

Perspectives on the Connection of Psychological Models of Emotion and Intelligent Machines

Johannes Feldmaier





Technische Universität München
Lehrstuhl für Datenverarbeitung

Perspectives on the Connection of Psychological Models of Emotion and Intelligent Machines

Johannes Feldmaier

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzende: Prof. Dr.-Ing. Sandra Hirche

Prüfer der Dissertation:

1. Prof. Dr.-Ing. Klaus Diepold
2. Priv.-Doz. Dr. Felix Schönbrodt

Die Dissertation wurde am 08.06.2017 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 21.11.2017 angenommen.

Johannes Feldmaier. *Perspectives on the Connection of Psychological Models of Emotion and Intelligent Machines*. Dissertation, Technische Universität München, Munich, Germany, 2018.

© 2018 Johannes Feldmaier

Institute for Data Processing, Technical University of Munich, 80290 Munich, Germany,
<http://www.ldv.ei.tum.de>.

This work is licenced under the Creative Commons Attribution 3.0 Germany License. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/3.0/de/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.

Abstract

Researchers from psychology and computer science consider artificial emotions as a missing component in cognitive systems. Such affective and cognitive systems are said to be the ideal partners in shared environments where assistive systems and service robots tightly work together with their human partners. The acceptance and tangibility of intelligent systems increase by integrating psychological findings into state-of-the-art machine learning algorithms. In current systems the cognitive component is central, and further optimization of their decision making algorithms and environmental recognition techniques often require huge effort, but deliver only few percent in the improvement on the general performance. Different approaches which consider the learning component of such an intelligent system from new perspectives are rare. Currently, the activity in the research area of affective computing increases. The idea of integrating psychological models into machine learning algorithms gives fresh impetus to the machine learning community. Generally, affective components in machine learning can be divided into the domains of emotion recognition, artificial emotion generation, and the final rendering process to express them in an appropriate way.

The focus of this dissertation lies on the first two aspects and sheds light on the answer to the question if psychological theories and models for emotions can inform and enhance the Human-Robot-Interaction. For this purpose, this general problem statement is subdivided into three related issues.

First, potential methods to calculate representations for the evaluation of the current performance and state of a machine learning process are considered. In three experiments, psychologically grounded emotion models are implemented and used to evaluate the performance of a learning agent. Besides the Zurich Model of Social Attachment and Fear, also two appraisal models (a version of the Component Process Model and a direct appraisal model) are implemented. The results show that quantitative figures and the temporal behavior of a machine learning process can be appraised in terms of artificial emotions and feelings by the developed algorithms. In a study, the expectations of human users on artificial emotions expressed by a service robot were investigated. The findings were compared to the simulated emotions of the developed algorithm, which revealed significant similarities.

Secondly, the implicit control of policies learned by a Reinforcement Learning agent is investigated. Therefore, the mechanism of reward shaping is considered and extended by a definition of an additional state. This state represents an internal affective state of the agent (e.g. an experience value) and is used to modify the weights of the reward functions. In an experiment, it is shown that different goals or actions can be biased in relation to the

affective state. This kind of reinterpretation of reward shaping as a method to affectively control Reinforcement Learning is the key contribution of this second part.

As a third contribution, the concept of Inverse Reinforcement Learning is modified to retrieve scalarization weights from multi-objective policies learned by a Reinforcement Learning agent. By comparing the retrieved weights with those intuitively set by a human designer reveal potential human biases. Also counter-intuitive effects, which are introduced through Reinforcement Learning can be detected with the developed algorithm.

Overall, the results show that the interpretation of psychological models in a way to deploy them in the machine learning domain can extend the understanding of artificial agents and foster the development of new approaches. Of course, there remain unanswered ethical and philosophical questions regarding the need to make machines more human like. However, this dissertation makes a step in affective computing in order to give answers to these questions by presenting examples of connections between psychology and machine learning.

The question is not whether intelligent machines can have any emotions, but whether machines can be intelligent without any emotions.

Marvin Minsky, 1986

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken die diese Dissertation begleitet haben und sie letztendlich zu dem gemacht haben was sie nun ist.

Allen voran gilt mein Dank Professor Klaus Diepold, an dessen Lehrstuhl diese Arbeit entstanden ist und ohne dessen Unterstützung und Rat dieses Vorhaben von vorneherein zum Scheitern verurteilt gewesen wäre. An seinem Lehrstuhl für Datenverarbeitung habe ich sehr viel gelernt – nicht nur fachlich, sondern auch in der Lehre und dem Umgang mit Studierenden.

Ebenfalls danken möchte ich allen Kolleginnen und Kollegen des Lehrstuhls. Besonderer Dank geht an Dominik Meyer, Martin Rothbacher, Martin Knopp und Philipp Paukner die sich aktiv bei den Korrekturen sowie mit Rat und Tat an dieser Dissertation beteiligt haben. Gerade zu Beginn der Promotion haben Martin Rothbacher, Julian und Tim Habigt ebenfalls maßgeblich meine Arbeitsweise und Vorgehensweise am Lehrstuhl beeinflusst – dafür ein extra Dankeschön! Ebenso möchte ich Ricarda Baumhoer danken, die einem das Leben am Lehrstuhl mit ihrer konstruktiven Art und Weise deutlich erleichtert und auch immer ein offenes Ohr für Problemchen hat.

Ein großer Dank gilt auch meinen Eltern die mir nicht nur das Studium ermöglicht haben, sondern mir auch regelmäßig motivierenden Beistand geleistet haben.

Nicht zuletzt, sondern am meisten danke ich dem liebsten Menschen in meinem Leben, Kathrin, die immer für mich da war und Motivationstiefs mit mir durchlebt hat und auch den ein oder anderen hilfreichen Tipp für diese Dissertation hatte – wir sind zusammen das beste Team.

Contents

1. Introduction	9
1.1. Motivation	9
1.2. Research questions	11
1.3. Contributions and Scope	12
2. Background	15
2.1. Theories and Models of Affective Computing	15
2.2. Affect Generation	20
2.2.1. Component Process Model	20
2.2.2. Ortony, Clore & Collins Model of Emotions	24
2.2.3. Zurich Model of Social Attachment and Fear	27
2.2.4. Model Implementation	30
2.3. Reinforcement Learning	35
2.3.1. Elements of Reinforcement Learning	35
2.3.2. Multi-objective Reinforcement Learning	44
2.3.3. Reward shaping	46
2.3.4. Human values	49
2.3.5. Preferences	50
2.3.6. Affective states in Reinforcement Learning	52
3. Affective Evaluation of Machine Learning Experiments	55
3.1. Bandit Simulation	56
3.1.1. Multi-armed bandits	56
3.1.2. Implementation	57
3.1.3. Experiment	60
3.1.4. Results	61
3.2. Gridworld	64
3.2.1. Implementation	64
3.2.2. Core Affect	65
3.2.3. Experiment and Results	68
3.3. Simultaneous Localization and Mapping	71
3.3.1. SLEmotion	72
3.3.2. Stimulus Evaluation Checks	74
3.3.3. Categorization Module	82
3.3.4. Experiment and Study	84

Contents

3.3.5. Results	87
4. Affective Control	91
4.1. Reinforcement Learning with Preferences	92
4.2. Experiment	94
4.3. Results	97
5. Human Value Retrieval	101
5.1. Inverse Reinforcement Learning for Human Value Retrieval	101
5.1.1. Inverse Reinforcement Learning	102
5.1.2. Adaptation to Scalarized Multi-objective Reinforcement Learning	106
5.2. Simulations and Test Cases	107
5.2.1. Environment Description	108
5.2.2. Test Cases	109
5.3. Results	110
6. Conclusion	115
6.1. Summary	115
6.2. Future perspectives	117
List of Acronyms	122
A. Appendix	123
A.1. Published algorithms	123
A.2. Questionnaire of the SLEmotion experiment	125
A.3. Derivation of the Block Matrix Form of the Inverse RL algorithm	129
Bibliography	143

1. Introduction

"The robots are coming for taking our jobs!" That is what often is written in economic publications and for many people this is a frightening scenario. As engineers we make steady progress in building new applications and systems equipped with highly developed artificial intelligence. The purpose is often to automatize traditional jobs in order to increase productivity. This automation process is not limited to specific sectors. Furthermore, the classical scenario where robots work autonomously behind security fences starts to crumble and new systems tightly working together with human co-workers are the current trend. In such *shared environments* a mutual and natural understanding is essential. In the following, this scenario is further motivated and related research questions are formulated.

1.1. Motivation

The current vision of researchers in the field of robotics are service robots tightly working together with humans in a real world environment. In this world, robots are able to sense their environment in real time and they react to dynamic changes. The robots render different services and the human users can give any command at any time. Depending on the current environment and the human user the corresponding robots react differently. Similar to humans, the service robots will have their own personality and show emotions giving the users the feeling of interacting with an intelligent *being*.

Such cognitive and social enabled robots will have huge impact on our everyday life. Intelligent systems will make decisions that might have a profound effect in our well-being. They might replace social contacts, take care of our children and family members, and support our health care. Similar to current smartphones, robotic agents will participate in almost every aspect of our daily life. Obviously, this will introduce massive problems in terms of acceptance and skepticism.

Therefore, research has to focus on aspects fostering the development of personality in artificial intelligent systems. Recent results of a preliminary study of Kate Darling of MIT (2017), reveal that assigning humans traits to robots increases the level of anthropomorphism of the machine. This simultaneously helps people to accept the machine and increase their tolerance for malfunctions. Experiences made with hospital robots arriving already dubbed with individual human names by the company show allegedly a higher tolerance of people in cases of errors or strange behavior as compared to the old, classical, squared-shaped, and non-anthropomorphically designed machines. While only assigning human names to machines looks like a shoddy trick, real personality relates to personal

1. Introduction

preferences, emotions, and individual decision making. Those additional factors of personality are also key components of trust between people and supposedly between humans and robots (Norman, 2013). Instead of simply imitating preferences and emotions, an autonomous system should be able to equally participate and cooperate with humans. It should be able to respond to gestures, body motions, the way (e.g. the speed, the forcefulness, etc.) an activity is performed, as well as to human feelings and thoughts. For this, emotions provide a convenient channel of communication to express the internal state of a system towards the user, or vice versa by recognizing the current emotions of the user to grasp her/his current mood. Additionally, emotions support the task of prioritizing goals and sub tasks.

Currently, there is no general or fully accepted definition that clearly separates the terms feeling, emotion, and mood from each other. In accordance to other definitions used in the domain of affective computing, feelings are often defined as an additional feature of the state representation (Marinier et al., 2009). Feelings use originally a task-independent format to combine current emotions and past mood, and thus are more general than a single emotion or the overall mood. There is always a limited set of distinct mood states, emotions, and feelings differentiating them from other objective state representations which can have an unlimited number of features and value ranges. The elicitation of distinct emotions and subsequent changes in mood and feelings, however, are state and task-dependent and involve previously made experiences. Feeling as an additional state representation can be used to guide control and the behavior of an artificial agent, as well as a simplification of the state representation. The influence of emotion, mood, and feeling is well-investigated by human emotion theories. For example, effects on the cognitive processing are investigated in the work of Forgas and George (2001) stating the pervasive influences on decision making and judgment in organizations. Also Phelps (2006) reviews the interaction of emotions with other cognitive processes of humans. She shows the impact of emotions on five topics: Emotion and memory, emotion's influence on attention and perception, emotional learning, processing emotion in social stimuli, and changing emotional responses. Her study clearly indicates that emotion and cognition appear to be intertwined at all stages. Furthermore, Gross and John (2003) consider the effects of emotions on the human coping behavior, and Frijda et al. (1989) investigate the affective implications on the action tendencies.

With these implications of feelings and emotions to human behavior in mind, it seems reasonable, that there is an evolutionary cause for human emotions. And yes, in literature it is said that emotions and feelings have helped us to survive. They are the reason for reacting quickly in dangerous situations without consciously grasping the situation (see Antonio Damasio, Joseph LeDoux, and Robert Trivers for extensive theories about emotions and evolution). While said to be essential for the evolutionary survival of human beings, the following question obviously arises: Are artificial emotions and feelings also beneficial in machine learning algorithms of future robots? The use of artificial emotions in machine learning algorithms could be as influential as the discovery of Slagle (1963), as he im-

plemented a method for autonomously determining the priority and the order of solving sub-problems in given symbolic integration problems.

This motivates investigations to further improve confidence and trust into autonomous systems by developing mechanisms that tightly incorporate machine learning and emotions. Those mechanisms can support new ways for intelligent systems to provide meaningful status, give explanations of their behavior, and ask for assistance when necessary. The key element in those algorithms might be an artificial emotion model and the effects caused by the calculated emotions.

1.2. Research questions

Considering the familiar question of choosing what and where to eat when you decide to go out for dinner, a typical decision making scenario unfolds. There will be a set of known restaurants and maybe some new recommendations as well as your own experiences you have made before. Furthermore, you cannot be sure if there are some changes in staff and ownership.

Do you choose the restaurant based on your experience, preferences for the location, style, or the people accompanying you? Or do you try out a new recommendation of your colleague? Why not taking pot luck and try a completely new restaurant randomly chosen while walking down the main street?

As one can see, a simple every-day question results in a dynamic decision making process. Describing all factors involved in this dynamic process touches several psychological and philosophical topics. People solve this questions a million times a day, but considering intelligent autonomous systems, the response on this well known question can be a short standard answer like "Sorry, I cannot answer this question.", or the systems start a time-consuming optimization process. While the optimization process tries to integrate all available user information and additional information sources (e.g. online reviews, rankings, facts), the calculated result may not be better than a random choice in the eyes of a human user.

On the other side, we as humans often either decide intuitively in a few seconds, or ponder on the choices for a while, discuss them and come to a reasonable rational decision. Those dynamics taking place during human decision making is a common research topic in the behavioral science in psychology.

Considering the motivation, the example above, and stepping back mentally and taking a broader view, this dissertation investigates how human decision dynamics can be transferred into machine learning. This problem statement can be further decomposed into the following questions:

- How to model the interaction of emotion and decision making in agent architectures so that artificial agents are capable of generating consistent emotional states and displaying believable emotional behaviors?

1. Introduction

- Is there a mapping between a machine learning algorithm and a model of emotions which intuitively improves the human grasp of the machine learning progress?
- Is it possible to model affectively controlled decision preferences within a machine learning framework?
- Can we adjust a multi-objective machine learning algorithm without introducing unintended human biases?

To answer these questions, this dissertation demonstrates machine learning algorithms which are extended with psychologically informed components. While the extensions not always deliver optimal decision policies, they enable the artificial agent to act more anthropoid and thus more believable and trustworthy. It might be a nontraditional approach not to calculate the optimal decision strategy. But as the human successfully has proven, also partially sub-optimal solutions lead to proper strategies in the long run.

1.3. Contributions and Scope

In this dissertation, I propose three directions of combining human values with machine learning algorithms in order to give necessary elements for the development of personality in intelligent systems.

First, I investigate psychological models and foundations for the integration of artificial emotions into technical systems. This results in simulations of machine learning agents able to *express* artificial emotions. Second, I extend the classical Reinforcement Learning framework with the ability to *control* it with an external affective state. This gives the learning agent the ability to develop specific preferences. Thirdly, the weighting for multiple objectives during decision making is strongly influenced by human factors and I present a method for *retrieving* these weights from human decision policies which implicitly preserve those biases.

Overall, this dissertation can be set into the context of Affective Computing. Affective Computing is an interdisciplinary endeavour between psychology, computer science, and cognitive science in order to simulate empathy. This requires that the machine is able to interpret the emotional state of humans and that it can respond to it accordingly by generating and expressing artificial emotions. In order to generate those artificial emotions the agent has to evaluate its current state and potential actions not only in terms of optimality, but also with artificial emotions. Therefore, in this dissertation several experiments are performed in which an agent interacts with its environment in order to achieve a goal. A popular learning algorithm which is widely used in computer science and engineering to let robots interact with the real world is Reinforcement Learning. Also, in this dissertation Reinforcement Learning (RL) was selected as the basis for most of the experiments, as

it represents a learning strategy that is also used by humans. Human decision making is generally a recursive process which (re-)evaluates changing external and internal decision variables. This principle of evaluating rewards which are basically external decision variables is also the basis of Reinforcement Learning. Additionally, Reinforcement Learning incorporates several aspects which are implicitly contained in a particular reward or value function. The value functions are used in RL to select optimal decisions (or actions) in already observed states while using trial-and-error methods in unknown states to gain experience. The decision variables are updated by rewards or punishments observed in each state. With this, Reinforcement Learning is a powerful decision making framework for artificial agents, delivering under certain circumstances optimal decision policies while reflecting basic components of human decision making:

- Uncertainty is reduced through exploration,
- experience is gathered by approximating state and action values,
- and preferences can be introduced by multi-objective techniques.

In Figure 1.1 these components are drawn in relation to decision making and Affective Computing. Decision making is related to the objectives, the experience, and uncertainty of the current scenario. The outcome depends on the policy followed by the decision maker and can be denoted as the extrinsic reward. Obviously, the perceived value of the reward

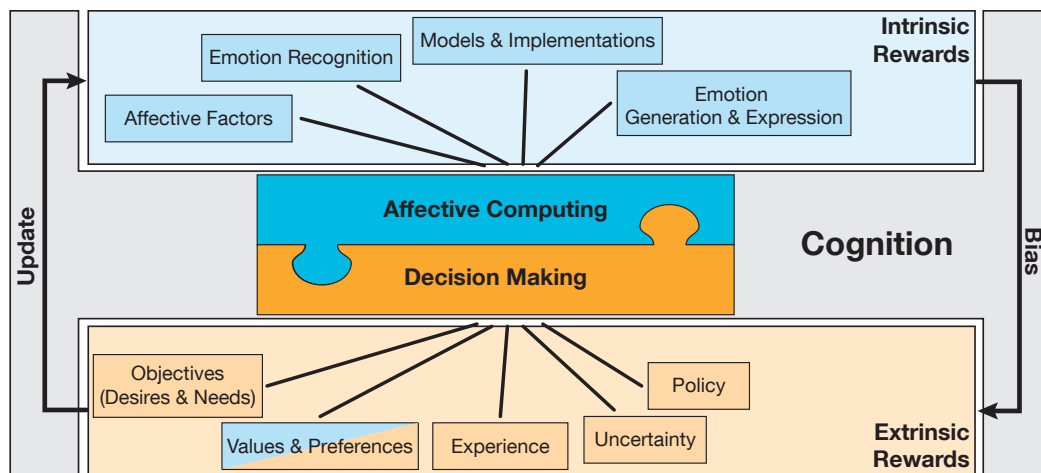


Figure 1.1.: Connections between decision making, Affective Computing, and cognition. The affective components are biasing the decision making process while outcomes are appraised and used to update emotional experiences.

depends on the (subjective) preferences of the decision maker. That means, the values and preferences for a decision outcome are related to both, internal and external factors. Consequently, the result of appraising a set of possible choices during decision making is a

1. Introduction

preference order according to internal values. Since these internal values and weightings for specific properties of a decision are the result of an affective process, the box *Values & Preferences* in Figure 1.1 is split into two halves, representing in this way the affective impact on this component. It is often assumed that humans are rational decision makers, however there is evidence that the human emotional system is biasing each decision (De Martino et al., 2006; Kahneman and Tversky, 1979). Especially in cases of incomplete information or uncertainty, people rely on heuristics and on learned rules of thumb. Those heuristics are subjected to former experiences and their related emotions, e.g. losses are correlated with aversive emotions while gains are associated with positive (appetitive) emotions. This fast evaluation of situations, events, or persons in relation to resulting emotions is often called *affective appraisal* in psychology. That means, emotions are a kind of *intrinsic reward* assigned to previous actions and outcomes of decisions. In case of Affective Computing, emotions are recognized or generated by an artificial agent during the interaction with humans. On the one hand, recognition of emotions should improve the agent's interpretation of human actions, while on the other hand artificial emotion expression is intended to enhance the communication abilities of the machine. The synthesis of artificial emotions and the interaction of cognitive and affective processes within artificial agents should replicate the intelligent behavior observed in humans and thus should improve the quality and believability of their expressions. However, there is still a lack in general architectures for the modeling and integration of emotion theories to technical cognitive architectures (Rodríguez et al., 2016).

The topic of expressing emotions and using them as an indicator for the performance of the underlying machine learning algorithm is part of this dissertation. In the following, Chapter 2 first provides an introduction to related topics. Then, in Chapter 3, three combinations of artificial emotion models with machine learning algorithms are presented. Chapter 4 sheds light on a method for biasing decisions in Reinforcement Learning in relation to an affective state. Finally, in Chapter 5 an algorithm to retrieve biased decision weights from human policies is described.

2. Background

In Affective Computing, many research directions exist and they are related to other scientific fields like psychology, computer science, and neuroscience. In this background chapter, some basic directions of this still young field of research are introduced. Due to the breadth of related topics only a limited sub-set will be addressed. The focus lies on the affect generation and modeling aspect of affective preferences and biases. This follows the main functions of emotions as proposed in psychology: Humans use emotions for communication, to adjust their motivations, and to guide attention-direction.

Affective Computing and sentiment analysis are inextricably bound to each other, and the detection of human emotions in given audio, video, or text data is an essential method for mutual communication in human-machine interaction. While this aspect of detecting emotions and sentiments in data is not in the focus of this dissertation, however it is a relevant and active research direction, and a comprehensive overview is provided in (Cambria, 2016).

As the development of affect-sensitive systems is intertwined with the century-long psychological research on human emotions, an overview of basic emotion theories relevant for Affective Computing is given. Besides the introduction of these emotion theories, three system theoretic models of emotions are described. Next, an introduction to Reinforcement Learning is given as almost all experiments in this dissertation are based on this machine learning method. Reinforcement Learning was selected as it represents a learning technique which is inherently used by humans, and nowadays RL is also a standard machine learning approach. The basic reward mechanism of the classical Reinforcement Learning framework is then extended by the concept of multiple objectives and reward shaping. Reward shaping is a method to guide learning and introduce preferences into the learning process. Finally, the additional topics of preferences, human values, and affective states in RL are briefly introduced.

2.1. Theories and Models of Affective Computing

Modeling of emotions is an interdisciplinary endeavor between psychology and computer science (Picard, 1997). There are two goals for improving the models of emotions: achieving a better theoretical understanding of human emotions and the enrichment of artificial agents with an affective core for understanding and generating human-like emotional expressions and reactions. The latter, the application of emotions in technical systems is a primary focus for computer scientists and the first goal is more theoretical and is mainly

2. Background

investigated by psychologists (Broekens, 2010; Reisenzein et al., 2013). Both goals need profound and faithful computational models of emotions. On both perspectives – psychological and technical – the best way to achieve a deep theoretical understanding of mental processes is the attempt to simulate (or synthesize) them in artificial agents (Boden, 1988; Rodríguez et al., 2016). This may also help to overcome a still existing problem in psychology: It is difficult to unify a general accepted theory of human emotions. With simulations, there exists a way to generate a lot of experimental data which may help to empirically validate the models.

Today, we distinguish three main types of emotion theories (Fontaine, 2013). They are the basis, except for a few specific cases for most of the existing computational models:

1. **Discrete emotion theories** base on the work of William McDougall and Charles Darwin and were taken up in the 20th century by Paul Ekman (1992) and Robert Plutchik (1991). The key assumption of this theory is that there exists a fixed number of discrete basic emotions, which are also recognized to some extent across different cultures (Russell, 1991). These emotions are the response to mainly *hard-wired* programs (often called affect programs) we have evolutionary learned. The most-well known theory is that of Ekman, which describes the existence of seven basic affective programs: happiness, surprise, contempt, sadness, fear, disgust, and anger. These programs are activated by suitable perceptions or appraisals and generate appropriate physiological reaction patterns and particular bodily (e.g. facial) expressions. Ekman's theory was extended by allowing blends of basic emotions (e.g. distress supports sadness), in order to support more complex emotions and different intensity levels. In the course of this extension, also the definition of the affect programs were modified allowing one affect program to modify the parameters of another program. This was an important step, since emotions are subjected to strong interplays, which could now be correctly taken into account. Since Ekman's and Plutchik's first publications, there is still no strong and widely accepted empirical evidence for their theory in psychology. However, basic emotions are often used and have broad acceptance in computer science and engineering, and therefore more and more evidence for this theory is gained in these domains.
2. **Dimensional or constructivist theories** represent the full spectrum of emotions with a small number of continuous dimensions. A first model, which is basically still used in more modern models was formulated by Wilhelm Wundt (1897). He has modeled the emotional experience of subjects with the dimensions pleasure–displeasure, arousal–calming, and tension–relaxation. The dimensional theories are often used in psychological studies, as the subjects can report their own affective experiences on the given dimensions. The subsequent derivation of a model gets simplified.

Russell and Mehrabian propose two very different dimensions in the affective space (Russell, 1978; Russell and Mehrabian, 1977). The first bipolar dimension is valence

2.1. Theories and Models of Affective Computing

on which negative and positive emotions are counterparts (cf. Figure 2.1). Valence is a measure for how content a person is, and high values indicate happiness or gratification (pleasant emotions), while low values correspond to unpleasant emotions like boredom and anger. On the other dimension low-arousal and high-arousal affective experiences are opposed to each other. Arousal corresponds to the state of how agitated or excited a person is, and the related intensity value is independent of whether it is a positive or negative excitation. High values are results of rage or surprise, while low values corresponds to boredom or well known situations. In Figure 2.1 discrete emotion terms are placed on a circle around the origin in the dimensional space.

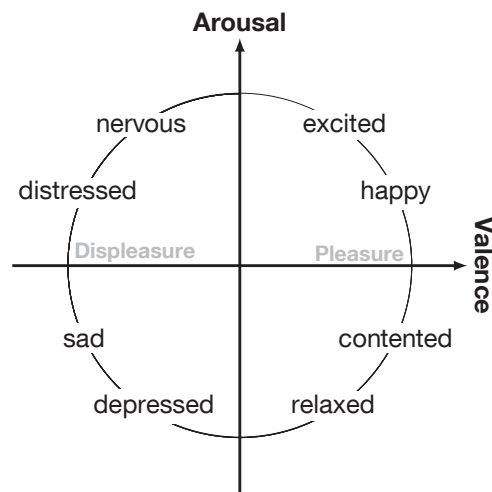


Figure 2.1.: Simplified Valence-Arousal space of Russell's dimensional emotion model

Psychological constructionists integrate dimensional and basic emotion theories and develop models assigning discrete emotion terms to specific combinations of basic psychological *ingredients* (the underlying dimensions). There exist other dimensional models sharing also two (or three) dimensions while using slightly different dimension names, but assessing similar psychological phenomena. In most two-dimensional models, the *arousal* dimension is common, but the *valence* dimension is used interchangeable with the dimension of *displeasure-pleasure*. In this dissertation, the Valence-Arousal space (VA-space) is used to describe affective phenomena in two dimensions. Sometimes, the models include a third dimension, which relates to power (dependency on someone else, also called potency, control, and dominance (Mehrabian, 1996)). The dimension of power is often found in large studies with large samples and a focus on interpersonal emotion terms, and is often not obvious in studies investigating intrapersonal emotions.

2. Background

3. **Appraisal theories** assume that perceived events are either positive or negative. This classification of events is the result of a comparison between the event's consequences and one's desired goals. Hence, an event is either positive if it is goal-congruent (fulfills a goal/desire) or it is negative if it is goal-incongruent. This kind of personal appraisal process was first explicitly described by Lazarus (1966) and has been kept up by succeeding appraisal theorists, e.g. Moors et al. (2013). In those following theories, emotions are characterized as processes interacting with each other and basic subsystems of human functioning (called components). Events are classified according to their goal-conduciveness and are appraised in relation to current motivational states. The most elaborated and systematic representatives of the modern appraisal theories are the Component Process Model (CPM) which was developed by Klaus Scherer (2010), and the OCC model (Ortony et al., 1990).

At a first sight, the different theories seem to be incompatible, and additionally, there is some ongoing debate between the authors of the different theories. Therefore, in the following their basic theory and weaknesses will be pointed out and their relevance from an engineering perspective is stated.

The discrete emotion theory mainly ignores neural correlates of emotions in the brain, and therefore the interrelation between different emotional states and mood is not well represented in the theory. Furthermore, cultural differences in the contextual sensitiveness causing the expression or suppression of particular emotions are not taken consistently into account (Sauter et al., 2010; Ekman et al., 1969). Also, in applications based on discrete emotion theory the contextual dependence of emotions is often neglected, resulting in improper emotional reactions of the system or bad emotion recognition performance. There is a general agreement, and findings of several studies show that at least four basic emotions can be cross-culturally utilized: joy, sadness, anger, and fear (Sauter et al., 2010; Pell et al., 2009; Scherer et al., 2001; Ekman et al., 1969). Depending on the final application, this limited set of emotions can be seen as a constraint in the development of cross-culturally accepted artificial agents.

In dimensional theories, the way a particular model is interpreted is crucial. Often a model is seen bidirectionally, and distinct emotional expressions are mapped onto the dimensions of the model and vice versa. Generally, the models are created in studies where subjects rate their feelings on given scales, and a dimensional representation or principal components are determined afterwards. In technical implementations, such models are then used the other way round: The computational model calculates values for each dimension and they are used together with a dimensional theory to translate them into distinct emotional expressions. From an engineering perspective this is an elegant and efficient way to map dimensional values to distinct emotion expressions, but often psychological evidence and empirical studies are missing. Despite this missing link, in Chapter 3, we will investigate such an approach and see the benefits of dimensional

2.1. Theories and Models of Affective Computing

emotion models in artificial systems.

Appraisal theories, however have been taken up by most approaches and demonstrators of artificial emotions in technical systems. They have been used to investigate how emotional *cues* can be generated and if they can be used to make an agent more socially intelligent and believable. One remaining question of appraisal theories in the engineering context is the detection and classification process of events and their elicited emotional reactions. It is complex to describe and to concretize how an emotional reaction is caused by an event. Often, there is a large space of different situations and related events causing the same emotional reaction. The available techniques for processing and capturing real world situations and creating suitable cues which cause appropriate emotional reactions in a technical system are not sufficient. The following example, provides a brief impression of this issue: A person reacts with joy on the arrival of a friend. The friend causes this particular feeling of joy, e.g. when he/she arrives at the person's home, but also, when he/she calls you just after landing, or if the person just receives a message of someone else, that the friend has arrived safely at the airport. In all of these examples, the situative context and the way the message of the friend's arrival is perceived or received by the person is significantly different, but the elicited feeling of joy is comparable.

Concluding remarks on emotion theories: *Basic and dimensional emotion theories seem incompatible from a psychological viewpoint. But several components like the component based appraisal of events, the mapping (dimensional reduction) to distinct dimensions, and the expression of dimensional patterns with basic emotions provide necessary foundations for a successful implementation. Most model descriptions are very clear on the definitions of components, signals, and appraisal variables for the different emotions. What is missing, however, are the formal definition of input signals and their formats. For applying human models of emotions in technical system, a mapping between sensors and system states as well as the inputs of the psychological models have to be defined. Such a mapping is often called appraisal-derivation model.*

2.2. Affect Generation

Affect generation means the ability of an artificial agent to generate synthetic emotions, and optionally to express them via verbal or nonverbal behaviors. In engineering, this topic plays an important role in building so called *Embodied Conversational Agents (ECAs)*. ECAs can be software based systems (like virtual characters) or physical robot platforms capable of expressing verbal or nonverbal emotions. The primary goal of development is the enhancement of the human-robot/computer interaction. As already mentioned, the artificial expression of emotions is not researched in this dissertation. Instead, it focuses on the underlying models of emotions which are necessary to generate the signals for the expression module of Embodied Conversational Agents. So, the structural relationship describing the mapping between appraisal variables, motivational states, and specific discrete emotions is investigated in the following. The result is a so called appraisal-derivation model, which transforms state representations into appraisal variables.

As the appraisal theory is the most influential theory of emotion in Affective Computing today, this section details the two prominent models: the Component Process Model (CPM) and the OCC Model. Both models deliver descriptions about representing emotions with formal languages and rules to manipulate them by well-defined operations. Additionally, in Subsection 2.2.3 the *Zurich model of attachment and fear* is introduced, as it represents a very well-defined and system theoretic oriented model of two basic feelings which are strongly related to emotions.

2.2.1. Component Process Model

The Component Process Model as introduced by Klaus Scherer (2009, 2010) is the result of his functional analysis of emotions in studies with individuals. It models the emotion elicitation process of human emotions and can be used to predict human like emotions for specific events. The dynamic and recursive model is based on appraisal theories in psychology. That means a relevant event (and its potential consequences) is appraised according to a set of criteria. Those criteria are not only considered simultaneously, rather they are ordered in a multi-layer principle creating the concept of appraisal components. There are four central appraisal objectives:

1. How relevant is the actual event for me?
2. What are the consequences/implications of this event and how do they affect my personal well-being and/or future goals?
3. Can I cope with the consequences?
4. How significant is the event for my self-concept and does it breach social norms and values?

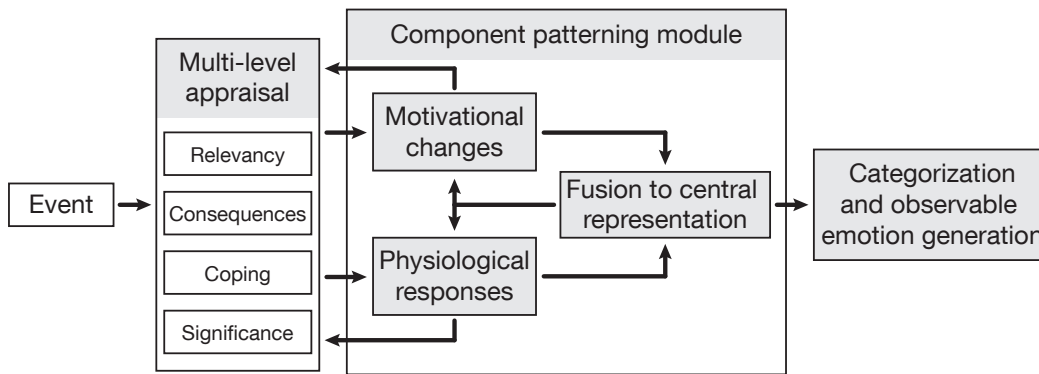


Figure 2.2.: Basic flow diagram of the dynamic and recursive architecture of the Component Process Model (CPM). A salient event is appraised and causes motivational and physiological responses which are finally expressed as emotions.

An evaluation for each objective is achieved by underlying *Stimulus Evaluation Checks (SECs)*. Depending on the particular check simple and fast or complex calculations are performed to create consistent representations of incoming stimuli. These calculations can additionally be grouped into different levels of processing. They range from low-level neural circuits, like modulated brain functions (e.g. increased activity of the amygdala in case of salient events) delivering unconscious cues for an event, to more complex considerations which involve memories, associations, and knowledge (Scherer, 2010). It should be noted, that the outcomes of the Stimulus Evaluation Checks are subjective as they base on the individual's inference which takes memories and knowledge into account. Furthermore, SECs are influenced by mood states, cultural differences, and group pressures (Manstead and Fischer, 2001). Therefore, in technical systems the simulated SECs should be seen as one representation of many possible implementations. The implemented appraisal process modeled with SECs is subject to the designer's bias and often bases on empirical findings which are not necessarily generally applicable.

The sum of the individual checks is the appraisal result and triggers changes in the intrinsic motivations and physiological responses. In this step, specific reaction patterns depending on the appraisal are simultaneously caused in different physiological and motivational subsystems (component patterning module). A central integrated representation is generated by taking together the appraisal results, the motivational changes, and the physiological responses. This central representation becomes in parts conscious and is assigned to fuzzy emotion categories. In Figure 2.2, the arrows indicate that all consecutive steps of the CPM recursively feed back their results and thereby influence the upstream components. The individual SECs and some of the feedback loops are omitted in the figure as they are not directly relevant for the practical implementation of the CPM.

From an engineer's viewpoint, the CPM has the benefit that it can be implemented as a decision tree and that computational resources can be optimized through the hierarchical

2. Background

order of the SECs. In cases where top level SECs calculate a result obviating the need for further checks in deeper levels of processing, a speed up of the appraisal process can be achieved. For example, if the relevancy check concludes that a stimulus is not goal relevant, the remaining SECs are obsolete and not carried out. Additionally, the hierarchical structure of the Component Process Model allows the integration of it into more complex cognitive architectures, such as a *model of affect and cognition* (cf. Section 3.2 and (Norman et al., 2003)).

In the following paragraphs two state-of-the-art implementations of the Component Process Model are described. The descriptions should point out the general ideas and the differences of the implementations. This supports and gives further understanding to the design decisions made for the implementation of the CPM in Section 3.3 of this dissertation.

WASABI In the *Affect Simulation for Agents with Believable Interactivity (WASABI)* architecture (Becker-Asano and Wachsmuth, 2010), the central point is the mapping of the appraisal outcome to the three-dimensional space of *Pleasure, Arousal, and Dominance (PAD)*. In this mapping, the dynamic of the three feelings is modeled so that a continuously changing and self-rebalancing internal state of the artificial agent is generated. The input of the mapping step is the result of appraised internal or external events.

Generally comparing the WASABI architecture with the CPM, the appraisal process and the categorization step is combined in the WASABI architecture. In the CPM, the appraisal process consists of multi-level appraisals capable of doing very sophisticated and detailed evaluations (the Stimulus Evaluation Checks). On the contrary, in the WASABI architecture this is represented with a less complex belief-desire-intention (BDI) approach, which models rational reasoning about possible events in the current situation. The BDI approach follows the work of Rao and Georgeff (1991), but is limited in the way it evaluates the normative significance of the events. It creates representations about the agent's goals and plans and evaluates current events and previous expectations in order to generate new expectations for achieving a desired goal. These expectations are used in the *reactive appraisal sub-module* to assess an event's intrinsic pleasantness by comparing the event's actual consequences with its calculated expectations. In a second sub-module (*cognitive appraisal*), events are classified according to their goal conduciveness as well as the agent's abilities to control the possible events of the current situation. Finally, a simple coping strategy is implemented in the *cognitive reappraisal module*, which lets the agent leave if it gets very angry and come back after it has calmed down again. Basically, with these appraisal checks the WASABI architecture addresses three of the four appraisal objectives given in the original CPM (the normative significance check is missing).

These appraisal checks generate so called *emotional impulses* which are represented in the PAD space and used to update the aware emotion of the agent in the subsequent categorization module. The categorization module of the WASABI architecture simulates emotion dynamics by shifting a reference point within the PAD space towards the newest

emotional impulse. The strength of this shift depends on a simulated general mood state, while simultaneously the reference point is pulled back to the origin of the PAD space with the simulated dynamic of a spring-mass system. This simulated dynamic in the mapping process of the WASABI architecture has no direct equivalent in the CPM, but covers some features that are described by Scherer regarding the component patterning module and the categorization module.

In summary, the WASABI architecture is psychologically well elaborated and implements basic aspects of the Component Process Model. It has its restrictions and limitations in the depth of the appraisal process, and the BDI approach has to be especially setup for each domain.

PEACTIDM In contrast to most other computational emotion models, PEACTION connects cognition and emotion and thus represents a more complete theory of cognitive control in artificial systems (Marinier et al., 2009). The abbreviation stands for *Perceive, Encode, Attend, Comprehend, Tasking, Intend, Decode, and Motor (PEACTIDM)* and introduces a cycle consisting of these eight cognitive functions enabling the simulation of cognition in artificial agents. These functions are implemented in a general architecture for artificial agents called *Soar* (Laird, 2012). The *Soar* framework provides the basic functions a cognitive agent needs: Memories (both long-term and short-term), processing components that combine knowledge and perception, as well as motor systems (its acronym *Soar* stands for the main components and functions namely *State, Operator, Apply, Result*). Together with the PEACTION cycle, the agent is able to calculate information about goals, needs, relevance, and expectations for events and situations. By implementing the appraisal checks as proposed by Scherer, the *Soar/PEACTIDM* architecture allows the calculation of so called appraisal frames for each situation and instance of time. The frames are subsequently segmented and assigned to categorical and linguistic emotion labels. An intensity value is also calculated by combining the numeric dimensions of the current appraisal frame using an intensity function. The intensity function is given by the average of multiple appraisal dimensions weighted by a surprise factor. The surprise factor is the result between the discrepancy from its expectations about the probability of a consequence of an event and the actual outcome. Furthermore, in the PEACTION architecture the categorical emotion is defined as the current feeling of the agent and the corresponding intensity value is called *feeling intensity*. Also the agent's mood is modeled by dynamically adjusting its dimensions according to the current emotion and a temporal decrease towards a neutral mood.

Compared to the previously described BDI approach, the *Soar/PEACTIDM* architecture allows a more general and extensible representation of the world (or domain) which, however, requires a complex description of the domain. The architecture also implements most features of Scherer's CPM and goes even further by integrating it to the cognition process of an artificial agent, and by including a simple mood model. As the original CPM does not provide sufficient details to implement all kinds of appraisals and a mood model, the

2. Background

PEACTIDM architecture introduces own constraints and assumptions leading to a slightly different underlying appraisal model.

2.2.2. Ortony, Clore & Collins Model of Emotions

As one of the most influential model, the model of Ortony, Clore, Collins (OCC) represents a structural model of criteria (appraisal variables) distinguishing between different emotions. The general structure of the model can be compared to a decision tree and therefore is often implemented in this way. The original model was proposed with only five positive and five negative types of emotions which could be classified according to a relatively simple and fixed scheme of causes (e.g. something good happened → reaction joy, happiness). Later the causes were formalized into different types and the appraisal mechanism got more detailed. In Figure 2.3 a shortened version of the extended model is depicted. The decision tree starts at the top level with a distinction between events,

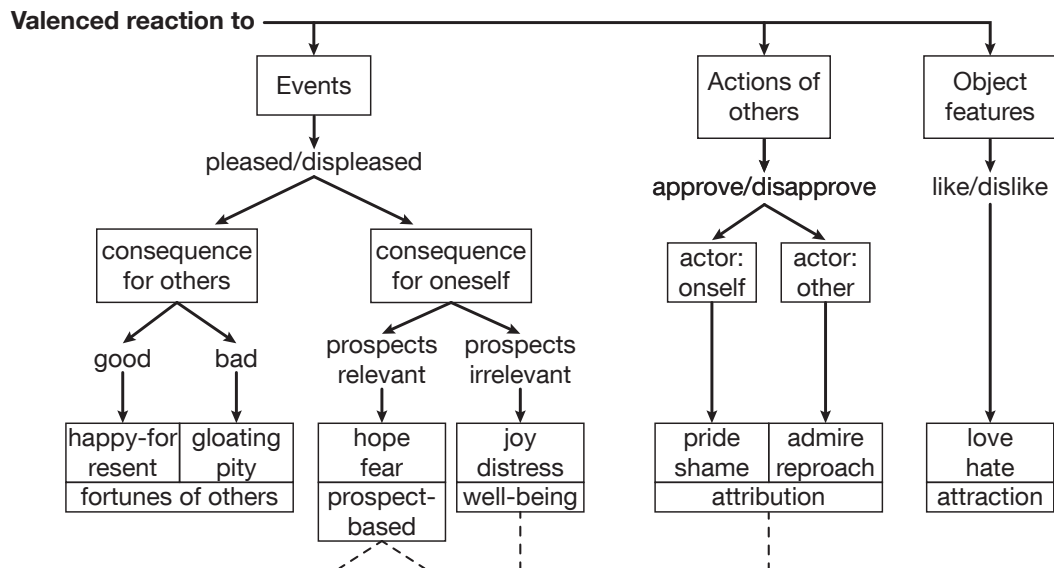


Figure 2.3.: Illustration of the theory of emotions according to the findings of Ortony, Collins, and Clark (OCC model). Events, actions, and objects are hierarchically appraised according to specific categories in relation to goals and motivations (adapted from Ortony et al. (1990)).

actions, and objects. In these three main branches, the emotions are classified in terms of (1) pleased or displeased events, (2) approved or not approved actions of others, and (3) subjective aspects of objects (like or dislike a thing). This general distinction is then further classified depending on the consequences for oneself or others, or in case of actions, depending on the actor of an action. Finally, emotions result as positively or negatively valenced reactions to one or another of these classifications. Altogether, including compound emotions the model describes 22 types of emotions in this way.

For this kind of emotion assignment, the agent needs a coherent and relatively stable value system of its environment. That means, the agent has to constantly appraise events, other agents or objects according to a (only slowly changing) system of goals, norms and its personal taste. For example, if someone responds with terror on seeing a mouse in his/her bedroom today, one generally expects that the person will also respond with terror on the next day as well. The basis for such a comprehensible expectation is a consistent system of values. Without such a consistent system of values that represents the basis of the agent's reaction, most of the resulting emotions and actions of the agents won't make sense to a human observer.

The definition of a consistent value system is not the only issue of the OCC model. Ortony et al. describe their model as *computational tractable* and compared to other psychologically informed models this account seems right. Attempting to implement the model ad hoc quickly reveals ambiguities which require workarounds to solve them. For example, the OCC model is often understood as an inheritance diagram by computer scientists and thus implemented in this way by Steunebrink et al. (2009). A final categorized emotion is then seen as a specialization of a parent type plus some subsequent child types (e.g. distress is specialization of a displeased event plus a relevant consequence for oneself). Steunebrink et al. also interpreted the OCC model in this way, but proposed a new logical hierarchy with an additional temporal order of the elicited emotions. This temporal ordering of the decision tree is new, and a corresponding description is missing in the original publication of the model. However, the results of Steunebrink et al. reveal that such a temporal interpretation of the model improves its performance. Such heavily differing details in the implementations of the OCC model are the main drawback of the original OCC model, as they elude comparison of the various implementations.

The *Fuzzy Logic Adaptive Model of Emotions (FLAME)* developed by El-Nasr et al. (2000), and *A Layered Model of Affect (ALMA)* developed by Gebhard (2005) are the most popular implementations basing on the OCC model. As the structure of the OCC model can be easily transferred to implementations, there exist quite a number of OCC-inspired systems, e.g. *EM*, *HAP*, *FearNot!*, *EMA*, *Oz*, *MAMID*, and *Greta* (recent reviews of those implementations were written by Calvo et al. (2015, pp. 101-104) and Marsella et al. (2010)). In the following, the basic principles of ALMA and FLAME are illustrated as these two implementations are popular and show significant differences.

ALMA ALMA is a layered architecture of cognitive decision making that involves emotions. It comprises three affect types which are distinguished according to their temporal characteristics. Short-term reactions are expressed as emotions, a mood state represents the medium-term-affects, and the personality of the artificial agent models long-term affects. ALMA is mainly used in conversational agents in order to improve the dialogue by adding non-verbal and verbal emotional cues. The core module modeling the affects is the *EmotionEngine* which bases directly on the OCC model. The OCC model is additionally combined with the five factor model of personality (McCrae and John, 1992). According

2. Background

to the theory of the OCC model, each of the 24 emotions is generated fulfilling a set of specific *emotion eliciting conditions (EECs)*. In case of ALMA, which is mainly focused on dialogue systems the EECs are checked upon meta-information extracted out of the current dialogue. But they can also be extracted from observations and situational appraisals.

The developer can assign personality profiles to the model, which is in turn used by the five component model of personality to alter the emotion intensities accordingly. Depending on the distance between the origin of the PAD space and a point corresponding to a mood state, the intensity is represented. Similar to the WASABI architecture above, all active emotions calculated by the EmotionEngine are used to drive the dynamic of the overall mood state. The mood state is altered by an elicited emotion which is mapped into the PAD space according to a fixed look-up table. The emotion represented as a point in the PAD space is used to push and pull the combined mood state in a particular direction. In this way, ALMA maps appraisal results to both, discrete emotion labels and to a dimensional core affect. The discrete emotion label is outputted in each cycle. The label is a high level representation of the internal appraisal result and can be used to drive non-verbal expressions of the agent. Besides these short-term expressions, the mood state lasts longer and is more stable and is thus used for more general behavior regulation (gazes, gestures, and postures).

In summary, ALMA exploits the OCC theory in terms of a hierarchical appraisal process and implements an additional dynamic mood model in order to simulate the temporal characteristics of emotions. The main drawback of ALMA is the strong focus on conversational agents, and the fact that subjective appraisal rules (EECs) for the characters have to be (manually) provided. This limits the general applicability of the approach, especially in scenarios where only implicit communication takes place.

FLAME The Fuzzy Logic Adaptive Model of Emotions (FLAME) bases on the OCC theory, too, but is also influenced by Roseman et al.'s (1990) event-appraisal models, and Bolles and Fanselow's (1980) inhibition model. Furthermore, the model was extended with a learning component which exploits inductive machine learning techniques like Reinforcement Learning to learn the impact of events, a probabilistic approach to learn action patterns, and a heuristic approach to learn specific properties of actions.

The authors of FLAME mainly criticize the lack of a complete picture of the emotional process in the existing appraisal models and their limited adaptability to dynamic situations. Therefore, they use fuzzy logic to represent emotions as fuzzy sets and fuzzy rules to map them to distinct emotions and behaviors. The use of fuzzy logic enables smooth transitions between the calculated emotions and allows a mixture of triggered emotions, which is not described in the original OCC theory. The elements of a fuzzy set are influenced by the perceptions, the expectations and the emotional state of the artificial agent (El-Nasr et al., 2000). Additionally, FLAME utilizes the aforementioned machine learning techniques to learn connections and relationships between objects, events, and expectations to increase the adaptability of the agent in dynamic environments.

Besides the emotional and the learning component, the FLAME model also includes a decision-making component which completes the architecture. By using these three major components, the agent perceives an external event and simultaneously passes it to the emotional and the learning component. In the emotional component, the appraisals are calculated while also (newly) learned expectations and event-goal associations are taken into account. Then the fuzzy rules are applied and a mixture of artificial emotions is outputted. A subsequent filtering step is applied to limit the emotionally influenced actions and behavior according to current motivational states of the agent. The motivational states in turn, are also influenced by the simulated emotions so that a motivational and emotionally driven regulation of the actions and behavior can be achieved. For example, while an agent is acting on fear it suppresses other motivational states (like charging the batteries) and tries to reach a safer place. Afterwards at the safe place, the appraisal of the situation changes accompanied by easing emotions and accordingly changing motivational states (the agent starts to check the battery state regularly). Besides this action and behavior regulation, also the calculated emotions are filtered using intensity based suppression of weaker emotions (e.g. a strong feeling of sadness suppresses a light feeling of joy).

In their experiments, El-Nasr et al. show that the adaptability of the artificial agent is a critical aspect of its believability during human interaction. In FLAME they use their learning component to achieve greater adaptability, but learning still depends on predefined objects and events which have to be recognizable by the agent. Also the necessary user feedback which is used for the appraisal of user actions is predetermined. Furthermore, FLAME only implicitly addresses personality through hand-tuned parameters and heuristics used in the calculations. Incorporating personality into FLAME is difficult due to its structure, however it is an essential and missing feature of the model. In spite of these weaknesses, FLAME is currently one of the most elaborated emotion model, which bases on the appraisal theory and incorporates memory, learning, and a temporal processing structure.

2.2.3. Zurich Model of Social Attachment and Fear

Besides the common emotion models (e.g. CPM or OCC) there exist different psychological models which were derived from empirical studies. The German psychologist Norbert Bischof proposed in his articles on the *Zurich Model of Social Motivation* (1989; 1975) a model of the human motivational systems. It is one of the most applicable psychological models of social motivation, since Bischof describes his model in a psychological manner and in a system theoretic way. Three basic motivational systems can be distinguished into (1) the security system, (2) the arousal system, and (3) the autonomy system. As the systems are precisely described in a system theoretic way, the model qualifies for a technical implementation. Basically, the structure of the model consists of three interconnected feedback loops comparing internal set points with the actual motivation levels. The simulation model receives its information about the surrounding environment through so called *detectors*.

In the following, the basic concept of the Zurich model is described, while further im-

2. Background

plementation details are postponed to the experiment section (Section 3.1). As already denoted, the general model structure consists of three subsystems. The first two systems – the security and arousal subsystem – will be explicitly described in this dissertation, the autonomy system is not considered in detail. Although the autonomy system plays a key role in the Zurich model as it controls the set points of the security and arousal subsystems, it was not simulated in the experiments and the set points were set manually to keep exact control.

Based on empirical studies, the model describes the behavior and actions performed by a child in the presence of surrounding objects. These objects could either be things like an ordinary ball or other humans. The recognition and classification of these objects thereby is not a part of the original model. Instead, so called detectors are assumed to assign two values – relevancy R^i and familiarity F^i – to each object i . In addition to these two values each object has a position z^i . The complete Zurich model is drawn in Figure 2.4.

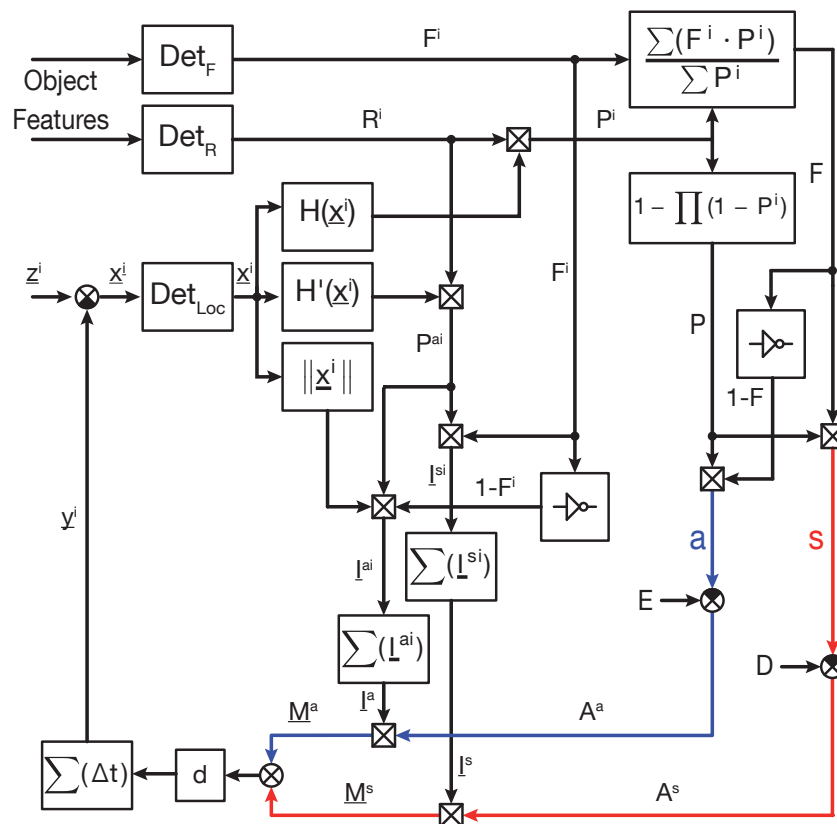


Figure 2.4.: Schematic realization of the Zurich model; the security value and activation is drawn in red and the arousal value and its corresponding activation is drawn in blue.

Although there is no extensive description of implementation details of these detectors, their characteristics are clearly defined in the original publications of Gubler and Bischof

(1989; 1991). The relevancy detector (Det_R) can be thought of as a sensor detecting the actual relevance of an object or person with regard of oneself. It is a measure of the potency of an object or person to be able to alter the situation. For example, in studies people showed that this value is highest for mature adults, lower for less potent siblings, and lowest for physical objects like a doll. In case of technical systems, the relevancy detector can be represented as an image recognition system capable of distinguishing adults, children, objects. Further, in more abstract examples like the simulated experiments in this dissertation the relevancy detector can be implemented as a mechanism evaluating the impact of an object or event on the situation. Large impacts result in high relevance values and negligible events cause small relevance values. The potency of an event has to be predetermined or learned with a suitable algorithm.

The familiarity detector (Det_F) senses the familiarity of a given object. Its output value is high if a lot of redundant information of an object are observed, but low if new features and information of a object are perceived. Considering again a technical scenario, the familiarity sensor can be implemented as a face recognition system with an attached database to record the detected faces and the duration of interaction (a similar approach is also possible with physical objects).

The third detector (Det_{Loc}) senses the physical and psychological distance of persons and objects. Generally, the distance measure corresponds to the physical distance between an object or other person and oneself, and is calculated e.g. by the Euclidean distance measure. Additionally, the Zurich model supposes that the distance is biased by psychological mechanisms, but does not deliver a concrete description of those mechanisms. Therefore, in the current implemented models, the psychological bias is modeled using hyperbolic functions $H(x^i)$ which let the proximity values exponentially decay.

Together, these three detectors create the security subsystem, which summarizes the three outputs of the detectors to a variable s which monotonically increases with the value of familiarity, relevance, and proximity (Gubler and Bischof, 1991). In the model, this is calculated by weighting the familiarity value F^i with a potency value P^i which is calculated by the relevance value multiplied by the distance. The arousal value a is determined by multiplying all potency values with the inverse familiarity values $(1 - F^i)$. Further implementation details are given in Section 3.1.

Both, the arousal and the security value are then compared with two set points, enterprise E and dependency D , respectively. The comparison is done by calculating the difference between the actual value and the set point, which results in a discrepancy called activation (A^s and A^a). Further, so called incentive vectors I^{si} and I^{ai} are determined by a scalar multiplication of the vectors pointing in the direction of arousal or security sources. The incentive vectors are superimposed and weighted by the corresponding activation values resulting in so called momenta, M^s and M^a . Finally, the security momentum and the arousal momentum are combined and damped by factor d , which smooths the movement vectors and increases the stability. The resulting movement vector points into a direction where a security and arousal equilibrium can be achieved. By following the movement vectors, a self-regulating feedback loop emerges altering all preceding values (since they

2. Background

depend on the positions) and finally converges to a position with the highest security value and lowest arousal value.

In summary, compared with the appraisal theory based emotion models, the Zurich model represents a very basic model of the two feelings security and arousal. The appraisal results in the Zurich model strongly depend on the set points and the detectors where pre-determined objects have to be defined. Simultaneously, the clear description of the model and its detectors also allows the integration of the model to a wide range of applications while keeping the complexity low. Modifying the set points allows an adjustment of the behavior of the implemented agent. Since the output of the model are two feelings, a suitable mapping into emotion expressions (e.g. Borutta et al. (2009)) has to be found in order to use the Zurich model in an Embodied Conversational Agent.

2.2.4. Model Implementation

In a technical system, the way of implementation and the system structure is a critical point as the overall complexity often exponentially increases with each new feature, and stability or security issues come to the fore. Humans have two systems of information processing: A fast and unconscious system and a slower and reflective information processing mechanism (Norman et al., 2003; Kahneman, 2011). Often they are referred to as *affect* and *cognition*. Human beings perceive their environment in a cognitive and an affective way.

Our cognitive information processing interprets, understands, makes sense, and reflects our environment. It also remembers previous perceptions and stores new information. But simultaneously, the system of affect which is a parallel and inseparable component of our perception system rapidly assesses events according to their overall value. It categorizes the events into good or bad, safe or dangerous, desirable or undesirable, hospitable or harmful, and many more. Both systems can be neuroanatomically seen as distinct systems, but there is some evidence in neurology and psychology, that humans only function optimally if both systems closely work together (Damásio, 1994). Additionally, Marvin Minsky, a pioneer in artificial intelligence suggests that both systems – affective and cognitive – are essential in future smart systems (Minsky, 2006).

Norman et al. published a paper about affect and machine design describing a model of affect and cognition. They propose a three-level theory of human behavior basically applicable to the architecture of affective computer systems (Norman et al., 2003). The three levels are the *reaction* level, the *routine* level, and the *reflection* level. These levels enable a processing of the surrounding world in terms of affective evaluation and cognitive interpretation of the environment.

Figure 2.5 depicts the three levels including different blocks at each level representing abstract functions which are performed at the particular level. With increasing level, the complexity increases and more abstracted data is processed. At the reflection level, the most computational expensive algorithms and processes are grouped. As the level decreases, the complexity decreases and computations are performed faster allowing fast

responses to external events. In the following, the general idea of each level is outlined and set into context:

The lowest level – the reaction level – processes low level information, performs rapid reactions to the current state (such as reflexes), and controls motor actions. In this level, all processing steps and actions are fixed and no autonomous learning occurs. The processes of the reaction level look like simple if-then rules and are directly executed on new perceptions (e.g. raw sensor readings) of the agent. Executed rules are allowed to interrupt ongoing processes and actions of higher levels.

In the routine level, *routinized* actions are performed. In human physiology, the routine level is responsible for most motor skills, language generation, and other skilled and well-learned behaviors (Norman et al., 2003). It has access to working memory and more permanent memory to guide decisions and update planning mechanisms. The routine level receives the outputs of the reaction level and the sensory system, but is also adjusted with inputs of the reflection level. With some training, the reactions upon inputs of the reaction level and its corresponding actions can be inhibited or reduced (e.g. get used, at least to some degree to pain). It is the routine level, where most of the appraisal steps executed by an emotional model (e.g. the OCC model or the CPM) can be found (or should be implemented). The implemented appraisal rules can be adjusted by a learning process, which is triggered in case of a mismatch between expectation and observation. Such learning mechanisms are part of the reflection level.

The highest level – the reflection level – calculates the most computationally intensive steps. Reflection is a *meta-process* involving all available data in order to create own internal representations about the surrounding environment. It reasons about the environment and interprets the preprocessed sensory inputs of lower levels. The main task of this level is planning, problem solving, and reasoning about facts. Thus, the reflection level is the home of update functions for lower levels and contains learning and reasoning techniques suitable for planning tasks to adjust goals and behaviors.

Keeping these three levels of affect and cognition in mind, there is some more evidence that fast and hard-wired calculations are separated from more complex and slower ones in the human brain. In the work of LeDoux (1998), neuroscientific results show that the information processing in the brain is divided into the *low road* and the *high road*. The low

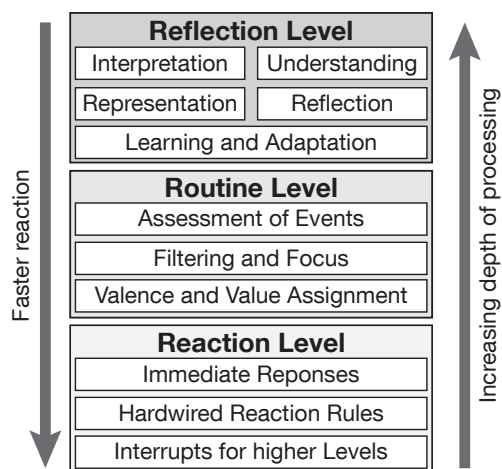


Figure 2.5.: Three level model of affect and cognition as basis for the implementation of affective artificial agents.

2. Background

road is a direct and fast pathway responsible for preconscious emotional processing and enables reflexes to events and causes low level responses like fear. While the high road is more indirect and involves more conscious and cognitive emotional processes. Compared to the low road the high road is much slower and more appraisal steps are performed until an emotional response is generated. Besides LeDoux's neural view, Kahneman and Tversky also propose a similar concept of two systems involved in our decision making process while affected by emotions (Kahneman, 2011). In the prospect theory *System 1* and *System 2* correspond to the *low road* and *high road*, respectively. System 1 reacts fast and without deeper cognitive processes while System 2 performs calculations and reasons about facts.

Based on the above considerations, the following technical implications result and create the basis for the three-layered model of affect. An affective artificial agent should be able to simultaneously sense its own state of operation and the surrounding (dynamic) environment. It should be able to compare system states and environmental changes according to an internal representation and react in cases where the expectations are not met. The reactions are indirect and cause automatic reconfiguration of appraisal and learning processes. This enables the agent to be aware of the own system state while not blindly following the most efficient way in achieving a goal. Such a supervision module should be completely separated from underlying operational tasks and should remain in operation in case of unexpected events. The priority of this module is more in the reliability than on fast computations which ideally fits to the reflection level. The reflection level is thus the place where a mood model and other mechanisms adjusting the setpoints of an emotion model should be implemented.

In the routine and reaction level below, more immediate functions are located. The routine level performs regular and complex calculations like updating models based on new sensory information. It also uses knowledge bases stored in memory for detecting known patterns in the sensory data. So, in the routine level the appraisal checks of the CPM and the update of the decision tree in case of an OCC based model should be implemented.

Finally, in the reaction level the most basic operations are hardwired to sensory inputs and corresponding outputs. It is the home of the sensors' preprocessing steps and provides functions for safely performing essential operations. Considering this level in the context of a model of emotion, some fast and expressive reactions should be implemented in this most prioritized level. Such reactions are e.g. pain and fear responses.

Concluding remarks on the state of the art of implemented affective models: *In this section, a comprehensive overview of existing models of emotions from an engineer's perspective has been presented. Such an overview can never be complete and one might miss one or another model. The selection of the proposed models was done according to their impact on the current research topic of affective computing. One important model, which was not extensively discussed above is the Belief, Desire and Intentions (BDI) software model. BDI models are often used to implement emotion models, but the principles*

of BDI models are the research results of human practical reasoning (decision theory) and are not related to any research results of emotion research. Although the presented models (e.g. CPM, OCC) above can also be formulated as a BDI structure (Steunebrink et al., 2012; Adam et al., 2006), such formulations unnecessarily increase complexity. In case of the WASABI architecture, the underlying BDI software model is intertwined with the corresponding Component Process Model which substantially increases the effort to adapt the WASABI architecture to other scenarios. This is similar with PEACTION which is integrated into Soar (Laird, 2012) and admittedly is an elegant way of connecting cognition and emotions, but the accompanying increase in complexity is out of proportion. Additionally, a remaining issue of PEACTION are the predetermined appraisal values without the possibility to dynamically adjust them. This problem is in turn addressed by FLAME, which is able to learn new expectations and relations between objects. FLAME uses external feedback to adapt its emotional reactions. This results in increased believability of the agent and improves the overall communication capabilities of the system, but simultaneously requires user interaction giving feedback. Since external feedback is not available in the considered scenarios of this dissertation, only models with a fixed appraisal structure will be investigated according to their ability to improve the communication between the agent and a human observer.

Comparing the CPM and OCC model with the Zurich model of social motivation, all three describe a human appraisal process resulting in distinct emotion categories or values representing artificial feelings. The models are all based on psychological studies and their results were formalized in a way to create simulation models. The main weakness in all current artificial emotion models is the interface to the real world. Necessary feature detectors delivering the input data for the models either base on predetermined rules or vague definitions. A remarkable exception is the Zurich model, which models two basic human feelings as connected control loops, and defines the corresponding inputs in a way that technical equivalents can be found (e.g. using image feature extraction algorithms). Another feature inherently covered by the Zurich model is the temporal structure of the outputted artificial feelings. In case of the CPM and the OCC model, temporal properties are not directly included in the original model conceptions and therefore are individually addressed by the authors of applications of the two models. As a consequence, the implemented emotional agents often use hypotheses to fill gaps (like a missing temporal structure of a model or the feature detectors), resulting in subjective interpretations of how particular details of an emotion model work. There is no common solution except for performing psychological studies on missing details to solve this issue. As an engineer, one has to keep in mind that the replacement of missing parts of a psychological theory results in subjectively biased models. In such cases, the authors should clearly state their modifications and refrain from claiming the generality of their models.

As an extensive applicability and generality of implemented emotion models is still not possible due to the fact that there is no universally accepted theory of emotions, the artificial intelligence designers are well-advised to implement their selected theories in a well structured manner and to ensure maintainability. For this purpose, the three levels of affect and

2. Background

cognition were presented in this section and thereby represent a structure for the technical implementation of cognition in combination with affective elements. The structure ensures the logical order of affective elements while remaining open for replacements of inevitably used heuristics.

2.3. Reinforcement Learning

In this section, an introduction to Reinforcement Learning (RL) summarizes its basic ideas, and outlines principle techniques. Basically, there are two tenets behind the main ideas of Reinforcement Learning. One perspective is that Reinforcement Learning represents a sampled version of Dynamic Programming (see below), the other is that RL is inspired by behaviorist scientists. In behaviorism, the central foundation is classical and operant conditioning. Classical conditioning is a learning process in which two stimuli are paired. The first stimulus is an unconditioned stimulus which results in an unconditioned action (often a natural action). The second stimulus is a neutral stimulus which originally results in an unspecific action. Ivan Pavlov, a behavioral scientist, figured out that an unconditioned stimulus can be paired with a neutral stimulus. After the pairing, the originally unconditioned stimulus becomes a conditioned stimulus which triggers a specific action (often related to the neutral action). Pavlov and Anrep (2003) demonstrated this behavior in an experiment with a dog. Each time food was given to the dog a bell sounded. After a few repetitions the dog learned the pairing between the stimulus of the bell sound and getting food. With the bell sound as conditioned stimulus, the sound itself caused that the dog started to salivate. A similar relation, the operant conditioning was discovered by Burrhus F. Skinner who was an American psychologist and behaviorist. He found out that rats learn very efficiently in situations where they are rewarded or punished (Skinner, 1938). So the difference between classical and operant conditioning is that in the classical case a relationship between behavior and stimulus is learned, in contrast to operant conditioning where a behavior is learned which is sensitive to or controlled by the consequence of this behavior. Both classical and operant conditioning are the results of animal experiments. However, behaviorists all over the world have found similar behavioral patterns in humans.

This basic idea of a learning algorithm that adapts its behavior in order to maximize a reward signal from the environment was also used by Richard S. Sutton and Andrew G. Barto to develop a computational approach to learning from interactions. They built a framework in which an artificial agent learns to achieve a goal by only observing the consequences of its actions in its surrounding environment. In other words Reinforcement Learning corresponds to a learning process where actions are selected to maximize a (numerical) reward signal (Sutton and Barto, 1998).

2.3.1. Elements of Reinforcement Learning

In the Reinforcement Learning framework, an *agent* interacts with its *environment* using a finite set of *actions* a_t . Figure 2.6 depicts the learning process and shows the agent's interaction with the environment. At each instant of time t the performed action of the agent causes a state transition to the next time instance $t+1$. This state transition is characterized by the state description s_t and a corresponding reward signal r_t . In the basic RL framework, there exists a finite set of states and actions. Reinforcement Learning is classically represented as a Markov Decision Process (MDP). MDPs are used for modelling sequen-

2. Background

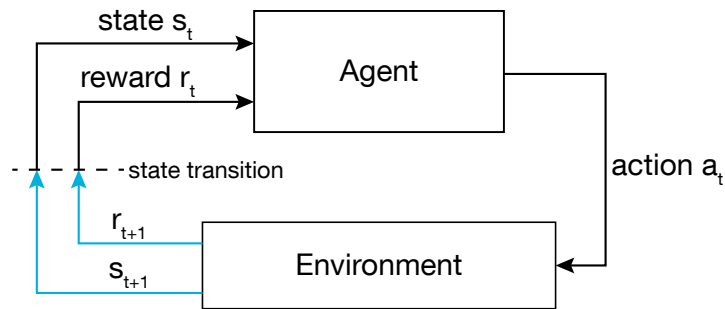


Figure 2.6.: An agent interacts (action a_t) with the environment while observing state information s_t and receiving a reward signal r_t (adapted from Sutton and Barto (1998)).

tial decision making in deterministic or probabilistic environments, cf. Bellman (1957). In the MDP setting the *Markov property* must be satisfied, which states that each transition between one state to another state only depends on the agent's action performed and the current state. The decision problem is then formalized as a state transition of the environment in consequence of the execution of an action determined by the intelligent agent. Each such transition can be rewarded with the observation of a scalar reward. Scalar reward signals are sufficient in most classic scenarios like simple robotic exploration tasks or economic decisions. However, there exist more realistic scenarios where the artificial agent has to consider multiple objectives at once, e.g.:

- Cooperation tasks where the agent can only achieve optimal results through cooperation while multiple possibly conflicting objectives have to be handled.
- Realistic exploration and exploitation tasks, where the agent has limited resources to explore while simultaneously exploiting known resources.
- (Traffic) Network control tasks, where simultaneously the highest possible throughput and minimal queue length shall be achieved.
- Reconfiguration problems e.g. in electric power systems, where reliability shall be maximized while minimizing power loss and costs.
- Optimizing a technically motivated objective function while at the same time ensuring that no ethical or subjective values are violated.

In order to state those decision scenarios formally, the following notation for finite Markov Decision Processes with single objectives is introduced:

- $s \in \mathcal{S}$ is a finite set of states,
- $a \in \mathcal{A}(s)$ is a finite set of actions, available in a particular state s ,

- transition probabilities $P_{ss'}^a = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$ which denote the probability that an action a in state s leads to the subsequent state s' (also known as *transfer function*),
- a reward function $R : \mathcal{S} \rightarrow \mathbb{R}$, assigning a reward r to a state s ,
- a deterministic policy $\pi(s) = a$ controlling the agent's actions in state s ,
- a probability distribution $\mu : \mathcal{S} \rightarrow [0, 1]$ over initial states, and a discount factor $\gamma \in [0, 1)$, discounting preceding rewards,
- and the notation for different instances of time with a subscript t .

This problem formulation extends to multiple objectives as follows:

- A vectorial reward function $\mathbf{R} : \mathcal{S} \rightarrow \mathbb{R}^q$ replacing the one dimensional reward function with a vector of q different reward components, so that each entry relates to one objective.

In this context, the agent tries to derive the policy $\pi(s)$ such that in each state an action is selected which maximizes the reward in the long run. Different policies can be valued using the so called *state-value function* $V^\pi : \mathcal{S} \rightarrow \mathbb{R}$, which denotes the expected cumulative discounted reward (the return) starting in state s and following the policy π . Assuming an infinite time horizon, the state-value function (for scalar rewards) is generally a sum over observed rewards

$$V^\pi(s) = \mathbb{E}_{s' \sim P_{ss'}^\pi} \left[\sum_{k=0}^{\infty} \gamma^k r(s_{t+k+1}) | s_t = s \right], \quad (2.1)$$

where $\mathbb{E}[\cdot]$ denotes the expected value given that the agent follows policy π and weights future values with the discount factor γ . Another important function in RL is the *action-value function*, assigning a value for each action in a particular state. Using the definition of the state-value function, the action-value function $Q^\pi(s, a)$ is defined as

$$Q^\pi(s, a) = \mathbb{E}_{s' \sim P_{ss'}^\pi} [r_{t+1} + \gamma V^\pi(s_{t+1}) | s_t = s, a_t = a]. \quad (2.2)$$

Both functions, $V^\pi(s)$ and $Q^\pi(s, a)$ can be approximated while an agent interacts with the environment. In case of the value function given in Equation (2.1) that means, that the agent has to infinitely often visit each state while maintaining an average for each state of the actual return that has followed that state. These averages would ultimately converge to the state's value $V^\pi(s)$. Such averaging methods are often called *Monte Carlo (MC) methods* as they average over many random samples of rewards until their value settles. Obviously, in case of large state spaces such averaging methods would be impractical. Therefore, in the action-value function $Q^\pi(s, a)$ below, the so called *Bellman optimality equation* was used to simplify the update (Sutton and Barto, 1998; Bellman, 1957). In

2. Background

the context of a MDP setting, the Bellman optimality equation describes the recursive relationship that the expected future discounted return in each state can be used to iteratively estimate this value for preceding states. This fact is consequently used in RL algorithms to approximate the state-value or action-value function in order to select the optimal action in a given state. There exist different strategies of selecting an action or approximating the value function. In the following paragraphs the most important techniques are presented.

Dynamic Programming

Dynamic Programming (DP) was introduced by Bellman (1957) and is still a state-of-the-art method to solve optimal control problems in cases where a system model is available and the Markov property is fulfilled. With the system model, the state transitions can be described with specific probability functions. These state transition probabilities in turn can be used to find the optimal policy for the given control problem. The available algorithms can be coarsely divided into two types: Value or policy iteration based methods (Bertsekas, 2000). Basically, they differ in the way the optimal policy is determined.

In case of value iteration, the value function for each state is recursively calculated on the basis of the given system model. With the converged value function, the optimal policy can be found by selecting the action in each state which transitions into the state with the highest subsequent value. Value iteration is intuitive and delivers optimal results, but it has two weaknesses: Although, the resulting optimal policy does not change anymore, the algorithm still improves the values for each state and therefore wastes time until it finally converges. Besides this undirected optimization process, the algorithm does only implicitly calculate the optimal policy by first determining the utilities for each state and then determining the optimal policy by selecting the highest expected utility values.

This time consuming calculation of the complete value function is overcome by policy iteration algorithms, which directly determine the optimal policy. Policy iteration means, that the algorithm iterates over policies. It starts with a random policy and computes the utility of each state given that random policy. In the next step, these utilities are used to determine a new optimal policy which is then used to compute the utilities again. This approach directly improves the final policy at each iteration and converges if the policy does not change anymore.

Both algorithms work fine and compute the optimal policy in given control problems assuming a sufficiently detailed model. In real world control problems, this assumption of a complete model can not be fulfilled which limits the applicability of DP. Additionally, the utility of DP is limited as the computational expense of the algorithms is quadratic in states and actions, and increases with the complexity of the models. Overall, Dynamic Programming provides the theoretical foundations for the understanding of classical Reinforcement Learning algorithms, as RL can be seen as an approach to approximate the optimal policy in cases where only an imperfect model is available or the models' complexity grows very rapidly.

Temporal Difference Learning

Temporal-Difference (TD) learning is a combination of Monte Carlo ideas and Dynamic Programming ideas and the key to implement efficient RL methods. Basically, TD learning is an unsupervised learning algorithm enabling the agent to learn to predict the value of future states. In RL this idea is used to update the state-value function during the online learning process after each state transition. The agent uses its observations to update the value for the current state and to propagate a discounted value back to prior states. With this, TD learning solves the problem of temporal credit assignment without waiting for a final outcome (bootstrapping), and without the need of a complete system model (cf. Dynamic Programming).

A classical example of TD learning demonstrating its principal method is the prediction of a time of arrival (Sutton and Barto, 1998). At the beginning of the prediction all possible events which may occur during the travel for which the time of arrival should be calculated are considered and summed up. This is the initial estimate (or prediction). Then, after the travel has begun and the situation changes by occurring events (e.g. arriving at a specific intersection or waiting at railway crossing) the time of arrival is continuously being updated. Each update is an estimate of the time left based on the actual state. Estimating the costs or value of the successor state considering the actual state and the past experiences is the core of TD learning and the main difference to MC methods which have to wait until the terminal state is reached. As TD methods learn new estimates on the basis of past estimates, they *learn a guess from a guess – they bootstrap* (Sutton and Barto, 1998). With this, TD methods have the advantage over Monte Carlo methods that they can be implemented in an online and incremental fashion and regarding Dynamic Programming methods that they do not require models of the system or the environment.

The definition of the TD error is important to understand the concept of a TD update. Generally, the TD error describes if things have gone better or worse than expected after an action was performed. The one-step TD error is defined as

$$\delta_t = r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t), \quad (2.3)$$

where $V_t(s)$ is the value function at time t and γ is the discount factor. In the simplest case, a one-step update of the value function is performed via the so-called $TD(0)$ update

$$V_{t+1}(s) = V_t(s) + \alpha \delta_t, \quad (2.4)$$

where $\alpha \in (0, 1)$ represents the *learning rate* and adjusts the step-size of the update. The update is always performed after the agent has executed an action according to the current policy and has received a new state observation.

A natural extension of this algorithm is the $TD(\lambda)$ update. With λ the use of eligibility traces is denoted. Eligibility traces are a temporal record (a type of memory) of the occurrence of events (such as the visit of states or taking an action). The basic idea behind eligibility traces is that each occurrence of an event triggers a short-term memory process

2. Background

which gradually fades out afterwards. For this, a memory variable $e_t(s_t)$ is associated with each state s_t for a particular time t and decays in each iteration in case of a non visited state by the factor of $\gamma\lambda$. Hence, the eligibility trace $e_t(S)$ is updated by

$$e_t(S) = \gamma\lambda e_{t-1}(S), \forall S \in \mathcal{S}, S \neq s_t, \quad (2.5)$$

for non visited states, where γ is the discount rate and λ the *trace-decay parameter*. For the actual visited state there exist different ways for updating the memory variable:

- **Accumulating trace:** Each time a specific state s_t is visited the memory variable $e_t(s_t)$ is incremented by 1.
- **Replacing trace:** Visiting a specific state s_t sets its eligibility trace variable $e_t(s_t)$ to the value of 1.
- **Dutch traces:** The *Dutch trace* is a mixture between replacing and accumulating traces. The memory variable is updated by $e_t(s_t) = (1 - \alpha)e_{t-1}(s_t) + 1$, where α is the step-size parameter (van Seijen and Sutton, 2014).

With the definition of eligibility traces a complete online tabular based $TD(\lambda)$ algorithm can be formulated as given in Algorithm 1. Basically, the algorithm runs continuously until a terminal state is recognized (e.g. a treasure is found). First, the agent has selected an action a_t according to a given policy (for example it greedily selects the action with the highest value), and performs it in the environment. After the execution it observes new

Algorithm 1 Basic implementation of tabular based online $TD(\lambda)$ learning.

```

1: Select an action  $a_t$ .
2: while  $s_t$  is not the terminal state do
3:   Take action  $a_t$ , observe reward  $r_{t+1}$  and next state  $s_{t+1}$ 
4:    $\delta \leftarrow r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$ 
5:    $e(s_t) \leftarrow e(s_t) + 1$  ▷ accumulating traces
6:   or  $e(s_t) \leftarrow 1$  ▷ replacing traces
7:   or  $e(s_t) \leftarrow (1 - \alpha)e(s_t) + 1$  ▷ Dutch traces
8:   for all  $S \in \mathcal{S}$  : do
9:      $V(S) \leftarrow V(S) + \alpha\delta e(S)$ 
10:     $e(S) \leftarrow \gamma\lambda e(S)$ 
11:   end for
12:   Advance to next state:  $s_t \leftarrow s_{t+1}$ 
13: end while

```

state information s_{t+1} and a reward signal r_{t+1} . Together with this new observations and the actual value function it calculates the temporal difference error δ and updates the eligibility trace $e(s_t)$ of the last state. Next, it updates the value function with the actual TD error and the eligibility traces for all states. Finally, the state variable is set to the new state.

Today, there exist modified versions of the TD update given in Equation 2.4, but the basic idea is still unchanged and the research focuses more on efficient state representations and function approximations for the value functions.

Q-Learning

Another popular group of Reinforcement Learning methods base on the update of the aforementioned state-action based value function, meaning that a utility value for each state-action pair is calculated online. Q-learning is such a model-free temporal difference method computing such a state-action value function, and it has shown to converge to the optimal policy for any given Markov Decision Process (Watkins and Dayan, 1992). Each state-action value $Q(s_t, a_t)$ is called Q-value and reflects the total discounted reward from time t for a particular state-action pair as

$$r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots + \gamma^n r_{t+n}, \quad (2.6)$$

where r is the reward at time $t, t+1, \dots, t+n$ and γ is the discount factor biasing the agent's total expected reward in favor of the present. The higher γ is, the more *myopic* the agent is giving less weight to rewards in the far future. The Q-value of any state-action pair can therefore be interpreted as a measure of how much total reward the agent can expect from choosing a particular action in any given state. The solution to this problem was remarkably simplified when Richard Bellman conceived the Bellman optimality equation (Bellman, 1957). Principally, it proved the optimality to estimate the total expected reward by recursively computing the value function from the immediate reward and the maximum expected total reward of the next state. Applied to Q-learning, the Q-value-function can thus be incrementally updated at every time step by the following rule

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

with s_t and a_t being the current state-action pair and s_{t+1} denoting the state after taking action a_t . $R(s_t, a_t, s_{t+1})$ is the received reward when transitioning from state s_t to s_{t+1} . The discount factor is denoted by γ and the learning rate is denoted by α_L . A fraction $1 - \alpha_L$ of the old Q-value for a given state-action pair is updated by a fraction α_L of the immediate transition reward and the maximum Q-value of the resulting new state. Even if state transitions are probabilistic, such an update rule will yield an optimal estimation of the expected total reward for any given state-action pair provided that $Q(s_t, a_t)$ is updated infinitely often for all states and actions. The learning rate α_L determines the smoothness of the updates, deciding how much weight will be given to the most recent experiences. For non-episodic tasks the discount factor γ has to be sufficiently small ($\gamma < 1$) to prevent Q-values from diverging.

Q-learning is an unsupervised Reinforcement Learning method and only receives feedback from the environment in terms of rewards. Because of that, the agent has to explore the environment and occasionally take low-valued actions to explore the state space. The

2. Background

agent faces the trade-off between exploiting already existing knowledge and exploring unknown terrain that could possibly yield higher rewards (see also next paragraph). The most common approach is to follow an ε -greedy policy. The agent takes the action with the highest Q-value, but with a probability of ε it selects a random action (Sutton and Barto, 1998).

For problems with moderately-sized discrete state and action spaces, look-up tables are used to store the Q-values for every state-action pair. This method quickly becomes impractical for continuous multi-dimensional state and action spaces. Memory requirements are too high and it takes too much time to visit all states often enough to derive a viable policy, let alone an optimal one. As a solution to this problem, one can use function approximation. Both, linear combinations of features and non-linear approximation techniques such as neural networks can be used. The primary advantage is, that the agent does not have to observe all states to take appropriate actions, but can infer from already seen states to similar new ones.

Overall, Q-learning is a powerful iterative value-function method in Reinforcement Learning allowing for a simple problem formulation and the flexibility to be applied in a wide variety of use-cases.

Planning

Dynamic Programming algorithms can be seen as planning steps in model based control problems. They calculate a policy using the given system model. In case of TD learning and Q-learning such a system model does not exist but can be approximated during learning from the system state transitions and actions. The state transitions and its corresponding actions, as well as a possible achieved reward are saved in a table and can be considered as a simple generative model. This generative model is then used in a subsequent step to sample random state transitions by selecting an arbitrary state and action in order to subsequently update the actual value function. The update is done by calculating the corresponding state values or Q-values, respectively, using the same update rule which is used during learning. In case of Q-learning the combination of planning and learning was introduced by Sutton (Sutton and Barto, 1998) and is called the *Dyna-Q algorithm*.

Overall, this kind of planning bases on looking ahead to future events and the computation of backed-up values which are then used to update an approximate value function. This improves the performance of the learning algorithms in problems with a high number of states and a comparable small number of states assigning rewards by propagating existing knowledge within the already learned model.

Action Selection

The introduced state-value or state-action-value function enables an efficient evaluation of learning situations an artificial agent is confronted with. Especially, the online evaluation

of a state-action-value function (like those represented by Q-values) introduce an important side issue in Reinforcement Learning which is not so evident in other kinds of machine learning algorithms – the exploration-exploitation dilemma (Sutton and Barto, 1998; Macready and Wolpert, 1998; Axelrod and Chowdhary, 2015).

In Reinforcement Learning, there exist several methods for selecting the best action on the basis of the state-action-value function. The action with the greatest expected utility is called the *greedy* action, while taking another action with a lower estimated utility corresponds to an *exploration* step. Exploration steps are sub-optimal in consideration of the current value function, but are necessary to gather new information about the environment which in turn can add new alternatives to the value function. In contrast, just taking the greedy action means to *exploit* the current knowledge contained in the value function without further exploration of the environment. Therefore, the exploration-exploitation dilemma unfolds as the designer or the agent itself has to decide how to adjust the fraction of exploration and exploitation steps. Only exploring the environment results in a near optimal value function entirely representing the environment, but the overall outcome is poor. By focusing on exploitation, which means only to select the greedy action, the overall outcome could be high but is strongly dependent on the initial quality of the value function. Intuitively, the agent should therefore first explore the environment and switch afterwards to an exploitation phase. In practice, the adjustment of this trade-off is a challenging topic and in Chapter 4 an approach on the basis of affective states is addressed.

Generally, in the RL domain the ϵ -*greedy* and *soft-max* action selection are the most common used strategies. The ϵ -greedy action selection strategy selects greedy actions most of the time, but occasionally it selects another random action. With the parameter ϵ the probability for selecting a random action is adjusted. Smaller values of ϵ account for less explorative behavior of the agent, while a high ϵ value frequently forces the agent to choose a random action which increases the probability to discover new states. The overall performance of the ϵ -greedy strategy depends on the task and the particular rewards. With an increasing variance of rewards it generally takes longer to explore the reward structure.

On larger state space problems, the fixed ϵ -greedy method will take too long to find the optimal policy, so instead a method interleaving exploration and exploitation is necessary. A well known representative of such a method is *Boltzmann* or soft-max action selection. It bases on Boltzmann distributions (Bridle, 1989; Salakhutdinov et al., 2007) for each action in a particular state s expressing a probability

$$P_s(a) = \frac{e^{\frac{Q_t(s,a)}{T}}}{\sum_{b \in \mathcal{A}_J} e^{\frac{Q_t(s,b)}{T}}} \quad (2.8)$$

for corresponding action values $Q_t(s, a)$. The temperature T controls the probability of executing explorative actions. If T is high (or if all action values are equal), random actions are generated by sampling over the distribution $P_s(a)$. If T is low and the action values are different, the fraction of a high action value is greater in the distribution than for smaller action values. A dynamic exploration-exploitation behavior of the agent can be achieved

2. Background

by starting the learning process with a high temperature which afterwards decreases with time. In case of an infinite time horizon, it can be shown that this method converges for stationary environments to the optimal state-action value function, while picking greedily the optimal action (Singh et al., 2000).

The above described two action selection strategies are most commonly used in the RL domain. There exist more advanced alternatives and domain specific solutions for the action selection step, but adapting them to new domains and the issue of finding the right trade-off between exploration and exploitation are still highly active research topics. In Chapter 4, the external control of RL with an affective state is described, which is an additional strategy to cope with these issues.

2.3.2. Multi-objective Reinforcement Learning

With the general definition of a finite Markov Decision Process and its extension to a multi-objective MDP, as introduced at the beginning of this section, the discounted sum of rewards for an infinite time horizon multi-objective Markov Decision Process is given by

$$\mathbf{G}_t = \sum_{k=0}^{\infty} \gamma^k \mathbf{R}(s_{t+k+1}), \quad (2.9)$$

where the actual reward vector at time step t is $\mathbf{R}(s_t)$. Generally, also undiscounted sums of rewards are possible in case of finite time horizon Markov Decision Processes, but to keep the learning framework as general as possible, an infinite time horizon is assumed which requires discounted rewards to bound the sum. This difference in episode length has to be kept in mind, as not all available algorithms can be applied in both domains (and vice versa).

The policy $\pi(s)$ determines the action an agent selects in each state and is going to be learned by the agent. With a given stationary policy, the multi-objective value function of a state can be formulated as

$$\mathbf{V}^{\pi}(s) = \mathbb{E}_{s' \sim P_{ss'}} \left[\sum_{k=0}^{\infty} \gamma^k \mathbf{R}(s_{t+k+1}) \mid s_t = s \right]. \quad (2.10)$$

In multi-objective environments, a policy is called *dominating* if the value in all dimensions is at least as high as the value of all other policies and strictly greater in at least one dimension. The *Pareto front*, also known as *Pareto curve* or *trade-off curve*, represents the set of multidimensional value functions \mathbf{V}^{Π} for a set of policies Π , such that no other dominating vector exists. It is used to evaluate optimal solutions in a multi-criteria optimization task. In case of a Pareto dominating value function this is written as

$$\mathbf{V}^{\pi} \succ \mathbf{V}^{\pi'} \Leftrightarrow \forall i, V_i^{\pi} \geq V_i^{\pi'} \wedge \exists i, V_i^{\pi} > V_i^{\pi'}, \quad (2.11)$$

where the policy π Pareto dominates another policy π' , when its value \mathbf{V}^{π} is at least as high in all objectives and strictly higher in at least one objective (Rojiers et al., 2014).

If the state transition probabilities of the environment are not known, solution techniques such as Dynamic Programming cannot be applied anymore. Instead, approximate versions which rely on sampling interactions with the environment have to be used. The most prominent sample based algorithms are the above introduced TD learning and Q-learning algorithm, however, they only work for the classical scalar reward setting.

In the multi-objective case, the learning algorithms can be additionally divided into two classes: Single and multiple policy approaches (Roijers et al., 2013). From the algorithmic perspective both classes are very similar, but they differ in the decision and execution phase. In case of a multiple policy algorithm, the agent calculates various policies based on previous observations or a model in order to approximate the Pareto front. Afterwards, the agent evaluates this set of calculated policies by comparing their values \mathbf{V}^π , and executes the one which promises the best compromise solution. There exist hybrid algorithms, first learning a transition or reward model which is then used afterwards for the calculation of multiple policies, cf. the works of Lizotte et al. (2012), Roijers et al. (2014), and Van Mofaert and Nowé (2014).

The single policy algorithms calculate only a single policy according to predefined preferences for the individual objectives. The preferences can be expressed in terms of specific weightings or as constraints for each objective. Single policy algorithms are typically model-free algorithms, and used in environments where sample costs are very high. In highly complex environments with a high number of objectives, also model-based single policy algorithms are used. Model-based approaches allow planning and simulation steps which speed up the learning process. In general, the following approaches for single policy algorithms are differentiated:

- **Known weights:** If a weighting of the different objectives is known beforehand, (linear) scalarization techniques can be used. Uniform and non-uniform weightings require different learning algorithms and the scalarization step can be performed before or after the action selection step. In case of uniform weights and linear scalarization the use of most classic RL techniques such as Q-learning or State-Action-Reward-State-Action (SARSA)-learning is possible (Sutton and Barto, 1998). Non-linear scalarization conflicts with the assumption that the rewards can be summed up additively, which prohibits the usage of algorithms based on the Bellman equation. More detailed surveys on scalarization algorithms can be found in the contributions of Vamplew et al. (2008) and Roijers et al. (2013).
- **Constrained objectives:** In cases where the weighting is not explicitly given, constraints for each objective can be introduced. The constraints are used to create a lexicographic ordering of the objectives which is then used to iteratively calculate various policies until all constraints are fulfilled. Generally, this requires a lot of iterations or a simulation model (Gábor et al., 1998).
- **Unknown weights:** In most real world applications, the weights for the individual objectives are unknown or hard to set a priori (Mannor and Shimkin, 2004). The

2. Background

unknown weights scenario is an attractive research topic, as in most scenarios the objective weights have to be optimized or adjusted while learning. Most approaches are based on geometric approaches like the maximization of the hypervolume, which is spanned by the outcomes of each objective. HB-MORL is such an algorithm and maximizes the hypervolume while the user can still adjust the policy to achieve the preferred compromise solution (Moffaert et al., 2013). A similar type of algorithm uses samples in order to approximate the convex hull of the optimal Pareto set and tries then to find the best trade-off solution (Barrett and Narayanan, 2008; Jin and Sendhoff, 2008; Perny and Weng, 2010). Approximating the Pareto front normally requires a large amount of samples or a simulation model.

In fact, finding policies which meet given constraints or reward target regions, or the task of learning optimal scalarization weights remain the basic challenge in multi-objective Reinforcement Learning. The different scalarization weights \mathbf{w} can be applied using the so called scalarization function $f(\cdot)$ which modifies the calculation of the *overall value function* by

$$V_{\alpha}^{\pi}(s) = f(\mathbf{V}^{\pi}(s), \mathbf{w}), \quad (2.12)$$

where \mathbf{V}^{π} is a vector value function. Selecting a good scalarization function depends on the problem and the desired solution. In cases where the weights per objective are already known, a single policy can be calculated. Using a strictly monotonically increasing scalarization function for policies on the Pareto front results in efficient decisions (Roijers et al., 2013). This fits in scenarios, where the rewards are related to monetary costs or profits.

In many other scenarios, not related to economic problems, the optimization of a multi-objective problem can only be achieved by adjusting the scalarization weights according to given user preferences, which in turn introduces the sub-optimality of learning human values, cf. Section 2.3.4. In order to analyze this sub-optimality in Chapter 5, an approach for inversely learning the scalarization weights implicitly specified by a given policy is proposed. In the following section, the process of reward shaping is introduced which can basically be used to apply scalarization weights during Reinforcement Learning.

2.3.3. Reward shaping

The term *shaping* was first introduced to science by the psychologist Skinner (1953). It describes a technique to train animals in solving problems by dissecting the task into related simpler sub-problems. These simpler problems are then solved first and rewarded separately. In consecutive steps, the reward for the simpler sub-problem is now withheld, and only sequences of correctly solved sub-problems are jointly rewarded at the end. With this technique, complex tasks can be trained by successive approximations. As a side effect, the well known *temporal credit assignment* problem of RL is reduced, as early phases of a learning task can already be rewarded. Conventional RL algorithms deal with this problem by employing delayed approaches which base on back-propagation. Such algorithms, like

Q-learning are comparatively time consuming in back propagating rewards within a high dimensional state space. In their paper, Gullapalli and Barto (1992) integrate the concept of shaping into the training of artificial learning systems. They describe the approach of facilitating learning of complex tasks by introducing domain knowledge into the Reinforcement Learning framework. The idea is to give additional (numerical) feedback to the agent in order to improve the convergence rate. This feedback is extra information (domain knowledge) and is incorporated by the designer of the learning agent. It can be used to steer the learning into a specific direction as this extra reward is exploited by the agent. Simultaneously, it is an elegant way to introduce preferences (biases) to the learning process which should be considered by the agent. To implement the concept of reward shaping, one way is to modify the state-value function of the RL agent which can be, e.g., expressed by

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r(s_t, a_t, s_{t+1}) + F(s_t, a_t, s_{t+1}) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right], \quad (2.13)$$

in case of Q-learning, and where $F(s_t, a_t, s_{t+1})$ is an additional reward component (shaping reward) that depends on the state transition. Ng et al. have formally analyzed the requirements on the shaping reward, and state that the optimal policy (in the model-free case) can be found by classical RL algorithms if and only if the shaping reward is defined as a difference of some potential function (Ng et al., 1999). The potential function is defined as a state dependent function $F : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ between a start state and a goal state. F should have the form of a difference of potentials, like

$$F(s, a, s') = \gamma \Phi(s') - \Phi(s), \quad (2.14)$$

where $\Phi(s)$ is again state dependent ($\Phi : \mathcal{S} \rightarrow \mathbb{R}$) and fulfills

$$F(s_1, a_1, s_2) + \dots + F(s_{n-1}, a_{n-1}, s_n) + F(s_n, a_n, s_1) = 0 \quad (2.15)$$

for following a policy circulating in the environment (starting and terminating in state s_1). With this definition of the potential function the optimal policy is preserved and potential oscillations are prevented. However, the constraints complicate the design or respectively the autonomous learning of a suitable potential function. In most state of the art approaches, an approximation of the potential function is done by computing a simpler abstraction of the value function (Grześ and Kudenko, 2010). These approximations use sub-sampled versions of the state space or aggregated state representations (e.g. clustering, approximation functions, neural networks are used in the work of Dean et al. (1997)).

Besides the approaches of generally calculating reward shaping functions as some kind of approximation, they can also be manually designed using *heuristics*. For example, such a heuristic rewards states in dependence to the physical distance to the goal state. Obviously, the knowledge of the goal state's position is required to implement this heuristic. As the goal of reward shaping is to speed up learning by introducing domain knowledge by an external *trainer* (in Reinforcement Learning also known as *critic*), this requirement is admissible. In literature, this more directed way of learning is often called *Informed*

2. Background

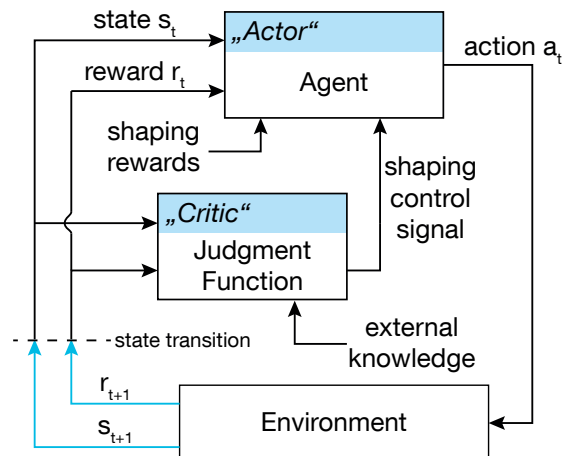


Figure 2.7.: Actor-critic architecture of an agent using reward shaping which is externally controlled by a judgment function.

Reinforcement Learning and is used in environments with high dimensional state spaces and limited training phases like dynamic environments or real world scenarios.

As in these particular cases, where reward shaping is used to reduce the state space hierarchically by switching between levels of abstractions, a *judgment function* is used to control the shaping process. The implementation can follow the actor-critic scheme, according to Sutton and Barto (1998), with the extension of a judgment function. The principal structure of this approach is depicted in Figure 2.7, where the shaping control signal (e.g. basing on the TD error or a manually implemented heuristic) adjusts the impact of the shaping rewards. Designing this central judgment function introduces another side issue to reward shaping which is very similar to the well known exploration and exploitation dilemma in basic Reinforcement Learning (see Section 2.3). In the exploration-exploitation dilemma, the control between randomly exploring states and exploiting the greedy action can be done via time-based functions, e.g. with ongoing time, the probability of explorative actions is reduced. Similar concepts are also used for reward shaping, however in case of high dimensional state spaces, the algorithms use more advanced methods like hierarchical decomposition of the task (e.g. Feudal RL by Dayan and Hinton (1993)) or the extraction of shaping rewards out of human policies (e.g. Griffith et al. (2013)). Especially, the hierarchical approaches require a function to balance the trade of between already mastered approximations of the tasks and the decision to switch to more complex state representations. By switching too quickly to the next more complex approximations, the overall learning progress can be slowed down.

Instead of time-based methods or approaches requiring external human feedback also affective models of an experience state can be used to control the shaping rewards. As described in Chapter 4, such an approach can introduce preferences into Reinforcement Learning in an intuitive way.

2.3.4. Human values

Human values are situational and contextual dependent appraisals of a situation or action. They relate to actual monetary or objective outcomes, but involve also affective values, which are actually related to emotions. Nowadays, the term *human-centered computing* is used to describe systems which consider those values in their calculations. According to Schwartz (1994), human values serve people as a motivational construct to achieve desirable goals while avoiding potentially adverse events. That is, they serve as standards or criteria a human uses during decision making. Advanced behavioral patterns of humans like morality base on the concept of human values as standards.

Introducing similar patterns into artificial agents means to design decision making agents in a way to share human values. In fact, that implicates to assign corresponding human values to individual decisions and consequently find out how to weight them during the final decision making stage. But how can such a system be built, that decides in a way that is in accordance with the general human understanding and its value system? Currently, artificial intelligence designers or programmers develop machine learning algorithms and systems by using general psychological and neurophysiological findings. This is usually a complex and often not viable task and sometimes introduces unintended biases. Additionally, the models used for programming are based on studies, which may also contain biases. Biases are natural tendencies of humans towards specific objectives or positions. For example, Muehlhauser and Helm (2012) conclude in their psychological survey about the ability of humans to write down their own values that they are not very accurate. Hence, handcrafted solutions implemented by an Artificial Intelligence (AI) designer might deliver sub-optimal solutions. For example, when one thinks of morality and tries to write down situations in which morality counts and in which not, the stated situations will share in most cases the same point of view and corresponding human values. However, there will also be some outliers which are the result of a person's background and current emotional state. That implies, that each situation and the corresponding human values have to be carefully investigated and modeled to create precise models. These resulting models might then be used in a technical system to adjust a learning algorithm, so that it behaves in accordance to human values. In general, however, such precise models do not exist.

A faster and more practical way would be to learn to imitate the behavior of a human in specific situations. Direct imitation learning introduces limited generalization in new unobserved situations. One solution, suggested by Ng and Russell (2000) is *Inverse Reinforcement Learning*, which tries to overcome the weaknesses of imitation learning by using the observed policies to recover the underlying (and mostly hidden or unknown) reward function. The reward function is the most robust and compact way to describe intended behaviors, but the reward function cannot be easily specified for most real world tasks. Soares (2015) argues that specifying just goals that an agent has to achieve cannot cover the real world complexity. On the contrary, describing the complexity of a decision scenario with a large number of variables which have to be maximized often leads to a large number of variables remaining unconstrained. Generally, those unconstrained variables are set to

2. Background

extreme values, which in turn can lead to undesirable solutions. Therefore, to improve the policies of artificial agents towards more believable and human-like behaviors, it is necessary to include additional factors like for example honesty, cooperativeness, or thoughtfulness. Those factors might be in contrast to the actual objective, and therefore reduce the general performance of an agent but improves the social acceptance of an artificial system (Livingston et al., 2008). Most of those factors are related to specific preferences (see Section 2.3.5) of humans and depend on a cultural context. A method for learning different preferences in a given task is described in Chapter 4 and bases on a combination of neurophysiological findings, Prospect Theory, and the classic Reinforcement Learning mechanism. The proposed framework is a multi-objective reward scalarization algorithm, which is driven by an external experience signal. It was designed for scenarios where the reward process gets unfold with increasing experience of the agent. The generation of this experience signal and the design of the corresponding weighting functions is done manually and therefore subjected to be sub-optimal and biased by the designer's value system and intention. To overcome this issue, in Chapter 5 an Inverse RL framework is developed, which enables to learn unknown scalarization weights for given policies.

2.3.5. Preferences

The term *preference* can be interpreted in several ways. For example, it can mean that an alternative a is more liked by a person than another option b , it can be used to compare an algorithm a that outperforms algorithm b on a certain problem task, or it can be used to compare an event a , that is more beneficial for achieving a desired goal than an alternative event b , etc. In all cases, a pairwise comparison between a and b results in a clear preference for one alternative. This is roughly speaking the economic theory of preference, which defines preferences as relations between alternatives and presumes a structure of a complete pre-ordering.

The expected utility hypothesis is commonly used to model human's decision making behavior in scenarios with uncertain outcomes, such as gambling. However, situations where preferences of individuals among same choices are important, are not handled properly by the classic expected utility theory of Bernoulli. The *Prospect Theory (PT)*, proposed by Kahneman and Tversky (1979) introduces the concept of a reference point to the expected utility theory of Bernoulli. This reference point enables to model preferences of individuals among same choices. That means, that an internal reference point for a specific decision is essential to model the decision making behavior of people. A similar concept also plays an important role in the psychological description of human behavior and its affect system. The human affect system is responsible to regulate the perception and assessment of events. It is able to rapidly assign emotions to occurring situations. The resulting affective representation of the situation is then used to influence the decision making process (Gigerenzer et al., 2011). In the work of Damásio (1994), the authors hypothesize that this affective representation is partly realized by so-called *somatic markers* in the brain. The somatic markers are specific patterns of affective states in the brain and are associated with

situations or events which the person has encountered. They encode past situations and events according to their rewards or punishments, enabling the person to compare them to the current situational appraisals. The comparison then causes a positive or negative feeling for the currently faced situation biasing the decision making process towards the most beneficial outcome. Keeping Damasio's theory in mind, the design of an autonomous agent requires both, an affective and cognitive component influencing the decision making.

The affective component in decision making is also related to subjective preferences which are often adjusted by the current affective state (e.g. mood state). Preferences are fundamental for the human choice behavior and an important component in learning. Over time, specific situations and their past outcomes are associated with particular emotions (and their corresponding bodily changes) and encoded with somatic markers in the brain. During decision making, these emotion-situation pairs are used as physiological signals to bias decision making towards certain policies while avoiding others. The whole set of somatic markers can therefore be seen as part of the (emotional) experience of a human and is gathered during life. The impact of past experiences not only depends on the encoded emotions, their individual components are also weighted by the current affective state or more commonly, the mood. Different states of mood adjust the weightings of particular goals and needs influencing their importance. This results in biases for specific options according to the internal state of mood and past experiences. Regarding the development of artificial life-long learning agents, such an emotional and experience based component is essential but currently treated only as a side issue. Examples of currently existing human like agents basing on the above ideas can be found in the papers of Vernon et al. (2007) and Velásquez (1998).

In machine learning, experience corresponds to past data, structured in a way fostering the learning progress. Specifically, experience is related to the goal conduciveness of actions in a particular situation. Therefore, for a set of potential options in a specific situation, the artificial agent has to build up a preference model over possible options. Considering Q-learning (see Section 2.3.1), experience is aggregated in the table of Q-values for each state and action. The preference ordering is done via the temporal difference update. Moreover, learning the order of choices corresponds to the technique of *preference learning* (cf. Doyle (2004) and Hüllermeier et al. (2008)) where binary preference relations between alternatives or a utility function for scoring alternatives are learned. Generally, with preference learning the technique of automatic learning, discovery, and adaptation of preferences is meant. A concrete approach combining RL with preference learning is described by Fürnkranz et al. (2012). In their approach, preference learning is used for learning a preference model from qualitative feedback in order to rank different policies of the RL process. The main drawback of this approach is the qualitative feedback which is used to evaluate already learned policies. This external qualitative feedback is given by human experts and therefore is not generally available. Also, the learning process is directly interrelated with the preference model and there is no possibility of control how much the preference model affects the decision afterwards.

In this dissertation, this kind of preference learning is not further discussed, rather a

2. Background

concept of steering the RL process with an affective state is considered (see Chapter 4). Accordingly, in the following sub-section related concepts of combining affective states with Reinforcement Learning and basic neuroscientific findings are briefly outlined.

2.3.6. Affective states in Reinforcement Learning

First of all, the work of Doya (2002, 2008) describes neuromodulatory systems and the signals that regulate the Reinforcement Learning mechanisms of the human brain. He argues, that specific signals control and regulate the meta-parameters (like randomness, action selection, reward prediction error, speed of memory update) for the RL process. However, there is no clear hypothesis regarding the generation of these signals in the human brain. It seems to be a process which is separated of the actual learning and simultaneously runs in different brain areas. This suggests that the brain has the capability of dynamically adjusting these meta-parameters towards new or dynamically changing environments. More recent findings of Dayan and Niv (2008) show that several learning strategies and neuromodulatory systems are simultaneously active in the human brain such as model-free and model-based learning, which are generally considered separately in machine learning. Although the results of cognitive neuroscience made by Dayan and Niv (2008) and Behrens et al. (2007), give evidence that humans utilize Reinforcement Learning, they also clearly suggest that the human brain simultaneously considers different strategies which are intertwined and influenced by other subsystems.

Results of neuroscience have been and still are a relevant and important source for the development of machine learning. Behavioral patterns and strategies discovered in the brain of animals and humans are modeled and implemented in technical systems. Besides neurophysiological findings, also behavioral science and economics influence the development of learning algorithms. For example, Prospect Theory as mentioned above shows the psychological influence of external and internal signals on the decision making behavior of humans. In the PT, a value function is described which is sensitive to deviations of the outcome (reward) according to a reference point in case of a risky choice. In such cases, the reference point is set by the current decision problem and depends rather on the losses and gains, than on the final net asset value. This is also the reason when framing of a choice problem becomes critical. The framing of the problem results in an external shift of the reference point which alters obviously the decision behavior (Tversky and Kahneman, 1981; De Martino et al., 2006).

A concrete combination of Reinforcement Learning and Prospect Theory is described by Ahn and Picard (2006). They have extended the conventional framework of RL for Markov Decision Processes with PT-based subjective value functions to model experienced-utility and predicted-utility functions. Furthermore, these functions vary dynamically according to the affective state of the decision maker (agent). This enables the agent to choose an action according to different risk attitudes and action tendencies on the basis of subjectively evaluated previous outcomes of decisions. The performance of the algorithm appears to be very good in the selected domains, however the results are difficult to reproduce due

to the strong parameter dependence (which were additionally optimized for each domain). Moreover, the reference point of the PT value function is automatically set by the algorithm and an external control is not intended. However, by considering the framing hypothesis and the presented neurophysiological findings, both, the external control as well as a strict separation of the reference point from the learning process seem to be essential.

In the work of Ahn and Picard, a kind of reward shaping is used to integrate the affective reference point into the learning process. As described in Section 2.3.3, reward shaping can steer Reinforcement Learning and thus is particularly suitable to integrate affective constraints into it. This is also shown in the work of Babes et al. (2008), where social accounts are integrated into RL by shaping the rewards accordingly. They present a multi-agent scenario, where agents behave more efficiently while using *social reward shaping*. The potential function for shaping the rewards was pre-calculated in this example and cannot be biased online by an internal (affective) state of the agent. This results in a limited dynamic of the proposed agents and the pre-calculated potential function is strongly domain dependent.

Beside external (or monetary) rewards there is evidence that also social rewards bias attention and preference for choices (Anderson, 2016). Social rewards provide an efficient teaching signal that can modulate behavior, and additionally increase motivations for correlated goals. In contrast, classical RL only maximizes the expected utility based on the monetary outcome and the probability of occurrence, and is therefore unable to incorporate affective or social aspects into the learning process in the basic setting. In order to model more human like behavior, additional factors have to be integrated into machine learning processes. For example, the study of Fehr and Camerer (2007) gives insight into the neural circuitry involved in human decision making. They hypothesize that many people exhibit social preferences in most decision scenarios. Social preferences in their context mean that people prefer choices based on a positive concern for the welfare of others, and on the assessment what other players might believe about them. With this, the central question of how the brain does construct decision utilities when faced a decision which is also governed by competing motives of others is formulated. The results of their study show that social rewards activate neural circuitry overlapping with circuitry that anticipate and represent general types of rewards like monetary outcomes. Hence, it seems reasonable that for humans social rewards are equally important as individual benefits. In contrast, in machine learning additional affective or social factors are not incorporated by default. In case of Reinforcement Learning, affective or social rewards can be integrated into learning by shaping the rewards accordingly (see Section 2.3.3). Also specific preferences can be learned in this way. Chapter 4 introduces a basic concept of learning policies biased with additional rewards and a method of controlling them by an additional (affective) state.

Concluding remarks on Reinforcement Learning: *Reinforcement Learning is a highly active research topic to the time of this dissertation. Since Richard Sutton has introduced the theoretic framework of RL to the broad audience of computer scientists and engi-*

2. Background

neers, a lot of learning strategies, optimization techniques, and approximation methods have been developed. A wide range of applications has been identified and successful demonstrations were published. Also in real world scenarios, Reinforcement Learning has successfully been implemented. Recently, with the introduction of deep neural networks the general applicability of RL in domains without predetermined features has been proved (Mnih et al., 2015). This general applicability renders RL attractive to the domain of robotics and artificial agents interacting in real world scenarios with human users. Less studied directions of RL are sub-optimal trade-offs, which typically arise in real environments, as well as the efficient communication of the learning progress towards non-expert users. The sub-optimal trade-offs are especially interesting in multiple-objective domains and if specific preferences (of the user) should be incorporated into the learning process. An efficient and intuitive communication of the learning progress gets important in human-robot interaction scenarios where non-expert users are confronted with the trial-and-error strategy of RL agents. Therefore, both topics are further investigated in the following, as they could potentially provide a contribution to these two less studied directions of Reinforcement Learning.

3. Affective Evaluation of Machine Learning Experiments

Nowadays, the complexity of machines and systems noticeably increases. Simple text displays reporting the current system status show an overwhelming amount of information and are often cryptic to read. Sometimes, the confusingly displayed quantitative figures require experts' knowledge to understand and interpret them. Therefore, alternative ways of communicating system information to naive users have to be found and evaluated. One of these possibilities are emotion aware systems, which are able to autonomously appraise their internal state, aggregate it, and calculate a representation which can be easily recognized by a human user.

Focusing on this aspect, this chapter tries to answer the question if an appraisal derivation model between a machine learning algorithm and a model of emotion can be found in order to improve the human grasp of the current system status and the learning progress of the machine. The appraisal derivation model is the layer between the technical machine learning algorithm and the psychological model of emotion. It translates figures of the machine learning algorithm into appraisal variables such that the emotion model can represent them as a general feeling or distinct emotion expression. In the following subsection, three examples of different machine learning scenarios are given and the corresponding appraisal derivation model is described. In experiments, the affect generation is shown and the results are discussed in respect to its contribution to increase the believability of the agent. Explicit human-machine interaction experiments were not carried out, as there is already evidence that non-verbal communication and interaction between robots and humans significantly improve the communication efficiency (Hogan and Stubbs, 2003; Cassell, 2000; Russell and Mehrabian, 1977). The implemented algorithms of this dissertation and their results have already been published at relevant conferences (Feldmaier and Diepold, 2013, 2014; Feldmaier et al., 2017).

A large variety of decision problems can be modeled with so called multi-armed bandits. Online advertising, news article selection, network routing, and medicinal trials, to name a few, can be transformed into an equivalent multi-armed bandit structure. In the first example, Section 3.1, an artificial agent interacts with a multi-armed bandit and appraises the outcomes and its learning progress using the Zurich model of Social Motivation (cf. Section 2.2.3).

In the next example, a robot navigates in a world represented by discrete tiles while seeking a goal location. Generally, this scenario is known as the *Gridworld* navigation task. In this Gridworld, the robot has no map for finding the treasure and is occasionally

3. Affective Evaluation of Machine Learning Experiments

confronted with obstacles during its search. A Reinforcement Learning based approach is used to learn the optimal path to the goal while the developed appraisal derivation model statistically evaluates the selected actions. The results are mapped into a dimensional emotion model and are used in turn by RL to improve the affective behavior of the robot.

Finally, in the third experiment, the considered scenario and correspondingly the machine learning component gets more complex and a version of the Component Process Model is implemented. The Stimulus Evaluation Checks of the CPM appraise a simulation of a *Simultaneous Localization and Mapping (SLAM)* process. The SLAM process represents the autonomous navigation and mapping of a robot which navigates through an unknown environment. Typically, the current status of such SLAM algorithms can be evaluated by co-variance measures and quantities of observed landmarks. The proposed combination with the CPM enables the representation of the current localization status of the robot in terms of artificial emotions.

In summary, this chapter introduces concepts for implementing psychologically informed models to appraise machine learning algorithms in an intuitive way. This can reduce users' annoyance and fosters an increased understanding of cyber-physical systems by average users. In this way, the findings contribute to the domain of social robotics. Social robots are defined inter alia as robotic systems acting in a way that can be intuitively understood by humans.

3.1. Bandit Simulation

In this section, an experiment combining a general decision making simulation with the Zurich model is presented. The objective is to investigate an approach which gives an artificial agent the ability to appraise its internal decisions with human like *senses* (in this particular case with the artificial feelings of security and arousal). First, the multi-armed bandit simulation is introduced, followed by implementation details of the Zurich model and the experiment description. Finally, the section ends with the results and first conclusions.

3.1.1. Multi-armed bandits

One-armed bandits are well known as slot machines and can be found in almost all casinos around the world. Those slot machines pay a reward according to an unknown probability distribution in each play. If a player is confronted with a row of slot machines and has to decide which machine he/she likes to play, then he/she is confronted with the so called multi-armed bandit problem. The Multi-Armed Bandit (MAB) research topic goes back until 1952 when Herbert Robbins considered his clinical trials as a bandit problem with more than two arms (Robbins, 1952).

In a MAB experiment, several trials are played on the same bandit machine consisting of more than one arm. Each arm has a specific probability distribution which determines the success probability for winning. At each trial, the agent tries to maximize its reward by

selecting the arm with the highest chance of success. During a game, the agent tries to find out which arm has the highest success probability. Then, the agent will keep playing this arm in order to maximize its reward. The optimal arm is found during the exploration phase. In this phase, the agent checks out each arm several times in order to estimate the success probability for winning. There is again an exploration-exploitation trade off, as the agent has to decide when to stop the exploration and start to exploit the arm with the highest success probability. Macready and Wolpert (1998) have investigated this dilemma in the context of bandit problems.

The success of each play is traditionally evaluated in terms of cumulative reward and the rate of optimal decisions. The performance of an agent (or learning algorithm) is evaluated in the long run by calculating the mean values of the cumulative reward and optimal decision rate over several plays with several trials. These mean values are also used for comparing different algorithms.

Besides many different learning algorithms for Multi-Armed Bandit problems, there are also many variations of the MAB problem itself. Different decision problems are modeled using specific numbers of arms. Bernoulli, Exponential or Poisson probability distributions are used to model underlying reward structures, and methods for modifying these probability distributions as a function of previous actions were also introduced.

For the following experiment, the class of Bernoulli distributed bandit problems has been chosen. This class represents decisions with only two possibilities (binary decisions) – success or failure, both cases with a distinct probability. Decision problems or scenarios with multiple choices or moves can be represented by extending the structure to multiple Bernoulli distributions so that each alternative has a distinct probability to win or lose. An efficient implementation of such multi-armed Bernoulli bandit problems and several learning algorithms has been presented in the work of Kaufmann et al. (2012) and was re-used in this dissertation for the conduction of the experiments.

3.1.2. Implementation

A four-armed Bernoulli bandit was implemented to represent a decision scenario with four possible actions. The random variable of the Bernoulli distribution becomes 1 with a success probability of p , and takes the value of 0 with a failure probability of $\bar{p} = 1 - p$. To learn an optimal policy in the setting of Bernoulli bandits, the Gittins index can be exploited (Gittins, 1979). The Gittins index is an approximation for the expected reward under the assumption of playing the optimal policy. An agent calculates the Gittins index for each arm after every iteration and selects in the subsequent round the arm with the highest value. As an interface between the agent and the Multi-Armed Bandit simulation only the number of the selected arm and the corresponding reward is used. With these pairs of decision and reward, the Zurich model as presented in Section 2.2.3 is updated. The Zurich model also needs a geometric relationship between objects to calculate the security and arousal value. Therefore, in order to represent the bandit arms in a two-dimensional space, each arm gets a virtual position so that all arms are arranged on a circle around the agent (Fig-

3. Affective Evaluation of Machine Learning Experiments

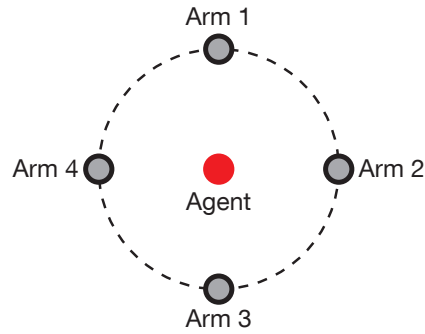


Figure 3.1.: Virtual positions of the bandit arms in relation to the agent.

ure 3.1). Besides the virtual position of an arm, the so called *Detectors* of the Zurich model are necessary to detect the familiarity (Det_F) and the relevance (Det_R) of an arm. Both detectors need to be individually adjusted to the corresponding machine learning scenario. For the Multi-Armed Bandit scenario *eligibility traces* are used in order to calculate time-dependent values for each arm. More specifically, the concept of accumulating traces are used (cf. Section 2.3) so that the familiarity value F_t^i for each arm i is updated at each time step t by adding one to the trace of the actual selected arm. This can be expressed by

$$F_{t+1}^i = \begin{cases} \lambda\gamma F_t^i + 1 & \text{if arm } i \text{ was selected} \\ \lambda\gamma F_t^i & \text{if arm } i \text{ was not selected} \end{cases}, \quad (3.1)$$

where λ and γ define the decay of the trace. Similarly, the relevance value R_t^i is calculated. The difference is that the amount of reward of a corresponding arm is taken into account, thus this is calculated by

$$R_{t+1}^i = \begin{cases} \lambda\gamma R_t^i + r_t^i & \text{if arm } i \text{ was selected} \\ \lambda\gamma R_t^i & \text{if arm } i \text{ was not selected} \end{cases}, \quad (3.2)$$

where r_t^i is the reward of the selected arm i . This results in high familiarity values for regularly selected arms and a high relevancy for familiar arms with high rewards. For the iterative simulation of this scenario, the agent selects in each round the arm with the highest Gittins index and perceives the resulting reward. The performed action and the obtained reward are presented to the detectors of the Zurich model. The model is then updated with the new familiarity and relevance values. Due to the fixed virtual arm positions only the agent's position influences the Zurich model. The agent's position, however is only changed by the Zurich model itself and therefore is directly dependent on the familiarity and relevance values of the surrounding arms. With these positions, the security and arousal system of the Zurich model (Figure 2.4) is updated resulting in the joint familiarity value F and the potency value P . They are compared with the reference values D and E , resulting in a security value s and an arousal value a . The dependency value D and the enterprise value E is set according to the empirical findings of Bischof (1975). Generally, the

dependency value determines the level of security someone needs in order to feel secure. In humans, this value is highest in childhood and falls to its minimum with adolescence, in order to rise again with age. The enterprise value is relatively high during infancy and falls to its minimum during early childhood, in order to continuously increase with age. Enterprise represents the initiative to undertake or try out something new, despite any risks. The Zurich model needs only these two values, dependency and enterprise to adjust the

Table 3.1.: Parameter settings used for the Zurich model, the detectors and the Multi-Armed Bandits.

Zurich Model	Detectors	Bandits	
$D = 0.75$	$\lambda \cdot \gamma = 0.95$	$p^1 = 0.1$	$p^2 = 0.3$
$E = 0.8$		$p^3 = 0.2$	$p^4 = 0.8$

agent's personality. In the present experiment, the values are set to a combination appropriate for an individual, which is highly dependent on sources of security ($D = 0.75$) and simultaneously receptive to changing situations ($E = 0.8$). Besides these two pre-defined values, the decay parameter of the eligibility traces is set to a value so that the trace almost vanishes after 50 iterations in order to represent a short term memory process. For the four bandit arms the success probabilities p^i are set arbitrarily, except that one arm has a clear maximum (p^4) which has to be found by the agent. All necessary parameters of the simulation and the Zurich model are summarized in Table 3.1.

The calculated artificial feelings of security and arousal are subsequently mapped by a heuristic to distinct emotional terms which are handier for describing the results. This mapping is based on the textual description of the Zurich model (Gubler and Bischof, 1991; Bischof, 1975), but should be further empirically evaluated. Both artificial feelings, security and arousal have to be interpreted in relation to each other. In Table 3.2 such a heuristic is given. It consists of relational operators, numerical gradient calculations (depicted with the ∇ operator), and Boolean operators comparing the values of arousal and security to each other. All rules of the heuristic are evaluated in parallel, hence more than one emotional term can be triggered. By way of example, the first rule compares the arousal and security value with a threshold and outputs *uncertainty* if both values are below. The second rule calculates the gradient of both values and compares their signs. In case of decreasing security and rising arousal *aversion* is triggered. A triggered emotion term is visualized in the results with a corresponding color mapping, which can be found in the first column of Table 3.2. The color mapping is motivated by Plutchik's wheel of emotions (Plutchik, 1991) and was slightly modified to create a clearly perceptible color mapping of the results.

3. Affective Evaluation of Machine Learning Experiments

Table 3.2.: Decision rules determining which emotional term is triggered (a : arousal, s : security). The color mapping in the first column is motivated by Plutchik’s wheel of emotions (Plutchik, 1991).

Color	Emotion	Rule
●	uncertainty	$a < 0.15 \wedge s < 0.15$
●	aversion	$\nabla s < 0 \wedge \nabla a > 0$
●	anger	$(a > 0.85 \wedge s < 0.15) \vee (\nabla a > 0.0075 \wedge s < 0.3)$
●	fear	$\nabla a > 0 \wedge \nabla s < 0 \wedge a > s$
●	anticipation	$\nabla a < 0 \wedge \nabla s > 0 \wedge a < s$
●	joy	$a < s \wedge \nabla a \leq 0$
●	trust	$a < s \wedge \nabla a < 0.0005$

3.1.3. Experiment

With the described implementation, several bandit simulations are conducted and appraised by the Zurich model. The learning policy is fixed and the resulting feelings of security and arousal are used to map them into distinct emotion expressions as shown in Figure 3.2 and 3.3. This mapping should show that a human appraisal model can be used to evaluate the decisions of an artificial agent. Furthermore, the implementation fulfills the requirement which has been stated in the introduction, that the core and the logic of the original psychological model should remain untouched, and only the appraisal derivation layer, in this case the detectors have to be adjusted in order to be compatible with the scenario.

Two different settings of the experiment are conducted. In the first experiment, the learning process is modified in order to extend the different parts of a typical learning phase. This helps to better visualize the results with the proposed heuristic. The experiment is repeated 100 times, while each time 1500 interactions with the Multi-Armed Bandit were done. In the first phase of the experiment, the agent is forced to select randomly an arm. After 300 trials, *Phase 2* starts and the agent is limited to select one of the two best arms. This second phase corresponds to an extended training phase. This training phase is manually ended after additional 300 trials and followed by a phase of optimal decisions (*Phase 3* between trial 600 and 900). At trial 900 a hidden disturbance of the Multi-Armed Bandit occurs, so that the success probabilities of the arms get mixed up and the agent has to relearn the optimal decision again. Until the end, between trial 1200 and 1500 the agent is set to exploit the arm with the highest reward. The five different phases are tagged in Figure 3.2.

In the second setting of the experiment, the agent uses the Gittins index to discover the optimal arm and the learning process is not manually slowed down. This setting is also repeated 100 times, each with 300 trials. After 100 trials the hidden disturbance of the first

experiment also shuffles the success probabilities, such that the agent has to adjust its policy.

3.1.4. Results

Figure 3.2 shows the resulting security and arousal values, as well as the overlaid emotions for the first experimental setting. The five simulation phases are separated with dashed lines. The first phase is characterized by uncertainty at the beginning followed by emotions like aversion and fear. This corresponds to the expected result in this phase. In the second phase, the amount of emotions like anticipation, joy, and trust increases but is still interrupted by feelings of aversion. Compared to *Phase 1* the arousal level in *Phase 2* slightly increases due to the changed situation, whereas the security level remarkably goes up as the success probability has doubled. The heuristic based mapping of arousal and security to emotional terms is inconsistent in these initial phases and changes frequently. In *Phase 3*, the ideal case is reached, the agent only shows emotions like anticipation, joy, and trust. The disturbance in *Phase 4* triggers emotions like aversion and fear, but no uncertainty due to the remaining amount of successful decisions. Finally, in the last phase the decision process has been relearned and the agent regains its trust, along with a decreasing arousal value and high security values. These results of the first setting of the

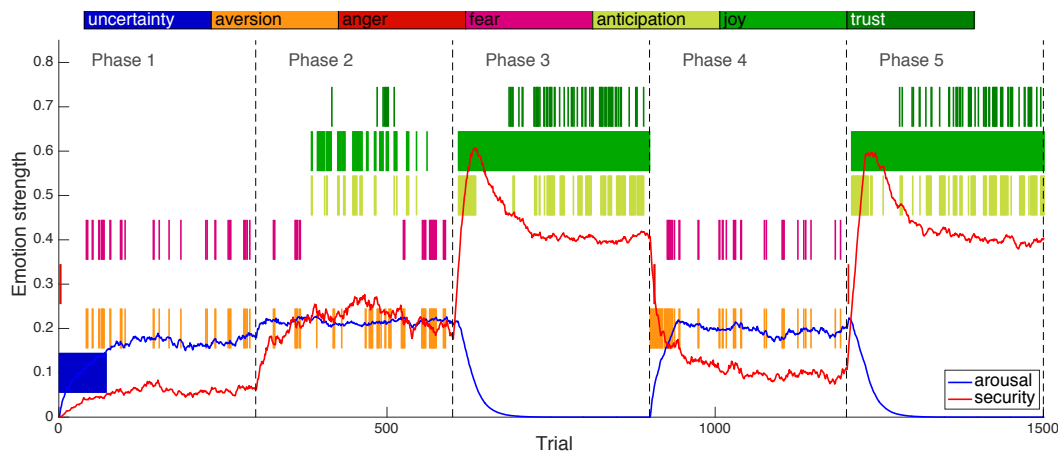


Figure 3.2.: Plot of the security and arousal feeling for the bandit experiment with prolonged learning phases and the overlaid emotional terms (color-coded according to Table 3.2).

MAB experiment show that an agent equipped with the Zurich model is able to appraise the different phases of a typical learning scenario. In order to further evaluate this concept, in the second setting the agent autonomously learns its actions using the Gittins index. The Gittins index generally learns the optimal choice within a few samples and stores a kind of experience as indexes for each arm. Therefore, frequent changes of the optimal choice without resetting the indexes results in decreased learning performance. Such a scenario

3. Affective Evaluation of Machine Learning Experiments

was modeled with the second setting of the experiment. Figure 3.3 shows the results. At the beginning, the standard learning progress is made and the agent learns the optimal choice within a few trials. With the changed success probabilities at trial 100, learning progress stops and relearning starts. Changing the success probabilities of the MAB process is unusual and results in poor learning performance. Generally, the performance in such a scenario would be evaluated according to figures like the cumulative reward which has been plotted in the second half of Figure 3.3. In terms of the cumulative reward, the changed learning progress can only be recognized by small changes in the slope of the curve. In contrast, the Zurich model recognizes the changed arm configuration very clearly and fast, as a kind of dynamic memory process is implicitly performed by calculating iteratively the relevance and familiarity values. After each change of the success probabilities the security value drops while the arousal value increases. During the relearning phases (after trial 100 and 200), the security value climbs up again and the arousal declines. In the presented experiment, multiple changes of the arm configurations occur which additionally reduces the learning performance at each change. This behavior is recognized by the detectors and processed by the Zurich model, hence after each additional change the security value climbs up more slowly and the arousal declines slower. The reduced security and higher arousal values are mapped more frequently to terms like uncertainty and aversion, while constant phases of joy and trust are shorter.

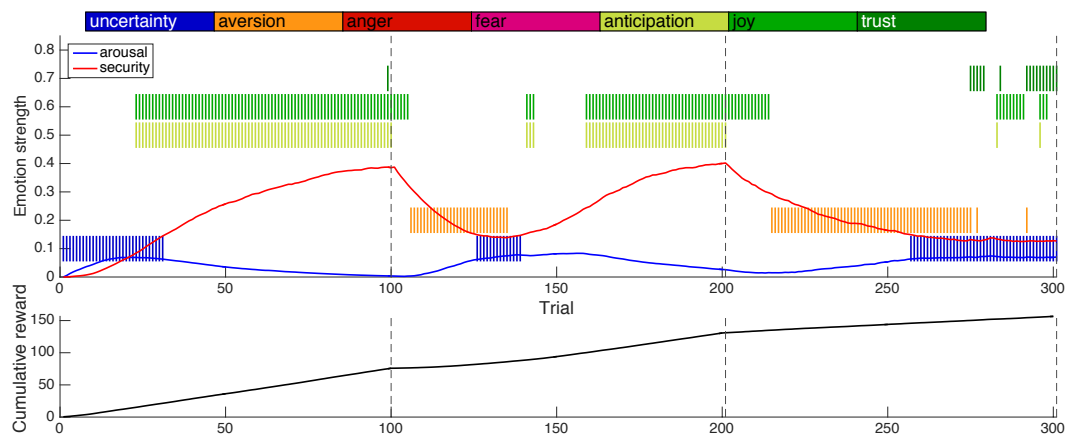


Figure 3.3.: Emotional evaluation (upper part) of a multi-armed bandit problem using the Gittins index policy. The lower part depicts the cumulative reward of 300 trials averaged over 50 independent trials.

Concluding remarks on the emotional evaluation of a bandit problem: *In this first experiment the principle approach to evaluate a machine learning algorithm with a psychological informed model was shown. The Zurich model has proven that it can be combined with an arbitrary machine learning scenario by just adapting the appraisal derivation layer (the detectors). In the experimental settings, the results show that the security and arousal values represent the actual learning progress and changed conditions. The emotional*

3.1. Bandit Simulation

terms can be mapped to more concrete emotional terms via a simple heuristic. Obviously, the same results might be generated by directly mapping the rewards to appropriate emotional terms. However, such a direct mapping does not fulfill the requirement of an extendable model. In case of the Zurich model, additional appraisals can be added by implementing corresponding detectors while the essential dynamic and logical properties of the Zurich model are maintained. In this way, a system can be developed which appraises several internal states or variables in order to express them in terms of artificial emotional expressions towards a human user.

3.2. Gridworld

In the previous section, a first approach of appraising the progress of a machine learning algorithm applying a psychological model was introduced. The presented Multi-Armed Bandit scenario is often used to model decision scenarios with immediate rewards. However, MABs are not suited to model scenarios with an episodic reward. For example, classical path finding problems are difficult to model within the MAB framework and therefore *Markov Decision Processes (MDPs)* are used. As stated in Section 2.3, MDPs model decision problems, where each state transition only depends on the current state and on the actions available in this state. Each state transition yields new state information and a possible reward. In this section, a canonical example of the Reinforcement Learning domain – the *Gridworld* – is used to model a path finding problem, while the MDP properties are essentially satisfied. The Gridworld is often used in the context of Reinforcement Learning to evaluate new learning algorithms. It is generally a rectangular maze consisting of surrounding walls and some obstacles. The maze is discretized into equal sized quadratic fields, and each field neither obstructed by a wall nor an obstacle corresponds to a state. Figure 3.4 shows an example of a Gridworld with a size of 9 times 6 fields, three obstacles

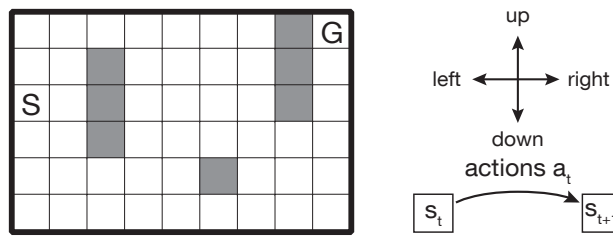


Figure 3.4.: An example of a *Gridworld* and the four possible actions a_t an agent can execute. Each action a_t causes a state transition $s_t \rightarrow s_{t+1}$ with a specific probability.

(dark grey), a start position S , and goal state G . A possible agent is then placed at the start position and has to find the goal location by just using the predetermined actions. In case of the example in Figure 3.4, the agent can only move into the Cartesian directions, namely up, down, left, and right. Each action causes a state transition from state s_t to state s_{t+1} with a particular probability $p(s_{t+1}|s_t, a)$. In general, assuming a deterministic environment, the transition probabilities are set to one for each unobstructed move and to zero in case of walls and obstacles. In this scenario, a Reinforcement Learning agent should find the optimal path to the goal state, while simultaneously appraising its current state with artificial emotions.

3.2.1. Implementation

A Q-learning agent, as described in Section 2.3, was implemented and a simulation of the Gridworld as depicted in Figure 3.4 was created. The experiment is con-

ducted in episodes and the agent starts in the start state S . Each episode is terminated after the agent has reached the goal state G . In the goal state, the agent receives a high positive reward ($r = 10$), while all other states are rewarded with a small negative reward ($r = -1$). The negative reward forces the agent to optimize its path towards the shortest path to the goal. Action selection is done according to the Q-values using an ϵ -greedy policy. Additionally, after each step the agent performs additional planning to improve the Q-representation of the environment. Besides the optional planning step, the RL process can be separated into several components. These components are then implemented according to the three level architecture of Affective Computing as proposed in Section 2.2.4. Therefore, Q-learning is split into a discretization step, the ϵ -greedy action selection, an updating function for the Q-table, and a reward perception function. In Figure 3.5, these sub-functions of RL are assigned to a corresponding level (light blue boxes). Planning is assigned to the reflection level, as it is an additional and optional step of RL and requires some computational effort. In the routine level, all necessary and periodic tasks, as the state discretization, action selection, and the Q-value update are placed. The reward detection is implemented as a fast and reactive function in the lowest level of the architecture, as it is just triggered in the goal state.

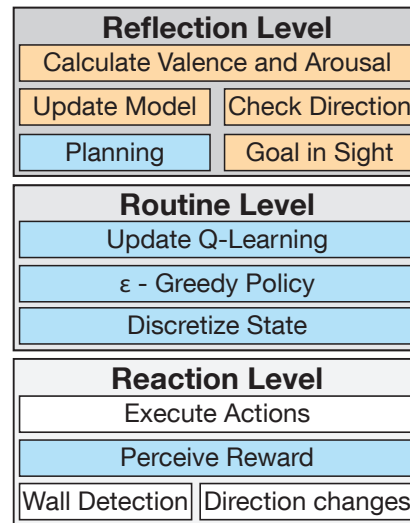


Figure 3.5.: Components of Reinforcement Learning (light blue) and a dimensional appraisal model (orange) integrated into the three level model of affect and cognition.

3.2.2. Core Affect

A dimensional model of emotions as introduced in Section 2.1 is used to appraise the decisions made by the Q-learning agent. Such a dimensional representation of the core affect is often selected in technical systems, as modeling the whole core affect of a system with a single point in a multi-dimensional space has some advantages (Russell and Barrett, 1999; Russell, 2003). On the one hand, discrete events can be used to modify the position of this point, while preserving a continuous change of the overall core affect. On the other hand, dimensional models output just this single point for each time step combining all obtained appraisal results. This efficient method of combining appraisal results is also often used as a final step in emotion models like the CPM or OCC model. Therefore, implementing a dimensional model in order to continuously represent the overall affective state of a technical agent is

3. Affective Evaluation of Machine Learning Experiments

not only intuitive but also fulfills important requirements for creating believable models of emotions.

Therefore, the three level architecture of Figure 3.5 is extended at the reflection level with functions which calculate a valence and arousal value appraising the Reinforcement Learning process. As calculating such an affective representation of the learning process requires information of lower levels and additional reasoning steps, these components perfectly fit into the reflection level (orange boxes). While the reaction level monitors changes in direction, collisions with walls, and obstacles, corresponding short term memory processes for each type of event are triggered in the reflection level. Each memory process is modeled by a variable corresponding to a frequency equivalent which models the rate of occurrence of an event. This is in accordance to findings of neurophysiology, which postulate that neuron activity and appraisal results in the human brain are related (Grandjean et al., 2008). That means, each frequency equivalent has a high value if the event occurs frequently and decreases to zero, if the event does not occur anymore (this concept is again similar to eligibility traces, however it uses a moving average function for calculating the frequency equivalent). Thus, in the current scenario a collision ($c_t = 1$) is reported to the reflection level which uses this information to compute a frequency equivalent f_t^{col} of the collisions. This can be expressed by

$$f_t^{col} = \begin{cases} f_{t-1}^{col} + (1 - f_{t-1}^{col}) \cdot \nu & \text{if } c_t = 1 \\ f_{t-1}^{col} + (-1 - f_{t-1}^{col}) \cdot \nu & \text{otherwise} \end{cases}, \quad (3.3)$$

which corresponds to a moving average filter, where ν describes the amount of residue that is added or subtracted from the previous frequency value. The filter responds to a collision in an exponential way. Its value does exponentially converge to one the more collisions occur, and decreases to minus one if no repeated collisions are observed anymore. This kind of representation is also used for the detection of changes in direction. The frequency equivalent f_t^{dir} is high, if the agent changes its direction in every step, and does decrease to minus one if long straight moves are executed. The change in direction is detected by comparing the previous action with the actual one. In case of a mismatch, the variable d_t is set to one, meaning that the agent must have changed its direction. A function in the reflection level monitors this variable and triggers the update for the change in direction frequency equivalent

$$f_t^{dir} = \begin{cases} f_{t-1}^{dir} + (1 - f_{t-1}^{dir}) \cdot \theta & \text{if } d_t = 1 \\ f_{t-1}^{dir} + (-1 - f_{t-1}^{dir}) \cdot \theta & \text{otherwise} \end{cases}, \quad (3.4)$$

with θ determining the amount of f_{t-1}^{dir} added to or subtracted from the previous value. These two monitor and update functions for the collision and the change in direction frequency equivalent are subsumed in the *Check Direction* component in Figure 3.5. A third frequency equivalent is regularly updated based on sightings of the goal state. Sighting means that a *Bresenham line algorithm* is used to calculate a line of sight between the agent and the goal state and if this line is unobstructed of a wall, the agent virtually sees

the goal state. Obviously, for this calculation the position of the goal is required, which would violate the model free assumption of the Reinforcement Learning agent. But, as the robot is just simulated in this experiment, the goal state detection would correspond to e.g. a camera based object detection in case of a real robot. The detection algorithm outputs a variable g_t which equals to one if the goal state is in sight, and is zero if the agent cannot directly observe the goal state. The frequency f_t^{goal} of this event is calculated by

$$f_t^{goal} = \begin{cases} f_{t-1}^{goal} + (1 - f_{t-1}^{goal}) \cdot \zeta & \text{if } g_t = 1 \\ f_{t-1}^{goal} + (-1 - f_{t-1}^{goal}) \cdot \zeta & \text{otherwise} \end{cases}, \quad (3.5)$$

where ζ specifies the amount of the residue that is added or subtracted from the previous value.

These three frequency equivalents are the actual appraisal results of the events caused by the RL process. They are used to calculate the arousal value A_t of the agent as a weighted average A_t given by

$$A_t = \frac{f_t^{dir} + f_t^{col} + g_t * f_t^{goal}}{2 + g_t}, \quad (3.6)$$

which can be further extended by averaging over additional appraisal variables. The valence value V_t is calculated using f_t^{goal} and the information if the distance between the agent and the starting point increases. If the distance increases ($e = 1$), the agent assumes to be on the right way, otherwise it assumes going the wrong way ($e = -1$). The variable e is also used to calculate a frequency equivalent f_t^{right} by

$$f_t^{right} = f_{t-1}^{right} + (e - f_{t-1}^{right}) \cdot \kappa, \quad (3.7)$$

where e sets the limit of the frequency value to which it converges to. The variable κ determines the slope. Similar to the arousal value, the valence component V_t is the (weighted) average of the corresponding appraisal variables. As the frequency of goal sightings f_t^{goal} and the fact going further away from the starting point f_t^{right} are the only indicators for the valence in this environment, V_t is calculated by

$$V_t = \frac{f_t^{goal} + f_t^{right}}{2}. \quad (3.8)$$

Both together, the arousal and the valence component represent the core affect of the RL agent in this experiment. The individual core affect components are calculated by weighting and averaging the appraisal variables, which base on features of the learning process itself and observable external variables. Expressing the appraisal variables as frequency equivalents enable smooth transitions between states. Overall, this approach represents an appraisal model and a dimensional representation of the core affect of the artificial agent and is implemented in the reflection level of the three level architecture of affect and cognition.

3. Affective Evaluation of Machine Learning Experiments

Table 3.3.: Preset parameters used in the Gridworld Q-learning experiment.

Parameter	Definition	Value
n	number of episodes	20
maxsteps	maximum number of steps per episode	2000
p_steps	number of planning steps	50
α	step-size	0.01
γ	discount-rate	0.95
ϵ	probability for random action in ϵ -greedy policy	0.1
$\nu, \theta,$	factor determining the slope of f^{col} and f^{dir}	0.15
ζ, κ	factor determining the slope f^{goal} and f^{right}	0.2

3.2.3. Experiment and Results

The proposed RL agent with its extended affective architecture is evaluated in the depicted Gridworld of Figure 3.4. All preset parameters used in this experiment for the residue factors and the Q-learning algorithm can be found in Table 3.3. The experiment was repeated ten times in order to smooth the results, as Reinforcement Learning together with the ϵ -greedy action selection produces slightly different results in each run. In each repetition of the experiment, the agent is placed at the starting point and has then to find the goal location. The Q-table is initialized with zeros for the first episode of each experiment. After the goal location has been reached or a maximum number of steps (*maxsteps*) per episode have been performed, the agent is reset to the start location. The already learned Q-table is preserved and the agent uses it in the subsequent episode to achieve a better result. After 20 consecutive episodes, the experiment is terminated. In Figure 3.6, the averaged results of the ten repetitions of the experiment are plotted. On the left side (Figure 3.6a), the resulting average per episode of the valence and arousal value is jointly plotted in the valence and arousal space. Each red dot corresponds to the overall average of the valence and arousal values of a complete episode. This average represents a measure for the overall performance of the agent for a complete episode. At the beginning of the experiment, during the first episodes (episodes 1 to 6) the overall valence value per episode is negative, which can be mapped to feelings of distress and sadness. During the experiment, the overall core affect per episode moves from quadrant Q4 towards Q2, where valence is positive and arousal is low, which can correspondingly be mapped to feelings of contentment and relaxation (cf. Section 2.1 and Figure 2.1). This movement of the core affect can be compared with the learning curve of the agent. The corresponding learning curve of this experiment is depicted in Figure 3.7. In contrast to this standard learning curve, the core affect can be read without any expert knowledge (e.g. there is no need to know how many steps are optimal) and therefore can be used to communicate the actual

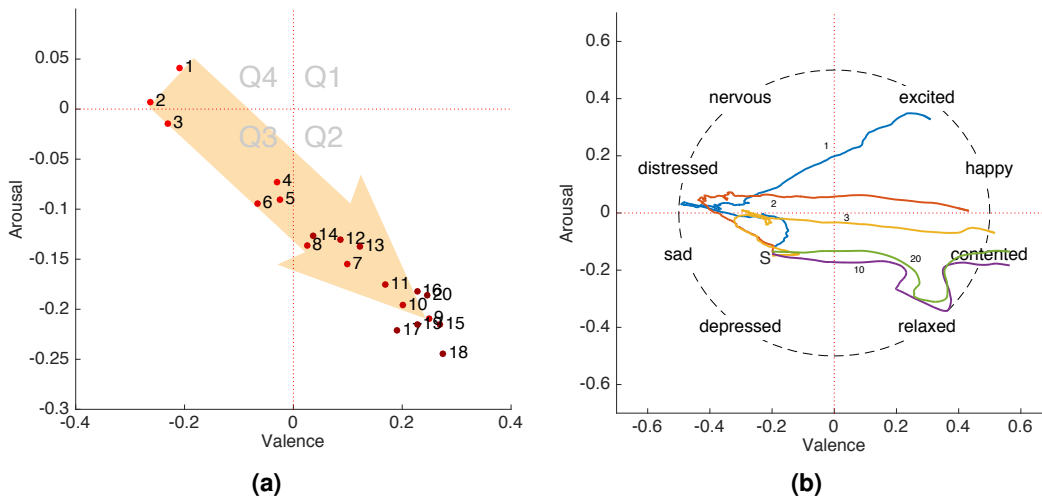


Figure 3.6.: (a) Average core affect of twenty subsequent episodes. The agent gets more relaxed with progress in learning. (b) Continuous core affect for five selected episodes. At very early episodes (1, 2, 3), the agent is more negatively aroused and gets relaxed and contented in later episodes (10, 20).

learning progress of this RL agent to average users. Additionally, the output of this appraisal model can also be used to drive an emotion display, like EDDIE (Sosnowski et al., 2006) or Kismet (Breazeal, 2003), which use arousal and valence values as a common interface.

The second plot on the right side (Figure 3.6b) shows the core affect's shift of complete episodes. For a better readability, only episode 1, 2, 3, 10, and 20 where selected and plotted. At the beginning of each episode, the frequency equivalents are also reset to zero and with the first appraisal the core affect takes a value of $V = -0.2$ and $A = -0.15$ (denoted with S in Figure 3.6b), which is determined by the residues factors ν , θ , ζ , and κ .

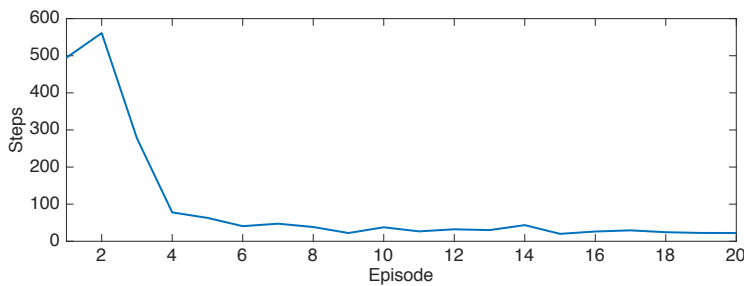


Figure 3.7.: Standard learning curve of 20 episodes of a Q-learning agent in the Gridworld scenario.

3. Affective Evaluation of Machine Learning Experiments

In the first episode, the agent spends a lot of time in negatively valenced states as the goal is not in sight and it explores the near environment of the starting location. The arousal value keeps neutral, as only some collisions occur and direction changes occur randomly. At the end of Episode 1 the agent gets more and more excited, as it is going into the right direction and the goal comes into sight. In subsequent episodes, the valence values already get positive after a few actions, as the agent can follow the policy determined by the Q-table. Also the arousal values go down, as only a low number of collisions and direction changes occur. The kink points just before the end in the graphs of episode 10 and 20 are caused by the necessary direction changes in order to avoid obstacles, and by the sighting of the goal.

Concluding remarks on the emotional evaluation of a way finding task: *Depending on the application, both the average or the immediate core affect can be used to efficiently and naturally communicate the current state of the Reinforcement Learning agent towards users. The proposed dimensional appraisal model uses just a few basic features of the learning process to calculate a joint representation of the agent's state. By using the valence-arousal space, the calculations of the valence and arousal values can be further extended by external appraisal results in order to affectively appraise additional processes of the agent. The mapping of the core affect into discrete emotion expressions is controversially discussed, but it allows a natural understanding of the two values. On the one hand, the discrete emotion expressions can be used to verbally express the current state of the learning progress, but on the other hand they can also be used to drive an advanced non-verbal emotion display which requires a continuous representation of the internal state. Overall, the approach has shown, that an appraisal derivation model based on frequency equivalents allows the appraisal of a canonical RL scenario and should therefore be applicable to other examples in this domain.*

3.3. Simultaneous Localization and Mapping

Most autonomous robots are faced with the problem of orientation in unknown and mostly dynamic environments. Normally, this problem is split into two sub-domains – localization and mapping. For both topics there exist many approaches and technologies. There are also algorithms combining both steps called *Simultaneous Localization and Mapping (SLAM)* algorithms. They are able to estimate the robot's position and simultaneously collect data to produce a representation of the environment. The results of those approaches are reasonable and can be quantitatively measured. Generally, the quality of the results of a SLAM algorithm is evaluated in terms of position accuracy and number of re-observed landmarks. However, in the following a way that enables the robot to mimic human emotions while moving around and exploring is described and should replace the classical *cold* and rational performance evaluation. In order to enable the robot to express its current state, whether it feels safe or uncertain in its environment, the following section describes another example of an appraisal derivation model which adapts the Component Process Model to a SLAM algorithm. The previously introduced concept of frequency equivalents is also used and extended.

SLAM algorithms are used for the navigation of mobile robot platforms. Especially, in indoor environments generally no map of the environment exists and the robot has to acquire mapping information in order to self localize. For this task, SLAM algorithms estimate a map and a location within this map based on odometry sensors of the platform and on landmarks, which are detected in the surrounding environment of the robot platform. In most cases, the position of the landmarks are optically detected (camera or LIDAR based), but also radio frequency beacons or similar techniques can be used. All of them deliver noisy measurements of the environment and use probabilistic filters to estimate a map and the ego position within this map. The probabilistic filters output position estimations which are accompanied by corresponding uncertainty measurements. The probabilistic filters used in SLAM algorithms can be coarsely categorized into Kalman Filters (KF), Particle Filters, Expectation Maximization techniques, and sparse extended information filters (SEIFs) (Grisetti et al., 2007; Bailey and Durrant-Whyte, 2006; Guivant and Nebot, 2001). Today, most state of the art SLAM algorithms base on Extended Kalman Filters (EKF) and fuse the positions of the landmarks with the data coming from the odometry sensors (Haykin, 2001). The EKF tracks the position of the robot and calculates uncertainty values for the robot's position and the positions of the landmarks. Then, the robot attempts to associate previously seen landmarks with actual detected landmarks. The re-observed landmarks are then used to correct the robot's position. The whole process is iteratively executed and enables a mobile robot to build a map of an environment and simultaneously use this map to estimate its location within this map. There exist many implementations of EKF-SLAM with various alterations and modifications. They share the basic idea of calculating a fully correlated posterior over the landmark map and the robot pose, but simultaneously suffer on the strong assumptions that have to be made on the sensor noise

3. Affective Evaluation of Machine Learning Experiments

and robot motion model during the implementation of the Extended Kalman Filter. Nevertheless, in this experiment, an EKF-SLAM simulator written by Tim Bailey and Juan Nieto¹ is used. The framework provides a straightforward implementation of the algorithm and permits quick access to a full SLAM simulator. The use of a simulation relaxes the necessary model assumption and allows to focus on the appraisal derivation layer. Also, the path planning of the robot is done manually, as the dynamic path planning is an additional and complex task and is not an essential part in the SLAM simulation. The following description of the appraisal derivation model can be adapted to other SLAM algorithms as well by identifying the corresponding uncertainty measurements of the used probabilistic filter or model.

3.3.1. SLEmotion

In contrast to the previous experiment, where a direct mapping between appraisal derivation model and dimensional representation (categorization) space was used, in this experiment an appraisal derivation model is implemented and the affective evaluation is done according to the structure of the Component Process Model (CPM) as introduced in Section 2.2.1. This takes account of the requirement to implement models following existing principles and being reusable by other researchers. The appraisal of a way finding and orientation task of a robot in an unknown environment in terms of emotions was selected as scenario. The various sensor inputs and internal representations of the SLAM algorithm are used to generate the corresponding stimuli for the CPM. In the present scenario, the emotional process is restricted to the localization and mapping task of the robot and does not consider any effects caused by external stimuli (like other agents, humans, or general motivational changes of the agent). Also, the implemented component patterning module is limited to the calculation of expressive emotions, and internal effects caused by the simulated appraisals do not influence subsystems of the robot. The simplification of the scenario corresponds to a situation in which the agent expresses its current emotional state while exploring an unknown environment without being obviously observed or influenced by anyone else. As motivational changes and physiological responses are omitted in this technical scenario, the component patterning module and the categorization module can be combined using the dimensional theory of a valence and arousal space to categorize the emerging emotions. Such a combination overcomes the difficulties that still exist in implementing a complete recursive and dynamic classification of the current appraisal and the affective state of an agent. Similarly, this approach is also proposed by Scherer (Scherer, 2004) and used in other implementations like the WASABI architecture (cf. Section 2.2.1). Additionally, as already mentioned the two-dimensional representation is used by other works as input for the visual (e.g. EDDIE (Sosnowski et al., 2006) or Kismet

¹http://www-personal.acfr.usyd.edu.au/tbailey/software/slam_simulations.htm [Accessed 1st September 2016]

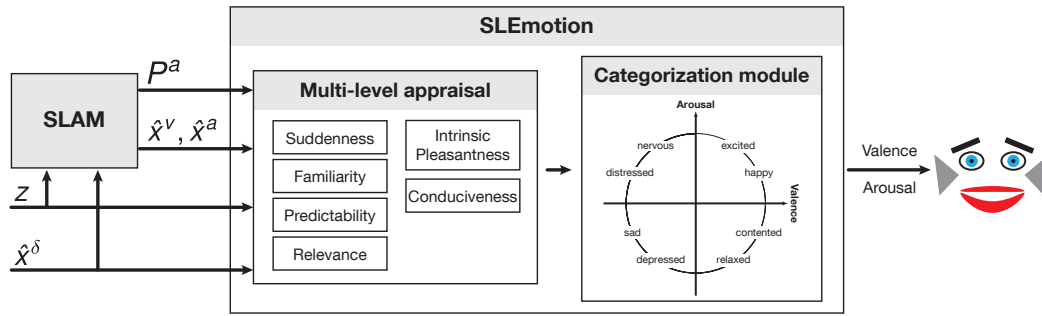


Figure 3.8.: Architecture of the Component Process Model (CPM) with its appraisal and categorization module (SLEmotion). The appraisal module determines the values of the appraisal registers based on the features of a Simultaneous Localization and Mapping (SLAM) process. The two-dimensional output of valence and arousal can be used to drive an emotion display like EDDIE or Kismet.

(Breazeal, 2003)) or behavioral (e.g. Bethel and Murphy (2008) and Miwa et al. (2003)) based rendering of the emerging emotions.

Figure 3.8 depicts the current CPM architecture, called *SLEmotion*, in the context of the SLAM algorithm. The SLAM algorithm bases on landmark observations $z \in \mathbb{R}^{n \times 2}$, where n is the number of actual detected landmarks, and on the odometry sensor values $\hat{x}^\delta \in \mathbb{R}^{1 \times 2}$. With these input data, the agent calculates and updates a covariance matrix $P^a \in \mathbb{R}^{m \times m}$ of the m successfully detected and associated landmarks, and estimates its current position $\hat{x}^v = (\hat{x}^v, \hat{y}^v) \in \mathbb{R}^{1 \times 2}$ and a map $\hat{x}^a = (\hat{x}^a, \hat{y}^a) \in \mathbb{R}^{m \times 2}$ of the environment. Observations and resulting estimations of the SLAM algorithm are the input variables for the appraisal derivation module, which performs multi-level appraisals on them – the so called Stimulus Evaluation Checks (SECs). The SECs are independent modules, in which the agent does perform its subjective assessment of the situation on a background of personal needs, goals, and values. In the original CPM several SECs are grouped into four major types or classes of information concerning the following different aspects: *Relevance*, *implications*, *coping potential*, *normative significance*. Coping potential and normative significance are not in the scope of the present scenario, as the SLAM agent is externally controlled and any interaction with tasks and other agents or humans is considered separately. Therefore, in the current implementation of SLEmotion the following SECs appraise the overall situation of the agent and each occurring event during the SLAM process:

- **Suddenness:** Measures the abruptness of onset of an event or a change in the situation.
- **Familiarity:** Checks if an event or the situation is familiar. Generally, this is performed by evaluating results of schema matching or the frequencies of occurrence.

3. Affective Evaluation of Machine Learning Experiments

- **Predictability:** Basing on past observations of regularities and probabilities for specific events the predictability is measured.
- **Relevance:** Evaluates the situation and events according to the agent's goals and needs. It depends on how many different goals are affected by an event and how often it occurs.
- **Intrinsic pleasantness:** Assesses a stimulus according to the resulting feeling of pain or well-being.
- **Conduciveness:** Evaluates the conduciveness of a situation or an event according to the goals and needs of the agent.

The first four SECs, suddenness, familiarity, predictability, and relevance are associated with the *relevance* group of the SECs, while intrinsic pleasantness and conduciveness are in the group of *implications*. All results of these Stimulus Evaluation Checks are stored in corresponding registers for each occurring event or situative context delivering a specific appraisal pattern at each time step. Each event and the environment is perceived by the agent with specific stimuli. In case of the considered SLAM agent, these stimuli are virtually restricted to the detection of new or already detected landmarks and changes of the situation within the scenario. But, as the multi-level appraisal module bases on discrete components for each SEC, the SLEmotion architecture can easily be extended to cover additional events and situations. In the following, the corresponding Stimulus Evaluation Checks appraising the Simultaneous Localization and Mapping task are described in detail.

3.3.2. Stimulus Evaluation Checks

The intuition of the Stimulus Evaluation Checks is defined in the model description by Scherer (2010), but the exact computational details are missing there. Therefore, according to the SEC descriptions of Scherer, a developer has to design a computational equivalent for a register in the actual domain. A convenient way to validate these equivalents is to compare the results with the expectations of humans (this is done in the results subsection of this section). In the present experiment, the SECs are exemplary stated for the used EKF-SLAM algorithm, and therefore represent one of many possible solutions for modeling the appraisal process of a robot navigation task. However, the underlying concepts of each SEC can be applied to other Simultaneous Localization and Mapping implementations by identifying the corresponding uncertainty measures of the particular SLAM algorithm.

As already mentioned, the Stimulus Evaluation Checks are working independently, and appraise both, single events and the whole situation. Therefore, the individual SECs are internally divided into a check routine for a single event and a routine for appraising the overall situation. As SLEmotion calls all SECs in parallel, in each iteration appraisal results for the overall situation are produced regularly, while events are only appraised if they

3.3. Simultaneous Localization and Mapping

have occurred in the actual time step. For this reason, in the following the situative and event-based component of each SEC is described separately. Basically, there exist more situative stimuli and events occurring during the runtime of a robot, but to illustrate the concept of integrating the SLAM task into the CPM architecture, these additional appraisals are omitted in this experiment.

Unless otherwise stated, as most other existing appraisal models the individual SECs compute values in a range between $[0, 1]$ or $[-1, 1]$ (e.g. Marinier et al. (2009); Gratch and Marsella (2005)). The implication is that the absolute value indicates the strength of a stimulus. That means, the "1"-end of the range is more intense than the "0"-start of the range. And for other dimensions, a range of $[-1, 1]$ is used to express a negative and a positive intensity for the same type of stimulus, e.g. for conduciveness both ends of the range are necessary. There are events and situations which can be highly conducive or (+1, e.g. passing an exam) or other events causing a very unconducive (-1 , failing in an exam) stimulus.

Intrinsic pleasantness: The check for intrinsic pleasantness evaluates with respect to the progress of achieving a goal, whether the current situation or a certain event will result in future to negative or positive outcomes (result in pain or well-being). In case of the SLAM scenario, the main goal of the agent is to achieve a stable and accurate localization in a self-generated map. Therefore, the concept of local and global security was created and represents the agent's feeling of security in the global map, and at the current position, respectively. Given the characterization of Scherer (2010) that pleasantness correlates with attractive situations, while unpleasantness results in avoidance, the feeling of security relates to this characterization. The agent aspires to feel globally and locally secure, while it simultaneously tries to avoid uncertain situations.

Consequently, local security is a measure for the detection quality of the immediate and actively observable environment, while global security evaluates the overall accuracy of the current process in relation to the runtime. Global security sec^{global} is therefore implemented by

$$sec^{global} = \left[\frac{a_{map}}{p_{dist}} \cdot \sqrt{\sigma_{xv}^2 + \sigma_{yv}^2} \right]_{-1}^1, \quad (3.9)$$

where a_{map} is the current map size, p_{dist} the overall traveled distance of the agent, and $\sqrt{\sigma_{xv}^2 + \sigma_{yv}^2}$ is the estimated standard deviation of the agent's position (it is assumed that the variances of the coordinate axis are stochastically independent). The standard deviation is estimated by the EKF used in the SLAM algorithm and would linearly increase if the update step repeatedly fails. The operator $[\cdot]_a^b$ limits the value of global security to a range between $[-1, 1]$, and is defined as

$$[x]_b^a = \begin{cases} a & \text{if } x > a, \\ b & \text{if } x \leq b, \\ x & \text{otherwise.} \end{cases} \quad (3.10)$$

3. Affective Evaluation of Machine Learning Experiments

This concept of global security neglects the fact that the agent never knows its absolute position in the world, and therefore should never have the feeling of absolute security. Nevertheless, in a controlled environment this simplification is acceptable. Further, it should be noted that time dependence in the above and all following equations is omitted for a better readability. Each variable, unless otherwise stated, is updated at each time step.

Local security bases on the detection accuracy of surrounding landmarks. As direct and regular observation of landmarks increases the confidence in them, the correlation values decrease correspondingly. This interrelation of landmarks is contained in the correlation matrix P^a and regularly updated by the EKF on each detection. While the correlation between two landmarks i and j with the estimated coordinates $\hat{\mathbf{x}}_i^a = (\hat{x}_i, \hat{y}_i)$ and $\hat{\mathbf{x}}_j^a = (\hat{x}_j, \hat{y}_j)$ is defined by the mean correlation coefficient

$$corr_{ij} = \frac{1}{2} (corr(\hat{x}_i, \hat{x}_j) + corr(\hat{y}_i, \hat{y}_j)), \quad (3.11)$$

where

$$corr(\hat{x}_i, \hat{x}_j) = \frac{\sigma_{\hat{x}_i, \hat{x}_j}}{\sigma_{\hat{x}_i} \sigma_{\hat{x}_j}}, \text{ and } corr(\hat{y}_i, \hat{y}_j) = \frac{\sigma_{\hat{y}_i, \hat{y}_j}}{\sigma_{\hat{y}_i} \sigma_{\hat{y}_j}}, \quad (3.12)$$

respectively, are the correlation coefficients for each axis of coordinates. The standard deviation for each coordinate component of a landmark is denoted with $\sigma_{\hat{x}_i}$ or $\sigma_{\hat{y}_i}$, and the covariance between two landmarks is denoted with $\sigma_{\hat{x}_i, \hat{x}_j}$ or $\sigma_{\hat{y}_i, \hat{y}_j}$, respectively. These pairwise interrelations are depicted in Figure 3.9a, where a black dot corresponds to a landmark and the lines in between visualize the correlation coefficient. The darker the color the stronger the correlation between both landmarks. In a subsequent step, a joint representation of the correlation coefficients is calculated by creating a regular grid equal to the area spanned by the detected landmarks. The present line representation is then quantized into this grid, resulting in a matrix which contains the sum of correlation coefficients at the positions of the sampling points of each line. This matrix is then circular interpolated and filtered with a dilation kernel. An exemplary result is depicted in Figure 3.9b, where the darker colors correspond to higher joint correlation values. The local security value sec^{local} is then derived by looking up the joint correlation value in this matrix according to the agent's current position, hence these calculations can be expressed as a function of

$$sec^{local} = f(\hat{\mathbf{x}}^v, P^a). \quad (3.13)$$

Finally, the global security value sec^{global} of Equation 3.9 is added to the local security value by

$$intp^{sit} = \left[\frac{1}{2} (sec^{global} + sec^{local}) \right]_{-1}^1 \quad (3.14)$$

to calculate the intrinsic pleasantness value $intp^{sit}$ of the actual situation. The operator $[\cdot]_{-1}^1$ limits it to a range between $[-1, 1]$. As mentioned above, there exist additional stimuli of the environment influencing the intrinsic pleasantness value of the overall situation and

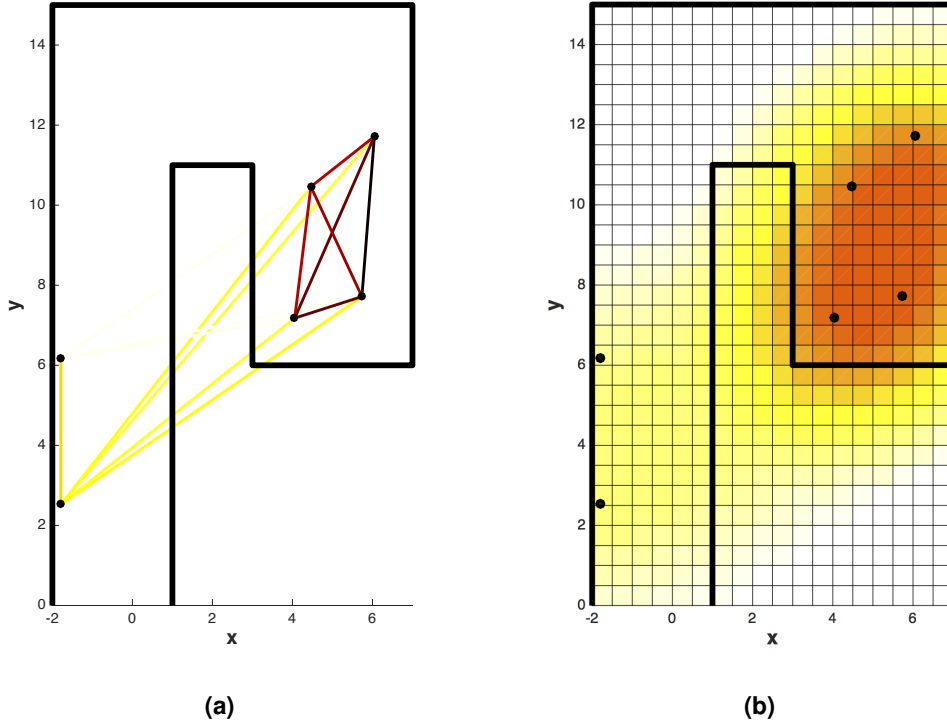


Figure 3.9.: Visual representation of the correlation coefficients between pairs of landmarks. In subplot **(a)** the correlation value between two landmarks is represented as a line with a corresponding color. The darker the color the higher the correlation coefficient. In subplot **(b)** the line representation is converted into a lookup table for the combined correlation value at each position.

they can be added into the calculation at this point. But, for the current experiment, these two variables were selected as they mainly influence the situation.

In contrast, the contribution of events to the intrinsic pleasantness value is primarily influenced by the detection and re-detection of landmarks. Most SLAM algorithms depend on precise sensor data and each new data sample improves the localization and map. Therefore, a high detection rate of landmarks and the reduction of the localization error is *pleasant* for the agent. A suitable modeling of this fact is again the correlation value between the agent and a particular landmark. So, the intrinsic pleasantness $intp^{lm}$ of a detection or update of a landmark can be expressed as

$$intp^{lm} = \begin{cases} \frac{1}{2} \left(\frac{\sigma_{\hat{x}^v, \hat{x}^{lm}}}{\sigma_{\hat{x}^v, \hat{x}^v} \sigma_{\hat{x}^{lm}, \hat{x}^{lm}}} + \frac{\sigma_{\hat{y}^v, \hat{y}^{lm}}}{\sigma_{\hat{y}^v, \hat{y}^v} \sigma_{\hat{y}^{lm}, \hat{y}^{lm}}} \right), & \text{for updating a landmark } lm; \\ 1, & \text{in case of a new landmark.} \end{cases} \quad (3.15)$$

In this equation, the correlation between the robot and a landmark expresses the precision

3. Affective Evaluation of Machine Learning Experiments

of the detection after each update. It is high if the agent's position estimation and the estimation of the landmark position is linear dependent, and low if the standard deviations of the estimations increase. A newly detected landmark is always considered as a positive event, and initially appraised with a fixed value of one. The correlation value might be an oversimplification for appraising the detection quality of landmarks, but in the simulation it delivers good results and can be extended in real world scenarios by confidence values of the real sensors.

Certainly, there are a lot of more events on a real robot, which contribute to the intrinsic pleasantness of the robot itself and the SLAM process, but as mentioned above, this experiment should show the concept of combining these processes and might be extended in future versions.

Conduciveness: In the Component Process Model, conduciveness is the appraisal of the situation or events with regard to the goals and needs of a human. Events favorable to achieve a desired goal or fulfilling a need have to be appraised positively while events taking one away from a goal should be negatively assessed. Similarly, a situation is conducive if the new situation feels subjectively safer than before, or if sub-goals are achieved on the way to a goal. This conception can be used within SLEmotion to model the perception of the situation by considering the change in local security in dependence to the variance of the agent's estimated position. Computationally, this can be expressed by

$$cond_t^{sit} = \frac{1}{2} \left(\frac{std_{t-1}^v - std_t^v}{std_{t-1}^v} + \frac{sec_{t-1}^{local} - sec_t^{local}}{sec_{t-1}^{local}} \right), \quad (3.16)$$

where the std_t^v corresponds the combined standard deviation of the agent at time step t , and $t - 1$ indicates that the value of the previous time step is taken. The second addend adds the change of the local security value sec^{local} at the agent's current position to the situative conduciveness value. Additional features could also be added, such as a measure for the current map size, a frequency equivalent of re-detected landmarks, and similar criteria that assess the increasing quality of the SLAM task.

Besides the overall situation, also single events can be appraised according to their conduciveness. Since in this experiment the focus is on the SLAM process, the only occurring events are the detection of landmarks. Therefore, each detection or re-detection of a landmark is appraised according to its impact on the improvement of the mapping. A suitable indicator of the utility of a detected landmark is the density of landmarks d^{lm} within a specific range. This can be approximated by

$$d^{lm} = \sum_{j=1}^N \exp \left(-\frac{\|\mathbf{x}^{lm} - \mathbf{x}^j\|}{\tau_{density}} \right), \quad (3.17)$$

where \mathbf{x}^{lm} is the position of the current (re-)detected landmark and \mathbf{x}^j the coordinates of the N surrounding landmarks within an adjustable range. The distance between the actual

landmark and its neighbors is calculated by the Euclidean norm $\|\cdot\|$ and shortened by an exponential function with a decay constant $\tau_{density}$. Then, all pairwise distances between the landmarks are summed up, resulting in a density value. This density can be used as conduciveness value $cond^{lm}$ for a (re-)detected landmark by

$$cond^{lm} = 1 - d^{lm}, \quad (3.18)$$

where d^{lm} ideally is limited to a range of $[0, 1]$ by setting $\tau_{density}$ and the calculation range in relation to the scenario (an example is given below). The setting depends on the detection range of the sensors, and should be selected slightly smaller than the actual detection range. In this way, the resulting conduciveness value is low in cases where a lot of landmarks are in the calculation range, and high if only a few or no landmarks are in range.

Suddenness: Estimating the direction and distance in an environment without clearly observable landmarks (like in a desert or a dark room), is only possible in relation to the starting location by counting the steps and direction changes. In robotics, this type of navigation which uses only internal motion sensors is called *odometry*. The odometry based position estimation can be corrected in case of encountering a landmark. Small deviations are common and are expected, but encountering an already known landmark without expecting it, requires a large correction of the internally estimated position. This is also the motivation for the in the following introduced calculation of the suddenness value.

As the SLAM algorithm uses an Extended Kalman Filter for predicting and correcting the agent's position, all building blocks are already in place. The odometry sensors of the robot deliver a coarse estimation of the agent's position and are corrected by the landmark sightings. Comparing an assumed position $\hat{\mathbf{x}}^{V,a}$ created by superimposing n_{odo} subsequent odometry sensor measurements with the estimated and corrected agent's position as calculated by the EKF, results in a good model for suddenness. The greater the deviation between pure odometry based position estimation and SLAM based position estimation, the larger the correction step $\delta_t^{correction}$. A visualization of the approach for calculating the

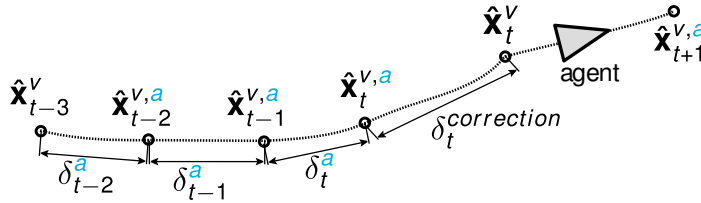


Figure 3.10.: Approach for calculating $\delta_t^{correction}$ which measures the probability of falsely associated landmarks.

distance $\delta_t^{correction}$ is depicted in Figure 3.10. The distances δ^a are the results of the e.g. $n_{odo} = 3$ subsequent odometry based estimation steps, resulting in a position $\hat{\mathbf{x}}_t^{V,a}$. This position is compared with the actual EKF based estimation by

$$\delta_t^{prediction} = \|\hat{\mathbf{x}}_t^V - \hat{\mathbf{x}}_t^{V,a}\|. \quad (3.19)$$

3. Affective Evaluation of Machine Learning Experiments

The smaller this prediction error the smaller the expected deviations in the detected landmarks, and the lower the astonishment created by falsely associated landmarks. This relation can be used to calculate the overall suddenness value sud^{sit} by

$$sud^{sit} = \frac{\delta_t^{prediction}}{\|\hat{\mathbf{x}}_t^{V,a}\| \cdot sud^{sit,max}}, \quad (3.20)$$

where $sud^{sit,max}$ is the overall maximum value of sud^{sit} and is used to normalize the suddenness value in relation to its maximum value.

Apart from this overall situation dependent suddenness value, also event based suddenness values are created for the sightings of landmarks. Landmarks can either be newly detected or re-detected. Re-detection can occur in case of landmarks within the field of view of the agent, or in case of already known landmarks which get associated again after coming into the field of view. As only new landmarks are surprising, the suddenness values for them are highest, while the value for landmarks permanently in the field of view is consequently lowest. The value for known landmarks coming again into view is neutral for the case they get associated correctly, otherwise they are treated as new landmarks. Hence, the suddenness value for the event of landmark sightings can be stated by

$$sud^{lm} = \begin{cases} 1, & \text{new landmark } lm; \\ 0, & \text{known and re-detected landmark;} \\ -1, & \text{permanently visible landmark.} \end{cases} \quad (3.21)$$

This suddenness model uses fixed values for each landmark detection, which might be a slightly coarse model. But as all event based calculations are averaged later on in the component pattering module, this is sufficient.

Familiarity: This SEC appraises familiarity of events and the situation using schema matching and frequencies of occurrence (Scherer, 1987). Familiarity depends on the frequency and duration of visiting and staying at a particular place. There are similarities between this definition and the pheromone trail of ants (Dorigo et al., 2006). The update rule for each visited position \mathbf{x}^v of the agent according to the model of a pheromone trail can be implemented by

$$i_t^{path}(\mathbf{x}^v) = \begin{cases} i_t^{path,init}(\mathbf{x}^v), & t = t^{visit} \\ i_{t-1}^{path}(\mathbf{x}^v) \exp\left(\frac{-t}{\tau_{path}}\right), & t > t^{visit} \end{cases}, \quad (3.22)$$

where τ_{path} is the time constant of the trail. For each visited position the values are updated at each iteration, while values below a one percent threshold of the original value are removed from the trail. Additionally, neighboring positions of the trail positions are also gradually initialized and updated with the pheromone value to continuously fade out the trail. The pheromone values for each particular position are topped up in case of repetitive visits. As this model of a pheromone trail depends on both, duration and frequency

3.3. Simultaneous Localization and Mapping

of a visited location, the situative familiarity value can be modeled by taking the sum of pheromone values at the agent's current position. Thereof it follows for the final situative familiarity value

$$fam_t^{sit} = \sum_{j^{path} \in \mathcal{I}(\mathbf{x}_t^V)} j^{path}, \quad (3.23)$$

where $\mathcal{I}(\mathbf{x}_t^V)$ is a set of pheromone values still updated at a particular position \mathbf{x}_t^V . A similar concept of gradually fading out a familiarity value is used for the update of detected landmarks. Regularly detecting a landmark increases its familiarity value, while longer observation pauses let the agent forget the landmark after a specific time. Therefore, the familiarity register for each observed landmark is updated by

$$fam_t^{lm} = \begin{cases} fam_{t-1}^{lm} + (1 - fam_{t-1}^{lm}) \cdot \eta_{rise}, & \text{during the time of regular detection,} \\ fam_{t-1}^{lm} + (-1 - fam_{t-1}^{lm}) \cdot \eta_{decay}, & \text{while the landmark} \\ & \text{lm is not directly observable,} \end{cases} \quad (3.24)$$

where η_{rise} and η_{decay} is the amount of residue added or subtracted respectively from the previous value at each time step. Besides the detection of landmarks, additional event based familiarity registers might be added to the Stimulus Evaluation Checks. There might be event triggers for loop closings in the traveled path or triggers for achieving particular thresholds of the correlation value. These additional events were omitted and subjected to future extensions.

Predictability: The predictability register of the situation should represent the vagueness or predictability of the current environment. In the context of the SLAM task, the environment would be predictable without noise and disturbances influencing the detection of landmarks and the odometry sensors. With increasing noise and unpredictable disturbances, the uncertainty increases and the positions of landmarks can only be estimated. As the disturbances of a system are hard to predict, a more suitable way is to compare them relatively to each other.

The standard deviation of the agent's position std^V represents the impact of noise and disturbances to the SLAM process. Therefore, using a running maximum of the standard deviation $std^{V,max}$ and comparing it to the current value, results in a fair model for predictability. The comparison can be implemented by

$$pred^{sit} = \left[2 \cdot \left(\frac{std^V}{std^{V,max}} \right) - 1 \right]_{-1}^1 \quad (3.25)$$

and is limited to a range of $[-1, 1]$.

Predictability might be the most challenging register to implement it in a technical system, as it appraises the ability to predict the environment. Generally, humans use their experience to appraise events and situations in relation to their predictability, while an artificial agent lacks this pool of experience. Such a pool of experience can be represented in

3. Affective Evaluation of Machine Learning Experiments

a technical system with a set of memory variables. Therefore, in Equation 3.25 the experience is modeled by storing a running maximum of the standard deviation in a persistent variable.

Also, the re-detection of a landmark can be appraised according to its predictability and linearly depends on the standard deviation of the agent's position. New landmarks however, are not predictable and therefore should always be appraised with the lowest possible value. Hence, the events triggered by observing landmarks are appraised with regard to their predictability by the following function:

$$pred^{lm} = \begin{cases} pred^{sit}, & \text{for actual observed landmarks} \\ -1, & \text{in case of a new landmark } lm. \end{cases} \quad (3.26)$$

Relevance: The relevance value is the most important register in regard to its impact and is calculated with the highest priority at the beginning of each appraisal process. It serves as a filter for checking the relevance of incoming stimuli. In case of the situation appraisal, the relevance value depends on the current task, and is highest (= 1) for all SECs related to the current task. Therefore, in the present scenario, the relevance register rel^{sit} for the situation is set to one while the agent moves or observes its environment. It is set to lower values for all situative SECs of the SLAM task while the agent performs another task.

The relevance values for events triggered by a landmark detection depend on the novelty of the landmark and therefore decrease with the duration of continuous observations. This can be modeled by an exponentially decaying short term memory process, which is expressed by

$$rel^{lm} = \begin{cases} 1 - \exp\left(\frac{t^{observed,lm} - t}{\tau_{rel}}\right), & \text{while a landmark } lm \text{ is observed} \\ \exp\left(\frac{t^{observed,lm} - t}{\tau_{rel}}\right), & \text{otherwise,} \end{cases} \quad (3.27)$$

where $t^{observed,lm}$ is the time the landmark lm was observed for the first time, and τ_{rel} the time constant for the decay of the relevance value of a landmark.

3.3.3. Categorization Module

The CPM postulates a recursive appraisal process constantly updating the appraisal results by evaluating or re-evaluating environmental changes. With these individual appraisal results a joint representation of the core affect can be created by mapping the different registers onto the two dimensional space of valence and arousal (see Section 2.1). The mapping calculates a point in the two dimensional VA-space for each appraisal result. In a subsequent step, all resulting points are combined with the previous core affect resulting in a single point within the VA-space representing the actual core affect of the agent at a specific time.

3.3. Simultaneous Localization and Mapping

Currently, in psychology there does not exist an extensive theory about the mapping from appraisal results to emotions. The reasons for this are the lack of comprehensive studies, as most researchers are focused on specific emotion components, and that currently most emotion research bases on self reports of subjects. Without such a theoretical foundation of the mapping process, most technical approaches base on integration rules using differential weighting of the various response components or in some cases non-linear functions to fuse the components together. In SLEmotion the mapping from the appraisal results to the valence and arousal dimensions is done by an averaging model. Averaging models are commonly used in psychology to describe relations between stimuli and measured responses (Anderson, 1989). Furthermore, Scherer (2009) suggests that there exists a direct correlation between the dimension of valence and the appraisals of intrinsic pleasantness and goal conduciveness, as well as a correlation between the arousal (activation) dimension and the appraisals related to relevance, familiarity, and suddenness. These correlations together with the prototypical appraisal profiles of Scherer are the basis for the following integration of the appraisal results. The combined valence value v_t of an appraised stimulus is calculated by

$$v_t = \frac{1}{v_{sit} + v_{lm}} \cdot \left(v_{sit} \cdot \frac{s_{intp}intp_t^{sit} + s_{cond}cond_t^{sit}}{s_{intp} + s_{cond}} + v_{lm} \cdot \frac{l_{intp}intp_t^{lm} + l_{cond}cond_t^{lm}}{l_{intp} + l_{cond}} \right), \quad (3.28)$$

using the registers of intrinsic pleasantness and conduciveness. The factor v_{sit} weights the influence of the situation, while v_{lm} weights the influence of the appraisals of landmark events (in the remainder of this section constants are denoted with a subscript, except time dependent variables which are denoted with a subscript t). Correspondingly, the arousal value a_t combines the following registers

$$a_t = \frac{1}{a_{sit} + a_{lm}} \cdot \left(a_{sit} \cdot \frac{s_{sud}sud_t^{sit} + s_{fam}fam_t^{sit} + s_{pred}pred_t^{sit} + s_{rel}rel_{sit}}{s_{sud} + s_{fam} + s_{pred} + s_{rel}} + a_{lm} \cdot \frac{l_{sud}sud_t^{lm} + l_{fam}fam_t^{lm} + l_{pred}pred_t^{lm} + l_{rel}rel_t^{lm}}{l_{sud} + l_{fam} + l_{pred} + l_{rel}} \right), \quad (3.29)$$

where the factors a_{sit} and a_{lm} correspond to the arousal dimension. During the experiments, an imbalance between the impact of the situation and the impact of appraised landmark observations has been detected. Thus, the weight factors v_{sit} , v_{lm} , a_{sit} , and a_{lm} had to be determined accordingly. Since landmark observations occur more frequently, the impact of these stimuli must be damped in respect to the appraisal of the overall situation. Calculating a running average of simultaneously observed landmarks could be used to adjust the weightings dynamically. For more deterministic results in simulations, an average value can be calculated offline.

The factors denoted with $s_{intp/cond/sud/fam/pred/rel}$ and $l_{intp/cond/sud/fam/pred/rel}$, respectively, are used to fine-tune the individual registers during the integration (the default value is 1). These values can be used to optimize the dynamic of the model by biasing specific registers.

3. Affective Evaluation of Machine Learning Experiments

Each new stimulus (a situation check or an event) results in a new appraisal, which is in turn mapped into the VA-space as a single point. Similar to the approaches of Miwa et al. (2003) and of Becker-Asano and Wachsmuth (2010), the individual points are used to dynamically shift the core affect of the agent, which is represented as a reference point in the VA-space. In each iteration, the core affect is shifted in the direction of the most recent appraisal result. The amount of each shift depends on the distance between the core affect point and the stimulus point. It is additionally weighted by w_v in the valence dimension and w_a in the arousal dimension. These two values mainly influence the dynamic and smoothness of the resulting core affect *course*, and can be set almost arbitrarily as they correspond to the personality values of the agent.

Obviously, the appraisal derivation model and its Stimulus Evaluation Checks contain some hand-tuned parameters. However, they were reduced to a minimum, but some of them have to be adjusted according to the sampling time of the simulation and to the number of simulation steps, or according to the desired *personality* of the agent. In the following experiment, the parameters were adjusted to support most of the situation the agent is faced with during the experiment.

3.3.4. Experiment and Study

The proposed SLEmotion architecture is used in a simulation of a robot exploring an office environment while calculating appraisals for events and the overall situation. The simulated robot uses the EKF-SLAM algorithm and the described SLEmotion model to generate the two artificial feelings of valence v_t and arousal a_t , which represent the current core affect of the robot at each time step t . Both feelings should be elicited at different intensity levels in each room. The environment consists of a map with eight rooms (six are used during the experiment) and a connecting hallway (Figure 3.11).

During the experiment, the agent visits six of these rooms on a predefined path (green line). In each of these six rooms, the number of recognizable objects (landmarks) differs significantly. The path was designed in a way that the robot encounters distinct situations. For these situations, the following hypotheses which base on the underlying ideas of the developed models for the individual Stimulus Evaluation Checks can be stated:

- H1:** In a situation with a high number of recognizable objects, the arousal value will be lower, and the valence value will be higher than in a situation without objects.
- H2:** In a situation with an adequate number of objects for a safe navigation, the arousal value will be higher and the valence value will be lower than in a situation with a lot of objects.
- H3:** In a situation with an adequate number of objects for a safe navigation, the arousal value will be lower and the valence value will be higher than in a situation without recognizable objects.

H4: Uncertain situations where no objects can be recognized will result in negative valence values.

In order to evaluate these hypotheses with respect to the expectations of humans in similar situations a survey was developed. The survey consists of a questionnaire and was split into two parts. In the first part, the participants should assess three situations: being in a dark room, coming home from a daily walk, and entering a familiar environment after a longer period of absence (e.g. holiday). In the different textual descriptions of the situations the number of familiar and recognizable objects was pointed out (the full descriptions and the questionnaire can be found in the Appendix A.2). The participants

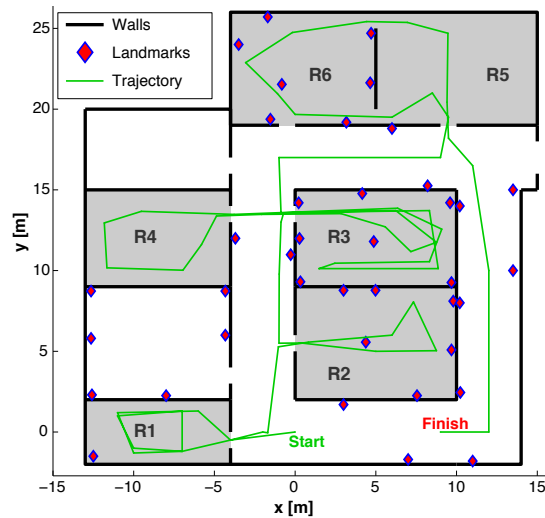


Figure 3.11.: Map used for the SLAM experiment with rooms $R1$ to $R6$, landmarks (red/ blue rhombi), and the given trajectory (green line).

were asked to assess each situation by writing down an emotion or feeling they would expect to have in such a situation. Additionally, they were asked to rate this feeling on the Self-Assessment-Manikin (SAM) scale (Bradley and Lang, 1994), which is able to directly quantify a feeling in the two dimensions of valence and arousal. This enables to directly compare the human's assessments with the simulated feelings of SLEmotion independently of a specific type of emotion display. As the first part was primarily designed to create self reports for the different situations and to prepare the participants for the second part, they were asked in the second part to report on their expectations of artificial feelings a service robot is supposed to express in similar environments. Unlike the textual descriptions in the first part, in the second part of the survey the robot scenarios are illustrated by plots of corresponding rooms which include a specific number of objects and the trajectory of the robot. Compared to the map used for the simulation (Figure 3.11), the illustrations for the questionnaire have a slightly increased level of detail (colored boxes of varying sizes). Illustration S1 represents rooms with a lot of landmarks (similar

3. Affective Evaluation of Machine Learning Experiments

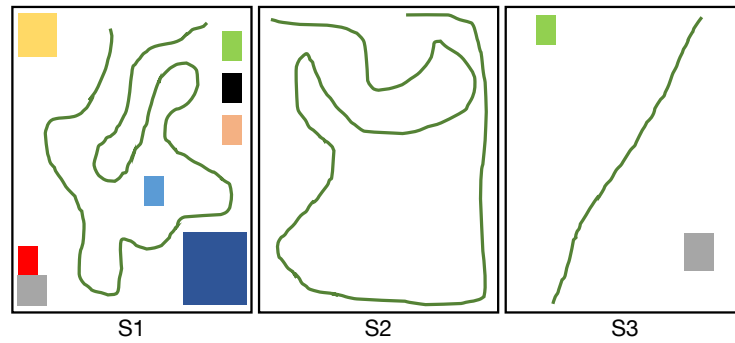


Figure 3.12.: Illustrations of the three rooms the subjects had to evaluate in the survey to support the hypotheses of the SLEmotion experiment. Colored boxes are objects and the robot's trajectory is plotted in green.

to $R2$, $R3$, $R6$ in in the simulation, Figure 3.11), $S2$ corresponds to a situation where no landmarks can be detected (e.g. $R4$, $R5$), and $S3$ is an illustration of a hallway. During the survey, the participants were asked to think about possible feelings a service robot would experience in such a scenario. It was additionally told, that the robot's navigation system and path planning ability depends on the number of objects the robot can recognize in its environment. The subjects should then write down their expectations on the robot's feeling in a text field and additionally rate them on the SAM scale. For the results, the mean values and the standard deviations of the ratings on the SAM scale where calculated. The given hypotheses were validated by performing an analysis of variance (ANOVA) and a corresponding Bonferroni post hoc analysis.

To compare the results of the study with the experimental data of the SLEmotion model, the model parameters are set to match the simulation environment and to adjust the dynamic of the agent conservatively. The simulation time was set to $T = 230$ seconds and a sampling rate of $f_s = 0.025$ seconds resulting in 9.200 simulation steps. According to the sampling rate the time constants τ_{path} , $\tau_{density}$, and τ_{rel} were set. The time constant for the decay of the familiarity value of the visited locations (Equation (3.22)) was set to $\tau_{path} = 100$, which means, that the way-points' impact vanishes (smaller than one percent of the original value) after about 460 simulation steps. For the landmarks, the time constants $\tau_{density} = \tau_{rel} = 5$ are comparatively small, as there are more simultaneous landmark events and only the constantly observed landmarks should influence the current appraisal result. The rise and fall times of the short term memory process, η_{rise} and η_{decay} , respectively, used in the calculation of the landmark familiarity register are adjusted to a rise time four times faster than the decay time. Generally, a landmark is only appraised if it is within the detection range d_m , which depends on the used sensors. In the present scenario, the size of the simulated environment is approximately 25x25 meters while the sensors can detect a landmark within a radius of $d_m = 5$ meters. With this detection range an

Table 3.4.: Parameters adjusting the dynamic of the arousal and valence signals according to the simulation time and desired agent behavior.

Parameter	Description	Value
T, f_s	simulation time, sampling time	230s, 0.025s
τ_{path}	time constant of the path decay	100
$\tau_{density}$	time constant of the decay of the density of landmarks	5
τ_{rel}	time constant of the relevance decay	5
η_{rise}	rise time for the short term memory process	0.04
η_{decay}	fall time of the short term memory process	0.01
d_{lm}	detection range of sensors	5m
v_{sit}, a_{sit}	weightings adjusting the impact of the situation	7
v_{lm}, a_{lm}	weightings adjusting the impact of landmark sightings	1
s_{fam}	weighting of the situational familiarity register	7
l_{intp}	weighting of the intrinsic pleasantness registers related to landmarks	2
w_v	amount of a valence appraisal added to the core affect per time step	$\frac{1}{10}$
w_a	amount of a arousal appraisal added to the core affect per time step	$\frac{1}{20}$

average number of seven detected landmarks per time step can be estimated. Therefore, the weighting for the situation was set seven times higher, $v_{sit} = a_{sit} = 7$, than for the landmarks, $v_{lm} = a_{lm} = 1$. These weights balance the impact of landmark appraisals and the overall situation appraisal and similarly can be calculated with a running average of simultaneously detected landmarks in an online scenario. To fine-tune the categorization module, each register can be biased with a weight which is 1 by default. Currently, only the familiarity register is biased with a factor of $s_{fam} = 7$ and the intrinsic pleasantness register of the landmarks has a gain factor of $l_{intp} = 2$. These two values were set in order to maximize the dynamic range of the simulation. Finally, the core affect is superimposed by fractions of the individual appraisal results using the factors $w_v = \frac{1}{10}$ and $w_a = \frac{1}{20}$.

All parameters, which have to be set manually are summarized in Table 3.4. Obviously, most of them are hand-tuned and currently require some expert knowledge to adjust them. However, most of these parameters can be replaced with variables calculated online as a function of appraisal frequency and simulation time.

3.3.5. Results

Figure 3.13 depicts the simulated valence and arousal values for a trajectory of the robot visiting the rooms $R1$ to $R6$. In rooms $R1$ and $R3$ the robot roams around to get more familiar with the environment, these loops are denoted with $R1loop$ and $R3loop$. At the

3. Affective Evaluation of Machine Learning Experiments

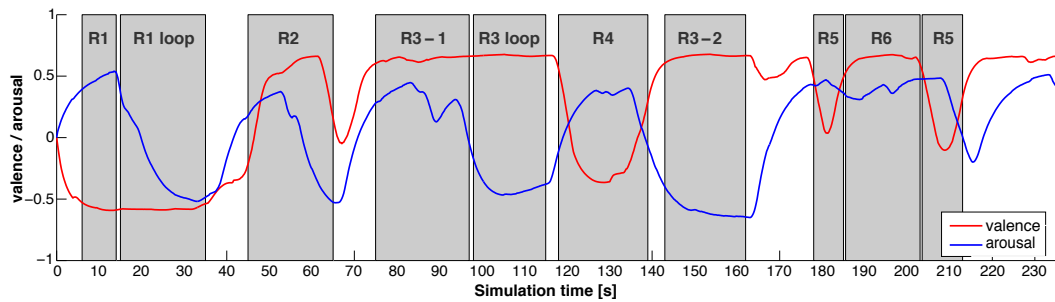


Figure 3.13.: Plot of the valence (red) and arousal (blue) values during the simultaneous localization and mapping process. The labels $R1$ to $R6$ correspond to the visited rooms of the map in Figure 3.11.

beginning, the initial values of valence and arousal are set to zero. From the starting point towards room $R1$ the robot is able to detect only one landmark. This results in a bad performance of the SLAM algorithm and the agent appraises its situation with a negative feeling of valence (following hypothesis H4) and a high value of arousal. After roaming around in room $R1$ the arousal decreases. Considering the dimensional theory of Russell and Mehrabian (1977), such a combination of valence and arousal values (the core affect) characterizes emotions like sadness and depression.

The accuracy of the SLAM algorithm increases when entering room $R2$ and $R3$ which feature a higher number of landmarks. This higher number of landmarks and the increased feeling of local security results in appraisal results shifting the core affect of the agent towards the first quadrant of the core affect space (cf. Figure 2.1). This fulfills hypotheses H1 and H2. Excitement and happiness are characteristic emotions of the first quadrant.

The situation changes drastically after the agent enters room $R4$ without any landmarks. This results in a fast transition of the core affect towards the second quadrant, where emotions of fear and anxiety are located. Hypotheses H3 and H4 are underpinned in this room. Leaving room $R4$ and directly re-entering room $R3$, which was already explored, results in small detection errors but high correlation values. This in turn increases the agent's feeling of local and global security, pushing the core affect into the fourth quadrant, that is characteristic for feelings of relaxation and contentment.

After leaving room $R3$ the agent travels along the hallway entering room $R5$. During this travel, the agent is able to detect a sufficient number of landmarks resulting in a fair accuracy of the SLAM algorithm and concomitant high values of valence. In contrast, the arousal values steadily increase due to the newly detected landmarks. Entering room $R5$ reduces the valence values for a moment, as the room does not contain any landmark. The transition to room $R6$ generates high valence values since there are many landmarks for a safe navigation. Simultaneously, there are elicited high arousal values, since most of the landmarks are newly detected. After leaving room $R6$ and $R5$ the simulation is ended.

Overall, the simulation results fulfill the model's underlying design considerations and

the extracted hypotheses H1 to H4. In the next step, these hypotheses will be verified according to the expectations of humans. For this reason, in a study, 30 voluntary participants (26 male, 4 female) mostly students in Electrical Engineering, ranging in age from 20 to 49 (MD = 27.0, SD = 5.2) were asked. They were divided randomly into three groups and a questionnaire with a rotating order of the questions was assigned to each group to avoid biases. For the following results only the ratings made on the SAM scale were used and mapped linearly into a value range between -1 and 1 in order to match with the output of SLEmotion. The additionally collected textual self reports for each surveyed situation were used to detect possible misunderstandings of the questionnaire.

The first situation $S1$, which was evaluated by the agent and by the human participants was a room with a lot of good recognizable objects. Rooms $R2$ and $R3$ were designed to represent such a situation. SLEmotion calculates an average valence value of $v = 0.293$, while the arousal values decay over time to lower values ($R2, R3$ on average $a = -0.19$). In comparison, the human participants reported on the SAM scale an average of $v = 0.45$ ($SD = 0.360$) and $a = -0.567$ ($SD = 0.469$) for being in a similar situation, and an average of $v = 0.283$ ($SD = 0.468$) and $a = -0.267$ ($SD = 0.469$) for their expectation on the feeling a service robot would express.

In the second situation $S2$, no landmarks can be observed ($R4$ and $R5$). This results in increasing uncertainties of the SLAM process and is evaluated by SLEmotion with (negative) valence values, relatively lower than the corresponding arousal values ($R4$ avg. $v = -0.144$, $a = 0.253$). These values are characteristic for negative emotions like sadness and fear. This is also in accordance with the self-reports of the participants, which have evaluated a situation of being in a dark and unknown basement storage room with an average of $v = -0.383$ ($SD = 0.387$) and $a = 0.35$ ($SD = 0.397$). Further, the average values for the expected emotion of the service robot in a room without any landmarks were $v = -0.517$ ($SD = 0.404$) and $a = 0.25$ ($SD = 0.537$).

The third situation $S3$, represents a standard situation with a sufficient number of landmarks for a stable navigation. This is the case for almost all rooms and the hallway in the simulation, except for $R4$ and $R5$. In those situations, SLEmotion calculates valence values relatively higher than the arousal values. In the survey, the participants were asked what they would expect of a service robot traveling along a hallway. The result was an average of $v = 0.133$ ($SD = 0.454$) and $a = -0.5$ ($SD = 0.347$). In the simulation, similar tendencies can be observed after the robot leaves a room.

Comparing the average values of the simulation have supported hypotheses H1 to H4. For the experimental data of the second part of the survey, an analysis of variance (ANOVA) was performed. Additionally, pairwise tests (using the Bonferroni correction) between the three situations, $S1$ - lot of landmarks, $S2$ - no landmarks, $S3$ - sufficient landmarks were performed. A significant main effect was found for arousal ($F[1, 29] = 16.30, p < 0.001$) and valence ($F[1, 29] = 41.87, p < 0.01$). Pairwise comparisons support hypotheses H1 and H3 by a significantly ($p < 0.001$) lower arousal value and a significantly higher valence value ($p < 0.001$) in $S1$ than in $S2$ (H1), respectively in $S3$ than in $S2$ (H3). The differences between $S1$ and $S3$ were not significant, therefore

3. Affective Evaluation of Machine Learning Experiments

hypothesis H2 is not fully supported by the conducted field study. One reason might be the misinterpretation of the robot's path within the illustration. The subjects reported in the closing interview that they were unsure how they should interpret the quirky kinks and curves of the green paths in the illustrations (Figure 3.12). Finally, hypothesis H4 was supported by a one-sample T-test revealing that the valence value in S2 is significantly lower than 0; $t(29) = -6.998, p < 0.001$.

Concluding remarks on the emotional evaluation of a SLAM algorithm: *The intention of this experiment was to extend the communication abilities of an artificial agent by an emotional component. The developed emotional component uses the two dimensional core affect to output the internal status of an underlying complex SLAM algorithm. In this way, the status can be mapped to distinct emotional expressions. All calculated emotions base on internal appraisal variables, which are calculated by utilizing only the decisions, observations and previous appraisal variables of the agent. The underlying emotion model is based on the Component Process Model which has proven to be implementable and modular. The simulated valence and arousal values change according to the accuracy and temporal behavior of the SLAM process and can be used to drive an appropriate emotion display. The results were compared with the expectations of humans observing a service robot in similar situations. The results support the idea that an artificial appraisal model enables the fusion of numerical features to a coherent and extensible representation, expressing the state of an underlying process in terms of emotions. This helps to improve the efficiency and intelligibility of communication of complex system states towards non-expert users. Obviously, the simulated emotions are not intended to represent the full emotional experience of the agent but can be seen as a component of the agent's emotional and cognitive architecture.*

4. Affective Control

Up to this point, the described methods improve the communication ability of the agent itself. By drawing on the concepts described in the background chapter, also the ability of the agent to react to its artificial emotions would be essential. Taking the assumption that an agent is able to calculate such artificial emotional states, this can not only enhance its communication abilities, it can also be used to control several conditions in its behavior. In the following, two scenarios where an emotion based assessment of the situation would be useful are briefly outlined.

The aforementioned exploration-exploitation dilemma is currently either treated by time-dependent or approximation procedures which reduce the percentage of exploration steps or subsume equal states. Most of the classic algorithms favor exploration at the beginning of the learning task and reduce exploration towards later stages of learning (Axelrod and Chowdhary, 2015). When the environment is non-stationary, like most real-world scenarios, state-visitation index based methods, or open-loop ϵ -greedy strategies will fail. One solution to change the agent's behavior and adapting its exploration-exploitation bias would be to utilize a meta-model on the expected change of the environment. In Section 3.3, a corresponding model for the assessment of the current environment (SLEmotion) with emotions was proposed. In support of such a SLAM scenario the agent could use the calculated core affect to control its behavior in an uncertain environment. For example, the agent could change its exploration behavior according to the current core affect. Since the valence value of a situation coarsely depends on the detection accuracy of the landmarks, and the arousal value similarly depends on the number of occurring events, a programmer of the agent could use these values to create control rules for the agent. These control rules could, for example, serve to switch between different detection modes, or regulate the approaching speed towards human users. In contrast to conventional control algorithms, such control rules can be formulated in an intuitive way expressing preferences for actions in specific regions of the core affect. Moreover, since the rules are defined according to the core affect, the set of control rules and the appraisal model can be extended independently of each other. In other words, implementing a meta-model such as the introduced SLEmotion enables the artificial intelligence designer to formulate preferences of the agent according to its intuitively understandable internal state.

In the second example, a virtual agent in the context of business intelligence is considered. The agent has no direct relation to the physical world and simulates its internal representation of an emotional state in dependence on the interactions with other users or virtual agents. The underlying decision logic of the agent bases on Reinforcement Learning and a cost signal is used as reward. By way of illustration, a negotiation between

4. Affective Control

several teams within a company is considered. Each team has different projects and corresponding goals to achieve, but they use similar machines to produce their goods. In case a machine breaks down, each team tries to get a replacement part as fast as possible. Therefore, they have installed an automated system for each team in form of an artificial virtual agent looking for necessary replacement parts. In the first instance, this agent tries to obtain the part within the company while respecting its own objectives and simultaneously the aims of the other teams. Like in the previous example, a meta-model is used to appraise the own and the objectives of the other teams resulting in combined and intuitively understandable appraisal results. That means, if the internal negotiations between the project teams fail, the agent negatively appraises the overall situation – the agent gets unhappy – and starts to look for the replacement part at external suppliers. In this *unhappy* state, the agent's preferences change, which is useful during the negotiations with external suppliers. External procurement of the spare part requires focus on price and delivery time, while internal procurement requires to find the optimal trade-off between all involved parties. As the agent uses Reinforcement Learning for selecting the optimal action (the decision to externally order the part, or to use an internally available part at the costs of another team) the calculated internal appraisal result can be used during the action selection process in order to adjust the multiple objective weightings.

Considering both examples, this leads to the following two questions: First, how can affective states be used to control Reinforcement Learning in case of non-stationary environments? And second, why are intuitive representations like affective states beneficial to control the learning process?

In the remainder of this chapter, these two questions will be treated by extending the reward representation of Reinforcement Learning with a multi-dimensional signal which is weighted on the basis of an additional state. This state is designed to be independent from the RL process so that it can be controlled by a process simulating the knowledge and experience of an agent while preserving all major properties of RL. Finally, numerical experiments show that the proposed method is capable to learn different preferences in a manner sensitive to the agent's level of experience.

4.1. Reinforcement Learning with Preferences

Decisions are mostly influenced by personal preferences which are the result of past experiences. In order to model preferences within the RL framework reward shaping is an essential component (cf. Section 2.3.3). The different components of the reward signal can then be shaped according to variable weights. This requires that the environment assigns different components of reward r^{ext} to a goal and the ability of the agent to add internal components r^{int} to the reward signal. With this multi-dimensional reward signal, the agent can learn to prefer specific reward components by assigning appropriate weights to it. There are different options for selecting these weights. In multi-dimensional RL algorithms it is tried to maximize the final reward by setting the weights to combinations resulting in

Pareto-optimal solutions. In contrast to that, the weights can also be set to fulfill individual preferences of the agent by adjusting them according to internal states of the agent. The problem lies in the definition of such internal states, and should consider the interrelations between the artificial emotional system, the preferences, and past experiences of the agent. Unfortunately, the currently existing emotional models are incomplete and cover only a subset of artificial emotions in the context of specific applications (cf. Chapter 3). They are by far not sufficiently complete to define a state that controls preferences in a reward weighting process of Reinforcement Learning in dependence to this state. Therefore, this general definition is still a future topic of research and the following algorithm assumes a proper definition of this affective state S^A .

S^A should intuitively represent a state, subsuming all underlying internal states of the agent and thereby e.g. represents the level of experience, arousal, happiness, or curiosity of the agent. Defined in this way, the state can be used to control the preference model of a given task and to calculate the reward weightings accordingly. Basically, S^A is the argument of a weighting function and there exists a set of such functions for each individual reward component. The result of evaluating this set of functions is a multi-dimensional weight vector in dependence to the affective state S^A . Finally, this vector can be applied to the multi-dimensional reward signal in order to weight the reward components according to the preference model and S^A . In Figure 4.1, the extended framework is depicted for such a

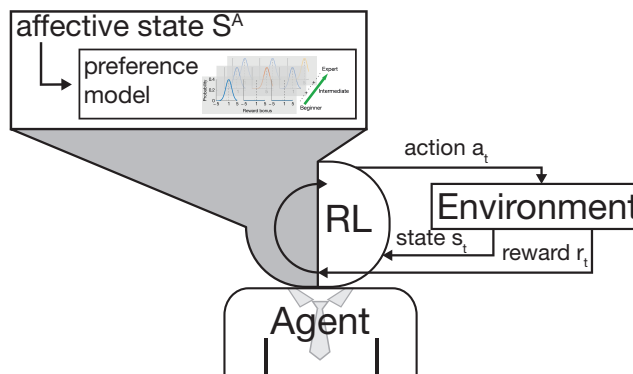


Figure 4.1.: Reinforcement Learning framework with an integrated preference model and affective state regulation. The rewards are modified by the preference model which is controlled by the affective state S^A .

one-dimensional affective state. The RL process itself is retained and only the reward function is replaced by a multi-objective reward function. Each reward component represents a specific property of the task. In the current setting, a one-dimensional affective state is chosen to demonstrate the principle method, however it is possible to use multiple states to control different properties of the task. As the affective state S^A compactly summarizes previous internal states and experiences of the agent, so that all state relevant information is retained, it still fulfills the Markov property. Therefore, the environmental state signal

4. Affective Control

s_t , and the external reward r_t^{ext} which is given by the environment can be used to drive RL algorithms. Reward scalarization as introduced by Vamplew et al. (2011), is used to derive a scalar reward r_t by weighting and summing up the individual reward components assigned by the environment in each time step.

Algorithm 2 RL with preferences

```
1: procedure TASK(1)
2:   Update  $S^A$  continuously
3: end procedure

4: procedure TASK(2)
5:   while  $s_t$  is not the terminal state do
6:     Take action  $a_t$ , observe external reward  $r_{t+1}^{ext}$  and new state  $s_{t+1}$ 
7:     Add internal reward components  $r_{t+1}^{int}$  to the reward signal
8:      $r_{t+1} = [\mathbf{w}^{int}(S_t^A) \quad \mathbf{w}^{ext}(S_t^A)] \cdot [r_{t+1}^{int} \quad r_{t+1}^{ext}]^T$   $\triangleright$  Calculate scalar reward using  $S_t^A$ 
9:     Update value function  $V(s_t)$  with  $r_{t+1}$ 
10:    Select next action  $a_{t+1}$  using  $V(s_{t+1})$ 
11:    Advance to next state:  $s_t \leftarrow s_{t+1}$ 
12:   end while
13: end procedure
```

In Algorithm 2, the corresponding weights for the internal and the external reward components are denoted with $\mathbf{w}^{int}(S_t^A)$ and $\mathbf{w}^{ext}(S_t^A)$, respectively. Note that these weights are functions of the affective state S^A , as they are dependent on the current affective state S_t^A . Similar to the basic idea of Prospect Theory, S_t^A can be seen as a reference point of a weighting function for a specific reward component. The correct choice of this weighting function depends on the desired behavior of the agent and on the implemented model of S_t^A . A concrete example for a model of S_t^A and the weighting function is shown in the following experiment.

4.2. Experiment

In this basic experiment, the design of a preference model and a corresponding affective state is shown. In contrast to the examples given at the beginning of the chapter, this experiment considers the complete learning process of an RL agent in a new environment, in which the agent additionally first has to learn how the reward process works. The following two vivid examples further motivate the experiment:

Example 1: With the assumption that there is a brand-new cooking robot which gets the order to cook fried eggs, the learning behavior can be described as follows. At the beginning, the robot is a novice with basic cooking abilities and has access to the necessary

equipment and ingredients, like eggs, a pan, and a flipper. The robot starts baking the eggs until the score for comparing the result with a picture of a fried eggs exceeds a certain threshold. Then the fried eggs are served, the owner of the robot tastes them, and tells the robot that it made indeed fried eggs, but that the yolk has to be cooked through. Up to now, the robot in its inexperienced state only has judged the result according to the appearance and basic recipe, but has not considered the new dimension which is now added: the consistency of the yolk. Next time the robot is to cook fried eggs (with its increased level of experience), it will take this new dimension into account and evaluate the result accordingly (e.g. by using an additional sensor to measure the yolk temperature).

Example 2: This example is related to a more complex assessment and a special case for machines: the taste of coffee. As a beginner in drinking coffee, one judges coffee according to the overall taste of bitterness or sweetness and probably the temperature. After drinking coffee regularly, one starts to taste different flavors within a specific kind of coffee. Consequently, with increasing experience and regularity in drinking coffee, new dimensions of possible *rewards* are added to one's preference model. For the perfect coffee experience, an expert chooses coffee according to a variety of different tastes and ways of preparation. A similar concept of an unfolding reward space according to an experience state is not considered in machine learning scenarios so far.

Keeping these two examples in mind, a more realistic reward process can be described by the three following components:

1. A primary reward that is assigned for completing the task and that is independent of the level of experience of the agent.
2. Additional reward components for the completion of sub-goals.
3. Subjective reward components that are internally generated by the agent according to its current affective state and level of experience.

The first two components are the external rewards, denoted by r_t^{ext} , while the third component subsumes the internal rewards r_t^{int} . These internal rewards are only generated by the agent at specific affective states (e.g. for being happy), or as a function of the current level of experience. That means, the internal rewards are only *perceived* by the agent in case of certain affective states, or in case the experience level exceeds a certain level so that the weightings of an underlying preference model influence the overall reward. An exemplary preference model which depends on S_t^A is depicted in Figure 4.2. In this model, three different weighting functions are assumed. The first one, $w^{ext,1}(S_t^A)$ assigns a constant weight of 1 to the first external reward component. This is also the default function, which is always used to weight the reward for the primary goal. The remaining two functions, $w^{ext,2}(S_t^A)$ and $w^{ext,3}(S_t^A)$, are applied to the second and third external reward component and increase exponentially with an increasing value of S_t^A towards their maxima. $w^{ext,2}(S_t^A)$

4. Affective Control

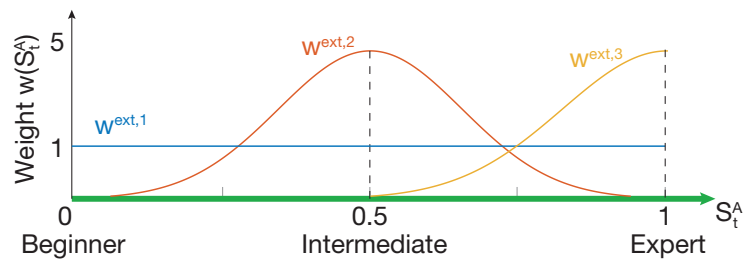


Figure 4.2.: Weighting functions for three different shaping rewards in relation to the affective state S_t^A .

decreases in turn exponentially once a specific value of S_t^A is reached. It is important to note, that the agent does not *perceive* a reward component if the corresponding weight is zero, thus in the example of Figure 4.2 the agent optimizes its learning behavior at the beginning ($S_t^A = 0$) only towards the first external reward component. The remaining components are then taken into account by the agent relatively to the affective state S_t^A and the corresponding weighting function.

Thus, the reward function is a combination of externally and internally assigned rewards, which are only perceived by the agent if a corresponding weight is greater than zero. As the weighting functions depend on the affective state S_t^A , specific reward components are biased according to this state. In this first approach, S_t^A is modeled as an experience signal which increases over time according to a sigmoid shaped function. Sigmoid shaped learning curves are often used to represent the gain in experience in non-trivial tasks which require numerous repeated trials to master them, and also motor skill experience curves take this shape (Leibowitz et al., 2010; Woodworth, 1938). The experience curve can be divided into three phases: beginner, intermediate, and expert. In the first phase, the level of experience increases exponentially, while slowing down in the intermediate phase, and finally settle at the expert level. This model of experience signal also takes into account that the experience level cannot decrease, which is on the one hand natural, but also implies that forgetting effects have to be modeled by the weighting functions. This also means, however, that new reward components can only be added to the reward signal and can never be removed (this is also called the *no-go-back-policy*). The only way to dampen the effect of a specific reward component after initially adding it to the reward signal is to reduce its weight, which can be done by modeling the weighting function accordingly.

The experience signal generation or a similar signal representing the affective state of the agent seems to be a substantial question. In fact, it is an extensive question and would go beyond the scope of this dissertation. There are several solutions conceivable for integrating the affective states, experiences of the agent, and externally given feedback into a combined signal for shaping the rewards. Generally, the requirements on this control signal are the correlation between the learned actions and its outcomes, the internal affective states, temporal effects, and external feedback. But as also neurophysiological findings suggest that a clear separation of this control signal from the actual learning process is

beneficial (LeDoux, 1998). Therefore, for the following modeling and the experiment, the described simple approximation of a monotonically increasing and time dependent experience signal is used.

With this basic model for S_t^A and the three weighting functions of Figure 4.2, a first simulation of learning preferences within the RL framework can be performed. For the simulation, a hypothetical decision problem of a three-armed bandit was chosen. The return of each arm of the bandit was set to a Gaussian distribution with $\mu_j = 1$ and $\sigma_j = 1$, where $j = 1, 2, 3$. The simulation was run independently 100 times and in each run 300 trials were played. Afterwards, the results were averaged. Algorithm 2 was used to learn the actions by updating the value function according to the Q-learning update rule ($\alpha = 0.8$, $\gamma = 0.4$), whereas the action selection was done by an ϵ -greedy strategy ($\epsilon = 0.05$).

4.3. Results

The following results illustrate the principal feasibility of modeling preferences within the classical RL framework by exploiting reward shaping. In order to show the basic idea of

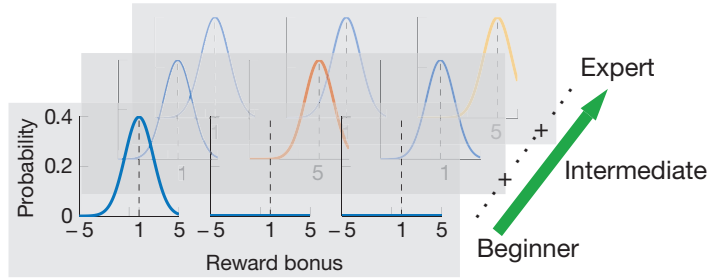


Figure 4.3.: The arising reward structure of a simulation with a preference model weighting the reward process of a bandit problem with Gaussian reward distributions. Each weighting function of the preference model is applied in relation to an additional external experience state. In the depicted structure, the current reward preference in the corresponding level is highlighted (orange/yellow).

this extension of the reward process within the RL framework, the weighting functions are concisely defined by

$$w^{ext,1}(S_t^A) = 1, \text{ and} \quad (4.1)$$

$$w^{ext,2/3}(S_t^A) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(S_t^A - \nu_{2/3})^2}{2\sigma^2}}, \quad (4.2)$$

where $\sigma = \frac{1}{5\sqrt{2\pi}}$ sets the maximum weight. Respectively, $\nu_2 = 0.5$ and $\nu_3 = 1$, is the value of S_t^A where the maximum is reached. With these weighting functions the reward structure of Figure 4.3 results. At the first level (beginner, $0 \leq S_t^A \leq 0.35$) the weightings for the second and third reward component is zero, while the first component is weighted by one (cf. Figure 4.2). With an intermediate level of experience ($0.35 < S_t^A \leq 0.65$), the

4. Affective Control

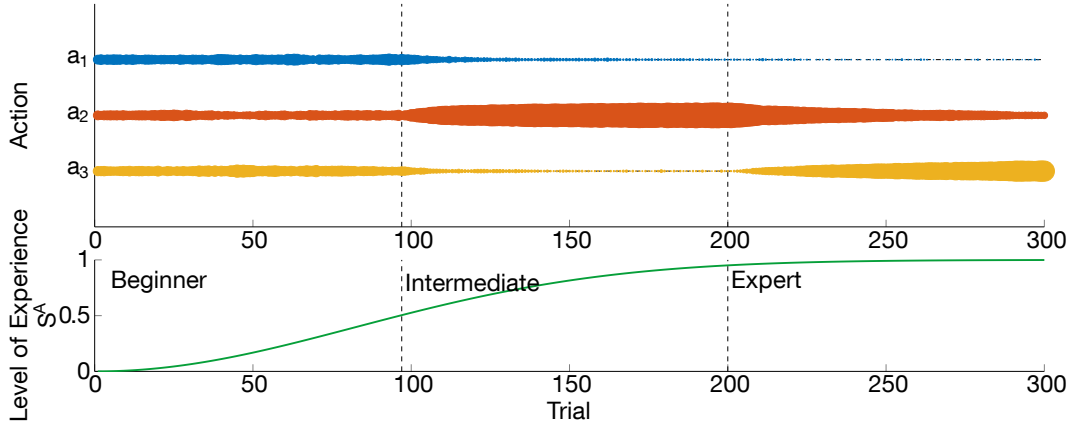


Figure 4.4.: Learned policy for a simulated increasing level of experience. First, the agent acts like a beginner and selects every action equally (the thickness of the bars corresponds to the frequency of selection). At trial 98, the agent enters the intermediate state and can perceive an additional reward model with a preference for the second action (a_2). In the expert state, another reward model is added with a preference for action three (a_3).

second reward component is preferred and weighted highest by $w^{ext,2}(S_t^A)$. In the expert level ($S_t^A > 0.65$), the second reward component gets less important, accompanied by an increasing preference for the third reward component. The experience level is simulated by

$$S_t^A = 1 - \frac{1}{\left(1 + \frac{t}{(0.5 \cdot n_{trials})}\right)^{1+e^1}}, \quad (4.3)$$

where t is the sampled step in time, and n_{trials} is the overall number of considered time steps. The resulting experience function is depicted in the lower part of Figure 4.4.

Learning on this reward structure using the given simulated experience signal results in preferences for actions correlated with the maxima of the weighting functions and the current value of S_t^A . In case of the chosen bandit scenario, each action directly corresponds to the respective reward component. In the upper part of Figure 4.4, the statistic of the selected actions in relation to the experience signal (lower part) is plotted. At the beginning (until trial 100), the agent is not able to differentiate the decisions and each action a_j is equally taken. After gaining some experience, in the intermediate phase, the second reward component gets more and more weighted, and thus is perceived more intensive by the agent. Now, the agent receives the first reward component of the bandit process and an additional reward bonus (the second reward component) which is added to the actual reward. As the weighting for the second reward component is remarkably higher than for the first component, the agent prefers the action (a_2) which maximizes the second reward component. In Figure 4.4, this clear preference is depicted by an increased linewidth for action a_2 , which represents the high selection frequency of this action.

Finally, in the expert level (trial 200 to 300) the weighting for the second reward component decreases again and the weighting function for the third component gets more intense. As the weighting for the first component is comparably low, the agent prefers only actions resulting in rewards for the second and third component. Interestingly, the preference for the second reward component does not vanish in direct correlation to the weighting function, as the RL process introduces a *forgetting phase* due to the back propagation of the outcomes. This results in a smooth and natural transition between one preference and another.

Given these results, the experiment shows that reward shaping in combination with an additional state representing experiences or affective states can be used to steer a classical Reinforcement Learning process. Obviously, the desired preference depends on the design of the control state and on the weighting functions. Since the goal of this chapter was to introduce a method for affectively controlling an RL based agent by an independent state, the considerations and results fulfill these requirements. The separation between control signals and decision making is preserved in the above algorithm, and thus is in accordance with the findings of neurophysiology.

The presented idea and the results were already published at a conference on Reinforcement Learning and Decision Making (Feldmaier et al., 2015) and discussed there controversially.

Concluding remarks on affectively shaping multiple rewards: *The difference between the above framework and approaches for multi-dimensional or dynamic reward environments surfaces in scenarios, where the reward process is initially unknown and the agent first has to learn how the process works. That means, instead of considering each reward component right from the beginning, the agent weights the different components according to an internal state. This enables to model a behavior that fulfills a primary goal first by exploiting the corresponding reward component and then advances by integrating additional reward components in relation to the control state. The neurophysiologically inspired and intuitive way of designing the required preference model is the advantage of the introduced method. Obviously, further research on the properties of the weighting function and the definition of the experience signal has to be done to complete the framework. More human biases like framing effects have to be considered and modeled by corresponding weighting functions and control signals. A major drawback of the described method is the lack in controlling the order of selected actions to achieve a preferred and optimal combination of reward components. Such a strict action order can be introduced by an additional action planning step which constrains the available actions in relation to the current state of the environment. Overall, the study has shown that the combination of existing Reinforcement Learning mechanisms and their interpretation in a psychologically informed way results in new model considerations. Those considerations are important in view of future developments of affective systems.*

5. Human Value Retrieval

In this chapter, a framework for combining inverse and multi-objective Reinforcement Learning is presented. The framework enables to learn linear scalarization weights for multiple reward signals. In multi-objective Reinforcement Learning scenarios, the weighting of the different objectives is crucial. Generally, the weights are manually assigned by the programmer which can introduce unintended biases. Inverse Reinforcement Learning was originally developed for approximating unknown reward functions and is now applied to learn hidden scalarization weights for given (expert) policies. The resulting weights can be used for comparison and verification of hand-crafted policies. In this way, also unwanted human biases inherently contained in hand-crafted policies can be detected. At the beginning of this chapter, the algorithm is derived and the retrieval of scalarization weights is described. Finally, results from simulations in a Gridworld scenario are presented.

5.1. Inverse Reinforcement Learning for Human Value Retrieval

In a setting where the reward function is not known, but an expert policy or several expert interactions with the system or the particular environment are available, so called Inverse Reinforcement Learning can be applied. A distinctive feature of this approach, compared to the manual design of a reward function, is that implicit human and animal learning strategies can automatically be detected and encoded. Additionally, Inverse RL is used for preference elicitation which tries to obtain a posterior distribution on the agent's preferences from observations (Rothkopf and Dimitrakakis, 2011). Numerous Inverse RL algorithms exist (Ng and Russell, 2000; Abbeel and Ng, 2004; Ramachandran and Amir, 2007) and they have been proven to be robust in deriving underlying reward structures and preferences. Therefore, it is a consequent step to extend the algorithm on multi-objective scenarios, where missing weights for the scalarization have to be determined. In addition to the ability to detect unwanted human biases of hand-crafted policies, the approach supports the creation of preference models as required by the algorithm of the previous chapter.

A prerequisite of the following algorithm is that there exists a sequence of sample policies, or a complete policy which is a result of successful interactions with the considered environment, and that this policy contains certain preferences. This means that the required policies should naturally express preferences of human or animal subjects for some of the components of the vectorial reward. The final reward used for selecting a decision and for updating value functions has a functional form related to the reward components.

5. Human Value Retrieval

It is also supposed that a policy to be analyzed contains the preferences a subject pursues if it behaves almost optimal according to its preferences.

There exist different approaches for Inverse RL which are distinguished according to the availability of training data. If the complete stationary policy for all states is given, the reward function can either be learned in a tabular form (in finite state spaces) or approximated as a linear function in the infinite state space. Hereby, a set of defined features $\phi(s) \in \mathbb{R}^q$ for each state is required. In contrast, if the policy is only partially observable or can only be sampled, the unknown reward function must be approximated. The latter case is assumed in this dissertation, as only a limited set of expert (human) policies is given, so that a linear function approximation is used in the following. Instead of observing different features for states and approximating the complete missing reward function, the existing multidimensional reward function of the environment is used together with the given policies to learn the weighting of the individual reward components. Hence, scalarization weights can be learned from observations and used for parametrization of learning agents without additional manual tuning.

In the following subsection, first the Inverse RL algorithm is introduced by rewriting the work of Ng and Russell (2000) using a more concise notation, while delivering a full and comprehensible derivation of the approach. Critical steps are explicitly described which are highly abstracted in the original and most other papers. Then the algorithm is extended to multi-objective scenarios and the ability to learn scalarization weights is illustrated.

5.1.1. Inverse Reinforcement Learning

Before formulating the Inverse RL algorithm for linearized reward functions, a simpler, discrete and finite state space version is derived. In the notation of the following algorithm, the i -th component of a vector \mathbf{x} will be denoted as $x^{(i)}$, and the individual (scalar) rewards for each state $R(s)$ are combined into a vector $\hat{\mathbf{R}} \in \mathbb{R}^N$, where N is the number of states. The goal of the Inverse RL is exactly to determine this unknown reward vector $\hat{\mathbf{R}}$. With this reward vector, the value function is defined as

$$\mathbf{V}^{\pi^*}(s) = \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k \hat{\mathbf{R}}(s_{t+k+1}) \mid \pi^*, s_t = s \right], \quad (5.1)$$

where the optimal policy is given by $\pi^*(s) = a^*$. The optimal policy states the optimal action a^* in state s . Using the above reward vector and the state transition probability matrix $\mathbf{P}_{ss'}^{a^*} \in \mathbb{R}^{N \times N}$, the well known Bellman equation is rewritten in vector form as

$$\mathbf{V}^{\pi^*} = \hat{\mathbf{R}} + \gamma \mathbf{P}_{ss'}^{\pi^*} \mathbf{V}^{\pi^*}. \quad (5.2)$$

Straightforwardly, this can be rearranged to

$$(\mathbf{I} - \gamma \mathbf{P}_{ss'}^{\pi^*}) \mathbf{V}^{\pi^*} = \hat{\mathbf{R}}, \quad (5.3)$$

5.1. Inverse Reinforcement Learning for Human Value Retrieval

where the matrix $(\mathbf{I} - \gamma \mathbf{P}_{ss'}^{\pi^*})$ is nonsingular. Hence, Equation (5.2) can be stated as

$$\mathbf{V}^{\pi^*} = (\mathbf{I} - \gamma \mathbf{P}_{ss'}^{\pi^*})^{-1} \hat{\mathbf{R}}. \quad (5.4)$$

Additionally, a so-called action-value function can be defined as

$$\mathbf{Q}^{\pi^*}(s, a) = \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k \mathbf{R}(s_{t+k+1}) \mid \pi^*, s_t = s, a_t = a \right], \quad (5.5)$$

which gives the expected future discounted reward starting in state s and taking action a , but following policy π^* afterwards. Two important relations between the policy and the action-value function as well as between the value function and the action-value function can be stated. The first is that the optimal policy can be derived from the action-value function by greedily selecting the next action due to the Bellman optimality condition, which is equivalent to

$$a^* = \pi^*(s) \in \underset{a \in \mathcal{A}(s)}{\operatorname{argmax}} \mathbf{Q}^{\pi^*}(s, a) \quad \forall s \in \mathcal{S}, \quad (5.6)$$

and the second relation is the transformation between the two value functions which is given by

$$\mathbf{Q}^{\pi^*}(s, a) = \mathbf{R}(s) + \gamma \sum_{s'} \mathbf{P}_s^a(s') \mathbf{V}^{\pi^*}(s'), \quad (5.7)$$

where $\mathbf{P}_s^a(s') \in [0, 1]$ is the probability of transition to state s' starting in state s and taking action a .

Now, instead of Equation (5.6) the equivalent condition for optimality

$$\sum_{s'} \mathbf{P}_s^{a^*}(s') \mathbf{V}^{\pi^*}(s') \geq \sum_{s'} \mathbf{P}_s^a(s') \mathbf{V}^{\pi^*}(s'), \quad \forall s \in \mathcal{S}, a \in \mathcal{A}, \quad (5.8)$$

can be compactly rewritten, as

$$\mathbf{P}_{ss'}^{a^*} \mathbf{V}^{\pi^*} \succeq \mathbf{P}_{ss'}^a \mathbf{V}^{\pi^*} \quad \forall a \in \mathcal{A} \setminus a^*, \quad (5.9)$$

where \succeq denotes entry-wise inequality of vectors, i.e. $x \succeq y \Leftrightarrow x^{(i)} \geq y^{(i)} \forall i$. By recalling the fact that $\mathbf{V}^{\pi^*} = (\mathbf{I} - \gamma \mathbf{P}_{ss'}^{\pi^*})^{-1} \hat{\mathbf{R}}$, Equation (5.2) can be stated as

$$\mathbf{P}_{ss'}^{a^*} (\mathbf{I} - \gamma \mathbf{P}_{ss'}^{\pi^*})^{-1} \hat{\mathbf{R}} \succeq \mathbf{P}_{ss'}^a (\mathbf{I} - \gamma \mathbf{P}_{ss'}^{\pi^*})^{-1} \hat{\mathbf{R}}, \quad \forall a \in \mathcal{A} \setminus a^*, \quad (5.10)$$

which is equivalent to

$$(\mathbf{P}_{ss'}^{a^*} - \mathbf{P}_{ss'}^a) (\mathbf{I} - \gamma \mathbf{P}_{ss'}^{\pi^*})^{-1} \hat{\mathbf{R}} \succeq 0. \quad (5.11)$$

The problem now lies therein, that the solution to Equation (5.11) can be trivial, i.e. $\hat{\mathbf{R}} = \mathbf{0}$. Additionally, there exists a multitude of valid solutions, from which the right one has to be chosen. To avoid the trivial solution and to get a unique result, additional penalty terms are introduced.

5. Human Value Retrieval

The demand that π should be optimal comes with a second imperative, namely that this policy should be as much better as possible than the second best policy. This imperative can be formulated as a maximization of

$$\sum_{s \in \mathcal{S}} \left(\mathbf{Q}^{\pi^*}(s, a^*) - \max_{a \in \mathcal{A} \setminus a^*} \mathbf{Q}^{\pi^*}(s, a) \right). \quad (5.12)$$

Additionally, the *simplest* solution should be preferred. This means, the reward function, that has as little nonzero entries as possible is preferable. Such behavior is achieved with an additional ℓ_1 -penalty. The ℓ_1 - and ℓ_2 -norm are commonly used as regularizers in machine learning, especially when the difference between zero and non-zero elements is very important. Their values increase every time an element of the considered vector moves away from 0.

The combined optimization problem is therefore formulated as

$$\begin{aligned} & \underset{\hat{\mathbf{R}}}{\text{maximize}} \sum_{i=1}^N \min_{a \in \mathcal{A} \setminus a^*} \left\{ (\mathbf{P}_{ss'}^{a^*}(i) - \mathbf{P}_{ss'}^a(i)) (\mathbf{I} - \gamma \mathbf{P}_{ss'}^{a^*} \hat{\mathbf{R}})^{-1} \right\} - \lambda \|\hat{\mathbf{R}}\|_1 \\ & \text{s.t. } (\mathbf{P}_{ss'}^{a^*} - \mathbf{P}_{ss'}^a) (\mathbf{I} - \gamma \mathbf{P}_{ss'}^{a^*})^{-1} \hat{\mathbf{R}} \succeq 0 \quad \forall a \in \mathcal{A}(s) \setminus a^* \\ & \mathbf{R}^{(i)} \leq \mathbf{R}_{\max}, i = 1, \dots, N, \end{aligned} \quad (5.13)$$

where $\mathbf{P}_{ss'}^a(i)$ denotes the i -th row of the $\mathbf{P}_{ss'}^a$ matrix. This means $\mathbf{P}_{ss'}^{a^*}(i) - \mathbf{P}_{ss'}^a(i)$ quantifies the same as written down in Equation (5.12).

In case of a non-deterministic system with a finite set of states, which can be stored in a tabular way, function approximation can be employed to approximate the value function. Function approximation is popular for describing the value functions itself, especially the linear version, where each state is transformed by a feature transform $\phi : \mathcal{S} \rightarrow \mathbb{R}^l$. The one-dimensional value function itself is then approximated via

$$\tilde{V}^\pi(s) \approx \boldsymbol{\theta}^\top \boldsymbol{\phi}(s), \quad (5.14)$$

where $\boldsymbol{\theta}$ denotes a parameter vector. This parameter vector $\boldsymbol{\theta}$ is iteratively learned by the approximation algorithm.

Keeping this scheme of approximating the value function in mind, the reward function for each state can also be expressed as a mapping $R : \mathcal{S} \rightarrow \mathbb{R}$, and therefore be approximated in a similar way by

$$\tilde{R}(s) = \boldsymbol{\alpha}^\top \boldsymbol{\phi}(s), \quad (5.15)$$

with the parameter vector $\boldsymbol{\alpha} \in \mathbb{R}^l$. In the following, $\phi^{(i)}(s)$ denotes the i -th component of the feature vector calculated for state s .

By the definition of the value function, and the fact that the approximation of the reward function \tilde{R} is also a linear combination of features, the value function approximation can be stated as

$$\tilde{V}^\pi(s) = \mathbb{E}_{s' \sim P_{ss'}^\pi} \left[\sum_{k=0}^{\infty} \gamma^k \sum_{i=0}^l \boldsymbol{\alpha}^{(i)} \phi^{(i)}(s_{t+k+1}) | s_t = s \right]. \quad (5.16)$$

5.1. Inverse Reinforcement Learning for Human Value Retrieval

Furthermore, by defining the specific value function V_i^π , which would arise if the reward was approximated by using only one feature as $\mathbf{R}^{(i)}(s) = \alpha^{(i)} \phi^{(i)}(s)$, a single dimension of the value function approximation can be derived by

$$\begin{aligned} \tilde{V}_i^\pi(s) &= \mathbb{E}_{s' \sim P_{ss'}^\pi(s)} \left[\sum_{k=0}^{\infty} \gamma^k \mathbf{R}^{(i)}(s_{t+k+1}) | s_t = s \right] \\ &= \mathbb{E}_{s' \sim P_{ss'}^\pi(s)} \left[\sum_{k=0}^{\infty} \gamma^k \alpha^{(i)} \phi^{(i)}(s_{t+k+1}) | s_t = s \right]. \end{aligned} \quad (5.17)$$

Rearranging Equation (5.16) and using Equation (5.17) together with the linearity of the expectation gives

$$\begin{aligned} \tilde{V}^\pi(s) &= \mathbb{E}_{s' \sim P_{ss'}^\pi(s)} \left[\sum_{k=0}^{\infty} \gamma^k \sum_{i=0}^l \alpha^{(i)} \phi^{(i)}(s_{t+k+1}) | s_t = s \right] \\ &= \sum_{i=0}^l \alpha^{(i)} \mathbb{E}_{s' \sim P_{ss'}^\pi(s)} \left[\sum_{k=0}^{\infty} \gamma^k \phi^{(i)}(s_{t+k+1}) | s_t = s \right] \\ &= \sum_{i=0}^l \alpha^{(i)} V_i^\pi(s), \end{aligned} \quad (5.18)$$

where again $\alpha^{(i)}$ represents the i -th component of a parameter vector. By using Equation (5.9) the following inequality for all i can be stated:

$$P_{ss'}^{a^*} V_i^\pi \succeq P_{ss'}^a V_i^\pi, \quad \forall a \in \mathcal{A}(s) \setminus a^*. \quad (5.19)$$

Then it is straightforward to replace the value function by its component-wise notation and rewrite the inequality as

$$\sum_{i=0}^l \alpha^{(i)} P_{ss'}^{a^*} V_i^\pi \succeq \sum_{i=0}^l \alpha^{(i)} P_{ss'}^a V_i^\pi, \quad \forall a \in \mathcal{A}(s) \setminus a^*, \quad (5.20)$$

and finally by following Equation (5.18), it can be concluded that

$$P_{ss'}^{a^*} \tilde{V}^\pi(s') \succeq P_{ss'}^a \tilde{V}^\pi(s'), \quad \forall s \in \mathcal{S}, a \in \mathcal{A}(s) \setminus a^*, \quad (5.21)$$

which is in parallel with the result of Equation (5.9), or entry-wise

$$\mathbb{E}_{s' \sim P_{ss'}^{a^*}} \left[\tilde{V}^\pi(s') \right] \geq \mathbb{E}_{s' \sim P_{ss'}^a} \left[\tilde{V}^\pi(s') \right], \quad \forall s \in \mathcal{S}, a \in \mathcal{A}(s) \setminus a^*. \quad (5.22)$$

Commonly, problems where linear function approximation is used have an infinitely sized state space, and thus an infinite amount of constraints of the type of Equation (5.22) would

5. Human Value Retrieval

have to be satisfied. This problem can be tackled by only evaluating the constraints on a large sampled subset of the states.

Also, since the true reward function might not exactly be representable by linear approximation, the constraints have to be relaxed. The overall constraint set can therefore be written down as

$$\begin{aligned} & \text{maximize } \alpha \sum_{s \in \hat{\mathcal{S}}} \min_{a \in \mathcal{A}(s) \setminus a^*} \left\{ p \left(\mathbb{E}_{s' \sim \mathbf{P}_{ss'}^{a^*}} [\tilde{\mathbf{V}}^{\pi^*}(s')] - \mathbb{E}_{s' \sim \mathbf{P}_{ss'}^a} [\tilde{\mathbf{V}}^{\pi^*}(s')] \right) \right\} \\ & \text{s.t. } |\alpha^{(i)}| \leq 1, i = 1, \dots, l, \end{aligned} \quad (5.23)$$

where $\hat{\mathcal{S}}$ is the set of the sampled states and p is an additional penalty function to enforce the required inequality constraints of Equation (5.22). In this case p is given by

$$p(x) = \begin{cases} x, & x \geq 0; \\ mx, & \text{otherwise.} \end{cases} \quad (5.24)$$

The weight is set to $m = 2$, as experiments by Ng and Russell (2000) have shown that the algorithms, starting from a value of 2, are insensitive to this parameter.

5.1.2. Adaptation to Scalarized Multi-objective Reinforcement Learning

In the multi-objective Reinforcement Learning setting, there are different ways of finding the ideal trade-off between the different reward components (cf. Section 2.3.2). However, scalarization of reward functions is mostly used. Therefore, the derivation of the following Inverse RL algorithm is restricted to this type of reward function.

Recalling that α contains the parameters for the scalarization function f and \mathbf{V}^π is the multi-objective value function containing the values for each objective (usually in vector form), V_α^π denotes the scalarized multi-objective value function of the problem. If certain restrictions apply, like in the example of Roijers et al. (2013), the model can even be further reduced. By additionally assuming a linearly scalarized multi-objective value function, the value function is finally written as

$$\begin{aligned} V_\alpha^\pi &= \alpha^\top \mathbf{V}^\pi = \alpha^\top \cdot \mathbb{E}_{s \sim P_{ss'}^\pi} \left[\sum_{k=0}^{\infty} \gamma^k (\mathbf{R}(s_{t+k+1})) \right] = \mathbb{E}_{s \sim P_{ss'}^\pi} \left[\sum_{k=0}^{\infty} \gamma^k (\alpha^\top \mathbf{R}(s_{t+k+1})) \right] \\ &= \mathbb{E}_{s \sim P_{ss'}^\pi} \left[\sum_{k=0}^{\infty} \gamma^k \tilde{R}(s_{t+k+1}) \right] = \mathbb{E}_{s \sim P_{ss'}^\pi} \left[\alpha^\top V_i^\pi \right], \end{aligned} \quad (5.25)$$

where \tilde{R} denotes the approximated reward and in this case $\alpha \in \mathbb{R}^q$ is an appropriate weight vector. As this scalarization is compatible with the above formulation, the linear version of Inverse Reinforcement Learning can directly be applied to determine the weights

of the linear scalarization in the multi-objective scenario by substituting the features with the individual dimensions of the multi-objective reward \mathbf{R} .

To solve the problem formulated in Equation (5.23), an off-the-shelf linear programming package cannot directly be used, as it is not yet formulated in its corresponding block matrix form. The detailed derivation of the block matrix form is stated in Appendix A.3. Using the resulting block matrix form, the weights can be learned with the algorithm depicted in Algorithm 3. In the following experiment, this algorithm is used and solved by a suitable solver (in this dissertation the *cvxopt* Python package¹ is used).

Algorithm 3 Inverse Reinforcement Learning for multi-objective RL

- 1: **Result:** Weight vector α
- 2: **Initialization:** Optimal policy $\pi(s) = a^* \quad \forall s \in \mathcal{S}$
- 3: Select a set $\hat{\mathcal{S}}$ of states from the environment.

- 4: Solve the linear program:

$$\begin{aligned} & \text{maximize } \sum_{s \in \hat{\mathcal{S}}} \min_{a \in \mathcal{A}(s) \setminus a^*} \left\{ \rho \left(\mathbb{E}_{s' \sim P_{ss'}^{a^*}} \left[\sum_{i=0}^q \alpha^{(i)} V_i^{\pi^*}(s') \right] - \mathbb{E}_{s' \sim P_{ss'}^a} \left[\sum_{i=0}^q \alpha^{(i)} V_i^{\pi^*}(s') \right] \right) \right\} \\ & \text{s.t. } |\alpha^{(i)}| \leq 1, i = 1, \dots \end{aligned}$$

5.2. Simulations and Test Cases

In the following simulations, the extraction of scalarization weights from given policies using the presented Inverse RL algorithm (Algorithm 3) is shown. For retrieving the scalarization weights, the considered policy should be consistent and the objective should be reachable in a replicable manner. These requirements are mandatory to ensure the convergence of the algorithm. In this context, consistent means that the same action in neighboring or similar states leads to similar subsequent states. For example in a Gridworld style experiment, this means that the optimal action of a policy in neighboring states moves the agent into the same or at least not into the opposite direction in the maze (see Figure 5.1). Therefore, retrieving scalarization weights from random or ambiguous policies is not possible and must be preempted beforehand by the user. The necessary policy that is entered into the linear program thus consists of contradiction free transition probabilities and the corresponding rewards of the individual objectives for each state.

In the following, two experiments are conducted. The first one considers manually designed policies for the Gridworld environment which directly reach the three different goals. In the second experiment, a Q-learning agent with linear scalarization is used to learn a policy. The learned policy is characterized by the weights defined by the programmer

¹<http://www.cvxopt.org/> [Accessed 18th January 2017]

5. Human Value Retrieval

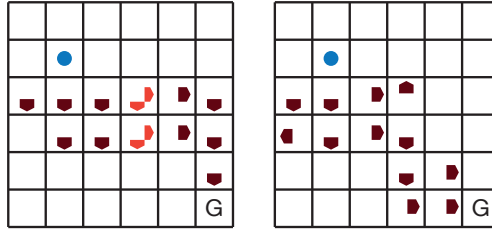


Figure 5.1.: Example of a consistent (left side) and an inconsistent (right side) policy in a two-dimensional Gridworld scenario. In case of an inconsistent policy, the optimal actions (small arrows) of neighboring fields differ substantially and point for instance into the opposite directions.

beforehand. With this learned policy and the same environment, the Inverse Reinforcement Learning algorithm is used to retrieve the programmer’s scalarization weights. Then, these retrieved weights in turn are reused to learn a new policy in the same environment. As a result, there exist two policies for each run of the experiments. In case of the first experiment, there are the predefined manual policy and a learned policy. In the second experiment, there are two learned policies of a Q-learning agent and additionally a set of scalarization weights. In the result section, the policies and scalarization weights are compared to evaluate the performance of the approach. The next section describes the Gridworld environment and the considered test cases.

5.2.1. Environment Description

As a benchmark environment, a multi-objective version of the already known Gridworld (cf. Section 3.2) is used. The agent starts out in the top left corner in state S of a 10×10 grid and is able to move according to the four cardinal directions (left, right, up, and down). As can be seen in Figure 5.2, there exist three different reward points $R^{(1)}$ to $R^{(3)}$ in the grid, and each of these points reflects one reward dimension. That means the vectorial reward $\mathbf{R} = [R^{(1)}, R^{(2)}, R^{(3)}, R^{(4)}]^T$ will contain a 1 in the position that corresponds to the goal that is reached. For example, if the agent reaches the terminal state ①, the reward components will contain $R^{(1)} = 1$ and $R^{(2)} = R^{(3)} = 0$. There is also a fourth reward dimension corresponding to a time penalty, which will always be set to $R^{(4)} = -1$ for each transition in Experiment 1 and to $R^{(4)} = -0.1$ for each transition in Experiment 2.

The three reward points ① to ③, can be interpreted as three different objectives in the environment between which the agent has to make a tradeoff. The fourth reward dimension can be interpreted to be in terms of *time*. Hence, if a large weight is put on the fourth dimension, the agent should prefer shorter paths in the environment taking less steps.

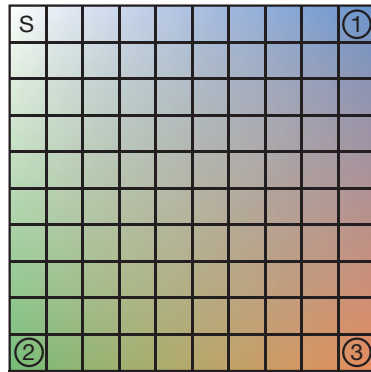


Figure 5.2.: Multi-objective Gridworld with three different goal states (1, 2 and 3) and the start position S . The color gradient depicts the influence of each goal in a Q-learning scenario with equal scalarization weights.

5.2.2. Test Cases

The first three deterministic test cases are characterized by the following three hand-crafted policies:

- D1** Always try to reach objective ①. This is done by moving to the right, when the agent is above the diagonal from the lower left to the upper right, and going up, when below the diagonal.
- D2** Always try to reach objective ② by doing the opposite of policy D1: going left below the diagonal and going down above the diagonal.
- D3** Try to reach the objective ③ on the shortest path. The shortest path from any point in the grid can again be characterized by a diagonal. This time the one from the upper left to the lower right. Above the diagonal move downwards and below the diagonal move to the right.

These manually designed policies are depicted in the upper row of Figure 5.3. Each red arrow indicates the direction that should be chosen in the corresponding state. It is noteworthy that by specifying the policies like this, the goal to reach a specific objective terminal state is explicitly given, however there is no explicit specification concerning the fourth dimension, which constrains time.

For the second experiment, the policies are not explicitly specified, but the scalarization weights α are set to specific values in a way how a typical designer (or programmer) would preset them to achieve a certain goal. The test cases are:

- L1** The weights in this case are set to $\alpha = [10, 0, 0, 1]^T$. This should encourage the agent to learn a policy to reach objective terminal state ①, while taking the least steps possible, because of a positive weight on the fourth reward dimension.

5. Human Value Retrieval

- L2** The second case is similar to the first one, namely $\alpha = [0, 10, 0, 1]^\top$, just that the agent should reach the second objective state.
- L3** The third one is again analogous to the two first ones with $\alpha = [0, 0, 20, 1]^\top$. The difference in the third weight is motivated by the fact that the positive reward of 20 in the third objective state should compensate for the longer path that has to be taken to reach this state.

In the next section, the results of the two conducted experiments using the policies specified above are described.

5.3. Results

The learned and designed policies are plotted in Figure 5.3 and Figure 5.4, respectively, and have to be interpreted in the following way: The shading of each square depicts how often the state has been visited during the learning process. The more red the shading, the more often this state was visited (similar to a heat map). The small arrows printed over indicate the relative probability of choosing one of the four cardinal actions. A redder shade and bigger arrow indicates that this action will be chosen more likely in this state, as the size and color depend on the action selection probabilities given by the Q-values.

In the top row of Figure 5.3 the hand-crafted policies **D1-D3** are depicted. By using these hand-crafted policies, the Inverse Reinforcement Learning algorithm retrieved the following weights:

$$\begin{aligned}\alpha_{D1} &= [1.0000, 0.0000, 0.0000, 0.9847]^\top, \\ \alpha_{D2} &= [0.0000, 1.0000, 0.0000, 0.9847]^\top, \text{ and} \\ \alpha_{D3} &= [0.0000, 0.0000, 1.0000, 0.0000]^\top.\end{aligned}$$

The learned weights for the three policies support the hypothesis, that the scalarization weights can be retrieved from given policies. As previously mentioned, the hand-crafted policies make only an implicit assumption about the fourth reward component – the time. In case of the first two policies **D1** and **D2**, the time component plays a role, as there exist sub-optimal paths between the start to the terminal state. By manually designing policies **D1** and **D2**, the human designer implicitly omits these sub-optimal and longer paths. The weight retrieval algorithm however takes account of this circumstance and assigns a weight of almost 1 to the fourth component. This forces the Q-learning algorithm to learn the shortest path. In case of policy **D3** an interesting detail of the approach is revealed, as the scalarization weight of the time component is zero. The reason for this counter-intuitive weighting is the so called *Manhattan* distance. In the experiment, only steps into the cardinal directions are possible, thus each path in a squared maze divided into an equally spaced grid has the same length. Therefore, it is not necessary to assign a weight to the time component, as all possible paths have equal length by definition. This

means in effect, that the proposed method for retrieving the scalarization weights reveals special properties of the environment and the learning algorithm.

The bottom row of Figure 5.3 depicts the policies which were learned by Q-learning with linear scalarization using the previously retrieved weights as stated above. For each learning experiment 200 000 episodes were conducted while the learning rate was set to $\beta = 0.6$, and the exploration parameter for the ϵ -greedy action selection was set to $\epsilon = 0.5$. The comparison of the resulting policies (bottom row) with the hand-crafted input policies (top row) shows some deviations. In case of policies **D1** and **D2** this behavior can be explained with the fact that in some cases, where the agent has collected an amount of negative reward in the fourth reward dimension, it is better to take the shorter path to the closest terminal state in order to end the episode, rather than to end the episode in the terminal state denoted by the actual policy. As this behavior is a special property of the Q-learning algorithm, it was not considered by the human-designer of the hand-crafted policies, and therefore is not represented in the input policies. The Q-learning algorithm always maximizes the expected utility, and as the reward in the intended objective state would not compensate for taking the longer path, the agent learns to terminate sub-optimal paths earlier instead of selecting a longer path to the desired goal. The difference between the hand-crafted and the learned policy in the third case (**D3**) can again be explained by the Manhattan distance. A human designer intuitively would require the algorithm to learn a policy resulting in a diagonal movement direction. However while keeping the characteristic of the Manhattan distance in mind, the paths learned in case **D3** are not longer with respect to reward dimension four than all the alternative paths in the original policy **D3**. To explicitly force such a behavior to be learned, the reward structure of these experiments is not expressive enough and can therefore not be retrieved by the Inverse RL algorithm.

In Figure 5.4 the results for the second experiment are depicted. In the top row, the policies learned by scalarized Q-learning are shown. The learning parameters were set to $\epsilon = 0.4$ for the ϵ -greedy action selection, and $\beta = 0.3$ for the learning rate. The algorithm was run for 100 000 episodes. It can be seen that the initially learned policies on the predefined weights already differ from the designer's intentions for cases **L1** and **L2**. Test case **L3** behaves as intended and a policy is learned that lets the agent move along the diagonal direction towards the lower right terminal state. The reason why **L1** and **L2** learn differently lies, once again, therein that in those cases a shorter path to a terminal state impacts less negatively on the total reward than an alternative longer path towards the desired terminal state. The longer path cannot be compensated by the relatively small positive reward of the desired terminal state. This behavior might be counter-intuitive to the human reward weight designer and therefore not intended. However, for case **L3** the behavior is clear, since the path to the objective state ③ is at the same time also the longest path.

5. Human Value Retrieval

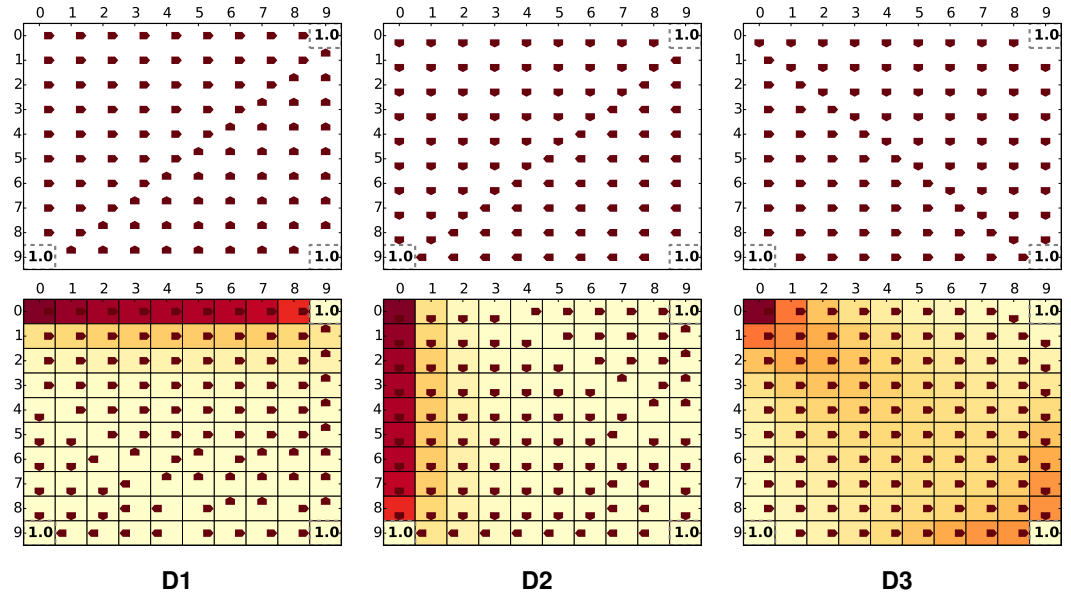


Figure 5.3.: Resulting policies of the first experiment. Top row: Hand-crafted policies D1-D3. Bottom row: Policies learned on retrieved weights.

When inspecting the retrieved weights,

$$\begin{aligned}\alpha_{L1} &= [1.0000, 0.9182, 0.4653, 1.0000]^T, \\ \alpha_{L2} &= [0.2516, 1.0000, 0.2473, 0.0736]^T, \text{ and} \\ \alpha_{L3} &= [0.4096, 0.0000, 1.0000, 0.0000]^T,\end{aligned}$$

which were calculated by averaging over the results of 10 independent runs, a relation to the original weights of the policies **L1** to **L3** can be observed. The intended objective of the original policy is represented by attaching a higher value to the corresponding reward component in the retrieved weight vector. Additionally, implicit trade-offs are included in the resulting scalarization vectors, which were chosen by the agent due to the problem and reward structure. Another interesting observation is that for case **L3**, the Inverse RL algorithm correctly discovers the fact that the number of steps does not play a big role in the path finding, as it is the longest path through the Gridworld anyway (again the characteristic of the Manhattan distance).

In general, it can be seen that the policies which were learned using the retrieved weights are similar to the original ones, but also differ in some ways (comparison between the upper and the bottom row of Figure 5.4). This reflects the hypothesis, that designed weights might be subject to human biases. In case of the first experiment (**D1** - **D3**), the hand-crafted policies result in mostly intuitively comprehensible combinations of scalarization weights. Using them in the subsequent step of the experiment, nevertheless results in different

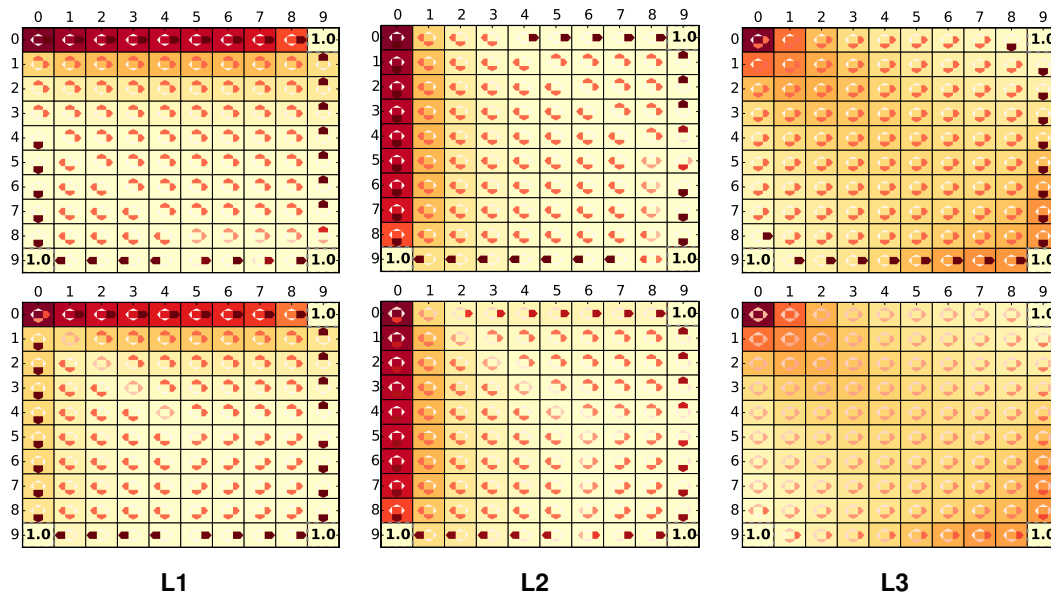


Figure 5.4.: Results of the second experiment: In the top row, the policies learned on predefined weights are depicted. Below, the policies are shown that have been learned from weights after retrieving them from the policies in the top row.

behaviors than intended, as the fact that the Q-learning agent maximizes expected utilities was not sufficiently considered by the human designer.

Overall, the described experiment and its results show that the developed inverse learning algorithm can be used to precisely detect discrepancies between the designer's intention and the interpretation of these intentions by a linear scalarization Reinforcement Learning approach. The formulated Inverse Reinforcement Learning framework can be applied to every multi-objective scenario if the individual reward functions for each objective and the transition probabilities can explicitly be formulated.

Concluding remarks on retrieving human biases from given policies: *In this chapter, an approach for determining scalarization weights out of given policies was motivated by describing the fact that a human designer introduces unintended biases. The results support the hypothesis that indeed such unintended biases are introduced by humans during manually designing policies or scalarization weights. In the performed experiments, intuitively designed policies or scalarization vectors do not meet the expectations, or result in differing policies accompanied by changed scalarization weights in the subsequent retrieval step. Although the presented experiment shows the results in a very simple and theoretical domain, the derived algorithm is theoretically applicable to all multi-objective Reinforcement Learning scenarios with linear reward scalarization. Currently, the precise formulations for the requirements in the shape of the expert policies and the prerequisites*

5. Human Value Retrieval

for the linear program are not fully stated. Additional theoretical analysis needs to be done to verify the promising results of this first study and to explicitly state the requirements on the non-contradictory policies (see Section 5.2). In sum, the framework is not only useful for designing linear scalarization functions but also for researchers to verify new models. For example, a psychologist can record human behaviors and input them as policies to the algorithm in order to calculate scalarization weights. Afterwards, an RL agent can be trained to reflect human behavior as close as possible, without manually designing the scalarization weights beforehand.

6. Conclusion

This chapter concludes this work. Section 6.1 reviews the research questions and briefly summarizes the answers provided in the various chapters of this work. Section 6.2 completes this chapter with an overview of open questions and avenues of future research.

6.1. Summary

In this dissertation, I have introduced three perspectives for connecting Reinforcement Learning with mechanisms that are inspired by human emotions and preferences. Without claiming general applicability, the presented algorithms are attempts to combine human values like artificial emotions and preferences with the classical and evidentially optimal Reinforcement Learning. The introduced models lower the pure performance of RL in achieving an optimal policy, however they simultaneously open new dimensions and perspectives in the advancement of Reinforcement Learning. The three key aspects which have been considered in this dissertation are:

- Artificial emotions as an intuitive and natural component of communication and their use to evaluate the learning progress of artificial agents.
- Reward shaping as an established method to modify learned policies with additional reward components and preferences.
- The application of Inverse RL to detect and analyse unwanted biases in given policies, and as a tool for determining scalarization weights.

The first two aspects improve the behavior of artificial agents towards human users, while the third point represents a tool for the designers of artificial agents.

In case of the presented artificial emotion models of Chapter 3, I have emphasized the relation of these models to existing psychological models and theories. Likewise, I have verified in a field test that the calculated artificial appraisals meet the expectations of human users.

The developed algorithms use appraisal derivation models to calculate meta-data of the underlying machine learning processes. These meta-data drive a subsequent appraisal model which bases on psychological findings calculating the final artificial emotions or feelings. The strict separation between appraisal derivation model and actual psychological appraisal model allows the adaption of the algorithms to new domains, without

6. Conclusion

touching the appraisal model and its logic itself. That means, in case of new domains only the appraisal derivation models are subjected to changes, as there are certain limitations concerning their generalisability. Primarily, the interfaces between the new learning algorithm and the appraisal derivation models have to be adjusted. This design step is supported and simplified by this dissertation as the variables which are calculated in the appraisal derivation models have been clearly motivated and defined. Moreover, to avoid a prevailing difficulty in the field of Affective Computing, the source code of all models and experiments of this dissertation are published as open source (see Appendix A.1). This should foster and speed up new projects, and is a contribution to the field of Affective Computing, as the source code of currently existing models is either kept under wraps (e.g. EMA¹, PEACTION², FAtiMA³), or is part of a more complex system architecture and cannot be simply reused standalone (e.g. ACT-R⁴).

Besides the fact that these artificial appraisals can be used to improve the way of communication between a system and a human user, they can additionally be used to steer decision making of the system itself. Therefore, in Chapter 4, I have researched possible methods for introducing preferences into Reinforcement Learning. The result bases on reward shaping, a technique to modify the reward function of scenarios with an additional potential function which is externally controlled by a shaping signal and adds a bias to the environmental reward. This external control signal and the interpretation of reward shaping as a method to integrate preferences into Reinforcement Learning is a new perspective in the Reinforcement Learning domain. In my experiment, the principle properties of the algorithm are demonstrated and a model for the affective control state is propounded.

Finally, in Chapter 5, the retrieval of scalarization weights out of given policies is considered, which is the remaining aspect of the three-sided approach in my investigation of connections between human emotions as well as biases and intelligent machines. The presented Inverse Reinforcement Learning algorithm is able to calculate scalarization weights out of previously created or recorded policies in order to analyze or compare the policies themselves on the basis of scalarization weights, or to compare the retrieved weights with those a human would intuitively set for certain problems. With this method, a kind of tool was developed to qualitatively compare scalarization weights of a multi-objective Reinforcement Learning process with those inherently contained in (human) policies. This partly solves the existing problem of artificial intelligence designers to assign certain weights to multiple objectives and it helps psychologists to analyze manually recorded human policies. The comparison between scalarization weights that were calculated, e.g. by Pareto optimal algorithms, and those retrieved by my Inverse RL algorithms helps to reveal human biases. Furthermore, the presented experiment and its results also

¹cf. Marsella and Gratch (2009)

²cf. Marinier et al. (2009)

³cf. Dias et al. (2014)

⁴cf. Anderson et al. (2004)

show that domain dependent properties (in the particular case the Manhattan distance) and its side effects on the weights can be revealed by analyzing the scalarization weights of hand-crafted policies.

I cannot render a final verdict on the benefit of the proposed extensions, as no generally applicable model of artificial emotions exists, and since the possible ways of interpreting the existing models for implementing them into artificial agents are nearly unlimited. However, the given examples demonstrate what can be done today, and simultaneously illustrate how young this field of research still is. Nevertheless, I have made some progress in combining the field of Reinforcement Learning and SLAM with components of Affective Computing. Extending Reinforcement Learning and a SLAM process with the ability to communicate the learning progress via a non-verbal channel increases the bandwidth of communication between artificial agents and humans. The control of Reinforcement Learning with an affective state extends the behavior of agents with preferences in order to imitate human decision making which in turn could increase the acceptance of intelligent systems in shared environments. Finally, to deeper understand policies created by RL agents, the Inverse RL approach delivers an important algorithm to analyze them.

6.2. Future perspectives

In the domain of Affective Computing, a lot of problems still need to be solved until a machine will be able to correctly interpret and process human emotions and additionally is able to react appropriately. By increasing the abilities of machines to imitate and understand humans, unanswered ethical and philosophical questions will come to the fore and need to be handled. From an engineering perspective, the combination of existing psychological models and machine learning algorithms is the basic groundwork to investigate those unanswered questions – today and in future.

The presented experiments of this dissertation represent a further step into this direction and show how psychology can be used to modify machine learning. However, larger human-in-the-loop studies are necessary to empirically verify the investigated algorithms. Additional scenarios in different domains should shed light on the general applicability of the introduced approaches. The implementation of the algorithms on physical hardware platforms is definitively the next consequent step. A small two-wheeled robot with an appropriate display, as depicted in Figure 6.1, has already been built and preliminary tests with human subjects were performed. This end-to-end experiment is intended to verify the results of the experiment in Chapter 3. End-to-end means that all effects, like the robot type, its movements, and the experimental setup are taken into consideration. The goal of this experiment is to show that the simulated emotions can be expressed with a low-resolution display, while still being recognized by humans. Using such a low-resolution display capable of visualizing dynamic color patterns, enables a broad area of application

6. Conclusion

in various domains. The displayed emotions should be recognized by children as well as by elderly people, and they should be widely cultural independent. Currently, the design of the dynamic light patterns makes progress, however the transition between two patterns and different intensity levels are still open research questions.



Figure 6.1.: A small two-wheeled robot which was developed at the Chair for Data Processing. The low-resolution led display on top is able to display basic emotions with dynamic light patterns (photographed by Martin Knopp).

Another open task is the application of the proposed reward shaping algorithm, as stated in Chapter 4, to solve the still open issue of finding the optimal trade-off between exploration and exploitation. In humans, the drive of exploration is strongly related to their mood, and the exploitation of situations is often controlled by short-term positive feelings. Similar concepts are modeled in state-of-the-art machine learning algorithms, but a central control of different learning and planning progresses using an affective signal within an artificial agent is still missing. The approach of Chapter 4 provides the basic algorithm for steering Reinforcement Learning with a centrally generated and affective state.

Lastly, as already mentioned, the algorithm of Chapter 5 gives the opportunity to analyze RL policies. Up to now, the focus in machine learning lies in finding and tweaking the optimal policy, but the essential behavior which is generated by an artificial agent and the learned policy is often not reflected in a greater context. It is well known that a classical RL agent maximizes the expected utility. A similar behavior was imputed to humans until Prospect Theory was discovered, which introduces prospects and needs besides pure maximization. More recent psychological findings add further dimensions to the behavior of humans, explaining reasons for sub-optimal decisions typical for humans. The analysis of policies, as presented in Chapter 5, is also the first step in order to understand policies generated by machine learning algorithms in greater depth, instead of over-optimizing them. Over-optimization is a main reason for lowering the performance of an algorithm in two ways: First, the generalization is bad, and secondly humans might get mistrustful and anxious in case of nearly perfect acting machines (see also *The uncanny valley* by Mori et al. (2012)).

6.2. Future perspectives

I became convinced you couldn't build a truly intelligent computer without having emotional capabilities like humans do.

Rosalind Picard, 2016

List of Acronyms

AI	Artificial Intelligence	49
ALMA	A Layered Model of Affect	25
BDI	Belief, Desire and Intentions	32
CPM	Component Process Model	18
DP	Dynamic Programming	38
ECA	Embodied Conversational Agent	20
EEC	emotion eliciting condition	26
EKF	Extended Kalman Filter	71
FLAME	Fuzzy Logic Adaptive Model of Emotions	25
MAB	Multi-Armed Bandit	56
MC	Monte Carlo	37
MDP	Markov Decision Process	35
OCC	Ortony, Clore, Collins	24
PAD	Pleasure, Arousal, and Dominance	22
PEACTIDM	Perceive, Encode, Attend, Comprehend, Tasking, Intend, Decode, and Motor 23	
PT	Prospect Theory	50
RL	Reinforcement Learning	12
SAM	Self-Assessment-Manikin	85
SARSA	State-Action-Reward-State-Action	45
SEC	Stimulus Evaluation Check	21
SLAM	Simultaneous Localization and Mapping	56
TD	Temporal-Difference	39
VA-space	Valence-Arousal space	17
WASABI	Affect Simulation for Agents with Believable Interactivity	22

A. Appendix

The following appendix contains supplementary material which was used to create this dissertation.

A.1. Published algorithms

The developed algorithms and simulation tools presented in this dissertation have been made publicly available on *GitHub*¹. This enables interested researchers to simply clone the repositories and reproduce the results shown in this work.

In the following, the different algorithms and repositories are listed according to the sections:

Bandit Simulation (Section 3.1): Implementation of a multi-armed bandit simulation together with an affective evaluation component which bases on the Zurich Model of Social Attachment and Fear.

<https://github.com/jfeldmaier/emoBandit>



Gridworld (Section 3.2): Simulation environment for arbitrary Grid-world scenarios, and a Q-learning agent that learns to find the goal state. The performed actions of the agent are evaluated using the described appraisal model of Section 3.2.

<https://github.com/jfeldmaier/pathEmotions>



SLEmotion (Section 3.3): The modified EKF-SLAM simulation of Tim Bailey and Juan Nieto with the implemented Component Process Model.

<https://github.com/jfeldmaier/SLEmotion>



¹<https://github.com> [Accessed 24st February 2017]

A. Appendix

Reinforcement Learning with Preferences (Section 4.1): A framework for learning specific preferences according to an externally controlled affective state in context of a bandit simulation.

<https://github.com/jfeldmaier/prefRL>



Inverse Reinforcement Learning for Human Value Retrieval (Section 5.1): Multi-objective Reinforcement Learning Benchmark Suite which additionally includes the developed inverse RL algorithm for retrieving previously unknown scalarization weights out of existing policies.

<https://github.com/RL-LDV-TUM/morlbench>



A.2. Questionnaire of the SLEmotion experiment

A research project of the Chair for Data Processing investigates Affective Computing. Affective Computing is a topic related to Artificial Intelligence and tries to add artificial emotions and affect to the cognitive abilities of a robot.

In order to verify and to investigate the applicability of the results in shared environments we need your (human) feedback. We would be pleased if you carefully read the following questionnaire and answer the questions. Overall, this will take about 10 minutes.

Your data will be anonymized and only used in aggregated form for the verification of our developed models. The personal data will be kept secure and only the aggregated data will be provided or otherwise made accessible to third parties. – Thank you for participation.

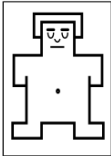
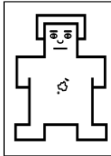
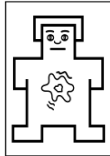
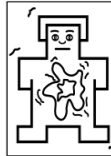

First, let's look which emotions are in your mind in the following three situations. Try to get mentally into the situation, and think on the emotions you would feel in it. There may be elicited just a single very strong feeling in your mind, but also several overlapping feelings or emotions.

1. You are in a dark, unknown cellar. With your hands you can feel walls and there are some larger pieces of furniture in this room. – You can only guess which additional dangers are present in this cellar. Now, think on your feelings you would feel in this situation and write the one that first comes to your mind in the following text box:

- 1.1. Please, try now to classify this feeling or emotion on the following two scales:

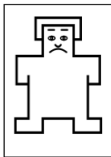
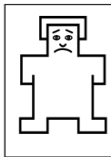
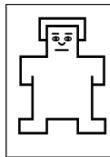
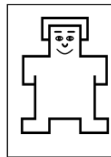
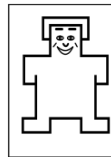
Scale 1:

(please check one box)

				
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Scale 2:

(please check one box)

				
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

A. Appendix

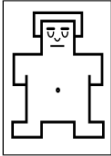
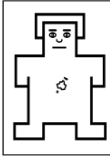
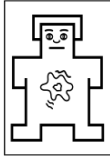
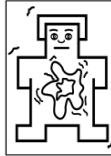
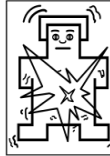
2. You arrive at home after a long vacation and enter the living room of your apartment or house. All your very familiar furniture and beloved things are on the right place – just as always.

Again think on your feelings in such a situation and write down the emotion or feeling that first comes to your mind in the following text box:

- 2.1. Please, try now to classify this feeling or emotion on the following two scales:

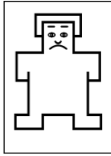
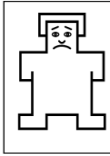
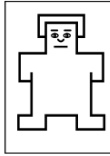
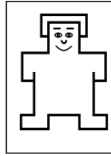
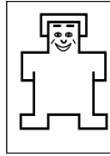
Scale 1:

(please check one box)

				
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Scale 2:

(please check one box)

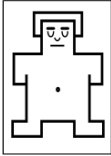
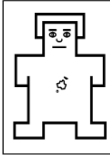
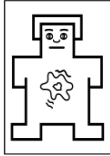
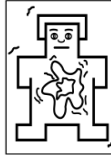
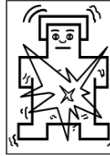
				
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

3. On a day off, you take a walk and choose your favorite trail or footpath. Think on your feelings you would have after returning. Please write down the feeling or emotion that first comes to your mind in the following text box:

- 3.1. Please, try now to classify this feeling or emotion on the following two scales:

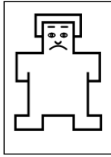
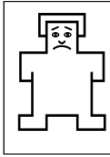
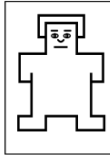
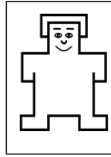
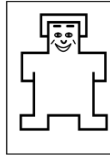
Scale 1:

(please check one box)

				
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

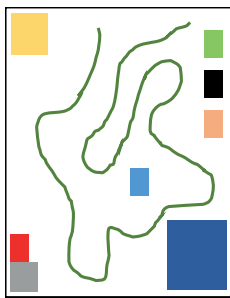
Scale 2:

(please check one box)

				
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

In this second part of the questionnaire we would like to deal with the feelings you would expect from a service robot in similar situations. Try to put yourself into each situation like before, and indicate your expectations on the feelings a robot should express in the particular situation. The service robot is able to express its feelings in a suitable way (like on a display) and it has the ability to calculate artificial feelings related to its situation.

In this part of the questionnaire, the individual scenarios are depicted as simplified sketches of rooms. Objects (such as furniture, plants, pictures) in the room are represented by differently sized squares, which the robot can recognize and use for navigation. The green line visualizes the path the robot moves within each room. The longer the robot stays in the room observing the objects, the more familiar it gets with its environment since the recognition rate of objects increases. A high recognition rate is required for a stable navigation of the robot.



4. In this first scenario, the robot slowly moves on its path through the room and along the objects to improve the performance of its navigation algorithm. How do you think does the robot feel in this room? Write down the first emotion that comes to your mind in the following text box:

4.1 Please, try to classify this feeling or emotion on the following two scales:

Scale 1:

(please check one box)

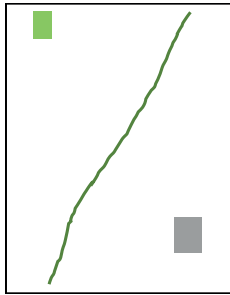
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Scale 2:

(please check one box)

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

A. Appendix

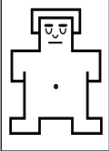
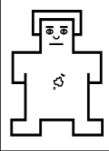
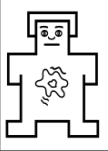




5. In this second scenario, the robot travels again on its path (green line) through a room and uses the objects for navigation. How do you think does the robot feel in this room? Write down the first emotion that comes to your mind in the following text box:

5.1 Please, try to classify this feeling or emotion on the following two scales:

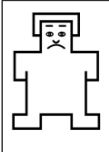
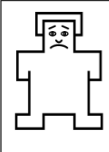
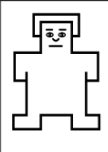
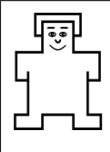
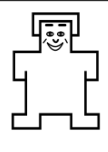
Scale 1:

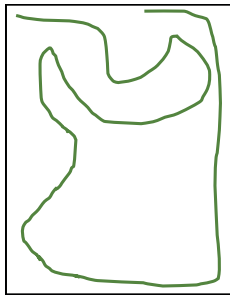
(please check one box)

				
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Scale 2:

(please check one box)

				
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

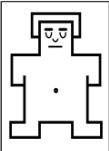
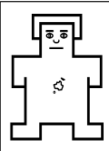
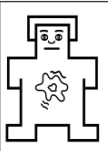

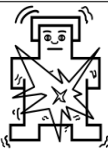


6. In this third scenario, there is a room without any objects detectable for the robot. The robot, however tries to detect objects and moves along the green line. Think again on the expressed feelings or emotions you would expect from the robot in this situation. Write down the first emotion or feeling that comes into your mind in the following text box:

6.1 Please, try to classify this feeling or emotion on the following two scales:

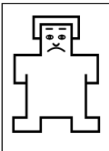
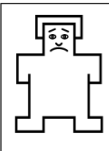
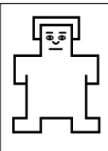
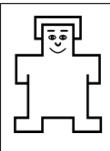
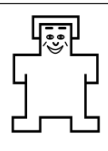
Scale 1:

(please check one box)

				
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Scale 2:

(please check one box)

				
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

A.3. Derivation of the Block Matrix Form of the Inverse RL algorithm

As shown in Section 5.1.2, the problem of finding the weights for the multi-objective scalarization boils down to solving the linear programming problem as stated in Equation (5.23), but it cannot be implemented straightforwardly in a linear programming solver package. In order to be able to use existing solver software, the formulation has to be rephrased and stated in a standard block matrix form. This rephrasing step is often omitted in similar studies, but is crucial for the verification of the results. The derivation will start out with the original formulation and then traverse the upper line of

$$\begin{aligned} & \underset{\alpha}{\text{maximize}} \sum_{s \in \hat{\mathcal{S}}} \min_{a \in \mathcal{A}(s) \setminus a^*} \left\{ \rho \left(\mathbb{E}_{s' \sim P_{ss'}^{a^*}} [V^{\pi^*}(s')] - \mathbb{E}_{s' \sim P_{ss'}^a} [V^{\pi^*}(s')] \right) \right\} \\ & \text{s.t. } |\alpha^{(i)}| \leq 1, i = 1, \dots, q, \end{aligned} \quad (\text{A.1})$$

from the outside to the inside, while adding in each step an appropriate constraint.

In the first step, the summation over the minimum over the non-optimal actions is replaced with a new variable $z_{s,a} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}| - 1}$ for each state $s \in \hat{\mathcal{S}}$ and action $a \in \mathcal{A}(s) \setminus a^*$. In order to reflect the desired minimum for this variable, the additional constraint that this variable has to be smaller than or equal to the inside of the curly brackets is added. After this transformation, the problem reads as

$$\begin{aligned} & \underset{\alpha}{\text{maximize}} \sum_{s \in \hat{\mathcal{S}}, a \in \mathcal{A}(s) \setminus a^*} z_{s,a} \\ & \text{s.t. } |\alpha^{(i)}| \leq 1, \quad i = 1, \dots, q \\ & z_{s,a} \leq \rho \left(\mathbb{E}_{s' \sim P_{ss'}^{a^*}} [V^{\pi^*}(s')] - \mathbb{E}_{s' \sim P_{ss'}^a} [V^{\pi^*}(s')] \right). \end{aligned} \quad (\text{A.2})$$

The next step is to reformulate the function $\rho(\cdot)$ in terms of two additional variables, in order to account for the components greater or equal to zero and for the rest, respectively. It is postulated that the sum of those two new variables $y_{s,a} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}| - 1}$ and $x_{s,a} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}| - 1}$ is equal to the argument of $\rho(\cdot)$. To ease notation, the individual quantities of this argument can be denoted as the vectors

$$v_{s,a} = \begin{pmatrix} (v_{s,a})^{(1)} \\ (v_{s,a})^{(1)} \\ \vdots \\ (v_{s,a})^{(q)} \end{pmatrix} = \begin{pmatrix} \mathbb{E}_{s' \sim P_{ss'}^{a^*}} [V_1^{\pi^*}(s')] - \mathbb{E}_{s' \sim P_{ss'}^a} [V_1^{\pi^*}(s')] \\ \mathbb{E}_{s' \sim P_{ss'}^{a^*}} [V_2^{\pi^*}(s')] - \mathbb{E}_{s' \sim P_{ss'}^a} [V_2^{\pi^*}(s')] \\ \vdots \\ \mathbb{E}_{s' \sim P_{ss'}^{a^*}} [V_q^{\pi^*}(s')] - \mathbb{E}_{s' \sim P_{ss'}^a} [V_q^{\pi^*}(s')] \end{pmatrix} \in \mathbb{R}^q. \quad (\text{A.3})$$

A. Appendix

Additionally, to acquire the constraints imposed by $p(\cdot)$, the variables $z_{s,a}$ now lower bounds the weighted sum of $x_{s,a}$ and $y_{s,a}$. Herewith, the overall problem, can be stated as

$$\begin{aligned}
 & \underset{\alpha}{\text{maximize}} && \sum_{s \in \hat{S}, a \in \mathcal{A}(s) \setminus a^*} z_{s,a} \\
 & \text{s.t.} && |\alpha^{(i)}| \leq 1, \quad i = 1, \dots, q \\
 & && z_{s,a} \leq y_{s,a} + 2 \cdot x_{s,a}, \quad s \in \hat{S}, a \in \mathcal{A}(s) \setminus a^* \\
 & && v_{s,a}^\top \alpha = x_{s,a} + y_{s,a}, \quad s \in \hat{S}, a \in \mathcal{A}(s) \setminus a^*.
 \end{aligned} \tag{A.4}$$

Up to now, the conditions $y_{s,a} \geq 0$ and $x_{s,a} < 0$ as required by $p(\cdot)$ are not in the formulation. Also the bound on the absolute value of the weights α_i is not properly reformulated yet. Including these constraints, the optimization problem can be stated as follows:

$$\begin{aligned}
 & \underset{\alpha}{\text{maximize}} && \sum_{s \in \hat{S}, a \in \mathcal{A}(s) \setminus a^*} z_{s,a} \\
 & \text{s.t.} && \alpha^{(i)} \leq 1, \quad i = 1, \dots, q \\
 & && \alpha^{(i)} \geq -1, \quad i = 1, \dots, q \\
 & && z_{s,a} \leq y_{s,a} + 2 \cdot x_{s,a}, \quad s \in \hat{S}, a \in \mathcal{A}(s) \setminus a^* \\
 & && v_{s,a}^\top \alpha = x_{s,a} + y_{s,a}, \quad s \in \hat{S}, a \in \mathcal{A}(s) \setminus a^* \\
 & && y_{s,a} \geq 0, \quad s \in \hat{S}, a \in \mathcal{A}(s) \setminus a^* \\
 & && x_{s,a} \leq 0, \quad s \in \hat{S}, a \in \mathcal{A}(s) \setminus a^*.
 \end{aligned} \tag{A.5}$$

This formulation can be directly rewritten in block matrix form as

$$\begin{aligned}
 & \text{maximize} && \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}^\top \begin{pmatrix} z \\ y \\ x \\ \alpha \end{pmatrix} \\
 & \text{s.t.} && \begin{bmatrix} 0 & I & I & -v^\top \end{bmatrix} \begin{bmatrix} z \\ y \\ x \\ \alpha \end{bmatrix} = 0, \\
 & && \begin{bmatrix} 0 & 0 & 0 & I \\ 0 & 0 & 0 & -I \\ 0 & -I & 0 & 0 \\ 0 & 0 & I & 0 \\ \Gamma_1 & -\Xi_1 & -2\Xi_1 & 0 \\ \Gamma_2 & -\Xi_2 & -2\Xi_2 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \Gamma_{|S|} & -\Xi_{|S|} & -2\Xi_{|S|} & 0 \end{bmatrix} \begin{bmatrix} z \\ y \\ x \\ \alpha \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix},
 \end{aligned} \tag{A.6}$$

A.3. Derivation of the Block Matrix Form of the Inverse RL algorithm

where \mathbf{x} , \mathbf{y} , \mathbf{z} and \mathbf{v} are vectors that contain all $x_{s,a}$, $y_{s,a}$, $z_{s,a}$ and $v_{s,a}^\top$ stacked into column vectors according to the same ordering, respectively. $\mathbf{1}$ and $\mathbf{0}$ denote vectors containing only 1 or 0 of appropriate length, while correspondingly the matrices denoted with \mathbf{I} and $\mathbf{0}$ represent square matrices of appropriate size only filled with 1 or 0, respectively. Additionally, in the bottom block the matrix Γ_s is defined as

$$\Gamma_s = \left. \begin{array}{c} \overbrace{\begin{bmatrix} 0 & \dots & 0 & 1 & 0 & \dots & 0 \\ 0 & \dots & 0 & 1 & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{bmatrix}}^{|\mathcal{S}|} \\ \end{array} \right\} |\mathcal{A}| - 1, \quad (\text{A.7})$$

which is a matrix consisting of zeros except for the s -th column. This column is filled with ones on the position that implicitly selects the entries of \mathbf{z} so that it corresponds to a particular state s . The remaining matrix Ξ consists of small diagonal matrices for each action except the optimal one starting from position $s \cdot (|\mathcal{A}| - 1)$. The final matrix Ξ then can be written down as

$$\Xi_s = \left. \begin{array}{c} \overbrace{\begin{bmatrix} 0 & \dots & 0 & 1 & 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & & \vdots & 0 & 1 & \ddots & 0 & \vdots & & \vdots \\ \vdots & & \vdots & 0 & \ddots & \ddots & 0 & \vdots & & \vdots \\ 0 & \dots & 0 & 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{bmatrix}}^{|\mathcal{S}| \times |\mathcal{A}| - 1} \\ \underbrace{\hspace{10em}}_{|\mathcal{A}| - 1} \end{array} \right\} |\mathcal{A}| - 1. \quad (\text{A.8})$$

This concludes the derivation of the linear program in the block matrix form. By entering the above specified block matrices into appropriate solver packages (like the already mentioned *cvxopt* Python package), a unique solution can be calculated.

Bibliography

- P. Abbeel and A.Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning*, pp. 1–8. 2004.
- C. Adam, B. Gaudou, A. Herzig, and D. Longin. OCC's emotions: A formalization in a BDI logic. In J. Euzenat and J. Domingue (eds.), *Artificial Intelligence: Methodology, Systems, and Applications*, volume 4183, pp. 24–32. Springer, Berlin, Germany, 2006.
- H. Ahn and R.W. Picard. Affective-cognitive learning and decision making: The role of emotions. In *Proceedings of the 18th European Meeting on Cybernetics and Systems Research*. 2006.
- B.A. Anderson. Social reward shapes attentional biases. In *Cognitive Neuroscience*, 7(1-4), pp. 30–36, 2016.
- J.R. Anderson, D. Bothell, M.D. Byrne, S. Douglass, C. Lebiere, and Y. Qin. An integrated theory of the mind. In *Psychological Review*, 111(4), pp. 1036–1060, 2004.
- N.H. Anderson. Information integration approach to emotions and their measurement. In R. Plutchik and H. Kellerman (eds.), *The measurement of emotion*, volume 4, pp. 133–186. Academic Press, New York, NY, USA, 1989.
- A. Axelrod and G. Chowdhary. The explore–exploit dilemma in nonstationary decision making under uncertainty. In L. Busoniu and L. Tamás (eds.), *Handling Uncertainty and Networked Structure in Robot Control*, pp. 29–52. Springer International Publishing, Cham, Switzerland, 2015.
- M. Babes, E.M. De Cote, and M.L. Littman. Social reward shaping in the prisoner's dilemma. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 3, pp. 1389–1392. 2008.
- T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (SLAM): Part II. In *IEEE Robotics & Automation Magazine*, 13(3), pp. 108–117, 2006.
- L. Barrett and S. Narayanan. Learning all optimal policies with multiple criteria. In *Proceedings of the 25th International Conference on Machine learning*, pp. 41–47. 2008.
- C. Becker-Asano and I. Wachsmuth. Affective computing with primary and secondary emotions in a virtual human. In *Autonomous Agents and Multi-Agent Systems*, 20(1), pp. 32–49, 2010.

Bibliography

- T.E.J. Behrens, M.W. Woolrich, M.E. Walton, and M.F.S. Rushworth. Learning the value of information in an uncertain world. In *Nature neuroscience*, 10(9), pp. 1214–1221, 2007.
- R.E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1957.
- D.P. Bertsekas. *Dynamic Programming and Optimal Control*. 2nd edition. Athena Scientific, Belmont, MA, USA, 2000.
- C.L. Bethel and R.R. Murphy. Survey of non-facial/non-verbal affective expressions for appearance-constrained robots. In *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(1), pp. 83–92, 2008.
- N. Bischof. A systems approach toward the functional connections of attachment and fear. In *Child Development*, 46(4), pp. 801–817, 1975.
- N. Bischof. *Das Rätsel Ödipus*. 5th edition. Piper, Munich, Germany, 1989.
- M.A. Boden. *Computer models of mind: Computational approaches in theoretical psychology*. Cambridge University Press, Cambridge, United Kingdom, 1988.
- R.C. Bolles and M.S. Fanselow. A perceptual-defensive-recuperative model of fear and pain. In *The Behavioral and Brain Sciences*, 3, pp. 291–301, 1980.
- I. Borutta, S. Sosnowski, M. Zehetleitner, N. Bischof, and K. Kühnlenz. Generating artificial smile variations based on a psychological system-theoretic approach. In *Proceedings of the 18th International Symposium on Robot and Human Interactive Communication*, pp. 245–250. 2009.
- M.M. Bradley and P.J. Lang. Measuring emotion: The self-assessment manikin and the semantic differential. In *Journal of behavior therapy and experimental psychiatry*, 25(1), pp. 49–59, 1994.
- C. Breazeal. Emotion and sociable humanoid robots. In *International Journal of Human-Computer Studies*, 59(1), pp. 119–155, 2003.
- J.S. Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In *Proceedings of the 1989 Conference on Advances in Neural Information Processing Systems*, pp. 211–217. 1989.
- J. Broekens. Modeling the experience of emotion. In *International Journal of Synthetic Emotions*, 1(1), pp. 1–17, 2010.
- R.A. Calvo, S.K. D'Mello, J. Gratch, and A. Kappas (eds.). *The Oxford Handbook of Affective Computing*. Oxford University Press, New York, NY, USA, 2015.

- E. Cambria. Affective computing and sentiment analysis. In *Intelligent Systems*, 31(2), pp. 102–107, 2016.
- J. Cassell. Embodied conversational interface agents. In *Communications of the ACM*, 43(4), pp. 70–78, 2000.
- A. Damásio. *Descartes' Error: Emotion, Reason, and the Human Brain*. Putnam Publishing, Kirkwood, NY, USA, 1994.
- K. Darling. *ROBOT ETHICS 2.0*, chapter “Who’s Johnny?” Anthropomorphic Framing in Human-Robot Interaction, Integration, and Policy. Oxford University Press, New York, USA, forthcoming 2017.
- P. Dayan and G.E. Hinton. Feudal reinforcement learning. In *Advances in Neural Information Processing Systems*, 5(1), pp. 271–278, 1993.
- P. Dayan and Y. Niv. Reinforcement learning: The good, the bad and the ugly. In *Current Opinion in Neurobiology*, 18(2), pp. 185–196, 2008.
- B. De Martino, D. Kumaran, B. Seymour, and R.J. Dolan. Frames, biases, and rational decision-making in the human brain. In *Science*, 313(5787), pp. 684–687, 2006.
- T. Dean, R. Givan, and S. Leach. Model reduction techniques for computing approximately optimal solutions for markov decision processes. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, pp. 124–131. 1997.
- J. Dias, S. Mascarenhas, and A. Paiva. FAtiMA modular: Towards an agent architecture with a generic appraisal framework. In T. Bosse, J. Broekens, J. Dias, and J. van der Zwaan (eds.), *Emotion Modeling: Towards Pragmatic Computational Models of Affective Processes*, pp. 44–56. Springer International Publishing, Cham, Switzerland, 2014.
- M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. In *IEEE Computational Intelligence Magazine*, 1(4), pp. 28–39, 2006.
- K. Doya. Metalearning and neuromodulation. In *Neural Networks*, 15(4), pp. 495–506, 2002.
- K. Doya. Modulators of decision making. In *Nature Neuroscience*, 11(4), pp. 410–416, 2008.
- J. Doyle. Prospects for preferences. In *Computational Intelligence*, 20(2), pp. 111–136, 2004.
- P. Ekman. An argument for basic emotions. In *Cognition & Emotion*, 6(3-4), pp. 169–200, 1992.

Bibliography

- P. Ekman, R.E. Sorenson, and W.V. Friesen. Pan-cultural elements in facial displays of emotion. In *Science*, 164(3875), pp. 86–88, 1969.
- M.S. El-Nasr, J. Yen, and T.R. Ioerger. Flame – fuzzy logic adaptive model of emotions. In *Autonomous Agents and Multi-agent Systems*, 3(3), pp. 219–257, 2000.
- E. Fehr and C.F. Camerer. Social neuroeconomics: The neural circuitry of social preferences. In *Trends in Cognitive Sciences*, 11(10), pp. 419–427, 2007.
- J. Feldmaier and K. Diepold. Emotional evaluation of bandit problems. In *Proceedings of the 22nd International Symposium on Robot and Human Interactive Communication*, pp. 149–154. IEEE, 2013.
- J. Feldmaier and K. Diepold. Path-finding using reinforcement learning and affective states. In *Proceedings of the 23rd International Symposium on Robot and Human Interactive Communication*, pp. 543–548. IEEE, 2014.
- J. Feldmaier, H. Shen, D. Meyer, and K. Diepold. Reinforcement learning with preferences. In *Proceedings of the 2nd Multidisciplinary Conference on Reinforcement Learning and Decision Making*. Edmonton, Alberta, Canada, 2015.
- J. Feldmaier, M. Stimpfl, and K. Diepold. Development of an emotion-competent SLAM agent. In *Proceedings of the Companion of the 12th International Conference on Human-Robot Interaction*, pp. 1–9. ACM/IEEE, New York, NY, USA, 2017.
- J.J.R. Fontaine. Dimensional, basic emotion, and componential approaches to meaning in psychological emotion research. In J.J.R. Fontaine, K.R. Scherer, and C. Soriano (eds.), *Components of Emotional Meaning: A sourcebook*. Oxford University Press, New York, NY, USA, 2013.
- J.P. Forgas and J.M. George. Affective influences on judgments and behavior in organizations: An information processing perspective. In *Organizational Behavior and Human Decision Processes*, 86(1), pp. 3–34, 2001.
- N.H. Frijda, P. Kuipers, and E. ter Schure. Relations among emotion, appraisal, and emotional action readiness. In *Journal of Personality and Social Psychology*, 57(2), pp. 212–228, 1989.
- J. Fürnkranz, E. Hüllermeier, W. Cheng, and S.H. Park. Preference-based reinforcement learning: A formal framework and a policy iteration algorithm. In *Machine Learning*, 89(1–2), pp. 123–156, 2012.
- Z. Gábor, Z. Kalmár, and C. Szepesvári. Multi-criteria reinforcement learning. In *Proceedings of the 15th International Conference on Machine Learning*, volume 98, pp. 197–205. 1998.

- P. Gebhard. ALMA: A layered model of affect. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 29–36. 2005.
- G. Gigerenzer, R. Hertwig, and T. Pachur. *Fast and Frugal Heuristics - Theory, Tests, and Applications*. Oxford University Press, New York, NY, USA, 2011.
- J.C. Gittins. Bandit processes and dynamic allocation indices. In *Journal of the Royal Statistical Society. Series B (Methodological)*, 41(2), pp. 148–177, 1979.
- D. Grandjean, D. Sander, and K.R. Scherer. Conscious emotional experience emerges as a function of multilevel, appraisal-driven response synchronization. In *Consciousness and Cognition*, 17(2), pp. 484–495, 2008.
- J. Gratch and S. Marsella. Lessons from emotion psychology for the design of lifelike characters. In *Applied Artificial Intelligence*, 19(3–4), pp. 215–233, 2005.
- S. Griffith, K. Subramanian, J. Scholz, C. Isbell, and A.L. Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in Neural Information Processing Systems*, 26(1), pp. 2625–2633, 2013.
- G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. In *IEEE Transactions on Robotics*, 23(1), pp. 34–46, 2007.
- J.J. Gross and O.P. John. Individual differences in two emotion regulation processes: implications for affect, relationships, and well-being. In *Journal of Personality and Social Psychology*, 85(2), pp. 348–362, 2003.
- M. Grześ and D. Kudenko. Online learning of shaping rewards in reinforcement learning. In *Neural Networks*, 23(4), pp. 541–550, 2010.
- H. Gubler and N. Bischof. A systems theory perspective. In M.E. Lamb and H. Keller (eds.), *Infant development: Perspectives from German-speaking countries*, pp. 35–66. Lawrence Erlbaum Associates, Hillsdale, NJ, England, 1991.
- J.E. Guivant and E.M. Nebot. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. In *IEEE Transactions on Robotics and Automation*, 17(3), pp. 242–257, 2001.
- V. Gullapalli and A.G. Barto. Shaping as a method for accelerating reinforcement learning. In *Proceedings of the IEEE International Symposium on Intelligent Control*, pp. 554–559. 1992.
- S.S. Haykin. *Kalman filtering and neural networks*. John Wiley & Sons, New York, NY, USA, 2001.

Bibliography

- K. Hogan and R. Stubbs. *Can't Get Through: Eight Barriers to Communication*. Pelican Publishing Company, Gretna, LA, USA, 2003.
- E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Brinker. Label ranking by learning pairwise preferences. In *Artificial Intelligence*, 172(16), pp. 1897–1916, 2008.
- Y. Jin and B. Sendhoff. Pareto-based multiobjective machine learning: An overview and case studies. In *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 38(3), pp. 397–415, 2008.
- D. Kahneman. *Thinking, fast and slow*. Farrar Straus & Giroux, New York, NY, USA, 2011.
- D. Kahneman and A. Tversky. Prospect theory: An analysis of decision under risk. In *Econometrica*, 47(2), pp. 263–291, 1979.
- E. Kaufmann, O. Cappé, and A. Garivier. On bayesian upper confidence bounds for bandit problems. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 22, pp. 592–600. 2012.
- J.E. Laird. *The Soar cognitive architecture*. MIT Press, Cambridge, MA, USA, 2012.
- R.S. Lazarus. *Psychological stress and the coping process*. McGraw-Hill, New York, NY, USA, 1966.
- J.E. LeDoux. *The emotional brain: The mysterious underpinnings of emotional life*. Simon & Schuster, New York, NY, USA, 1998.
- N. Leibowitz, B. Baum, G. Enden, and A. Karniel. The exponential learning equation as a function of successful trials results in sigmoid performance. In *Journal of Mathematical Psychology*, 54(3), pp. 338–340, 2010.
- S. Livingston, J. Garvey, and I. Elhanany. On the broad implications of reinforcement learning based AGI. In *Proceedings of the 1st Artificial General Intelligence Conference*, pp. 478–483. 2008.
- D.J. Lizotte, M. Bowling, and S.A. Murphy. Linear fitted-Q iteration with multiple reward functions. In *The Journal of Machine Learning Research*, 13(1), pp. 3253–3295, 2012.
- W.G. Macready and D.H. Wolpert. Bandit problems and the exploration/exploitation trade-off. In *IEEE Transactions on Evolutionary Computation*, 2(1), pp. 2–22, 1998.
- S. Mannor and N. Shimkin. A geometric approach to multi-criterion reinforcement learning. In *The Journal of Machine Learning Research*, 5, pp. 325–360, 2004.
- A.S.R. Manstead and A.H. Fischer. Social appraisal: The social world as object of and influence on appraisal processes. In K.R. Scherer, A. Schorr, and T. Johnstone (eds.), *Appraisal processes in emotion: Theory, methods, research*, pp. 221–232. Oxford University Press, New York, NY, USA, 2001.

- R.P. Marinier, J.E. Laird, and R.L. Lewis. A computational unification of cognitive behavior and emotion. In *Cognitive Systems Research*, 10(1), pp. 48–69, 2009.
- S. Marsella, J. Gratch, and P. Petta. Computational models of emotion. In K.R. Scherer, T. Bänziger, and R. Etienne (eds.), *Blueprint for Affective Computing: A Sourcebook*, pp. 21–46. Oxford University Press, New York, NY, USA, 2010.
- S.C. Marsella and J. Gratch. Ema: A process model of appraisal dynamics. In *Cognitive Systems Research*, 10(1), pp. 70–90, 2009.
- R.R. McCrae and O.P. John. An introduction to the five-factor model and its applications. In *Journal of Personality*, 60(2), pp. 175–215, 1992.
- A. Mehrabian. Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in temperament. In *Current Psychology*, 14(4), pp. 261–292, 1996.
- M.L. Minsky. *The Emotion Machine: Commonsense Thinking, Artificial Intelligence, and the Future of the Human Mind*. Simon & Schuster, New York, NY, USA, 2006.
- H. Miwa, T. Okuchi, K. Itoh, H. Takanobu, and A. Takanishi. A new mental model for humanoid robots for human friendly communication introduction of learning system, mood vector and second order equations of emotion. In *Proceedings of the International Conference on Robotics and Automation*, volume 3, pp. 3588–3593. 2003.
- V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski et al.. Human-level control through deep reinforcement learning. In *Nature*, 518(7540), pp. 529–533, 2015.
- K. Moffaert, M.M. Drugan, and A. Nowé. Hypervolume-based multi-objective reinforcement learning. In *Proceedings of the 7th International Conference on Evolutionary Multi-Criterion Optimization*, pp. 352–366. 2013.
- A. Moors, P.C. Ellsworth, K.R. Scherer, and N.H. Frijda. Appraisal theories of emotion: State of the art and future development. In *Emotion Review*, 5(2), pp. 119–124, 2013.
- M. Mori, K.F. MacDorman, and N. Kageki. The uncanny valley. In *IEEE Robotics & Automation Magazine*, 19(2), pp. 98–100, 2012.
- L. Muehlhauser and L. Helm. The singularity and machine ethics. In A.H. Eden, J.H. Moor, J.H. Søraker, and E. Steinhart (eds.), *Singularity Hypotheses: A Scientific and Philosophical Assessment*, pp. 101–126. Springer, Heidelberg, Germany, 2012.
- A.Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning*, volume 99, pp. 278–287. 1999.

Bibliography

- A.Y. Ng and S.J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning*, pp. 663–670. 2000.
- D.A. Norman, A. Ortony, and D.M. Russell. Affect and machine design: Lessons for the development of autonomous machines. In *IBM Systems Journal*, 42(1), pp. 38–44, 2003.
- D.A. Norman. *The design of everyday things: Revised and expanded edition*. Basic Books, New York, NY, USA, 2013.
- A. Ortony, G.L. Clore, and A. Collins. *The cognitive structure of emotions*. Cambridge University Press, New York, NY, USA, 1990.
- I.P. Pavlov and G.V. Anrep. *Conditioned reflexes*. Dover Publications, Mineola, NY, USA, 2003.
- M.D. Pell, L. Monetta, S. Paulmann, and S.A. Kotz. Recognizing emotions in a foreign language. In *Journal of Nonverbal Behavior*, 33(2), pp. 107–120, 2009.
- P. Perny and P. Weng. On finding compromise solutions in multiobjective markov decision processes. In *Proceedings of 19th European Conference on Artificial Intelligence*, pp. 969–970. 2010.
- E.A. Phelps. Emotion and cognition: Insights from studies of the human amygdala. In *Annual Review of Psychology*, 57, pp. 27–53, 2006.
- R. Picard. *Affective computing*. MIT Press, Cambridge, MA, USA, 1997.
- R. Plutchik. *The emotions*. University Press of America, Lanham, MD, USA, 1991.
- D. Ramachandran and E. Amir. Bayesian inverse reinforcement learning. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pp. 2856–2591. 2007.
- A.S. Rao and M.P. Georgeff. Modeling rational agents within a BDI-Architecture. In *Proceedings of the International Conference on Principles of Knowledge Representation and Planning*, pp. 473–484. 1991.
- R. Reisenzein, E. Hudlicka, M. Dastani, J. Gratch, K. Hindriks, E. Lorini, and J.J. Meyer. Computational modeling of emotion: Toward improving the inter- and intradisciplinary exchange. In *IEEE Transactions on Affective Computing*, 4(3), pp. 246–266, July 2013.
- H. Robbins. Some aspects of the sequential design of experiments. In *Bulletin of the American Mathematical Society*, 58(5), pp. 527–535, 1952.
- L.F. Rodríguez, J.O. Gutierrez-Garcia, and F. Ramos. Modeling the interaction of emotion and cognition in autonomous agents. In *Biologically Inspired Cognitive Architectures*, 17, pp. 57–70, 2016.

- D. Roijers, J. Scharpff, M. Spaan, F. Oliehoek, M. de Weerdt, and S. Whiteson. Bounded approximations for linear multi-objective planning under uncertainty. In *Proceedings of the 24th International Conference on Automated Planning and Scheduling*, pp. 262–270. 2014.
- D.M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. A Survey of Multi-Objective Sequential Decision-Making. In *Journal of Artificial Intelligence Research*, 48, pp. 67–113, 2013.
- I.J. Roseman, M.S. Spindel, and P.E. Jose. Appraisals of emotion-eliciting events: Testing a theory of discrete emotions. In *Journal of Personality and Social Psychology*, 59(5), pp. 899–915, 1990.
- C.A. Rothkopf and C. Dimitrakakis. Preference elicitation and inverse reinforcement learning. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 34–48. 2011.
- J.A. Russell. Evidence of convergent validity on the dimensions of affect. In *Journal of Personality and Social Psychology*, 36(10), pp. 1152–1168, 1978.
- J.A. Russell. Culture and the categorization of emotions. In *Psychological bulletin*, 110(3), pp. 426–450, 1991.
- J.A. Russell. Core affect and the psychological construction of emotion. In *Psychological Review*, 110(1), pp. 145–172, 2003.
- J.A. Russell and L.F. Barrett. Core affect, prototypical emotional episodes, and other things called emotion: dissecting the elephant. In *Journal of Personality and Social Psychology*, 76(5), pp. 805–819, 1999.
- J.A. Russell and A. Mehrabian. Evidence for a three-factor theory of emotions. In *Journal of research in Personality*, 11(3), pp. 273–294, 1977.
- R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*, pp. 791–798. 2007.
- D.A. Sauter, F. Eisner, P. Ekman, and S.K. Scott. Cross-cultural recognition of basic emotions through nonverbal emotional vocalizations. In *Proceedings of the National Academy of Sciences of the United States of America*, 107(6), pp. 2408–2412, 2010.
- K.R. Scherer. Toward a dynamic theory of emotion. In *Geneva studies in Emotion*, 1, pp. 1–96, 1987.
- K.R. Scherer. Feelings integrate the central representation of appraisal-driven response organization in emotion. In A.S.R. Manstead, N. Frijda, and A. Fischer (eds.), *Feelings*

Bibliography

- and Emotions: The Amsterdam Symposium*, pp. 136–157. Cambridge University Press, Cambridge, United Kingdom, 2004.
- K.R. Scherer. The dynamic architecture of emotion: Evidence for the component process model. In *Cognition and emotion*, 23(7), pp. 1307–1351, 2009.
- K.R. Scherer. The component process model: Architecture for a comprehensive computational model of emergent emotion. In K.R. Scherer, T. Bänziger, and R. Etienne (eds.), *Blueprint for Affective Computing: A Sourcebook*, pp. 47–70. Oxford University Press, New York, NY, USA, 2010.
- K.R. Scherer, R. Banse, and H.G. Wallbott. Emotion inferences from vocal expression correlate across languages and cultures. In *Journal of Cross-Cultural Psychology*, 32(1), pp. 76–92, 2001.
- S.H. Schwartz. Are there universal aspects in the structure and contents of human values? In *Journal of social issues*, 50(4), pp. 19–45, 1994.
- H. van Seijen and R. Sutton. True online TD(λ). In *Proceedings of the 31st International Conference on Machine Learning*, pp. 692–700. 2014.
- S. Singh, T. Jaakkola, M.L. Littman, and C. Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. In *Machine Learning*, 38(3), pp. 287–308, 2000.
- B.F. Skinner. *Science and Human Behavior*. Macmillan, New York, NY, USA, 1953.
- B.F. Skinner. *The behavior of organisms: An experimental analysis*. Appleton-Century, New York, NY, USA, 1938.
- J.R. Slagle. A heuristic program that solves symbolic integration problems in freshman calculus. In *Journal of the ACM*, 10(4), pp. 507–520, 1963.
- N. Soares. *The value learning problem*. Technical Report, Machine Intelligence Research Institute, Berkeley, CA, USA, 2015.
- S. Sosnowski, A. Bittermann, K. Kuhnlenz, and M. Buss. Design and evaluation of emotion-display EDDIE. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3113–3118. 2006.
- B.R. Steunebrink, M. Dastani, and J.J.C. Meyer. The OCC model revisited. In *Proceedings of the 4th Workshop on Emotion and Computing*, pp. 146–154. 2009.
- B.R. Steunebrink, M. Dastani, and J.J.C. Meyer. A formal model of emotion triggers: An approach for BDI agents. In *Synthese*, 185(1), pp. 83–129, 2012.

- R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. 1st edition. MIT Press, Cambridge, MA, USA, 1998.
- A. Tversky and D. Kahneman. The framing of decisions and the psychology of choice. In *Science*, 211(4481), pp. 453–458, 1981.
- P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker. Empirical evaluation methods for multiobjective reinforcement learning algorithms. In *Machine Learning*, 84(1–2), pp. 51–80, 2011.
- P. Vamplew, J. Yearwood, R. Dazeley, and A. Berry. On the limitations of scalarisation for multi-objective reinforcement learning of pareto fronts. In W. Wobcke and M. Zhang (eds.), *Advances in Artificial Intelligence*, pp. 372–378. Springer, Berlin, Germany, 2008.
- K. Van Moffaert and A. Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. In *The Journal of Machine Learning Research*, 15(1), pp. 3483–3512, 2014.
- J.D. Velásquez. When robots weep: Emotional memories and decision-making. In *Proceedings of the 15th National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference*, pp. 70–75. 1998.
- D. Vernon, G. Metta, and G. Sandini. A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents. In *Transactions on Evolutionary Computation*, 11(2), pp. 151–180, 2007.
- C.J.C.H. Watkins and P. Dayan. Technical note: Q-learning. In *Machine Learning*, 8(3), pp. 279–292, 1992.
- R.S. Woodworth. *Experimental psychology*. Holt, Rinehart and Winston, New York, NY, USA, 1938.
- W. Wundt. *Outlines of psychology*. Engelmann, Leipzig, Germany, 1897.