learning, the rewards and transition probabilities of the stationary system are not known *a priori*, and the policy has to be updated incrementally based on sample observations of state transitions and received rewards.

In the field of reinforcement learning, MDPs are usually classified as either *episodic*, having one or more designated terminal states, or *continuing*, in which case the process never terminates naturally. This terminology is used in *e.g.* Sutton and Barto (1998), whereas in MDP literature the terms *finite-horizon*, *indefinite-horizon* and *infinite-horizon tasks* are more widely used. It should be sufficient to note that indefinite-horizon and infinite-horizon tasks correspond to episodic and continuing tasks, respectively[2], and are often used interchangeably in RL literature; for a more detailed treatment, see Sutton and Barto (1998, 3.11).

A Markov decision process extended to the multi-agent case is known as a *stochastic game*; conversely, an MDP is the special case of a one-player stochastic game (Bowling and Veloso, 2000).

2.2 The Value Function

The main problem in reinforcement learning is trying to maximise the so-called *state-value function*, defined as

$$V^\pi(s) = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s \right\}, \tag{2.2}$$

where $\gamma \in [0, 1]$ (or $[0, 1)$ for continuing tasks) is a discounting factor, assigning more weight to future states as it approaches 1. Thus, the state-value function denotes the expected cumulative discounted reward (the *return*) starting in state s and following the policy π. Under an optimal policy π^*, and assuming that the MDP has a finite number of states, the optimal value function, denoted V^*, is the unique solution to the MDP's Bellman equation (Sutton and Barto, 1998; Baird, 1995).

[1] Sometimes, as in for example Bellman (1957a), instead of maximising reward one is trying to minimise the cost, which is just negative reward.

[2] The third class, finite-horizon tasks, refers to tasks in which interaction terminates after a fixed number of steps, and is usually of little interest in reinforcement learning.

Another important function is the *action-value function*, giving the expected return for taking action a in state s and thereafter following the policy π. Equation (2.3) gives the action-value function defined in terms of the state-value function:

$$Q^{\pi}(s,a) = E\left\{ r_{t+1} + \gamma V^{\pi}(s_{t+1}) \middle| \, s_t = s \right\} \tag{2.3}$$

Reinforcement learning methods that aim to estimate the optimal policy by directly approximating the optimal action-value function Q^* are called *control* methods, as opposed to *prediction* methods concerned with estimating the state-value function V^{π} for a given policy π (Sutton and Barto, 1998).

2.3 The Policy

The agent stores a mapping from each state-action pair $\{s, a\}$ to the probability $\pi(s,a)$ of taking action a when in state s known as the *policy*. This policy can be implemented in various ways, and is often stochastic. In the simplest case, called the *greedy policy*, the agent simply selects the action with the highest value at each state. However, this is not as good an idea as it might seem at first glance, as it does not allow the agent to try new actions with apparently low value that could actually lead to better unexplored states. Finding a good balance between exploiting the current knowledge of the environment and exploring new states to gain additional knowledge is a difficult and important problem in on-line reinforcement learning, and is known as the *exploration vs. exploitation dilemma*.

2.3.1 Exploration vs. exploitation

Several methods have been proposed to handle the exploration vs. exploitation problem, but the most widely used ones are ϵ-greedy and Gibbs, or Boltzmann, softmax action selection. Using ϵ-greedy action selection, with probability ϵ a random action is chosen, else the greedy action is chosen. As learning progresses and the agent gains more accurate knowledge of the environment, ϵ should preferably be gradually decreased (Sutton and Barto, 1998). A Gibbs softmax policy selects actions using equation (2.4):

$$\pi(s,a) = \Pr(a|s) = \frac{e^{Q(s,a)/\tau}}{\sum_{b \in A(s)} e^{Q(s,b)/\tau}}, \tag{2.4}$$

where τ is the *temperature*, a parameter specifying the amount of exploration the agent should do. This scheme assigns higher selection probabilities to actions with higher estimated value, and there is always a small chance of even a very lowly valued action being selected. A higher temperature means the difference in selection probabilities decrease, approaching equiprobable selection for all actions in the limit. As with ϵ in ϵ-greedy selection, it is often

a good idea to decrease the temperature parameter over time to maximise reward.

2.4 Temporal-Difference Learning

In many cases it might be undesirable to wait until the end of the episode before performing an update of the value function; for example, one might want the agent to learn while interacting with the environment, or the task might not be episodic at all, but continuing. In such a task, off-line learning would not make sense, as the agent would continue on forever, or at least until terminated, without reaching a natural terminal state where it could backup. In these cases, one has to turn to *on-line* learning algorithms, which perform smaller updates after every state transition based on the current knowledge of the environment. Such on-line algorithms make use of a method called *temporal-difference learning*. The ideas behind temporal-difference, or TD, learning are old, but the method was formalised and thus named by Sutton (1988).

The main underlying innovation in TD learning is the use of a quantity called the *TD error*, the difference between the expected reward based on the current value function and the actual received reward. At every time step, then, the value function is updated in the direction of the TD error towards the actual observation. The TD error δ_t for time step t is defined as

$$\delta_t = r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t), \tag{2.5}$$

where $V_t(s)$ is the value function at time t and γ the usual discounting factor. Then, in the simplest TD method, called TD(0) and introduced in Sutton (1988), the value function is updated according to the following equation:

$$V_{t+1}(s) = V_t(s) + \alpha\delta_t, \tag{2.6}$$

where $\alpha \in (0, 1)$ is a step-size parameter called the *learning rate*.

Note that these updates can be saved and performed in batches, such as in the Least-Squares TD method presented in Bradtke and Barto (1996), or all at once at the end of the episode, in which case one gets an off-line algorithm (Sutton and Barto, 1998).

From the above, the TD-learning algorithm (after Sutton and Barto (1998)) is given by algorithm 2.1.

2.4.1 Q-learning

One of the most popular and important TD-learning approaches is so-called Q-learning, first introduced by Watkins (1989). Q-learning attempts to directly learn the optimal action-value function $Q^*(s, a)$, using the update rule

Algorithm 2.1 TD Learning

Initialise $V(s)$ arbitrarily
for each episode **do**
  $s \leftarrow$ initial state
  repeat {for each step of episode}
    $a \leftarrow$ action selected by π for s
    Execute action a; observe reward r and next state s'
    $\delta \leftarrow r + \gamma V(s') - V(s)$
    $V(s) \leftarrow V(s) + \alpha\delta$
    $s \leftarrow s'$
  until s is terminal
end for

(for *one-step Q-learning*)

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]. \quad (2.7)$$

Since the update is independent of the policy, Q-learning is termed an *off-policy* TD control algorithm. This means that one can follow a policy designed to ensure a good level of exploration while still updating towards the optimal policy.

Algorithm 2.1 adapted for Q-learning is given by algorithm 2.2, also after Sutton and Barto (1998).

Algorithm 2.2 Q-Learning

Initialise $Q(s, a)$ arbitrarily
for each episode **do**
  $s \leftarrow$ initial state
  repeat {for each step of episode}
    $a \leftarrow$ action selected by policy derived from Q
    Execute action a; observe reward r and next state s'
    $Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$
    $s \leftarrow s'$
  until s is terminal
end for

2.4.2 Actor-Critic Methods

Learning the action-value function, the agent can implicitly derive a greedy policy by selecting the action with the highest Q-value at each state. This approach has several benefits, such as being easy to combine with function approximation (chapter 3). Another approach, introduced in Sutton (1984), is the *actor-critic* architecture, which explicitly stores the policy in a separate

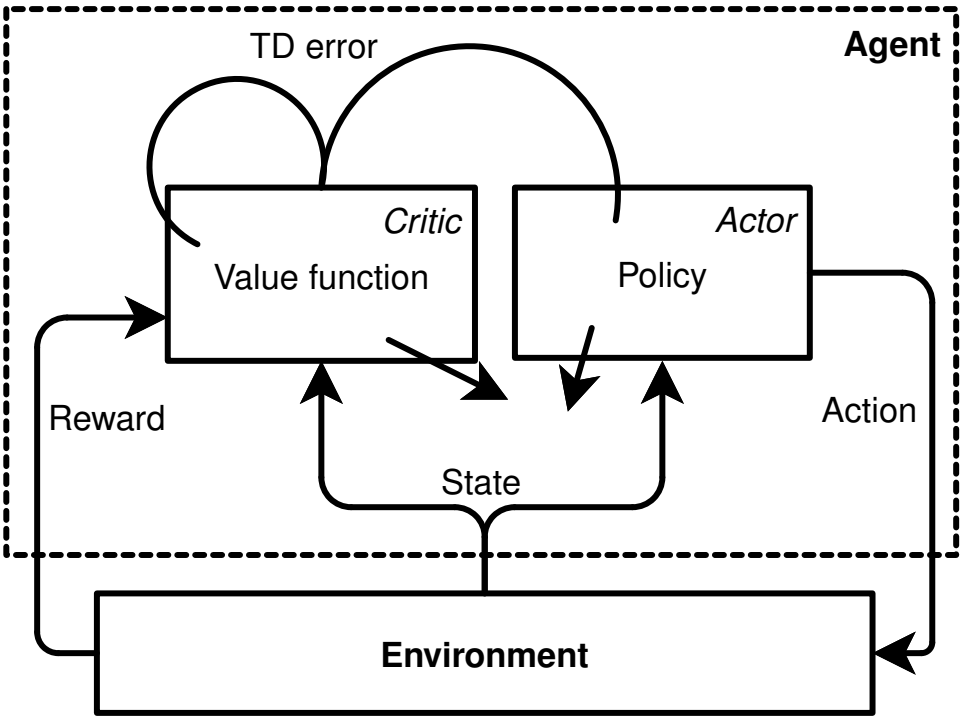

Figure 2.1: The Actor-Critic architecture

data structure, called the *actor*. The actions chosen by the actor is then criticised by the value function, known as the *critic*. Critique is in the form of the TD error (2.5); if positive, the action was a good one, and its selection probability should be strengthened. Conversely, the probability of selection should be weakened if the TD error is negative.

The policy can, for example, be stored as a vector of *preferences* $P(s, a)$ for selecting action a when in state s, and action selection probabilities being generated using a form of the Gibbs softmax method (equation (2.4)), replacing the Q-values with the preferences:

$$\pi(s, a) = \Pr(a|s) = \frac{e^{P(s,a)/\tau}}{\sum_{b \in A(s)} e^{P(s,b)/\tau}}, \tag{2.8}$$

Since the policy is stored explicitly, the actor-critic architecture allows for example an explicitly stochastic policy to be stored. This is useful in competitive multiagent environments, where in a given state there might not be a single best action, but rather a best probability distribution over available actions, in game theory known as an optimal *mixed strategy* (Shoham and Leyton-Brown, 2009).

The actor-critic architecture is illustrated in figure 2.1.

2.4.3 Eligibility Traces

Since TD-learning algorithms update their predictions incrementally based on the most recent observations in a bootstrapping procedure, the value predictions for recently visited states are likely to be more accurate than those for more distant states. A way of making use of the latest knowledge to backup previous states instead of waiting until they are encountered again is *eligibility traces*. The idea is that the values for previously visited states are *eligible* for alteration such that value predictions of more recently visited states are changed more, using the weighting λ^k (for $\lambda \in [0, 1]$) for an update of a state visited k steps in the past (Sutton, 1988). To denote the use of eligibility traces, therefore, such algorithms are called *e.g.* TD(λ) or $Q(\lambda)$.

In the case of TD(λ), due to Sutton (1988), the eligibility traces are updated as

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) + 1 & \text{if } s = s_t \text{ and} \\ \gamma\lambda e_{t-1}(s) & \text{otherwise.} \end{cases} \tag{2.9}$$

This gives the update rule

$$\forall s \in S, V_{t+1}(s) = V_t(s) + \alpha\delta_t e_t(s) \tag{2.10}$$

Note that $\lambda = 0$ gives exactly the update in (2.6). Extending algorithm 2.1 to include the use of eligibility traces produces algorithm 2.3.

Algorithm 2.3 TD(λ)

```
Initialise V(s) arbitrarily
e(s) ← 0, ∀s ∈ S
for each episode do
  s ← initial state
  repeat {for each step of episode}
    a ← action selected by π for s
    Execute action a; observe reward r and next state s'
    δ ← r + γV(s') − V(s)
    e(s) ← e(s) + 1
    for all s do
      V(s) ← V(s) + αδe(s)
      e(s) ← γλe(s)
    end for
    s ← s'
  until s is terminal
end for
```

Although equation (2.10) requires value estimates and eligibility traces to be updated for every state on every time step, it is clear that such an

implementation would not be feasible for any realistic application on a serial computer. However, as noted by Sutton and Barto (1998), it is not necessary to perform updates for all states, as almost all states will have eligibility traces very close to zero. Those states, then, would not need to be updated, saving considerable computation time. Furthermore, using function approximation, the added complexity from eligibility traces becomes significantly smaller.

There exist several variations on the update given in equation (2.9); for example, Singh and Sutton (1996) suggest using so-called *replacing traces*, which instead of incrementing the trace by 1 when a state is encountered again sets it to 1.

Finally, it might be worth noting that eligibility traces, by introducing a memory effect, can interfere with learning in non-stationary systems such as multi-agent systems, where transition probabilities might have changed the next time a state is encountered (Geist and Pietquin, 2010).

2.5 Parameter Modulation

Tuning the learning rate, discount factor and, when using softmax action selection, the temperature, is often done manually by the programmer specifically for the application at hand. In many environments, however, it is desirable to let some of the parameters change as learning progresses. As will be discussed in section 3.1, approximations based on gradient descent will converge slowly for small values of α, while high values will cause oscillation around the optimum. Therefore, the learning rate should be relatively high at the onset of learning in order to quickly bring the approximation close to the optimum, but approach zero in the limit to assure convergence; in fact, convergence proofs are predicated on the latter condition being satisfied. In particular, the convergence proof for stochastic approximation in Robbins and Monro (1951) requires

$$\sum_{k=1}^{\infty} \alpha_k(a) = \infty \quad \text{and} \quad \sum_{k=1}^{\infty} \alpha_k^2(a) < \infty, \tag{2.11}$$

where $\alpha_k(a)$ denotes the learning rate used following the kth selection of action a, following the presentation in Sutton and Barto (1998). However, it should be noted that algorithms fulfilling condition 2.11 are seldom used in practice, as they assume a stationary environment; an assumption that rarely holds in real systems.

As explained in section 2.3.1, whether ϵ-greedy or Gibbs softmax action selection is used, it is necessary to decrease the relevant parameter in order for learning to converge. High randomness in selecting actions is useful for exploring unknown courses of action, while in later stages of learning it interferes with exploitation of gained knowledge.

Commonly, simple heuristics are employed to have the temperature and sometimes also learning rate parameters decay over time. Although such algorithms usually suffice for stationary environments, they will cause problems if applied to non-stationary environments, where state transition probabilities can change during learning. However, also in non-stationary environments there may be benefit in adapting the parameters to the dynamics of the environment. For example, if the value function is found likely to be inaccurate, learning rate should be increased to compensate; conversely, if predictions are good, learning rate can be gradually decreased to further improve the approximation of the value function. An algorithm that learns parameters of an underlying learning algorithm is termed a *meta-learning* algorithm.

The Incremental Delta-Bar-Delta (IDBD) algorithm (Sutton, 1992) finds the optimal learning rate for a linear base learning system through on-line gradient descent. Ishii et al. (2002) describe a method for modulating the temperature parameter by using a model-based RL method where the environment is estimated using Bayes inference. A more general approach for control of any learning parameter is presented in Schweighofer and Doya (2003), who expand on the Stochastic Real Value Units (SRV) reinforcement learning algorithm by Gullapalli (1990) to achieve meta-learning with stochastic gradients.

Chapter 3

Function Approximation

For small toy problems, the value function can be stored explicitly in a table of values for each state. This is known as the *tabular* case, and allows convergence to the true value function. However, it is obvious that the potential size of the table will increase enormously with the dimensionality of the state space, due to the *curse of dimensionality* (Bellman, 1957b; Sutton and Barto, 1998). Therefore, in order to decrease memory usage and computation time, some form of function approximation is usually desired.

An approximated value function (being, after all, an approximation) does not generally compute the exact value of a state or state-action pair; while potentially affecting the accuracy of convergence (see Sutton (1996) for a discussion and some reassuring results), this is not necessarily detrimental to learning speed. Rather, using value function approximation can speed up learning as one gets some degree of generalisation between similar points in state space, allowing the system to generalise from and make use of previous knowledge of the environment.

3.1 Gradient Descent

For a parameter vector $\boldsymbol{\theta}$ and a (continuous, differentiable) function $f(\boldsymbol{\theta})$, the *gradient* of f with respect to $\boldsymbol{\theta}$ is defined as the vector of partial derivatives

$$\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) = \left[\frac{\delta f(\boldsymbol{\theta})}{\delta \theta_1}, \frac{\delta f(\boldsymbol{\theta})}{\delta \theta_2}, \ldots, \frac{\delta f(\boldsymbol{\theta})}{\delta \theta_n} \right]^T . \tag{3.1}$$

The gradient is thus a vector field that in any point points in the direction in which f increases fastest with length proportional to the rate of increase, as illustrated in figure 3.1. Then, *gradient descent* is performed by going in the direction of the negative gradient with some small step size α:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \nabla_{\boldsymbol{\theta}_t} f(\boldsymbol{\theta}) \tag{3.2}$$

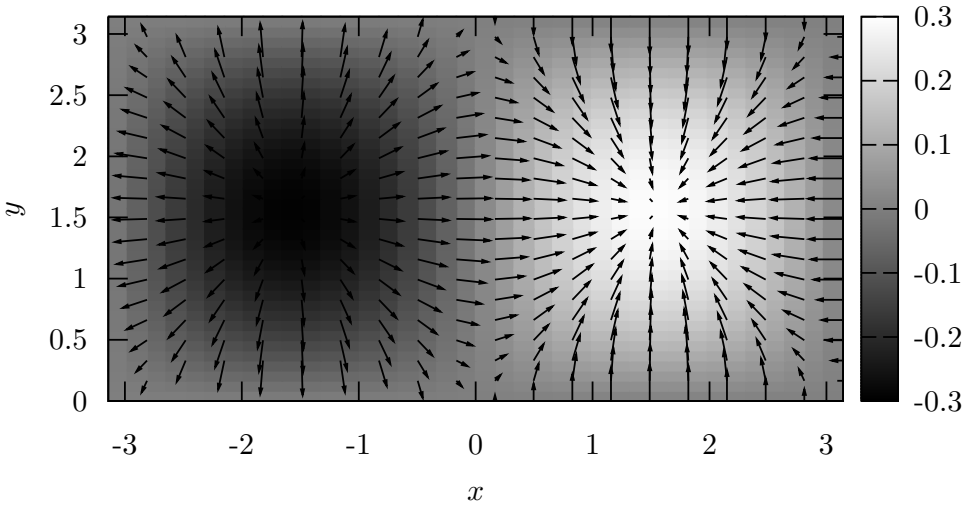

Figure 3.1: The gradient of the function $f(x, y) = 0.3 \sin x \sin y$ as a vector field plotted over a density map of f.

α should be set to a value less than 1 in order to guarantee convergence, as adjusting the parameter vector by the full length of the gradient will lead to oscillation around the optimum without converging to a single point. Still, the step size should not be set too low, or learning will take too long time. (Rojas, 1996, ch. 7-8) shows examples of oscillation around the optimum and discusses methods to increase learning rate while still keeping good bounds on convergence in the chapters on backpropagation, a popular learning algorithm based on gradient descent for artificial neural networks[1].

If, instead of minimising, one wants to maximise f, the parameter vector is adjusted in the positive direction of the gradient. The method is then called *gradient ascent*.

In reinforcement learning, the objective function is usually the mean squared error of the parameters, defined as

$$\mathrm{MSE}(\boldsymbol{\theta}_t) = \frac{1}{2} \sum_{s \in S} P(s) \left[V^{\pi}(s) - V_{\boldsymbol{\theta}_t}(s) \right]^2 . \tag{3.3}$$

Here, P is a distribution over the state set used to produce a weighted average of the errors. As Sutton and Barto (1998) explains, this distribution is important as one can generally not hope to reduce the error to zero for all states. Usually, this distribution is the *on-policy distribution* obtained by using on-policy sampling of the state space, meaning the states actually visited following the policy π up to time step t are used.

[1] More accurately, backpropagation is a way of computing the gradient of the network weights by running the network "backwards", and learning is then achieved through gradient descent.

However, since the true value function V^π is not known beforehand, one usually has to use some estimate of it. For the example of TD-learning, equation (3.2) becomes

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha\delta_t \nabla_{\boldsymbol{\theta}_t} V_t(s_t),$$

giving algorithm 3.1, once again adapted from Sutton and Barto (1998).

Algorithm 3.1 Gradient-Descent TD(λ)

```
Initialise θ arbitrarily
for each episode do
  e ← 0
  s ← initial state
  repeat {for each step of episode}
    a ← action selected by π for s
    Execute action a; observe reward r and next state s'
    δ ← r + γV(s') − V(s)
    e ← γλe + ∇_θ V(s)
    θ ← θ + αδe
    s ← s'
  until s is terminal
end for
```

3.1.1 Global and Local Optima

The ideal goal of any function approximator is to find a *global optimum*, defined by Sutton and Barto (1998) as a parameter vector $\boldsymbol{\theta}^*$ such that $\forall\boldsymbol{\theta}, \text{MSE}(\boldsymbol{\theta}^*) \leq \text{MSE}(\boldsymbol{\theta})$. However, in most cases, it is too optimistic to expect to find a global optimum, and the best one can hope for without searching the entire parameter space is to converge to a *local optimum*. A locally optimal parameter vector $\boldsymbol{\theta}^*$ fulfils $\text{MSE}(\boldsymbol{\theta}^*) \leq \text{MSE}(\boldsymbol{\theta})$ for all $\boldsymbol{\theta}$ for some δ such that $\|\boldsymbol{\theta} - \boldsymbol{\theta}^*\| \leq \delta$. As a local optimum can be quite far from the global optimum (or, possibly, optima), a guarantee merely of convergence to a local optimum is usually of little comfort. Fortunately, this is not a problem when using linear function approximation, treated in the following section.

3.2 Linear Methods

A method for function approximation is said to be linear, or linear in the parameters, if the approximate function can be written as a linear combination of the parameters. If the approximate value function V is a linear function of the parameter vector $\boldsymbol{\theta}$, one gets

$$V_t(s) = \boldsymbol{\theta}_t \cdot \boldsymbol{\phi}_s = \sum_{i=1}^{n} \theta_{ti}\phi_{si},$$

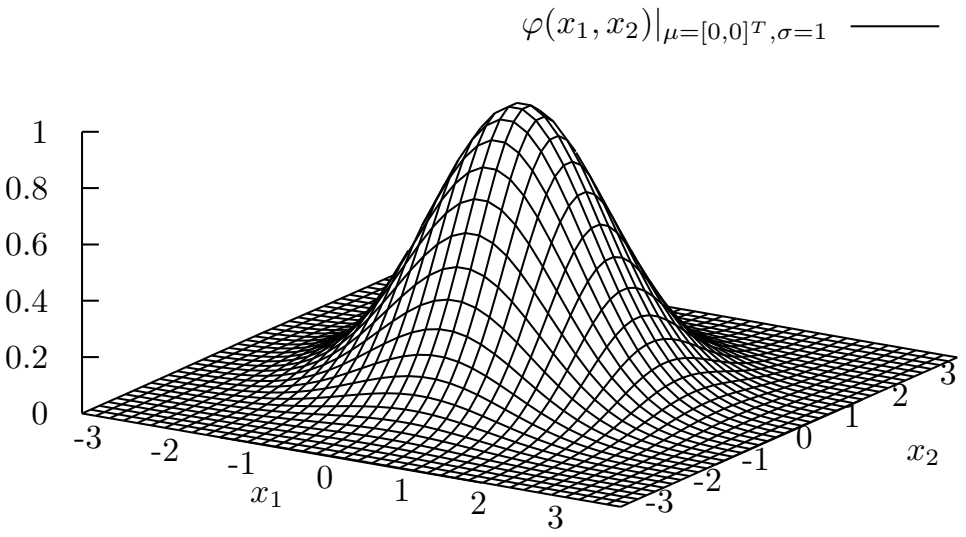

Figure 3.2: The shape of the Gaussian radial basis function in two dimensions centred at $[0, 0]^T$ with unit width.

where $\boldsymbol{\phi}_s$ is a feature vector constructed from the state s by some method for function approximation using a set of basis functions $\phi_1(s), \phi_2(s), \ldots, \phi_n(s)$ (Doya, 2002; Sutton and Barto, 1998). In this case, the gradient can be written as

$$\nabla_{\boldsymbol{\theta}_t} V_t(s) = \boldsymbol{\phi}_s.$$

A major advantage of linear approximation is that there exists only one optimum, or at least a set of equally good optima, avoiding the problem of convergence to local optima (Sutton and Barto, 1998).

3.2.1 Radial-Basis-Function Networks

Hartman et al. (1990) define radial basis functions as the set of functions $\varphi : \Re^n \to \Re, \varphi(\boldsymbol{x}) = \Phi(\|\boldsymbol{x} - \boldsymbol{\mu}\|)$, where $\boldsymbol{\mu}$ is the *centre* of the radial basis function in input space. In words, any function of the distance between the input vector and the centre vector is a radial basis function.

For the common case of a Gaussian basis function, the *activation* in the point $\boldsymbol{x}$ is given by

$$\varphi(\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{\mu}\|^2}{2\sigma}\right), \tag{3.4}$$

where σ is the *width* of the radial basis function (Jang and Sun, 1993). The shape of this function in two dimensions is shown in figure 3.2.

The first use of radial basis functions with the semantic meaning of artificial neural networks for function approximation is variously ascribed to

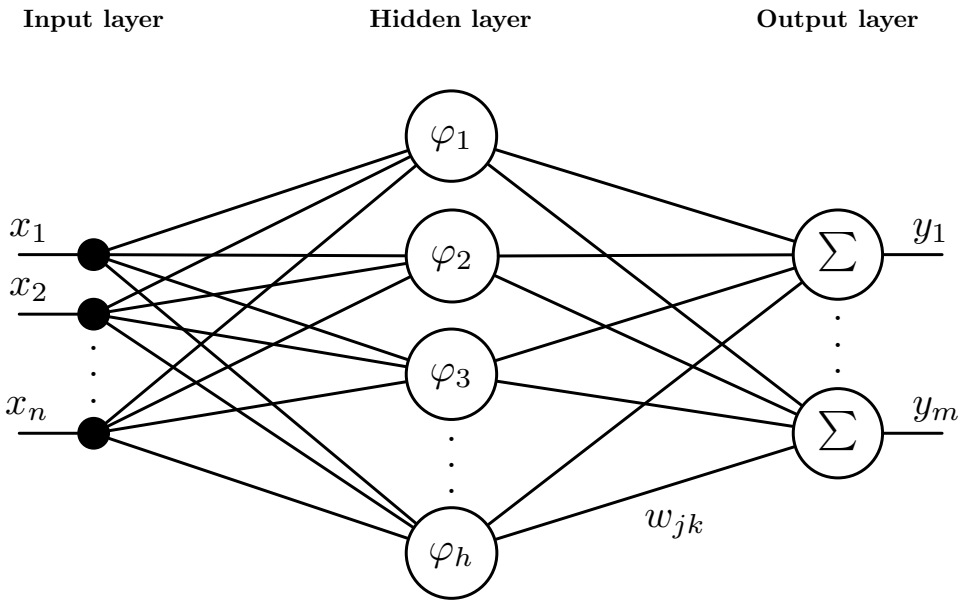

Figure 3.3: The topology of a radial-basis-function network.

Broomhead and Lowe (1988) and Moody and Darken (1989), and the theory of such *radial-basis-function networks* (RBFN) were developed in the following years by authors such as Poggio and Girosi (1989, 1990). They have since become one of the most important classes of ANN for function approximation, much due to the fact that they learn fast and are linear in the parameters, allowing guarantees to be made on convergence to a global optimum.

The structure of an RBF network is shown in figure 3.3; the network is layered with a single hidden layer, containing h computational units. Each hidden unit j computes its output for an input vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T$ using a radial basis function $\varphi_j(\boldsymbol{x})$. The output from the hidden layer is linearly combined at each output unit y_k with weights w_{jk}:

$$y_k = \sum_{j=1}^{h} w_{jk}\varphi_j(\boldsymbol{x}). \tag{3.5}$$

RBF networks with one hidden layer have been shown by Park and Sandberg (1991) to be capable of universal function approximation on a compact subset of $\Re^n$. At about the same time, Hartman et al. (1990) proved universal approximation capability of RBF networks with a single layer of Gaussian units, the most common form of RBFN. Lippman (1989), in a review of different types of neural networks for pattern classification, reports that RBF classifiers typically learn a lot faster than classifier networks using backpropagation, while achieving similar error rates at the cost of just a few times as many connection weights.

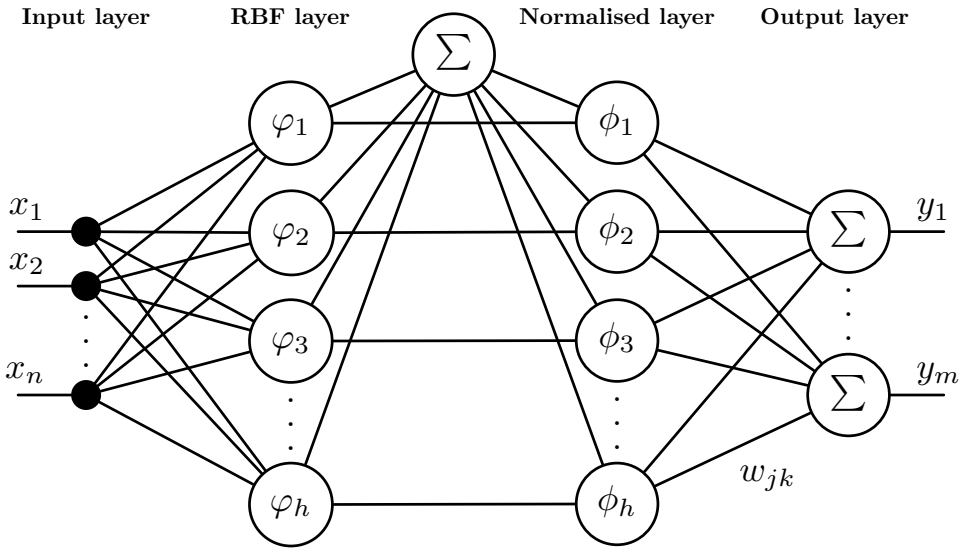

Figure 3.4: The topology of a normalised radial-basis-function network.

Normalised RBFN

The output of each hidden unit in a normalised radial-basis-function network is described by

$$\phi_j(\boldsymbol{x}) = \frac{\varphi_j(\boldsymbol{x})}{\sum_{j'=1}^{h} \varphi_{j'}(\boldsymbol{x})}, \tag{3.6}$$

as illustrated in figure 3.4. Note that $\forall \boldsymbol{x}, \sum_{j=1}^{h} \phi_j(\boldsymbol{x}) = 1$. Similarly to equation (3.5), the output at unit y_k in the output layer is given by

$$y_k = \sum_{j=1}^{h} w_{jk} \phi_j(\boldsymbol{x}). \tag{3.7}$$

The shape of two normalised radial basis functions for a single input dimension can be seen in figure 3.5.

Normalisation of radial basis functions was suggested by Moody and Darken (1989), and normalised RBFN were shown by Benaim (1994) to be satisfactorily capable of function approximation. The main reason for normalisation, as given in Shorten and Murray-Smith (1994, 1996) is that it results in a *partition of unity* across the input space, making the RBFN less sensitive to poor selection of the centre and width parameters as the basis functions sum to unity at every point in input space, thus equally covering all points. Wang et al. (2007) also note that normalisation gives better interposition performance at points located in overlapping receptive fields of the basis functions. However, Shorten and Murray-Smith (1994,

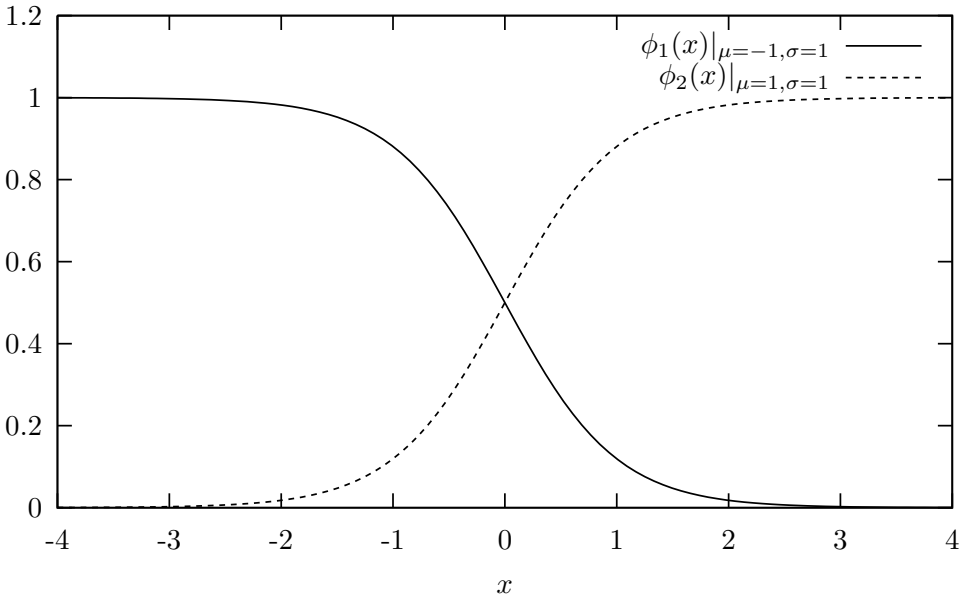

Figure 3.5: Two normalised Gaussian radial basis functions in one input dimension.

1996) go on to warn that normalising non-compact basis functions leads to certain side effects such as stability problems at the edges of the input space, shifting of unit maxima away from the centres (visible in figure 3.6) and reactivation of units far away from their centres, possibly leading to unpredictable behaviour.

Fuzzy RBFN

The functional equivalence of Gaussian radial-basis-function networks and Takagi-Sugeno fuzzy inference systems (due to Takagi and Sugeno (1985); fuzzy logic is due to Zadeh (1973)) was proven by Jang and Sun (1993), whose results were later generalised and expanded upon by Hunt et al. (1996). Rules in a Takagi-Sugeno model are of the form

$$R_j : \text{if } x_1 \text{ is } A_{j1} \wedge x_2 \text{ is } A_{j2} \wedge \ldots \wedge x_n \text{ is } A_{jn} \text{ then } y_j = f_j(x_1, x_2, \ldots, x_n),$$

where $f_j(x_1, x_2, \ldots, x_n)$ is usually a linear function, so that $y_j = a_{j0} + a_{j1}x_1 + \ldots + a_{jn}x_n$.

Inference is performed by first computing the truth value, or *firing strength* $|y = y_j|$ of the premise $y = y_j$:

$$|y = y_j| = \mu_{A_{j1}}(x_1) \wedge \mu_{A_{j2}}(x_2) \wedge \ldots \wedge \mu_{A_{jn}}(x_n),$$

where μ_{A_i} is the membership function of the fuzzy set A_i and $\wedge$ is a t-norm (*triangular norm*) fuzzy conjunction operator, usually multiplication or the

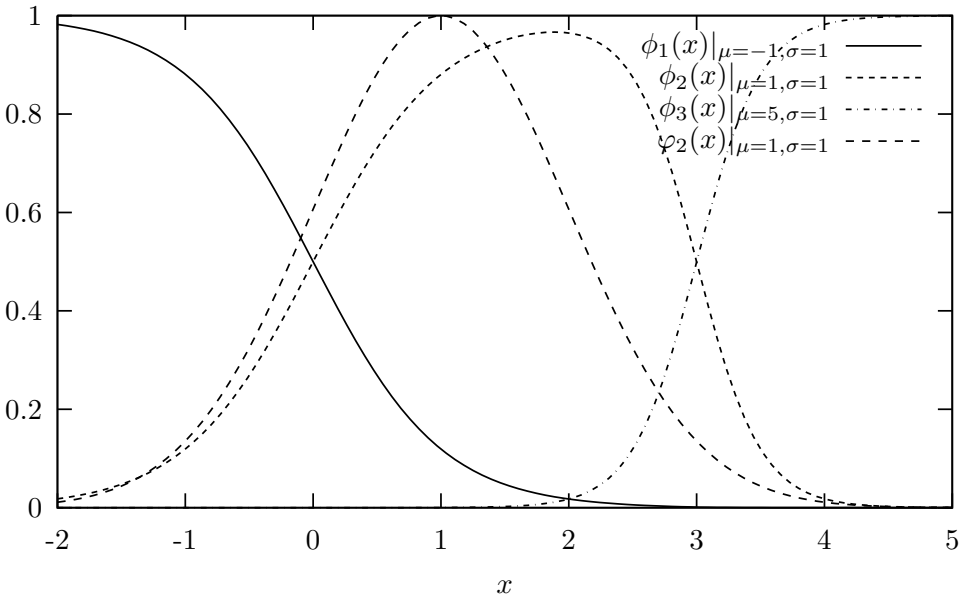

Figure 3.6: An example of the side effects of normalisation of radial basis functions. Note how the functions have been distorted, and the maximum of ϕ_2 has been shifted from the centre, compared to the corresponding unnormalised function φ_2.

min operator such that

$$\mu_{A_i}(x_i) \wedge \mu_{A_{i'}}(x_{i'}) = \begin{cases} \mu_{A_i}(x_i)\mu_{A_{i'}}(x_{i'}) & \text{or} \\ \min(\mu_{A_i}(x_i), \mu_{A_{i'}}(x_{i'})). \end{cases}$$

While Takagi and Sugeno (1985) use the min conjunction operator, in order to achieve functional equivalence with radial-basis-function networks, the multiplication operator must be used (Jang and Sun, 1993; Hunt et al., 1996). The full list of conditions given in Hunt et al. (1996) for functional equivalence between Gaussian RBF networks (including a more generalised class of networks than the model presented here) and the T-S fuzzy inference model is:

1. The number of RBF units is equal to the number of fuzzy if-then rules.
2. Gaussian basis functions are used as membership function within each rule.
3. The multiplication operator is used to compute each rule's firing strength.
4. The same method (normalised or unnormalised calculation) is used in both the RBF network and the fuzzy inference system to compute their overall output.

Hence, the ith membership function of the jth rule in a fuzzy RBF network is described by equation (3.8). To avoid confusion with the centre $\boldsymbol{\mu}$ and its elements, the membership function $\mu(x)$ has been relabelled $\mathrm{MF}(x)$, following the example of Wang et al. (2007).

$$\mathrm{MF}_{ij}(x_i) = \exp\left(-\frac{(x_i - \mu_{ij})^2}{2\sigma_{ij}^2}\right) \tag{3.8}$$

Then, the output of unit j is given by

$$\varphi_j(\boldsymbol{x}) = \prod_{i=1}^{n} \mathrm{MF}_{ij}(x_i) = \exp\left(-\sum_{i=1}^{n} \frac{(x_i - \mu_{ij})^2}{2\sigma_{ij}^2}\right). \tag{3.9}$$

To compute the overall output of the network, equation (3.5) is used like in a normal RBF network, or the network can be normalised using equation (3.6) and (3.7).

Chapter 4

Modelling Emotion

Many computational models of emotion have been proposed, although due to the difficulty of conducting experiments to verify them and the problems of relating such a complex phenomenon as emotions to a set of functions, real-valued or otherwise, the field is still far from maturity. In the field of reinforcement learning, Malfaz and Salichs (2006) apply a categorical model, keeping a set of basic emotional and physical parameters which are computed from the agent's observations of the environment, to agents in a multiagent setting to provide an intrinsic reward function, while on the other hand Marinier and Laird (2007); Marinier et al. (2008) base their work on appraisal theory. Both approaches will be expounded upon in the following sections.

4.1 Categorical Models

In computational models of emotion, categorical approaches that define a set of discrete emotions that each can vary in intensity are commonly employed. Tomkins and McCarter (1964) identifies eight basic *affects*, defined by the authors as the outer bodily responses, primarily facial behaviours, to emotions. These eight primary affects, as given in Tomkins and McCarter (1964), are recounted below. Each affect has been named also at high intensity, so Interest, for example, at high intensity is labelled Excitement.

- Interest - Excitement
- Enjoyment - Joy
- Surprise - Startle
- Distress - Anguish
- Fear - Terror
- Shame - Humiliation

- Contempt - Disgust
- Anger - Rage

Ekman and Friesen (1969) later narrowed the above list of primary affects down to a suggested seven:

- Happiness
- Surprise
- Fear
- Sadness
- Anger
- Disgust
- Interest.

Although Ekman and Friesen (1969) note that their list might be incomplete, and Ekman (1999) proposes a revision and expansion to fifteen basic "emotion families", it is still the above list that is most commonly used in computational models of emotion.

4.2 Appraisal Models

Appraisal theories of emotion posit that emotion emerges as a result of a person's *appraisal* of a situation rather than as a function of the situation itself. This process of assessing a situation takes into account future goals as well as physiological response and personality traits.

Computational models of emotion based on appraisal theory make use of some set of so-called *appraisal dimensions* (also called *appraisal variables*), computed for each event (or for events with an absolute utility larger than a small threshold value, as suggested by Gratch and Marsella (2004)) and stored in a data structure known as an *appraisal frame*. For the current state, an *emotion* frame is computed, but since emotion can fluctuate wildly depending on the state, it is combined with a moving history of emotion termed *mood* to create a *feeling* frame, which is what the agent actually "feels". Mood is "pulled" towards the current emotion, and at the same time decays by a small factor at each transition, so that were there no emotion, all dimensions in the mood frame would eventually approach zero (neutral).

Marinier and Laird (2007); Marinier et al. (2008), basing their work on Scherer (2001), use a set of eleven appraisal dimensions, listed in table 4.1.

Dimension	Range
Suddenness	$[0, 1]$
Relevance	$[0, 1]$
Intrinsic pleasantness	$[-1, 1]$
Conduciveness	$[-1, 1]$
Control	$[0, 1]$
Power	$[0, 1]$
Unpredictability	$[0, 1]$
Discrepancy from expectation	$[0, 1]$
Outcome probability	$[0, 1]$
Causal agent	$\{Self, Other, Environment\}$
Causal motive	$\{Intentional, Negligence, Chance\}$

Table 4.1: Appraisal dimensions

To combine two appraisal dimensions v_1 and v_2, Marinier and Laird (2007) use the following function, developed from a similar function for combination of intensity values proposed by Neal Reilly (2006):

$$C(v_1, v_2) = 0.1\text{Sign}(S)\log_b(|S + \text{Sign}(S)|), \tag{4.1}$$

where

$$S = \sum_{i=1}^{2} \text{Sign}(v_i)(b^{10|v_1|} - 1),$$

$$\text{Sign}(v) = \begin{cases} 1 & \text{if } v \geq 0 \text{ and} \\ -1 & \text{otherwise} \end{cases},$$

$$\text{and } b = \begin{cases} e & \text{if Sign}(v_1) = \text{Sign}(v_2) \text{ and} \\ 1.1 & \text{otherwise.} \end{cases}$$

The intensity I of an appraisal frame, using the dimensions given in table 4.1, is then computed using the function

$$I = [(1 - OP)(1 - DE) + (OP \cdot DE)] \cdot \frac{S + UP + \frac{|IP|}{2} + GR + Cond + \frac{|Ctrl|}{2} + \frac{|P|}{2}}{7},$$

where $[(1 - OP)(1 - DE) + (OP \cdot DE)]$ is called a "surprise term" by Marinier and Laird, and the remaining term is an average of the appraisals.

4.2.1 Emotion as Reward

For emotion to be of use in learning, it should serve as a reinforcement signal. As the appraisal model derives its dimensions with regards to the

Neuromodulator	Likely RL equivalent
Dopamine	TD error δ
Serotonin	Discount factor γ
Noradrenaline	Inverse temperature τ^{-1}
Acetylcholine	Learning rate α

Table 4.2: The four main neuromodulators and the reinforcement-learning parameters they have been indicated to control.

goals and estimations of the agent, it seems intuitive that it should be fit for generating reward for reinforcement learning. Marinier and Laird (2007) suggests computing reward by multiplying the feeling intensity I_F with the Conduciveness dimension, yielding

$$R = I_F \cdot \text{Conduciveness}. \tag{4.2}$$

However, it is important to note that, since the feeling is produced by combining emotion (which is a function of only the current state) and mood (which depends on the history of previously visited frames), the environment loses the Markov property. Hence, the theory of classical reinforcement learning cannot be effectively applied, and convergence proofs are not available. It might, however, be possible that this property helps the agent cope with an already non-Markovian environment such as in a competitive task by keeping the agent from being sidetracked by events it cannot exert control over.

4.3 Effect of Emotion on Learning

Doya (2000, 2002), relating to previous work by, among others, Montague et al. (1996) and Schultz et al. (1997), discusses the effect of emotion on learning in the human brain and attempts to relate the observed effects to reinforcement learning in the machine-learning sense. He considers the TD error δ and what he terms *metaparameters*, referring to parameters such as the learning rate α, discount factor γ and Gibbs/Boltzmann temperature τ, and how they are controlled by so-called neuromodulators in the human brain during the process of reinforcement learning. The main findings of Doya are summarised in table 4.2.

Ishii et al. (2002) relate their algorithm for modulating the temperature parameter to Doya (2000) and investigate a possible implementation of exploration/exploitation balance control by the noradrenaline system in the brain. A more in-depth analysis of the role of noradrenergic neurons in the locus coeruleus in controlling the level of selective attention can be found in Usher et al. (1999).

Chapter 5

System Design

The proposed algorithm was implemented on the *Virtual Alter Egos* (VAE) platform, previously called *Video Agents* (Asai et al., 2007; Kitamura et al., 2008; Asai et al., 2005); specifically, a rewritten and updated version of the system presented in Rong et al. (2010) was used. Put briefly, the VAE system allows users to easily create autonomous virtual agents in their own likeness that "live" and interact with each other, the environment and the users in a virtual world, as illustrated in figure 5.1.

5.1 Learning

Because of its theoretical advantages in stochastic and multiagent environments, the actor-critic model was chosen instead of the more common Q-learning or SARSA. The *Fuzzy Actor-Critic Reinforcement Learning Network* (FACRLN) by Wang et al. (2007) is used for function approximation of the state-value function and the policy. This algorithm builds on previous work in Cheng et al. (2004) and makes use of a normalised fuzzy radial-basis-function network. As shown in figure 5.2 (simplified to show a radial-basis-function network; the radial basis functions are computed using equation (3.9)), the state vector $\boldsymbol{s}$ is input to the network, whose output consists of action preferences $A_k(s)$ and the estimated value $V(s)$ for state s. The usefulness of the algorithm stems in large part from its ability to adaptively add and remove computational units, as well as adapt the centre and width parameters of the radial basis functions during the learning process as required by the task at hand.

For a detailed description of the FACRLN algorithm, the reader is directed to Wang et al. (2007).

Figure 5.1: A scene from *Virtual Alter Egos: Osaka Developing Story* (Rong et al., 2010), used for testing the proposed algorithm.

5.1.1 Merging and Deletion of Units

Because of the added computational complexity of the subroutine for merging similar computational units, disproportionate to the observed frequency of calls using reasonable parameter settings, this functionality was left out of the finished system. Another related technique for keeping the size of the network, and in extension learning and computation time, to a minimum as given in Cheng et al. (2004) is to remove inactive units that does not exceed a certain small activation threshold for a specified number of time steps. This technique seems to be more effective in removing non-contributing units, as well as less computationally demanding than determining when to merge units, but it is difficult to know beforehand what is a suitable period of inactivity before the unit is removed.

5.1.2 Action Selection

The softmax action selection scheme suggested by Cheng et al. (2004) and Wang et al. (2007), adding a random number from a normal distribution to each output value $A_t(\boldsymbol{s})$ of the actor and selecting the action with the greatest sum, is not used. Although useful in theory, as the width $\sigma_V(t)$ of the normal distribution from which the noise term is drawn depends on the estimated value of the current state s_t according to $\sigma_V(t) = \frac{1}{1+\exp(2V(s_t))}$, better empirical results were obtained using simple Gibbs softmax selection (equation (2.8)), wherefore the latter scheme is used instead.

Li et al. (2009), implementing a system similar to the one described in Cheng et al. (2004), use a directed form of Gibbs softmax selection that is more likely to select actions that have not been tried as many times pre-

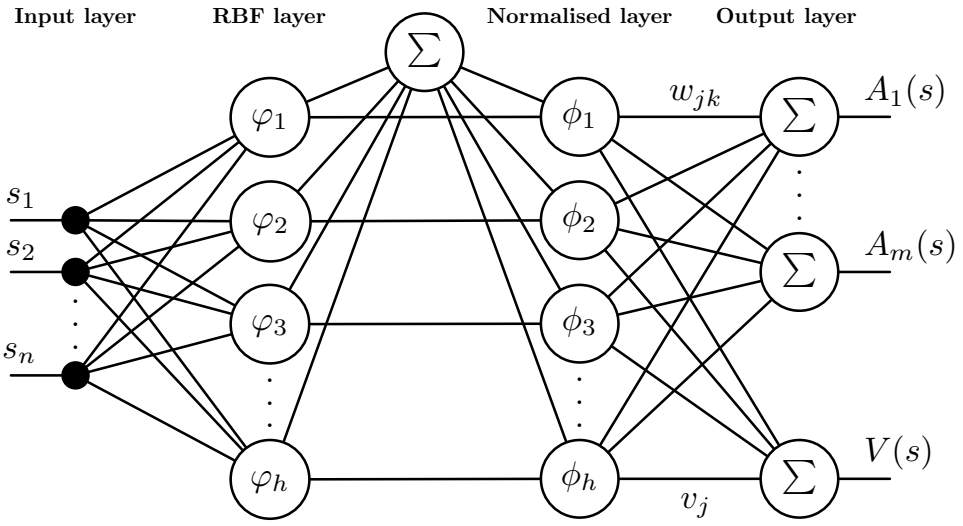

Figure 5.2: The topology of the actor-critic normalised radial-basis-function network.

viously. Despite its theoretical benefits, however, that variant was also discarded as it introduces additional parameters that need to be selected beforehand, already a problem in radial-basis-function networks for reinforcement learning.

5.1.3 Extensions

The algorithm was extended by the author so that, instead of merely using a single learning rate α_A for the actor, each computational unit j has a set of parameters α_{jk}, one for every action a_k. At every state transition, the actor weights are updated with $\Delta w_{jk} = \alpha_{jk}\alpha_A\delta\phi_j(s)$ and the corresponding stepsize parameters, each initialised to 1, according to $\alpha_{jk} \leftarrow \alpha_{jk}(1 - \beta\phi_j)$, where β is a small positive number. The motivation for this is that (in a stationary environment) the learning rate should fulfil condition 2.11.

5.2 Emotional Model

Each agent keeps a set of primitive physiological, mental and emotional parameters as listed in table 5.1. Each parameter can take values in $[0, 1]$. The choice of emotional parameters follows Ekman and Friesen (1969), although Interest for practical reasons has been classified as a physiological parameter. The physiological parameters can be directly influenced by interaction with the environment: eating food reduces hunger, resting reduces fatigue, and interacting with other agents reduces interest. The mental parameters are hidden from the user, but represent the agent's conscious or unconscious

Table 5.1: Agent internal parameters.

Physiological	**Mental**	**Emotional**
Hunger	Comfort	Happiness
Fatigue	Satisfaction	Sadness
Interest	Caution	Anger
	Impression	Surprise
	Stress	Fear
	Unexpected	Disgust

assessment of received stimuli with regards to its current goals and model of the environment. As such, they function as an intermediary step between the physiological and emotional parameters, as the latter are computed exclusively from the mental parameters using fuzzy logic.

5.3 Interaction with the Environment

The actor can choose between four basic actions: picking up food, socially interacting with another agent, resting, or doing nothing, which in practice translates to randomly walking aimlessly or playing an animation suitable for their current mood. Whether or not all actions are available depends on the observed state; for example, the agent can obviously not pick up food if there is no food available. Actions may result in failure if, for example, the agent is interrupted by a nearby event, the item targeted for retrieval is taken first by another agent, or if the other party declines a request for social interaction. Failure will have negative emotional impact on the agent, although the magnitude depends on various factors such as the reason for failure and the estimated reward missed out on.

A new state is obtained upon completion, or interruption, of an action, and hence different agents operate in different time frames. The internal state is represented in a rather minimalistic fashion as detailed in table 5.2. The agent is often not aware of the full true state of the environment, but creates an internal state from what it can perceive at that time. Therefore, and considering the unpredictable risk of action failure, the environment can be seen as a noisy partially observable Markov decision process (POMPD).

Reward is computed by taking the difference between current and previous values of the Happiness parameter as sampled at state transitions; hence, the reward r_t at the transition from state $t-1$ to t is given by

$$r_t = \Delta Happiness = Happiness_t - Happiness_{t-1}.$$

A penalty is subtracted from the reward if the previous action would have caused a physiological parameter to drop below zero, to prevent the agent from learning *e.g.* to rest even when not tired.

Table 5.2: The internal state representation.

Dimension	Possible values
Hunger	$[0, 1]$
Fatigue	$[0, 1]$
Interest	$[0, 1]$
Other agent present	$\{0, 1\}$
Food present	$\{0, 1\}$

Table 5.3: Modulation of learning parameters.

Parameter	Modulating emotion
Learning rate α	$1 -$ Surprise
Discounting factor γ	$1 -$ Fear
Temperature τ	Anger

5.3.1 Modulation of Learning Parameters

The parameters whose modulation is of interest are the learning rate α, the discount factor γ and the Gibbs/Boltzmann temperature τ. While the approach described in Akiguchi and Maeda (2006) was tried, as was one directly inspired by Doya (2002), neither approach produced satisfactory results. This should, however, be blamed on differences in the emotional model and incomplete implementation, respectively, rather than any flaws in the cited works. Instead, the method shown in table 5.3 was settled upon for metaparameter modulation; each parameter is linearly interpolated by the amount of the modulating value between a maximum and a minimum value set by the programmer.

The choice of $1 -$ Surprise rather than simply Surprise for modulation of the learning rate may warrant further explanation, as it may seem wiser to take high Surprise to mean learning rate should be increased to compensate for the observed discrepancy between the actual and the expected outcome. This is likely to be the case in a truly Markovian environment if Surprise were derived from a continually updated estimation of state-transition probabilities based on observed transitions (the critic would be a prime candidate) or if conditioning by the user was primarily desired, but in this case Surprise is increased by a static function upon action failure. As failure depends on unpredictable factors upon which the agent can exert little or no control, excessively updating the value function and policy to adjust for surprising outcomes is likely to interfere with learning. In a sense, surprising outcomes really are surprising, and should therefore to some extent be ignored.

Chapter 6

Results

Based on the theory and ideas expounded in the previous chapters, a system was implemented aiming to achieve efficient learning in the face of competition while driven by emotional feedback. In the following sections, the implementation details, particulars of the experiment enacted in order to test the system, as well as analysis and discussion of the results and suggestions of future work are presented.

6.1 Experiment Design

In order to test the performance of the proposed algorithm, a script was written for execution on the *Virtual Alter Egos* system. The script was designed to at all times provide a single item of food for the agents; upon consumption of one item of food, another was spawned at a random location. Each experiment round lasted for 2000×5 seconds; during the first 1500×5 seconds, at each update there was a small probability of fire spawning at a random location, forcing nearby agents to abandon their current actions and escape. Being too close to a fire inflicts significant stress upon an agent, which will also be fatigued by running away. This introduced an additional element of randomness into the experiment, for the purpose of testing the robustness of the behavioural models. The spawn probability was set to give an expected value of 30 seconds between fires.

The experiment was performed with four agents all using the same character so as to avoid such problems as discrepancies in action execution speed due to animations differing between characters. The average of the values of the happiness parameter of all four agents was recorded every five seconds. First, the experiment was run 20 times without emotional feedback; then, 20 times with the emotional feedback loop activated. The sampled data sets were then averaged separately over both sets of 20 runs, and the resulting graphs evaluated.

6.1.1 Parameter Settings

While the fuzzy radial-basis-function network described in Wang et al. (2007) has the capability of self-allocating computational units, a network with preallocated blank units was used for the experiment to simplify comparison of the results; however, the network could add additional units during learning as deemed necessary. As the use of eligibility traces may interfere with learning in non-stationary systems such as multi-agent systems, as noted by Geist and Pietquin (2010), the eligibility parameter λ was set to zero. For similar reasons, and so as not to interfere with the experiment, the learning-rate decay parameter β, described in section 5.1.3, was also set to zero, turning also that functionality off.

In order to allow for fair comparison of the two algorithms, the fixed parameters of the unmodulated algorithm were set equal to the "base" values (*i.e.* the values taken when Surprise, Fear and Anger are all zero) assigned to the modulated algorithm. The settings $\alpha_A = 0.2, \alpha_C = 0.1, \gamma = 0.3, \tau = 0.01$, where α_A and α_C are the learning rates for the actor and the critic, respectively, were empirically found to give good results, and were thus chosen as base values. The modulated algorithm was allowed the ranges $\alpha_A \in [0.05, 0.2]\,, \alpha_C \in [0.01, 0.1]\,, \gamma \in [0.1, 0.3]$ and $\tau \in [0.01, 0.1]$.

6.2 Outcome

The experiment results are shown in figure 6.1. The sharp increase in both curves at $t = 1500$ is due to the cessation of disturbing elements in the form of fires at this point. The emotion-modulated algorithm can be seen to perform slightly better than its unmodulated counterpart following the removal of fires; while the advantage is slight, is was consistent across experiment rounds. This seems to hint that a better policy was learnt compared to that of the unmodulated algorithm, which still expects otherwise optimal actions to sometimes be interrupted by fires and thus assigns them lower preferences, instead sometimes choosing actions whose interruption would cause less disappointment.

The modulated algorithm appears to be learning somewhat slowlier in the beginning, which might be attributed to the higher value of the temperature parameter early on, leading to greater exploration and thus less greedy action selection.

6.3 Conclusion

An algorithm implementing the actor-critic model of reinforcement learning was employed in an environment with four individually optimising and potentially conflicting agents, who could be interrupted and disturbed by fires

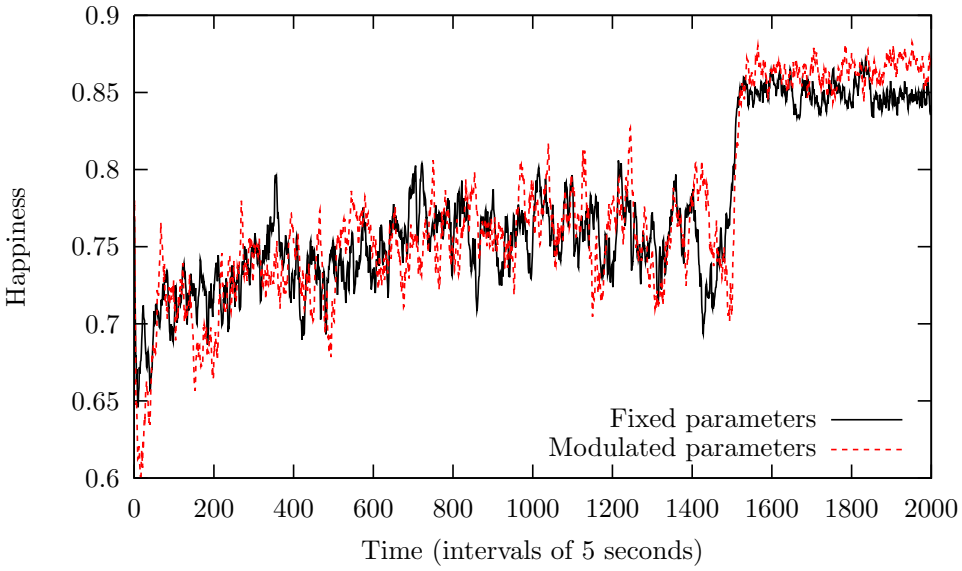

Figure 6.1: Average values of the Happiness parameter over all four agents using the fixed-parameter and the parameter-modulation scheme, plotted over time. Averaged over 20 runs.

randomly spawning near them at any time. With learning thus interfered with until fires were turned off after a set time towards the end of each test run, a parameter modulation scheme that would adjust the reinforcement-learning parameters depending on the current emotional state of the agent was evaluated and compared to a fixed-parameter scheme otherwise using identical settings.

Altough the increase in performance over the fixed-parameter algorithm using the emotion-modulated algorithm was minor, it is nevertheless apparent and indicates that some advantage could be gained in the test environment by using the proposed algorithm. The reason is believed to be mainly that the modulation scheme allows learning to focus on more useful input by assigning less weight to input that is considered less useful and that might interfere with learning, although increased exploration following unsatisfactory outcomes may also have contributed in the long run by keeping the agent from pursuing suboptimal strategies.

6.4 Future work

The principal weakness of the proposed algorithm is that it assumes a more or less correct evaluation of the input state as basis for the emotional feedback; this requires more domain knowledge than is generally available. It would be desirable to design an algorithm that makes use of the approximated state evaluation function already available through the critic to generate

emotion, preferably through appraisal. Such a method would likely require a somewhat different approach to modulation than is presented in this paper, so the modulating algorithm would need adjustment as well.

Furthermore, it would be interesting to attempt to make a more realistic implementation of the mechanisms described in Doya (2002), perhaps coupled with the model proposed by Lövheim (2012) to generate emotions from simulated neurotransmitters.

Bibliography

Shunsuke Akiguchi and Yoichiro Maeda. Autonomous pet-type robot with emotion behaviour-learning system based on neuromodulators. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 220(8):717–724, 2006. doi: 10.1243/09596518JSCE150. Cited on pages 4 and 31.

Kazuhiro Asai, Atsushi Hattori, Katsuya Yamashita, Takashi Nishimoto, Yoshifumi Kitamura, and Fumio Kishino. Agents from reality. In *Proceedings of IFIP International Conference on Entertainment Computing (ICEC)*, pages 519–522, 2005. Cited on pages 4 and 27.

Kazuhiro Asai, Yoshifumi Kitamura, Takashi Nishimoto, Yoshinori Hirano, Emiko Hama, and Fumio Kishino. Video agents. In *ACM SIGGRAPH Emerging Technologies*, 2007. Cited on pages 4 and 27.

Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 30–37. Morgan Kaufmann, 1995. Cited on page 6.

Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72:81–138, January 1995. ISSN 0004-3702. Cited on page 5.

Richard E. Bellman. A markovian decision process. *Indiana University Mathematics Journal*, 6:679–684, 1957a. ISSN 0022-2518. Cited on pages 3, 5, and 6.

Richard E. Bellman. *Dynamic programming*. Rand Corporation research study. Princeton University Press, 1957b. ISBN 9780691079516. Cited on pages 3 and 14.

Michel Benaim. On functional approximation with normalized gaussian units. *Neural Computation*, 6:319–333, March 1994. ISSN 0899-7667. doi: 10.1162/neco.1994.6.2.319. Cited on page 19.

Michael Bowling and Manuela Veloso. An analysis of stochastic game theory for multiagent reinforcement learning. Technical report, Computer Science Department, Carnegie Mellon University, 2000. Cited on page 6.

Steven J. Bradtke and Andrew G. Barto. Linear least-squares algorithms for temporal difference learning. In Leslie P. Kaelbling, editor, *Machine Learning*, pages 22–33, 1996. Cited on page 8.

Dave S. Broomhead and David Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988. Cited on pages 3 and 18.

Yu-Hu Cheng, Jian-Qiang Yi, and Dong-Bin Zhao. Application of actor-critic learning to adaptive state space construction. In *Proceedings of the Third International Conference on Machine Learning and Cybernetics*, volume 5, pages 2985–2990, August 2004. doi: 10.1109/ICMLC.2004.1378544. Cited on pages 3, 27, and 28.

Kenji Doya. Metalearning, neuromodulation, and emotion. In *The 13th Toyota Conference on Affective Minds*, pages 101–104, 2000. Cited on pages 4 and 26.

Kenji Doya. Metalearning and neuromodulation. *Neural Networks*, 15:495–506, June 2002. ISSN 0893-6080. doi: 10.1016/S0893-6080(02)00044-8. Cited on pages 4, 17, 26, 31, and 35.

Paul Ekman. Basic emotions. In Tim Dalgleish and Mick J. Power, editors, *The Handbook of Cognition and Emotion*, chapter 3, pages 45–60. John Wiley & Sons, Ltd., 1999. Cited on page 24.

Paul Ekman and Wallace V. Friesen. The repertoire of nonverbal behavior: Categories, origins, usage, and coding. *Semiotica*, 1(1):49–98, 1969. Cited on pages 24 and 29.

Sandra Clara Gadanho. *Reinforcement Learning in Autonomous Robots: An Empirical Investigation of the Role of Emotions*. PhD thesis, University of Edinburgh, 1999. Cited on page 4.

Sandra Clara Gadanho. Learning behavior-selection by emotions and cognition in a multi-goal robot task. *Journal of Machine Learning Research*, 4: 385–412, 2003. Cited on page 4.

Sandra Clara Gadanho and John Hallam. Emotion-triggered learning in autonomous robot control. *Cybernetics and Systems*, 32:531–559, 2001. Cited on page 4.

Matthieu Geist and Olivier Pietquin. Revisiting natural actor-critics with value function approximation. In V. Torra, Y. Narukawa, and M. Daumas, editors, *Proceedings of 7th International Conference on Modeling Decisions for Artificial Intelligence (MDAI 2010)*, volume 6408 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 207–218. Springer Verlag - Heidelberg Berlin, Perpinya, France, October 2010. URL `http://www.metz.supelec.fr/metz/personnel/geist_mat/pdfs/Supelec624.pdf`. Cited on pages 12 and 33.

Jonathan Gratch and Stacy C. Marsella. A domain-independent framework for modeling emotion. *Journal of Cognitive Systems Research*, 5:269–306, 2004. Cited on pages 4 and 24.

Vijaykumar Gullapalli. A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks*, 3(6):671–692, 1990. Cited on page 13.

Eric J. Hartman, James D. Keeler, and Jacek M. Kowalski. Layered neural networks with gaussian hidden units as universal approximations. *Neural Computation*, 2:210–215, 1990. doi: 10.1162/neco.1990.2.2.210. Cited on pages 17 and 18.

Kenneth J. Hunt, Roland Haas, and Roderick Murray-Smith. Extending the functional equivalence of radial basis function networks and fuzzy inference systems. *IEEE Transactions on Neural Networks*, 7(3):776–781, May 1996. ISSN 1045-9227. doi: 10.1109/72.501735. Cited on pages 20 and 21.

Shin Ishii, Wako Yoshida, and Junichiro Yoshimoto. Control of exploitation-exploration meta-parameter in reinforcement learning. *Neural Networks*, 15:665–687, 2002. Cited on pages 13 and 26.

Jyh-Shing Roger Jang and Chuen-Tsai Sun. Functional equivalence between radial basis function networks and fuzzy inference systems. *IEEE Transactions on Neural Networks*, 4(1):156–159, January 1993. Cited on pages 3, 17, 20, and 21.

Yoshifumi Kitamura, Rong Rong, Yoshinori Hirano, Kazuhiro Asai, and Fumio Kishino. Video agent: Interactive autonomous agents generated from real-world creatures. In *Proceedings of ACM Symposium on Virtual Reality Software and Technology (VRST)*, pages 30–38, 2008. Cited on pages 4 and 27.

Chun-Gui Li, Meng Wang, Zhen-Jin Huang, and Zeng-Fang Zhang. An actor-critic reinforcement learning algorithm based on adaptive rbf network. In *2009 International Conference on Machine Learning and Cybernetics*, volume 2, pages 984–988, July 2009. doi: 10.1109/ICMLC.2009.5212431. Cited on page 28.

Chin-Teng Lin and C. S. George Lee. Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems. *IEEE Transactions on Fuzzy Systems*, 2(1):46–63, February 1994. ISSN 1063-6706. doi: 10.1109/91.273126. Cited on page 4.

Richard P. Lippman. Pattern classification using neural networks. *IEEE Communications Magazine*, 27:47–64, 1989. Cited on page 18.

Hugo Lövheim. A new three-dimensional model for emotions and monoamine neurotransmitters. *Medical Hypotheses*, 78(2):341–348, 2012. ISSN 0306-9877. doi: 10.1016/j.mehy.2011.11.016. URL `http://www.sciencedirect.com/science/article/pii/S0306987711005883`. Cited on pages 4 and 35.

Maria Malfaz and Miguel A. Salichs. Emotion-based learning of intrinsically motivated autonomous agents living in a social world. In *Fifth International Conference on Development and Learning*, May 2006. Cited on pages 4 and 23.

Robert P. Marinier and John E. Laird. Computational modeling of mood and feeling from emotion. In *Proceedings of the 29th Annual Conference of the Cognitive Science Society*, 2007. Cited on pages 4, 23, 24, 25, and 26.

Robert P. Marinier, III, John E. Laird, and Richard L. Lewis. A computational unification of cognitive behavior and emotion, 2008. Cited on pages 4, 23, and 24.

Stacy C. Marsella and Jonathan Gratch. Ema: A process model of appraisal dynamics. *Cognitive Systems Research*, 10:70–90, 2009. doi: 10.1016/j.cogsys.2008.03.005. Cited on page 4.

P. Read Montague, Peter Dayan, and Terrence J. Sejnowski. A framework for mesencephalic dopamine systems based on predictive hebbian learning. *Journal of Neuroscience*, 16:1936–1947, 1996. URL `http://www.gatsby.ucl.ac.uk/~dayan/papers/mds96.html`. Cited on pages 4 and 26.

John Moody and Christian J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1:281–294, June 1989. ISSN 0899-7667. doi: 10.1162/neco.1989.1.2.281. Cited on pages 3, 18, and 19.

Scott Neal Reilly. Modeling what happens between emotional antecedents and emotional consequents. In *Proceedings of the 18th European Meeting on Cybernetics and Systems Research*, pages 607–612. Austrian Society for Cybernetic Studies, April 2006. Cited on page 25.

Jooyoung Park and Irwin W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computing*, 3:246–257, June 1991. ISSN 0899-7667. doi: 10.1162/neco.1991.3.2.246. Cited on page 18.

John Platt. A resource-allocating network for function interpolation. *Neural Computation*, 3:213–225, June 1991. ISSN 0899-7667. doi: 10.1162/neco.1991.3.2.213. Cited on page 3.

Tomaso Poggio and Federico Girosi. A theory of networks for approximation and learning. *Laboratory, Massachusetts Institute of Technology*, 1140, 1989. Cited on pages 3 and 18.

Tomaso Poggio and Federico Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, September 1990. ISSN 0018-9219. doi: 10.1109/5.58326. Cited on pages 3 and 18.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951. Cited on page 12.

Raúl Rojas. *Neural Networks - A Systematic Introduction.* Springer-Verlag, Berlin, 1996. URL `http://www.inf.fu-berlin.de/inst/ag-ki/rojas_home/pmwiki/pmwiki.php?n=Books.NeuralNetworksBook`. Cited on pages 2 and 15.

Rong Rong. Interactive video content using captured real human videos. Master's thesis, Osaka University, 2011. Cited on page 4.

Rong Rong, Yoshifumi Kitamura, Shinya Kitaoka, Hirokazu Kato, Keisuke Kishi, Kei Nakashima, Kazuhiko Yamazaki, Sadayoshi Horiuchi, Tatsuro Shioda, Kazuto Kamakura, and Shinya Matsuyama. Osaka developing story: An application of video agents. In *Proceedings of the International Conference on Advances in Computer Entertainment Technology (ACM ACE)*, pages 3–6, 2010. Cited on pages 4, 27, and 28.

Klaus R. Scherer. *Appraisal considered as a process of multi-level sequential checking*, pages 92–120. Oxford University Press, New York and Oxford, 2001. Cited on pages 4 and 24.

Wolfram Schultz, Peter Dayan, and P. Read Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, March 1997. doi: 10.1126/science.275.5306.1593. URL `http://www.sciencemag.org/content/275/5306/1593.full`. Cited on pages 4 and 26.

Nicolas Schweighofer and Kenji Doya. Meta-learning in reinforcement learning. *Neural Networks*, 16(1):5–9, January 2003. Cited on page 13.

Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations.* Cambridge University Press, New York, USA, 2009. URL `http://www.masfoundations.org/`. Cited on pages 2 and 10.

Robert Shorten and Roderick Murray-Smith. On normalising radial basis function networks. In *4th Irish Neural Networks Conference*, September 1994. Cited on page 19.

Robert Shorten and Roderick Murray-Smith. Side effects of normalising radial basis function networks. *International Journal of Neural Systems*, 7(2):167–179, May 1996. Cited on pages 19 and 20.

Satinder Singh and Richard S. Sutton. Reinforcement learning with replacing eligibility traces. In Leslie P. Kaelbling, editor, *Machine Learning*, pages 123–158, 1996. Cited on page 12.

Richard Sutton. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *Proceedings of Tenth National Conference on Artificial Intelligence AAAI-92*, pages 171–176. MIT Press, 1992. Cited on page 13.

Richard S. Sutton. *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, University of Massachusetts Amherst, 1984. Cited on pages 3 and 9.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, August 1988. ISSN 0885-6125. URL `http://www.incompleteideas.net/sutton/publications.html#TD_paper`. Cited on pages 3, 8, and 11.

Richard S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems*, volume 8, pages 1038–1044. MIT Press, 1996. Cited on page 14.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, March 1998. ISBN 0262193981. URL `http://www.incompleteideas.net/sutton/book/the-book.html`. Cited on pages 5, 6, 7, 8, 9, 12, 14, 15, 16, and 17.

Tomohiro Takagi and Michio Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 15(1):116–132, February 1985. URL `http://takagiken.com/ts-model`. Cited on pages 3, 20, and 21.

Gerald Tesauro. Practical issues in temporal difference learning. *Machine Learning*, pages 257–277, 1992. Cited on page 3.

Gerald Tesauro. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, pages 215–219, 1994. Cited on page 3.

Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, pages 58–68, 1995. Cited on page 3.

Silvan S. Tomkins and Robert McCarter. What and where are the primary affects? some evidence for a theory. *Perceptual and Motor Skills*, 18, 1964. Cited on pages 4 and 23.

Marius Usher, Jonathan D. Cohen, David Servan-Schreiber, and Janusz Rajkowski. The role of locus coeruleus in the regulation of cognitive performance. *Science*, 283(5401):549–554, January 1999. Cited on page 26.

José M. Vidal. *Fundamentals of Multiagent Systems*. e-book, 2010. URL `http://multiagent.com/2010/02/multiagent-systems-textbook.html`. Cited on pages 2 and 6.

Xue-Song Wang, Yu-Hu Cheng, and Jian-Qiang Yi. A fuzzy actor-critic reinforcement learning network. *Information Sciences*, 177:3764–3781, September 2007. ISSN 0020-0255. doi: 10.1016/j.ins.2007.03.012. Cited on pages 3, 19, 22, 27, 28, and 33.

Christopher J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, University of Cambridge, 1989. Cited on pages 3 and 8.

Thomas Weise. *Global Optimization Algorithms - Theory and Application*. it-weise.de (self-published), 2009. URL `http://www.it-weise.de/projects/book.pdf`. Cited on page 3.

Xinjie Yu and Mitsuo Gen. *Introduction to Evolutionary Algorithms*. Decision Engineering. Springer, 2010. ISBN 9781849961288. URL `http://books.google.com/books?id=rHQf_2Dx2ucC`. Cited on page 3.

Lotfi A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3:28–44, 1973. Cited on pages 3 and 20.