Towards Adaptive Human–Robot Interaction

PhD thesis proposal

Ján Magyar, MSc

Supervisor: prof. Peter Sinčák

Department of Cybernetics and Artificial Intelligence
Faculty of Electrical Engineering and Informatics
Technical University of Košice

March 2019

Declaration

I hereby declare that this thesis is my own work and effort. Where other sources of information have been used, they have been acknowledged.

Ján Magyar, MSc
March 2019

Acknowledgements

I would like to express my sincere gratitude to my supervisor, prof. Ing. Peter Sinčák, CSc., for giving me the opportunity to grow as a researcher and for providing valuable insights from the field of artificial intelligence.

I would further like to thank Ing. Gergely Magyar, PhD for his advice in writing this thesis proposal and for his invaluable help in my research.

Special mention should go to my parents who have always supported me in my studies, and have encouraged me to follow my goals. This wouldn't be possible without you.

Last but not least, thanks to all my friends who reminded me not to take myself too seriously.

Abstract

This mid-term thesis serves as an overview of the theoretical background behind adaptive human–robot interaction and its use in educational settings. It contains a short introduction to skill development theory as well as a description of social robots and their use in education. The core of this work is an overview of reinforcement learning and how it can be used in adaptive human–robot interaction. The work further makes a case for and provides examples of using cloud computing in robotics research. The thesis proposal also presents the scientific and technological goals of the PhD thesis along with the proposed testing scenario.

Table of contents

List of figures

List of tables

Introduction

Knowledge and skill acquisition are an integral part of human life and human nature. In their first years, people learn to talk and walk, and they get familiar with the social norms that prepare them to be a part of society throughout their lives. They are taught to read and write in formal education, and they learn necessary basic and in many cases advanced knowledge. Learning, however, also takes place outside and beyond formal education, especially in a professional setting. In today's information-driven society and economy, the ability to acquire new knowledge and develop new skills quickly and efficiently is more important than ever.

Psychological research exploring the science behind learning successfully determined the characteristics of an optimal learning environment. Apart from the learner's motivation to learn and improve, the material being taught must be designed in a way that takes into consideration the learner's previous knowledge and experience, and the learner must come across immediate informative feedback based on her performance. Although there is a consensus about what an efficient learning process should look like, implementing it in a real setting might prove to be problematic, since it requires personalized tutoring and dedicated attention from the teacher, something that is rarely plausible in formal education.

In past years, technological developments have been used to address this problem and to optimize the learning process of students of all ages. Artificial intelligence provides the possibility of personalizing the learning process for a larger number of students, while research in human–robot interaction explored the viability of using social robots in education, most of this research focusing on knowledge acquisition. Cloud computing offers an ideal environment for computationally heavy operations connected to personalization, and it also promotes higher accessability and speed.

In our work we want to use methods of artificial intelligence, reinforcement learning in particular, to help to create a personalized and optimized learning experience for learners developing a new skill, and to train adaptive tutors for cognitive stimulation therapy.

Main goals and structure of the mid-term PhD thesis

The goals of this mid-term thesis are the following:

- provide an introduction to the theoretical background of human learning, and cognitive stimulation therapy
- give an overview of intelligent tutoring systems and how they are used in education
- provide an overview of social robots and their application in educational settings
- describe reinforcement learning and some of its methods and algorithms
- list the advantages of cloud computing to robotic research and tutoring systems
- define the scientific and technological goals of the PhD thesis

Based on the goals of the thesis, this mid-term thesis is structured as follows:

- Chapter 1 discusses the theory of skill development and types of practice. The problem of knowledge and skill retention is described, and spaced repetition is shown to be an effective solution. The chapter contains an introduction to cognitive stimulation therapy, and closes with the description and a short overview of intelligent tutoring systems, software applications used in personalized education.
- Chapter 2 focuses on the use of robots in social settings, especially in education and cognitive stimulation therapy. It lists issues that are necessary to consider when testing and evaluating interactive robotic systems, and outlines three possibilities for adaptive human–robot interaction.
- Chapter 3 is a short introduction to reinforcement learning, an approach of machine learning that can be used to adapt the learning process to the student's needs. It touches on problems and questions that must be considered when applying reinforcement learning, and it describes a few basic algorithms and methods.
- Chapter 4 comprises a short introduction to cloud computing, and the potential benefits of its use in robotics research, with special emphasis on personalization.
- Chapter 5 presents the scientific and technological goals of the PhD thesis.
- Chapter 6 describes the scenario in which the proposed outputs of the doctorate thesis will be tested and evaluated.
- Chapter 7 concludes the work.

Chapter 1

Human learning and interactive tutoring systems

Learning is inherent to human life and human development be it in a formal or an informal setting. As such, understanding the human learning process and optimizing it for the highest learning gain and rate of retention has been subject of research for decades. Educational applications also became part of artificial intelligence research. In this chapter, we look at the process of skill development, and different types of practices used in skill development. We describe how spaced repetition can be beneficial for long-term retention. The chapter concludes with a list of some intelligent tutoring system architectures.

1.1 Skill development

Besides knowledge acquisition, learning also entails the acquisition and development of new or existing skills. Most learning processes, such as language learning, consist of both knowledge acquisition (e.g. vocabulary) and skill development (e.g. sentence structures). Even though skill development is an essential part of human development, in most cases, it is done in an ad hoc and naive way [26]: after some initial carefully devised lessons (given by a teacher/tutor/coach or following a formal and structured course), people reach an acceptable level after which they practice the same things over and over. While the general belief is that such practice will lead to improved performances, research has shown that beyond reaching a level of competence, simple repetition does not help people to get better at a skill. On the contrary, repetition might lead to performance slightly worse than that of beginners [26][28].

1.1.1 Stages of competence

The steps of skill development were formally described by Broadwell in 1969 as the four levels of teaching [17]. The four stages of competence (*hierarchy of competence* or *four stages of learning*) describe basic stages in the learning process and identifying them can help to create a more efficient learning process. The four stages are [33]:

1. **unconscious incompetence** – It is associated with the wrong intuition from the learner who is not aware of the existance and/or relevance of the skill area and does not recognize her deficit in the area which hinders any development. The learner must first recognize her incompetence, and the value of the new skill. The time needed for moving on to the next stage depends on the incentive to learn.

2. **conscious incompetence** – It is associated with the wrong analysis from the learner who understands the importance and relevance of the skill and also perceives her incompetence (usually by attempting to perform the skill). Initial learning of the skill takes place here. The learner still does not know how to do something, but she sees the value of the new skill and why she should address her deficit. At this stage, making mistakes is an integral part of the learning process.

3. **conscious competence** – It is associated with the right analysis from the learner who already understands or knows how to do something but performing the skill requires concentration. This stage comprises the majority of a learner's time spent on skill development. The learner usually breaks the skill down into smaller steps, and relies on heavy conscious involvement in executing the new skill.

4. **unconscious competence** – It is associated with the right intuition from the learner who, thanks to practice, is now able to perform the skill easily. Performance is fully automatic and can be done while executing other tasks. The learner is able to teach others in the skill, although due to the unconscious performing of the skill, she might find it difficult to explain the basic steps. Frequent practice with the aim to improve is necessary to maintain this level and prevent deterioration of the skill.

1.1.2 Purposeful practice

Purposeful practice is more thoughtful and focused than simple repetition of the skill [26]. While simple repetition means performing the skill without any further consideration, purposeful practice has well-defined specific goals that are easy to check at the end of practice.

Furthermore, purposeful practice is focused and involves feedback. While all practices have some inherent belated feedback (e.g. the result of performing the skill), purposeful practice involves feedback already during training and is closely connected to the attention the learner gives to the task, and it engages the learner's ability to approach his learning in an analytical way. Feedback is needed to identify weaknesses and to find out which aspects of the skill must be improved to achieve one's goals.

Finally, purposeful practice requires the learner to always perform right beyond the limits of his comfort zone. This characteristic is expected to prevent the learner from getting comfortable and stalling his improvement by constantly challenging his physical and/or cognitive capacity. In this regard purposeful practice is also necessary to maintain one's abilities.

1.1.3 Deliberate practice

Although purposeful practice is efficient in developing one's skills beyond the limits that can be reached through simple repetitive practice, it is still not enough to maximize one's performance levels, something which is highly dependant on the mental representation one uses to perform a skill [20]. Ericsson et al. in [29] defined another type of practice called deliberate practice. They observed such practice especially in domains where education is highly developed and formalized, such as classical musical training [30], and sports [27]. In these domains, performance and achievements increased over time, and this improvement is connected to the development of teaching and training methods used in the domain. Generally, learners engage in personalized training that is devised with consideration to their current abilities and are intended to push the learner just beyond their current skill level [26].

The difference between purposeful and deliberate practice is twofold [29]. First, deliberate practice is only possible in a field with clearly distinguishable elite and beginner performers; these fields are usually developed and competitive with objective or semiobjective ways to measure performance. Second, deliberate practice requires a teacher or trainer to provide learners with personalized activities that improve their performance the most. Thus, in addition to being purposeful, deliberate practice is also informed, since learners understand the top performers' accomplishments and their methods to reach such performances. In contrast with performance, deliberate practice has almost no external rewards, and it is not inherently enjoyable [29].

In [26] and [29] Ericsson defines the following traits of deliberate practice:

- Deliberate practice develops skills already mastered by others where a set of efficient training techniques have been established.

- The practice is overseen by a teacher or a coach.
- Deliberate practice requires the student to try things just outside his or her comfort zone.
- Similarly to purposeful practice, deliberate practice targets a specific goal.
- Deliberate practice requires the learner's full attention; the learner must be able to observe his or her own practice to be able to make adjustments.
- Based on feedback acquired from the learner or teacher, deliberate practice is modified.
- Mental representations play a crucial role in the student's ability to analyze his or her performance; improvements in performance are only possible through improvements in mental representations.
- Deliberate practice involves improving particular aspects of a skill, consisting of a series of step-by-step improvements.

1.2 Retention

After knowledge or skill acquisition ends, retention of the new information or of the ability to perform a skill becomes desirable. Somewhat counterintuitively, for long-term retention, some degree of short-term forgetting is beneficial, since it strengthens the ability to be able to recall information the most. This reality stems from a well-observed phenomenon called the *spacing effect*, whereby the amount of learned material is greater when learning takes place over a longer period of time and multiple study sessions instead of a single instance [25].

Spaced repetition is a learning technique that exploits the spacing effect and also ensures effective long-term retention. The idea behind spaced repetition is that bits of information that the learner recalls easily should be revised less frequently and in growing intervals while the material that causes problems should be revisited again.

Spaced repetition as a learning technique was first described by C. A. Mace in 1932 [77]. He pointed out the importance of distributing study sessions over a longer period of time and proposed that revision be spaced in gradually increasing intervals. Research in spaced repetition gained new momentum in the 1970s when psychologists showed its effectiveness in improving recall [69][85]. A general spaced repetition learning system using flashcards was proposed in 1973 by Sebastian Leitner [38].

In the Leitner system (see Figure 1.1), flashcards are sorted into categories based on how hard the learner finds to recall information described on the cards. During each study session,

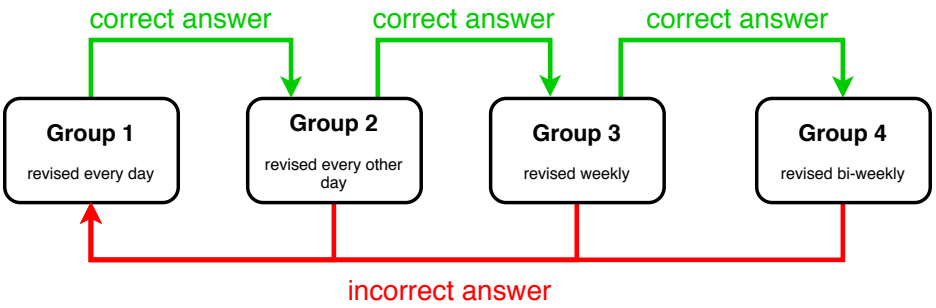

Fig. 1.1 The Leitner system

the student tries to recall the information on a given flashcard and if he is successful, he moves the flashcard into the next group which he revises at longer intervals. If the recall of a flashcard is incorrect, the flashcard is moved to the first group of cards, revised on a daily basis (or, in some variations of the system, into the previous group of cards).

The original Leitner system was further developed and adjusted by Wozniak [135]. The advent of spaced repetition computer softwares gave rise to personalizable and adjustable algorithms that can use neural networks to determine the rate of revision.

1.3 Intelligent tutoring systems

The incorporation of computer technology and artificial intelligence in education gave rise to the new field of computer-assisted instruction from which the research of intelligent tutoring systems (ITSs) emerged. While some research in computer-assisted instruction envisioned a more radical change in formal education and its environment [102], intelligent tutoring systems are created for the current educational systems [98]. The goal of ITSs is to reproduce the behavior and teaching strategy of a human tutor. Therefore, ITSs must know what they teach, whom they teach and how to teach. The research of ITSs lies at the intersection of computer science, artificial intelligence, cognitive psychology, and educational research. The motivation behind ITSs is that they can provide presonalized one-on-one tutoring which was shown to be the most effective learning approach for most subjects.

1.3.1 Architecture of intelligent tutoring systems

The general architecture of an ITS (see Figure 1.2) consists of four basic components: the domain model, the student model, the teaching model, and a user interface or learning environment [36][98].

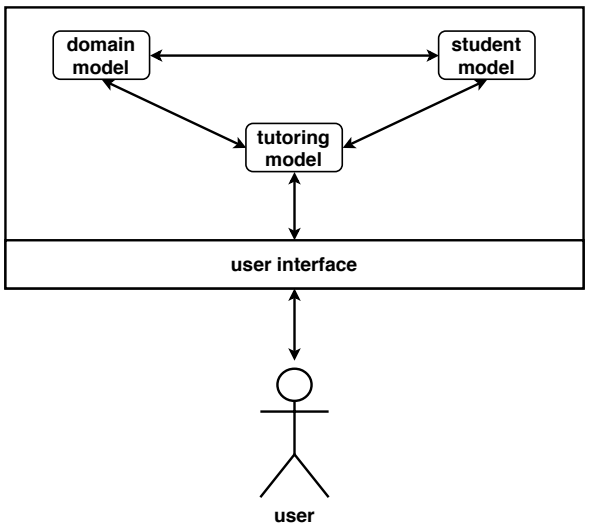

Fig. 1.2 Architecture of an intelligent tutoring system (recreated from [98])

The **domain model** contains the facts and rules of the domain being taught. It is the source of knowledge to be presented to the student, and it also has the information necessary to evaluate the student's answers and performance. The domain model must be able to generate questions and problems, their solutions, explanations, and responses. The component must also have the ability to identify mistakes the student makes, so it can draw the student's attention to them, just like to any gaps in her knowledge. The domain model represents an expert's view on the subject, this might lead to problems with interpretation of instructions by the student or with interpretation of student answers by the component [98].

The **student model** is the dynamic representation of the student's knowledge and skills (for a more detailed description of implementation see sections 1.3.2 and 1.3.3). This component is necessary to enable the understanding of the student and the personalization of the learning experience. The model can include any aspects of student behavior that might have an impact on student performance, although this might not be possible because of the constricted nature of communication between student and system. [115] identifies six types of student model functions:

- corrective – to address misconceptions in the student's knowledge
- elaborative – to correct incomplete student knowledge
- strategic – to change the tutorial strategy in a significant way
- diagnostic – to identify misconceptions in the student's knowledge
- predictive – to determine the student's most likely responses to the system's actions

- evaluative – to assess the student or the ITS

The **teaching model** or **tutoring model** designs and controls the interaction with the student. It uses information from the domain and student models to infer appropriate pedagogic actions. Based on the pedagogic strategy, action selection can take multiple forms. Two of the most common decision making processes are *elicit/tell* and *justify/skip-justify* [22]. A system can *elicit* a step from a student, meaning that the student must explicitly express the next step in solving a problem, or it can *tell* the student the next step without further explanation. The *justify* action asks the student to explain why he used a certain step in the solution as this is expected to deepen the student's understanding of the domain. On the other hand, a discussion of a problem solving step might not be necessary or helpful to the learning process, such decisions are called *skip-justify*.

The **user interface** or **learning environment** is the communication channel between the system and the student. It bridges the difference between the system's internal representation and an interface language understandable to the student, and also preprocesses input from the student.

[4] presents an alternative architecture that contains a bug catalogue of common misconceptions and errors instead of a student model (see Figure 1.3). The domain model also serves as the model that the student should ideally reach. The architecture presented in [101] contains five components: student history, student model, teaching strategy, teaching generation, teaching administration (communicating with the student). ITSs also incorporate the idea of gradual self-improvement over time. Their architecture is shown in Figure 1.4; the adaptive teaching system has the architecture of an ITS, while the self-improving component applies experimental changes gathered from study sessions to improve the system's tutoring abilities.

In the following two sections we look at two commonly used methods of student knowledge modeling: model tracing, and constraint-based modeling.

1.3.2 Model tracing

Model tracing tutoring (MTT) stems from the adaptive control of thought theory, which states that "acquiring cognitive knowledge involves the formulation of thousands of rules relating task goals and task states to actions and consequences" [3]. The aim of MTT is to recreate the thought processes that lead a student to a solution of a problem. The thought process is represented through a series of rule executions whose result is the same as the student input. An MTT system is composed of expert rules, buggy rules, a model tracer and a user interface (see Figure 1.3). Expert rules model the steps used to solve the problem at hand and can

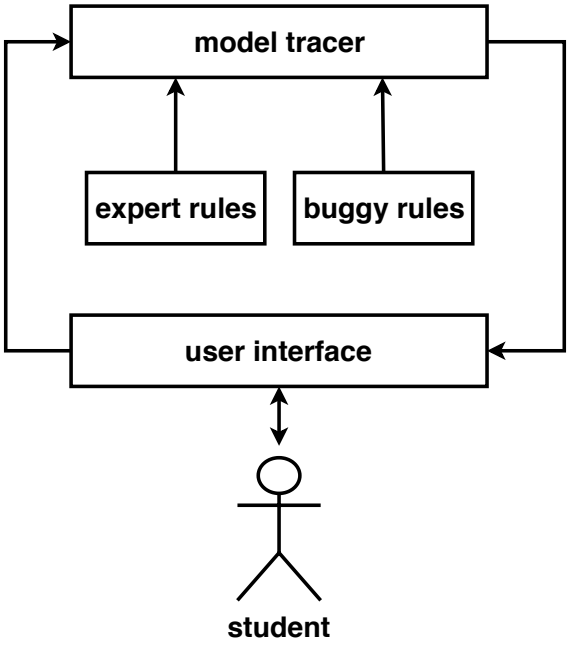

Fig. 1.3 Anderson's ITS architecture

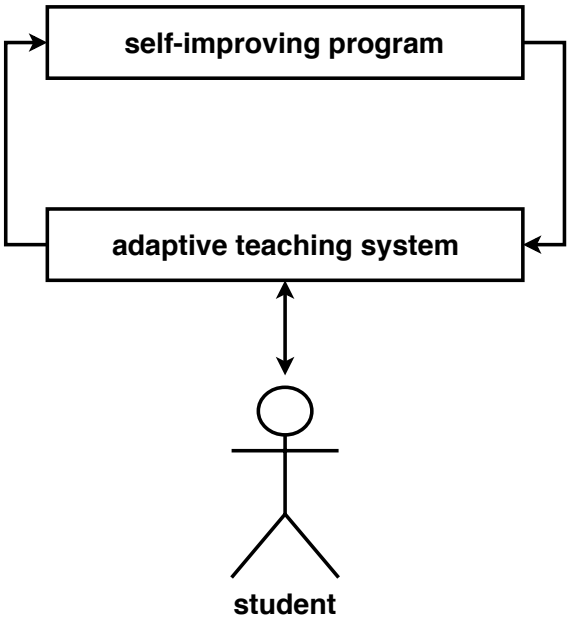

Fig. 1.4 Architecture of an adaptive tutoring system (recreated from [98])

be sorted into two categories: planning rules that describe the process of decomposing a problem into subproblems (see example I below), and operator rules that address the solution of atomic subproblems (see example II below). Buggy rules reflect common misperceptions and misconceptions (see example III below).

I IF goal is to prove that number is divisible by six
THEN prove that number is divisible by three
prove that number is divisible by two

II IF goal is to prove that number is divisible by two
and division by two has zero remainder
THEN infer that number is divisible by two

III IF number ends in 2, 4, 6, 8, or 0
THEN infer that number is divisible by four

The model tracer's goal is to retrace the student's reasoning through the application of these rules. If the trace contains buggy rules, the system concludes that the student lacks the necessary understanding of the given knowledge and it provides remediations associated with the found buggy rules. If the student input is incorrect, but the model tracer cannot find the buggy rule applied by the student, the system provides a general feedback pointing out the existence of the mistake. When several equally acceptable strategies can lead to the correct solution, the tutoring system must have rules for all of them, and must be able to map the student's reasoning to a particular strategy [114].

Model tracing is the most widely used approach to develop ITSs [87]. Their successful application is well-documented and generally their use resulted in as much as a one standard deviation improvement in student performance [65]. In [66], the introduction of the PUMP algebra tutor improved student performance by one standard deviation, similar improvement was observed in the domain of geometry and programming [5]. The Andes physics tutor improved student letter grade by a third on average [37].

1.3.3 Constraint-based modeling

Constraint-based modeling (CBM) of students was proposed by Ohlsson in [99], and it relies on detecting the student's errors to build a student model and to provide remediation. A CBM system is based on contstraints, a structure that specifies conditions that must be satisfied

for all correct solutions. Therefore, if one of the student's answers violates this condition, the system can detect that the student has made a mistake. In contrast with model tracing, CBM systems do not try to emulate the student's thinking process and do not contain a solver component that is capable of solving the problem presented to the student. Because of this, CBM systems do not require a sophisticated set of rules that can trace the student's thinking process as is the case with model tracing systems.

Formally, a constraint is an ordered pair $< C_r, C_s >$ (see example IV below), where C_r is the relevance condition that specifies when a constraint is relevant to the student's answer, and C_s is the satisfaction condition that defines a condition that should be true for all student answers. Not all constraints are relevant to all student answers, their relevance is ascertained based on the student input and additional variables integral to the presented problem. If the student's answer meets C_r but does not satisfy C_s, the system can detect that the student has arrived at an incorrect solution. Otherwise, the system lets the student proceed in solving the problem.

IV	C_r	IF	student says that number is divisible by three
	C_s	THEN	the sum of the number's digits must be divisible by three

CBM tutors were implemented for introducing students to SQL database commands [86], punctuation [81], and database modeling [121].

Chapter 2

Social robotics

Social robotics, a new field of robotics, addresses the need for socially interactive robots. The main difference between social and service robots is in their approach to human–robot interaction (HRI): while service robots don't necessarily interact with people (most often they only observe them as obstacles), social robots must interact with humans to reach their goal. Another difference is that social robots must have characteristics that make the interaction natural for humans. In this chapter we discuss questions of social robot design, social aspects of human–robot interaction, the Wizard of Oz methodology, and we present some results of our research in adapting human–robot interaction along with use cases of social robots in education and cognitive stimulation therapy from other research studies.

2.1 Design of social robots

In [32] and [34], the authors identified characteristics and capabilities such as emotion expression and perception, high-level dialogue communication, learning and recognizing models of other agents, establishing and maintaining social relationships, use of natural cues (e.g. gaze, gestures), and distinctive personality and character that are specific to social robots. It is important to note that not all social robots must have all of these qualities, although the more they have, the more natural the human–robot interaction will be.

In accordance with the above-named characteristics, [34] names key research problems in social robotics:

- design – During HRI, it is desirable that the human be put at ease and relax and so social robots often have human-like faces, support speech recognition and further functionalities that make the HRI natural to humans. Robots can be built in two ways [34]: **biologically inspired** (robots try to internally simulate social intelligence through

cognitive, behavioral, motivational, motor and perceptual systems) and **functionally designed** (the robot acts as if it was socially intelligent, but there are no internal mechanisms similar to those of biologically inspired robots).

- embodiment – A robot is embodied if there is a possibility of "mutual perturbation between system and environment" [105]. Social robots can often fit into four categories based on their level of embodiment: **anthropomorphic** (imitates human behavior, it must be able to facilitate social interaction), **zoomorphic** (imitates animals), **caricatured** (some features of the robot are highlighted to intentionally create biases), and **functional** (the robot's functionality is reflected on its appearance).

- artificial emotions – They help to facilitate HRI and provide feedback to the human about the robot's internal state. They can be expressed through speech, facial expressions, body language, and further non-verbal communication channels.

- dialogue – For successful dialogue the human and robot must share a set of symbols describing common concepts. There are three main levels of human–robot communication: **low-level**, **non-verbal**, and **natural language**.

- personality – It is believed that a robot with a personality can encourage interaction and establishment of relationship between human and robot. From the point of view of robot's having personality, we can categorize them into the following groups: **tool-like** (robots performing services), **pet/creature** (robots acting as domesticated animals), **cartoon** (robots with caricatured personalities), **artificial being** (robots with artificial – non-human – characteristics), and **human-like** (robots with human personality traits).

- human–oriented perception – In HRI it is not sufficient for robots to perceive only their environment, they must be able to correctly perceive the human they communicate with. Therefore, capabilities tied to human–oriented perception, such as people tracking, speech recognition, gesture recognition, facial expresssion assessment, and gaze tracking, are essential.

- user modeling – For natural HRI social robots must also be able to detect, recognize and predict human action. This can be done through user modeling, which enables the robot to expect and react to human behavior, understand human behavior and dialogue, and adapt the robot's behavior to the user. User models can be static or dynamic and can be built from explicitly (through questioning) or implicitly (inferred through observation) acquired data.

- socially situated learning – Social robots are not limited to a pre-defined set of possible actions but can also learn new skills and behaviors during interacting with a human. These skills are not necessarily social skills but are all useful in accomplishing the aims of HRI.
- intentionality – It is closely bound to understanding and predicting human behavior and it has three levels: **physical stance** (body movement and physical characteristics), **design stance** (design and functionality of the robot), and **intentional stance** (the robot's beliefs and desires).

2.2 Social aspects of human–robot interaction

One of the goals of social robotics is to facilitate human–robot interaction that is as natural for the human interacting partner as possible. Since the observing, understanding, and conveying of social input is an inherent part of human–human interaction, social robotic systems must also support the processing of social input to some degree. Social input comprises verbal utterances and non-verbal cues, such as voice pitch, emphasis, tone, facial emotion, mood, gestures, gaze direction, etc. Dealing with social input has a high computational cost which may present a problem for robotic systems, the use of cloud computing is therefore often beneficial (see section 4.2.4).

The processing of dialogue is the subject of research in natural language processing, but it also touches on the problem of speech-to-text transcription. In social robotics, however, transcribing and processing a human's speech is insufficient. The robot must also understand what the human says and transform this information into a representation that the robot can then use to construct its reaction. In creating the reaction, the robot can also rely on natural language processing to accomplish personalized and targeted language use from the robot. Generating speech can be done either by using a robot's implicit text-to-speech capabilities, which is provided for a number of social robot platforms, or by applying third-party implementations.

Observing and detecting non-verbal communication poses a bigger problem for social robotics, and a distinction must be made between the way humans process non-verbal cues and how robots try to imitate this recognition. In the case of robots (or any computer), it is more appropriate to talk about assessment rather than recognition, since robots do not view verbal and non-verbal communication channels as part of the same coherent communication, but as individual channels that must be processed independently. Because of this, the results of processing non-verbal cues can be seen only as approximation rather than true recognition.

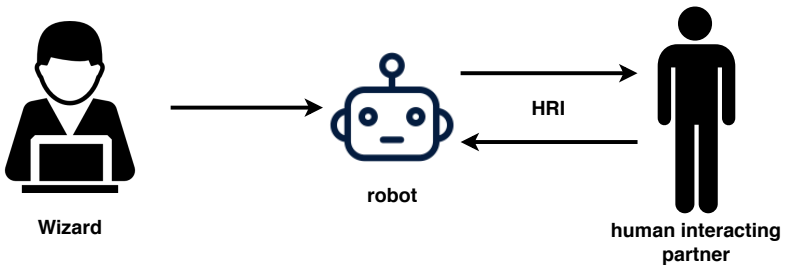

Fig. 2.1 The Wizard of Oz methodology

Non-verbal information from speech is usually detected from the transcribed text (e.g. mood based on the words being used by the human), but it can also be done by analyzing the human speaker's pitch, tone, and emphasis. In observing gaze direction, the most relevant piece of information is the amount of time the human interacting partner spends looking directly at the robot. For gestures and other aspects of body language, the acceptance of the robot by the human can be approximated. By far the most often used non-verbal cue in human–robot interaction is facial emotion, where the human's emotion is to be approximated from his or her face. This is done first through the detection of landmarks or marker points on the human's face (e.g. corners and middle of eyebrows), combining these points into expressions (e.g. furrowed eyebrow), and then mapping these expressions into emotion (e.g. anger).

2.3 Wizard of Oz

Because of the relative difficulty of processing social input, as well as the complexity of reasoning required to make an appropriate robot reaction, the Wizard of Oz (WoZ) methodology is often used in the context of human–robot interaction research. First described in [62], the Wizard of Oz refers to "a person remotely operating a robot, controlling any number of things, such as its movement, navigation, speech, gestures, etc." [107], see Figure 2.1. To the human participating in HRI, the robot seems to work and act autonomously. WoZ is necessary because of the limited capacities of robots in interacting in a socially appropriate way, and it is primarily used to test hypotheses or to simulate capabilities and functionalities not yet implemented.

Some researchers raised ethical concerns of using WoZ and social deception of the subjects of experiments [35][108], while others questioned its methodological insufficiencies [133] pointing out that a WoZ controlled robot is just a proxy of the teleoperator and the interaction is therefore more a human–human interaction via a robot than a real human–robot interaction. However, it is important to note that in a WoZ setting, it is possible for the Wizard

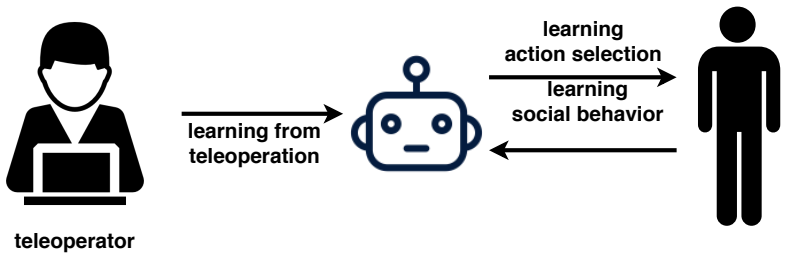

Fig. 2.2 Dimensions of adaptive human–robot interaction

to observe the interaction only through the sensors of the robot, resulting in a more accurate simulation of the robot's decision making, since the Wizard is only shown information that would also be available to the robot.

2.4 Adaptive human–robot interaction

As described in 2.2, implementing an appropriate social behavior for social robots is a computationally heavy task that can be solved by having the robot controlled by a human teleoperator, or Wizard. The WoZ methodology on the other hand has an insufficiency connected to its nature and its applicability for long-term use. Teleoperating the robot represents a high cognitive load on the Wizard which is a problem especially in a setting where the accomplishment of the goal of the interaction requires the full cognitive facility of the Wizard (e.g. cognitive stimulation therapy where the therapist should focus on the content of the therapy session and not its execution). Furthermore, teleoperation can quickly become monotonous that will have a negative impact on the quality of the interaction. To solve this problem, it is desirable to make social robotic systems adaptive, which means that social robots will adapt their behavior to the human interacting partner based on observations they make during the interaction. In this section, we describe three dimensions of adaptive human–robot interaction (as shown in Figure 2.2), namely learning from the teleoperation, learning social behavior, and learning action selection. We also present our research in using these adaptations.

2.4.1 Learning from teleoperation

Learning from teleoperation addresses the problems with the Wizard of Oz methodolgy and aims to move from full teleoperation to a lower level of teleoperation where the Wizard's role becomes that of a supervisor and the robot acts autonomously almost every time it has to select an action. This reduces the load on the Wizard regarding teleoperation, meaning

that the Wizard can focus on other aspects of the interaction that cannot be tackled by the robotic system.

The notion learning from teleoperation was first mentioned in [64] and it considers the Wizard a teacher whose behavior is the desirable (or optimal) behavior in the context of the given interaction. The robotic system's aim, then, should be to learn this behavior, so as to imitate the Wizard's action selection process, emulating the Wizard's reasoning, and reaching a higher level of autonomy. We must emphasize that in a teleoperation setting it is often still necessary for the Wizard to be present to supervise the interaction and to be able to interrupt and override the robotic system's decision if necessary. This is especially important in interactions that can have a lasting negative effect on the human participant, or with physically or mentally vulnerable human participants (e.g. therapeutic settings).

A robot system that is learning from teleoperation was implemented and tested in a cognitive stimulation therapy setting [79]. In our experiments, the social robot acted as a cognitive stimulation therapy coach in an elderly care facility, and used reinforcement learning to learn proper behavior from the Wizard in the context of cognitive games. The games consisted of the robot choosing a letter and asking the human participant to name a fruit or animal whose name started with the given letter, respectively (e.g. apple/antelope for A). If the answer was correct, the robot cheered, otherwise it gave a hint for a possible answer. Each game lasted until the subject gave a correct answer for all the letters. At the beginning of games the robot introduced itself and explained the rules of the game, and at the end it wrapped up the game and said goodbye.

The robot could choose from six types of actions:

1. easy question – naming a letter for which there are a number of common animals or fruits whose names start with the letter
2. medium difficulty question – naming a letter for which it is harder to find an animal or fruit beginning with the letter, but there are still multiple options
3. hard question – naming a letter for which it is hard to find an animal or fruit beginning with the letter, often there was just one possible answer
4. give a hint – give the participating human a short description of a possible correct answer
5. applaud – to be executed after a correct answer; six different expressions were defined
6. sorrow – to be executed after each incorrect answer; six different expressions were defined

From the point of view of learning from teleoperation, we measured and analyzed the proportion of autonomously selected correct actions from the robot. Since the game constituted a structured interaction, we expected a high rate of learning, which could be observed by comparing the percentage of autonomously selected actions approved by the teleoperator between two sessions witch each participating human. On average, the system selected actions autonomously in 73.75% of the time during the first sessions, and 84.36% during the second sessions, which represents a 10.61% average increase. The highest increase for an individual participant was 26.64%, the lowest was 5.71%. For one participant, we didn't observe an increase in the proportion of autonomously selected actions.

2.4.2 Learning social behavior

By learning social behavior we refer to the process of adapting a robotic system's behavior by observing the human interacting partner's emotional state without the need for a human teleoperator to teach the system the proper social behavior. The aims of learning social behavior is to adjust the robot's gestures, expressions, etc. to better please the human participating partner; it doesn't consider the actual content of the interaction. Learning social behavior was the subject of a research awaiting publication in cooperation with the Advanced Telecommunications Research Institute International in Japan.

The impemented robotic system considered two dimensions of the interacting human partner's state: emotion level and motivation level, both with three possible values: positive, neutral, and negative. The emotion level was determined using facial emotion assessment, while motivation was derived from the length of the human's speech. Three actions were available to the robot: short response (e.g. nodding, or a simple affirmation of listening), long response (a question that urges the human to elaborate on a topic), and topic change. The goal of the learning was to keep interaction time longer than 30 minutes, and, if possible, keep the human in a state of positive emotion and positive motivation.

Learning took place considering two aspects. First, since the robot was initially operated by a human, learning from teleoperation took place, and second, the learning algorithm also considered the state of the human interacting partner. Although when comparing the social behavior of a robot being teleoperated, and that of a robot acting autonomously displaying the learned social behavior, we did not observe a significant increase in the accuracy of the trained model (meaning that it should display the same behavior as a robot being teleoperated), the action selection process was not random, and the resulting interactions fulfilled all expectations. Interaction time increased, as well as the proportion of time when the human participant was talking because of higher motivation, and the human participants accepted

the social robot better than they accepted the teleoperated robot, as shown by an increase in the amount of time when the human participant was looking directly at the robot.

2.4.3 Learning action selection

Although the robotic system adapts its action selection process to reach a more desirable behavior in all adaptive human–robot interaction, in this subsection we specifically refer to the adaptation of the action selection with regards to the content of the interaction without the need for a human teleoperator to teach the system the proper behavior. In the context of education or cognitive stimulation therapy, for example, this would mean selecting questions and problems that keep the human participant just beyond his or her comfort zone of skills and knowledge.

We created a robotic system in which the robot plays a simple *Can you show me the picture of ...?* game with visitors of a museum where the robot names a category and the visitors have to find and show the robot an image corresponding to the named category. The images are laid out in front of the robot (see Figure 2.3). For the initial testing and evaluation we selected eight categories of robots as the subject of the game with three levels of expected difficulty: *easy* (robotic dog, drone); *medium* (domestic robot, industrial robot, mobile robot); *hard* (android, humanoid, nanobot).

We tested the hypothesis that the expected difficulty of the question along with the player's age could contribute to an adaptation of the question selection process with the aim of maximizing the number of ideal games, which are defined as ones where the player answers two questions correctly and one incorrectly (to ensure that the visitors acquire new knowledge). The system has been deployed for four months for an initial phase of data collection, for this purpose, questions were selected randomly without the use of a learning module.

Table 2.1 Player answers by robotic categories

category	**correct**	**incorrect**	**no answer**	**total**
mobile robot	52	54	28	134
industrial robot	94	18	21	133
robotic dog	106	9	26	141
humanoid	68	39	18	125
android	52	34	30	116
nanobot	71	34	16	121
domestic robot	94	27	21	142
drone	135	5	11	151

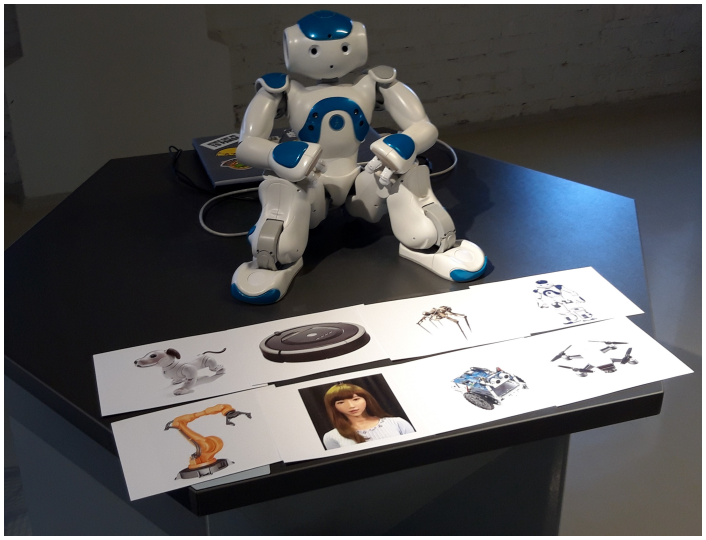

Fig. 2.3 Robotic tutor in a museum

The player's age was approximated using the Microsoft Face API facial emotion assessment service. From the 403 detected players 162 were male and 241 were female, their ages ranged between 2 and 66. However, after analysis of data we found that the service overestimates the players' ages, since the museum is mostly visited by school groups and the percentage of players under the age of 18 was only 13.9%. Therefore we must conclude that in this context, approximation of the player's age cannot be effectively used for our purposes.

Our second hypothesis, that question difficulty can be adjusted, however, was supported by the gathered data. Table 2.1 shows the number of correct, and incorrect answers for each robot category. Although the rate of correct answers does not necessarily correspond to our expectations, it is evident that players have a harder time giving the correct answer for some questions.

A learning system was successfully trained using the gathered data, which consitutes off-line learning. The learning system is currently being integrated with the robotic system for supporting learning during interaction. The adaptive robotic system will be subject to further testing and evaluation in a real-life setting, and our results await publication.

2.5 Social robots in education

The application potential of social robots in education was apparent very early on [11][70]. In addition to the general advantages of human–robot interaction, social robots present a number of further benefits to the human learner. Since robots can observe and map the human learner's skill level just like intelligent tutoring systems, they can offer a personalized learning experience while also having the additional advantage ensured by embodiment [73]. Human learners tend to react more positively toward a robot than toward an intangible computer software, even though both rely on the same algorithms [43][45]. Furthermore, in HRI, the novelty effect of robots has been observed many times and was shown to increase the level of motivation in humans. Robots can also offer valuable characteristics for a long–term learning experience, since they have no problem maintaining a consistent level of attention unlike human tutors.

In [95], the authors describe five dimensions along which research in educational robotics can be categorized:

- learning activity domain – Four broad domains are listed: robotics/computer education, science education, language learning, and cognitive development.
- learning location – Intra-curricular (formal education part of the syllabus) or extra-curricural (after school hours at school, at home or in public places). Extra-curricular activities are usually less strict and defined, and easier to set up, although they are usually restricted to one or a couple of sessions and their long-term effects are unclear.
- robot role – The robot can take on a number of roles in the learning process depending on the expectations defined by the learning domain, instructor, student, and the learning activity. In general [117] there are three categories of robots based on their role: tool (robots have a passive role and are only used as teaching aids), peer (role of co-learner), and tutor (active and autonomous role). Role choice is highly dependent on the learning task; whereas for some basic learning tasks, cooperative robots are preferred [100], for language learning tutors are clearly at advantage [111].

- robot type – Based on the level of embodiment, a progressive scale can be devised: mechanical kits with a single function, programmable electronic robotic kits (e.g. LEGO Mindstorms, Arduino), and fully embodied robots (e.g. humanoids, pet animals, toy characters). In this dimension, the age and the requirements of the students should be taken into consideration, along with the aims of the educational use.

- pedagogical theory – Three pedagogical theories are prevalent in research: constructivism [103], active learning [46], and learning by design [40]. Social constructivism [130] applies to most peer or tutor-based scenarios in robotics education.

Educational research showed that one-on-one tutoring is more effective than standard lectures and results in bigger learning gains and higher rates of information retention. Because of this fact, research in assistive social robots in education also focused on a setting with robotic tutors. Traditional methods in research either used predefined behaviors from which the robot could select, or adapt the difficulty of a class to meet the human learner's knowledge [116]. A more general approach promising more efficient robotic behavior is to learn from a human teacher teleoperating the robot and demonstrating appropriate behavior, or to learn directly from the interaction with the human learner.

Another dimension of social robot research in education focuses on improving the learning experience through the robot tutor's social behavior. Such research is based on studies done on the impact of human social behavior on learning. In general, social behavior is thought to increase the learner's interest which is expected to lead to greater learning gains [6][80], and personalized language can also lead to improved knowledge transfer [15][49].

These positive effects of social behavior were also observed in a HRI setting [72]. [124] showed that robot gestures can be used to attract student attention and improve recall. Maintaining eye contact has also been discussed as a way to engage students [54]. In [111] the authors compared neutral and socially supportive robots with the use of the latter leading to improved learning. According to [117] younger children were satisfied with robots being their peers during learning but older children saw robots more as teaching aids or tools.

On the other hand, research conducted in [63] showed that social robot tutors (see Figure 2.4) can lead to lower learning gains than asocial robot tutors, although the use of either robot resulted in significantly improved learning compared to a setting without a robot.

In [71] the authors used a social robot not in the role of the tutor but in the role of the learner. This approach is based on the learning by teaching paradigm, which engages the student in the role of a teacher and has been shown to produce motivational, meta-cognitive, and educational benefits [109]. Children who had to undergo remediations for handwriting participated in the research. Their task was to teach a "bad writer" robot to write (as shown in

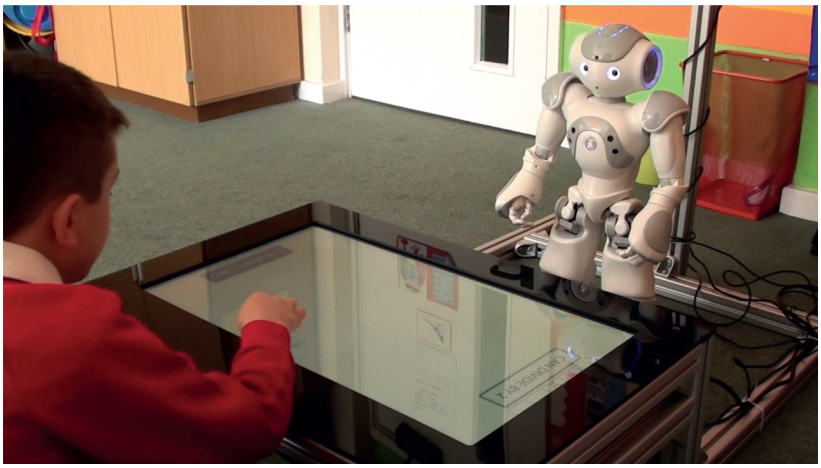

Fig. 2.4 Using a robotic tutor in an educational setting [63]

Figure 2.5). The reversed roles had positive psychological effects on the children (increased self-esteem and motivation) in addition to pedagogical effects. The robot's behavior was still adapted to the child's needs and was used to control the learning curve. The study showed that the learning by teaching paradigm can be successfully transposed to social robotics and that blending machine learning techniques with human–robot interaction allows for believable agents, leading to social commitment that induces cognitive engagement on the child's side.

A large portion of research in educational robots focuses on using robots in language teaching [10]. Robots were used to teach English in Japan [58], Korea [44], and Taiwan where researchers pointed out the fact that children are less hesitant to talk to a robot in a foreign language [18]. Research in [118] showed that social interaction is imperative in a language learning setting by conducting a survey of two robots – an emotionless humanoid and a robotic dog – over a period of four-week long home use, with both children and parents preferring the robotic dog. This conclusion reflected results similar to those obtained in [92].

In [2] a NAO robot was used to support teaching English as a foreign language in a classroom setting. The students' knowledge was assessed before and after five lessons; children learning with a robot showed superior retention rate and students learned more vocabulary. In one-on-one tutoring settings, however, research shows mixed results [134][50]. It is also important to note that language tutoring usually requires accurate speech recognition, which poses a problem for the successful use of robots [94].

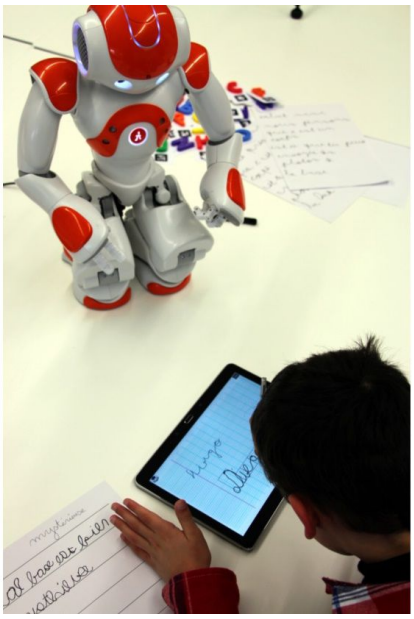

Fig. 2.5 Using a robotic peer in an educational setting [71]

2.6 Social robots in elderly care

Cognitive stimulation therapy (CST) is an intervention program and treatment effective in slowing down and mitigating the effects of mild to moderate dementia or Alzheimer's disease. Its goal is to stimulate and engage the cognitive capabilities of a human suffering from dementia in order to help them to continue to learn and to stay socially engaged. The therapy consists of a series of themed activities, such as physical games, word association, orientation, number games, and word games; and it is usually carried out in groups, but individual treatment is also possible, and it also makes a more personal and targeted therapy possible.

CST was developed in the late 1990s and early 2000s, as an evidence-based treatment [120]. Research showed that CST led to significant benefits in people's cognitive functioning. Therapy involves at least 14 sessions, typically twice weekly, and has been shown to have comparable effects to those of anti-dementia drugs. It shows the biggest positive effect in promoting language function (naming, word-finding, and comprehension) [119]. CST also improved quality of life significantly, as rated by partipicants of various research. No side-effects of CST were reported. An optimalized learning environment is essential to the success of CST, and so applications of artificial intelligence and robotics research were tested in elderly care settings.

The social robot PARO was used in anti-dementia treatment in [19]. The robot interacted with people with different levels of cognitive impairment and with their therapists. The therapy increased the subjects' activities in social interaction, and their activity levels.

In [31] a social robot was used to engage elderly users in physical exercise. The aim of the therapy was to achive helth benefits and improve quality of life, and the system used personalized interaction to achieve its aim. Two systems were compared, one with social behavior and one without, with the former one meeting a higher level of acceptance among participants.

A robotic system was used in the context of cognitive stimulation therapy in [125]. The robot played games to increase the participants' cognitive attention and slow down their cognitive decline. By recording the subjects' performance the system was able to adapt the level of difficulty for games, resulting in games tailored to the needs and capacity of each participant. A summary of our research in robotic tutors in cognitive stimulation therapy can be found in section 2.4.1 or in [79].

Chapter 3

Reinforcement learning

In section 2.4 we named three aspects of human–robot interaction adaptation: learning from teleoperation, learning social behavior, and learning action selection. These dimensions have some specific characteristics that define the problem. First, it is often desirable for the learning to take place on-line, during the interaction. Second, the robot can be considered to be an agent of the interaction that must observe the reaction of its interacting partner, the human. In this chapter, we describe reinforcement learning and its methods, a machine learning approach most suited to the problem of adaptive HRI because its representation of the problem is similar to the nature of HRI.

Reinforcement learning is one of the three main approaches of machine learning along with supervised and unsupervised learning. Both supervised and unsupervised learning require a dataset for training, but while in supervised learning the expected output for all inputs must be available, unsupervised learning is able to identify some subsets of data with the same characteristics without knowing the output. Reinforcement learning is typologically closer to supervised learning in that it considers both the input and output of a problem, however, unlike supervised learning, it does not require to know the outputs in advance, it discovers them instead during training. The classification of machine learning methods and their connection is depicted in Figure 3.1.

Reinforcement learning (RL) represents the problem as situations and then maps these situations to actions in a way that maximizes numerical rewards [123]. In a RL setting, the learning model (**actor** or **agent**) is interacting with an environment. This environment has a state and reacts to the actions undertaken by the actor. From the changes observed in the environment's state, the actor can deduct the reward of executing a given action: the higher the reward, the more appropriate the action. In many use cases, actions affect not only the immediate reward but all subsequent rewards. The sum of subsequent rewards is the **value** which represents the amount of expected reward in a given state. The actor has no explicitly

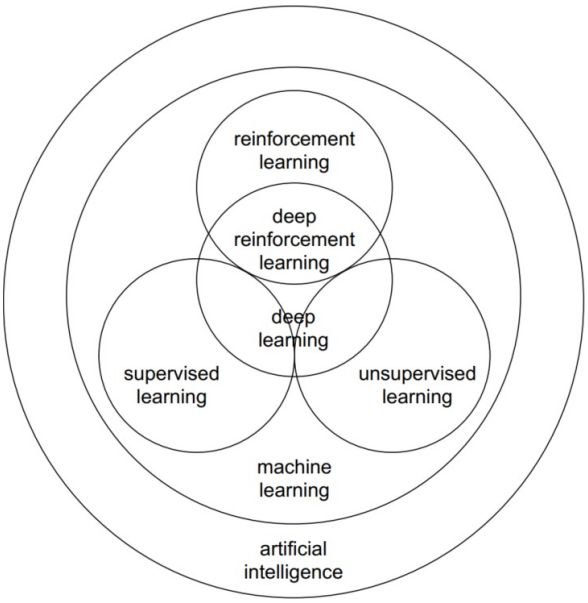

Fig. 3.1 Overview of a machine learning approaches [74]

programmed rules for action selection, it learns the correct behavior by trial and error which along with delayed reward distinguishes reinforcement learning from other machine learning methods [123].

The problem of reinforcement learning can be formalized as the optimal control of an incompletely-known Markov decision process. In such a setup, the actor is able to observe its environment and through executing actions can affect the state of the environment [123]. A RL model consists of three basic elements:

- a state space $S = \{S_1, S_2, ..., S_n\}$ that contains all possible states of the environment as described by a finite set of features
- an action space $A = \{A_1, A_2, ..., A_n\}$ that contains all actions available to the actor
- a reward function $R = r(s_t, s_{t+1}, a_t)$ that assigns rewards to state transitions where s_t is the state before executing action a_t and s_{t+1} is the subsequent state

The goal of RL is to find an optimal policy π^* which maps a state to the action yielding the highest possible reward. To reach this goal, a number of issues need to be taken into consideration.

One of the most common problems in RL is the tradeoff between **exploration** and **exploitation**. Exploration is an attempt to discover more about the world by choosing actions randomly hoping that it will lead to the discovery of a more rewarding behavior. Exploitation is using already mapped behavior to get the best results the actor has discovered. A lack of

exploration will lead to a suboptimal policy while too much exploration hinders learning as the learning model does not exploit its knowledge of the environment. Action selection that always exploits current knowledge is called greedy since it always selects the action with the highest expected reward. A more sensible approach is to use an almost greedy action selection function which acts greedily most of the time and selects the action randomly with a low probability ε (ε-greediness).

Two further parameters that need to be considered for RL are **learning rate** (α) and **discount factor** (γ). Learning rate or step size defines the extent to which newly acquired experience affects the overall behavior of the RL model. A learning rate $\alpha = 1$ will cause the model to consider only recent information and not remember previous action selections (ideal for deterministic environments), while with $\alpha = 0$ the model is not learning (full exploitation). For stochastic environments, the learning rate must be sufficiently big at the beginning of learning and must converge to zero with time so that the model exploits its past experience.

Discount factor is used to create an appropriate balance between immediate rewards and long-term values. A discount factor $\gamma = 0$ will make the actor consider only the immediate reward, while $\gamma = 1$ makes the actor strive for long-term high value. It is important to note that there is a theoretic possibility that with a discount factor of 1 or higher, the action values might diverge and that a small discount factor at the beginning of learning will accelerate learning. Therefore, it is desirable for the discount factor to be low at the start of the learning and gradually increase toward a value lower than 1.

3.1 Model-free and model-based reinforcement learning

There are two major categories of RL that differ in their approach to considering the environment and state transitions: model-free algorithms and model-based algorithms [56].

Model-free algorithms learn a value function and policy from interaction with the agent. This means that at the end of learning, the agent knows how to act but does not have an explicit representation of the environment (as if playing chess without knowing the rules, and only knowing which moves will result in a positive outcome).

Model-based algorithms first construct a model approximating the environment's behavior by replicating state transitions and outcomes, and then they search this model to find appropriate actions. This means that model-based algorithms do learn state transitions and know what to expect from the environment if a certain action is executed, while model-free algorithms only make these decisions based on the expected rewards for a given action in a given state.

Apart from their different approaches to learning, these two types of algorithms are suited for different learning scenarios. Model-free methods are appropriate for use cases where collecting data is not a problem and their algorithms include Monte Carlo methods (relying on random sampling), and temporal difference methods (bootstrapping from estimates of the value function). Model-based methods are better suited for domains where collecting data is expensive or otherwise problematic, and their algorithms include dynamic programming (breaking down a problem to sub-problems).

3.2 Feature selection

Effective state representation is essential for successful learning through reinforcement. Ideally, the learning algorithm would consider only features that describe the state of the environment in sufficient detail but is also complex enough not to simplify the environment too much. The importance of careful feature selection becomes apparent when considering the number of possible states an environment has based on the number of features we select. The selection of a new feature multiplies the number of possible states by the number of values this new feature can take. Therefore, the number of selected features must be considered carefully as a number too small might lead to an overgeneralized state of the environment, while selecting too many features will hinder learning by being too specific in modeling the environment.

Feature selection is problematic because human intuition alone might not always result in the selection of the most appropriate features for RL. At the beginning of the process, the best is to consider all features that might affect the learning process and then use algorithmic feature selection methods to identify the ones that contribute to the action selection process the most.

In [22], the authors listed twelve feature selection methods in the use case of pedagogical systems. In their methodology, they first considered all possible features and then determined the maximum number of features ($\hat{m}$) in a way that ensured that all states are represented in the training data but large enough not to lose information necessary to making good system decisions.

There is a large variety of feature selection methods from which we will describe four categories here:

- random feature selection methods – used mostly as benchmarks for other feature selection methods. They select features in a random manner from a set of possible features ω.

- RL-based feature selection methods – based on RL metrics such as Expected Cumulative Reward (ECR), lower-bound (worst-case scenario for acquired long-term value), upper-bound (best-case scenario for acquired long-term value) or hedge ($Hedge = \frac{ECR}{UpperBound - LowerBound}$). In [21], it was shown that for $\hat{m} \leq 3$, ECR is the best metric, while for $4 \leq \hat{m} \leq 6$, upper-bound performed the best for pedagogical systems. This is due to the fact that in an educational scenario, the student will most likely not lose information, hence there is no real chance of a negative reward.

- PCA-based feature selection method – in a set of features where features are correlated, it might come to redundant feature selection. To tackle this problem, it is possible to use Principal Component Analysis [55] for which the features must first be normalized. From the normalized features, the principal components and their eigenvalues are generated. From such a list, we can select only the components with an eigenvalue of 1 to be considered for feature selection (feature set ω_{PCA}).

- hybrid methods – a combination of the above-listed methods can also be used for feature selection. A possible combination could be to first apply PCA to the set of features ω to get rid of redundant features and then apply RL-based or random feature selection to ω_{PCA}.

3.3 Reinforcement learning methods

In the following subsections we describe selected methods of reinforcement learning. These methods can be categorized into value-based and policy-based methods (see Figure 3.2). Value-based methods learn the value function V^{π} during training; they derive their policies from this function. Policy-based methods on the other hand learn the optimal policy π^* directly, or they approximate it if the optimal policy isn't available. We first describe Q-learning, a value-based method, then we turn to deep reinforcement learning, and end the chapter with the consideration of hybrid reinforcement learning methods.

3.3.1 Q-learning

Q-learning is a model-free RL technique that finds an optimal policy π^* for any finite Markov decision process as proved in [84]. Q-learning uses the Q-function to reach an optimal policy. The Q-function calculates a reward for any given state-action combination ($Q : S \times A \rightarrow R$) according to the value iteration update shown in Equation 3.1.

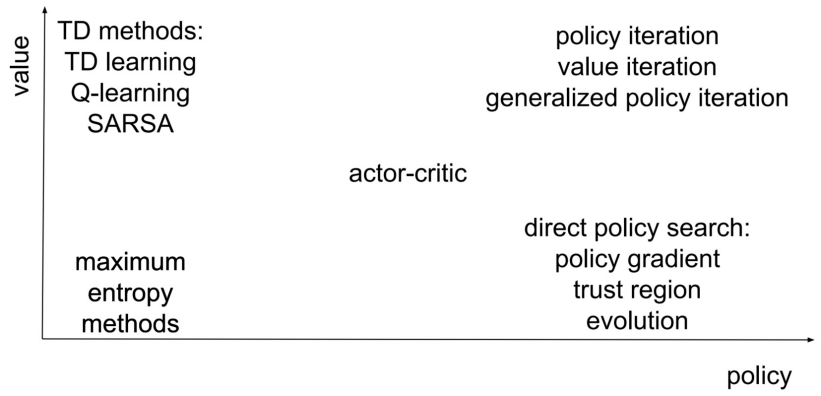

Fig. 3.2 Value- and policy-based reinforcement learning methods [74]

$$Q(s_t,a_t) \leftarrow (1-\alpha)\cdot Q(s_t,a_t) + \alpha\cdot(r_t + \gamma\cdot \max_a Q(s_{t+1},a)) \tag{3.1}$$

where

- $Q(s_t,a_t)$ is the Q-value for the state-action pair at timestep t,
- α is the learning factor,
- r_t is the immediate reward acquired after executing action a_t in state s_t,
- γ is the discount factor,
- $\max_a Q(s_{t+1},a)$ is the maximal expected Q-value for state s_{t+1}.

Training of the model consists of multiple episodes with one episode lasting until state s_{t+1} is a terminal state. The Q-values for terminal states are never updated but set for the reward observable in the given state. The learned Q-values are represented in a Q-table that consists of two sides: the left side describes the state and the right side consists of the expected Q-values for executing an action in the given state, the update of these Q-values constitutes the learning process (compare Table 3.1 (a) and (b)).

A number of further RL algorithms were developed based on Q-learning from which we mention two here. Double Q-learning targets an inherent weakness of Q-learning, namely that in some stochastic environments the algorithm overestimates action values because of a positive bias [47]. This is caused by the Q-learning algorithm using the maximum action value as an approximation for the maximum expected action value. Double Q-learning introduces a double estimator that might underestimate the action value but is still convergent

Table 3.1 Q-table before (a) and after (b) training. The model represents an actor moving in a simple 2×2 maze with the end state at position [1, 1]. Action names: N – move north, E – move east, S – move south, W – move west. Reward function: 10 if actor reaches position [1, 1], -1 otherwise.

(a)

state		action			
x	**y**	**N**	**E**	**S**	**W**
0	0	0	0	0	0
0	1	0	0	0	0
1	0	0	0	0	0
1	1	0	0	0	0

(b)

state		action			
x	**y**	**N**	**E**	**S**	**W**
0	0	6.19	7.99	7.99	6.19
0	1	6.19	10.0	7.99	7.99
1	0	7.99	7.99	10.0	6.19
1	1	6.19	10.0	10.0	7.99

to the optimal policy. In this setup, two different policies are used to estimate the action value and to select the next action.

While Q-learning and double Q-learning are off-policy algorithms, meaning that they update Q-values considering action selection with the optimal policy, another alternative, SARSA, is on-policy. SARSA was proposed in [110] and its name reflects the differences in the Q update function compared to traditional Q-learning. The SARSA update function does not consider the optimal action value (the highest value) but the one that the algorithm would actually choose in the next state, as shown in Equation 3.2.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot (r_t + \gamma \cdot Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \tag{3.2}$$

All of these algorithms suffer from another inherent limitation, that is, learning is effective only when the features describing the various states take discrete values. Otherwise, the state space would be infinitely large and the policy deduced from the Q-table would not converge to an optimal one.

3.3.2 Deep Q-learning

Neural networks were combined with reinforcement learning already in the 1990s when one of the most well-known successful application was **TD-gammon**, a blackgammon-playing program that reached super-human level of play thanks to reinforcement learning and self-play [126]. TD-gammon used a model-free RL algorithm based on Q-learning, and a multi-layer perceptron with one hidden layer to approximate the value function $V(s)$ instead of the action-value function $Q(s,a)$ as compared to Q learning. However, further attempts to repeat TD-gammon's success with other games such as chess, Go and checkers were not

fruitful. This resulted in a more systematic analysis of combining model-free RL algorithms with non-linear function approximators. In [127], it was shown that such a Q-network could diverge because of the sequential nature of observations where later observations have a larger effect on the policy. This issue was partially addressed by gradient temporal-difference methods which are proven to converge [14][78].

The most recent breakthrough in deep reinforcement learning was presented in [89] where the authors' aim was to connect a RL algorithm to a deep neural network operating directly on RGB images. The model was trained using experience replay [75] so that the observation is saved at each time-step ($e_t = (s_t, a_t, r_t, s_{t+1})$) creating a dataset $D = e_1, e_2, ..., e_N$. This dataset is then pooled over training episodes into a replay memory. Finally, Q-learning updates are carried out through minibatch updates over samples of experience ($e \subset D$) selected randomly from the stored observation samples. Such an approach results in a new algorithm, which the authors named deep Q-learning (see Algorithm 1).

Initialize replay memory D to capacity N;
Initialize action-value function Q with random weights;
for *episode = 1, M* **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$;
  for *t = 1, T* **do**
    With probability ε select random action a_t;
    otherwise select $a_t = max_a Q^*(\phi(s_t), a, \theta)$;
    Execute action a_t in emulator and observe reward r_t and image x_{t+1};
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$;
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D;
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D;
    Set $y_j = \begin{cases} r_j & for\ terminal\ \phi_{j+1} \\ r_j + \gamma max_{a'} Q(\phi_{j+1}, a'; \theta) & for\ non-terminal\ \phi_{j+1} \end{cases}$;
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$;
  end
end

Algorithm 1: Deep Q-learning algorithm [89]

Using this algorithm, the neural network initialized with random weights will be trained to represent the action-value function $Q(s_t, a_t)$ where the output nodes represent the Q-values calculated for action a_t in state s_t. Although in the original paper the authors used raw images as input (state representation) for the neural network it is possible to use the values of the features describing the state as input to the network. In such a scenario, the number of input nodes is equal to the number of features. The number of output nodes is equal to the number

of actions available to the actor. The gradient descent step is performed using the gradient shown in Equation 3.3

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot);s'\sim\varepsilon}[(r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i))\nabla_{\theta_i} Q(s,a;\theta_i)], \tag{3.3}$$

where

- θ is the set of weights of the neural network
- $Q(s,a;\theta)$ is the value function
- $\nabla_{\theta_i} Q$ is the gradient of the value function
- $L_i(\theta_i)$ is the loss function
- $\nabla_{\theta_i} L_i(\theta_i)$ is the gradient of the loss function

This model shows a number of advantages over standard online Q-learning. First, greater data efficiency is accomplished since each step of experience is potentially used in the weight updates. The algorithm also addresses the inefficiency of learning from consecutive samples that are strongly correlated. By randomizing the samples, such correlations are broken. Third, experience replay also minimizes the effect of the current parameters since the behavior distribution is averaged over many of its previous states [89]. It is important to note, however, that sampling is completely randomized and done using uniform distribution from the dataset D. A sophisticated sampling strategy that targets experiences from which the model can learn the most might lead to more effective learning, such as in [91]. One more advantage of deep reinforcement learning is that it is more resilient to non-discrete feature values, given that states with similar feature values are similar.

The deep reinforcement learning algorithm is model-free, it uses samples from direct communication with the environment without constructing an estimate of the environment. Similarly to the standard Q-learning, it is also off-policy, and uses an ε-greedy strategy to ensure adequate exploration of the state space. Apart from ε, the deep Q-learning model has two further main hyperparameters: learning rate (α) and discount factor (γ). In practice, ε has a very high value (near to one) at the start of the training process and is then decreased until it reaches its final value that ensures a sufficient proportion of exploration.

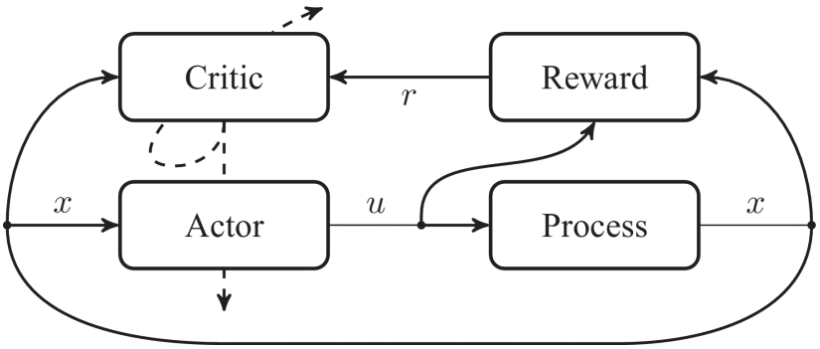

Fig. 3.3 Schematic overview of an actor-critic algorithm [42]

3.3.3 Actor-critic methods

There are two main approaches to reinforcement learning: value-based and policy-based. Q-learning and deep Q-learning are both value-based, since they calculate the expected value for executing an action in a given state to determine whether the action is optimal. Policy-based methods, on the other hand, do not use a value function to optimize the policy, an approach most suitable for continuous or stochastic action spaces, although its drawback is that it requires a good score function to assess how good a policy is. **Actor-critic methods** [9] combine these two approaches to train a RL model.

Actor-critic methods use two models to train the proper behavior: the **actor** is responsible for producing the actions through a policy function $\pi(s,a,\theta)$, while the **critic**'s role is to evaluate the current policy determined by the actor through a value function $\hat{q}(s,a,w)$. This evaluation is often done by temporal difference [122], least-squares temporal difference [16], or residual gradients [7]. The critic approximates the value function based on the samples gathered from observations. This value function is then used to update the actor's policy parameters to improve performance through proper action selection. Convergence is ensured by a small step size in the policy gradient, meaning that a change in the value function results in only a small change in the policy [8].

The interaction between the actor and critic is shown in Figure 3.3. Given the current state (x), the actor generates a control input (u). The reward acquired from the change in the environment's change (r) is processed by the critic, which uses the reward to evaluate the

quality of the current policy by adapting the value function estimate. The actor is updated only after a few evaluation steps performed by the critic (as indicated by the dashed line in Figure 3.3). The actor and critic networks are updated using the gradients showed in Equations 3.4 and 3.5 respectively.

$$\Delta\theta = \alpha \, \nabla_\theta (log \, \pi_\theta(s,a)) \, \hat{q}_w(s,a) \tag{3.4}$$

$$\Delta w = \beta (R(s,a) + \gamma \hat{q}_w(s_{t+1}, a_{t+1}) - \hat{q}_w(s_t, a_t)) \, \nabla_w \hat{q}_w(s_t, a_t), \tag{3.5}$$

where

- α and β are the learning rates applied to the policy and value functions
- $\hat{q}_w(s,a)$ is the estimated action value
- $R(s,a) + \gamma \hat{q}_w(s_{t+1}, a_{t+1}) - \hat{q}_w(s_t, a_t)$ is the temporal difference error
- $\nabla_w \hat{q}_w(s_t, a_t)$ is the gradient of the value function

To avoid high variability of value-based methods, the advantage function $A(s_t, a_t)$ can be used instead of the standard value function. The advantage function (see Equation 3.6) reveals the improvement accomplished by taking one action compared to taking other actions in the same state ($V(s_t)$). If the value of $A(s_t, a_t) > 0$, it means that the given action is better than the average expected value in the given state and so the gradient is pushed in that direction. $A(s_t, a_t) < 0$ indicates that the action does worse than the average value in the given state and so the gradient is pushed in the opposite direction.

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t) \tag{3.6}$$

By substituting $r_t + \gamma V(s_{t+1})$ for $Q(s_t, a_t)$, we get Equation 3.7 representing the temporal difference error which can be used as a good enough approximation of the advantage function.

$$A(s_t, a_t) = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{3.7}$$

There are two different strategies for implementing an actor-critic agent:

- **Advantage Actor-Critic** (A2C) – in this setup, there is a global network that is being updated by multiple instances of agents (workers) synchronously. The workers first finish their training through interacting with their own copy of the environment. Then, the gradients calculated by the individual workers are averaged and this average is used to update the global network. A2C has been shown to more efficiently utilize GPUs and work better with large batch sizes while achieving the same or better performance than A3C [136].

- **Asynchronous Advantage Actor-Critic** (A3C) [88] – in this setup, there is a global network that is being updated by multiple instances of agents asynchronously. The agents interact with their own version of the environment in parallel and without affecting each other's experiences, resulting in a speedup in the speed of training. Since the agents copy the global network parameters independently, it is possible that some agents use different versions of the policy which will make the aggregated update sub-optimal.

3.3.4 Recurrent reinforcement learning

Recurrent reinforcement learning is a combination of two neural networks: deep Q-networks, and recurrent neural networks. Since recurrent neural networks have their own memory, and can model time series, recurrent reinforcement learning models are able to select actions based on the previous steps and actions selected. Therefore, they are a suitable model for any problem where action selection has a lasting effect, e.g. in human–robot interaction.

Formally, networks without a memory will observe time series as a Partially-Observable Markov Decision Process, which describe an environment that has incomplete and/or noisy state information. Deep recurrent Q-networks were first described in [48] in the context of game play. The input of the network was screenshots from classic Atari games. After convolution, an LSTM layer was used as a model for remembering temporal change, before a regular deep Q-network would select actions (see architecture in Figure 3.4). An evaluation of the recurrent network showed better performance than standard deep Q-networks, but the recurrent nature of the network raised a technical issue when training the model.

In section 3.3.2 we described the advantages of experience replay that are still desired for a recurrent deep Q-network (RDQN). However, random sampling of past experience is not viable for a recurrent network because the whole time series must be considered when training the model. Therefore, two options are available when training a RDQN. With bootstrapped sequential updates the model remembers whole episodes that are then selected randomly from the replay memory, the LSTM's hidden state (or memory) is carried forward

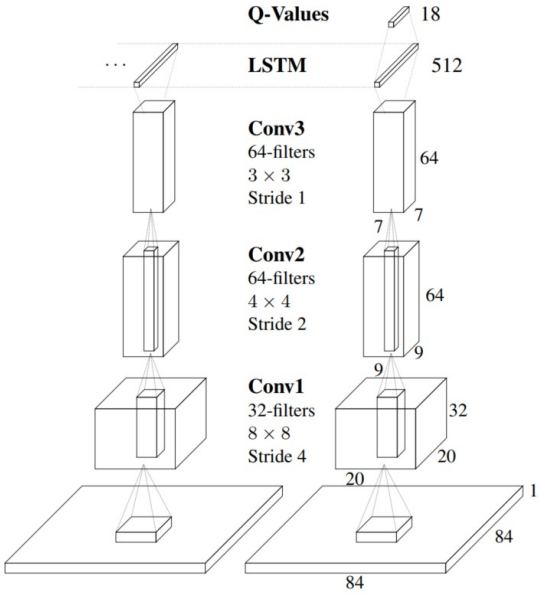

Fig. 3.4 Architecture of a deep recurrent Q-network [48]

throughout the episode. In bootstrapped random updates, episodes are selected randomly, but not the entire episode is replayed, instead just a part of it, starting at a random time step. The LSTM's hidden state is zeroed at the start of each update. Sequential updates are better for training the LSTM, but they violate random sampling policy. Random updates fulfil the requirements for random sampling, but are inefficient in training the LSTM because of the zeroing of the hidden state.

3.3.5 Fuzzy reinforcement learning

Another hybrid reinforcement learning method is fuzzy reinforcement learning, which combines fuzzy logic with a reinforcement learning network. One of the most well-known implementations is the Generalized Approximate Reasoning based Intelligent Control (GARIC) structure (see Figure 3.5) [12].

The GARIC architecture has three main components: the Action Selection Network (ASN), the Action Evaluation Network (AEN), and the Stochastic Action Modifier (SAM). ASN maps its input, a state vector, into a recommended action F, using fuzzy inference. AEN can be implemented as a neural network or a fuzzy system, and it has two inputs, a state vector, and a failure signal, which it combines to calculate a scalar score indicating the desirability of the state. The scalar score is later used to calculate an internal reward r'. SAM considers the outputs of ASN and AEN, the proposed action F and the internal reward r', to reach a final selected action F' that is than applied onto the physical system or environment.

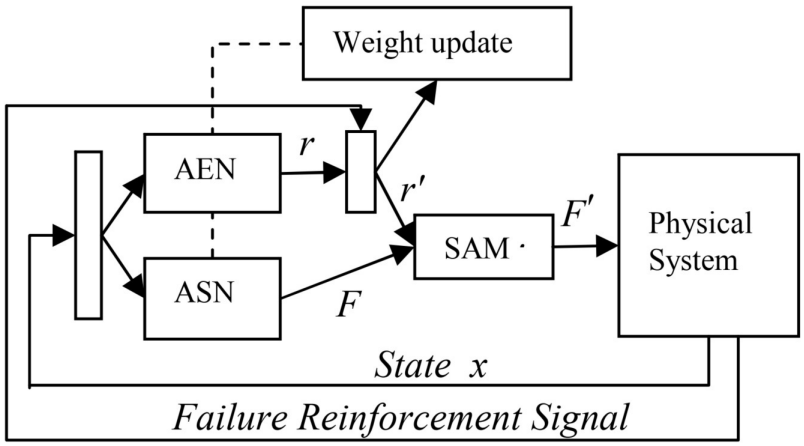

Fig. 3.5 GARIC structure [12]

Training in GARIC is done through updating the ASN and the AEN. If the AEN is implemented as a neural network, its weights are updated, if it is implemented as a fuzzy system, the fuzzy parameters are adjusted. Either update takes place using temporal difference learning method (hence the combination of fuzzy systems and reinforcement learning). In the ASN, the parameters describing the fuzzy membership function are updated through the gradient descent approach.

In [112] the authors present a solution similar to GARIC, which combines reinforcement learning with a recursive fuzzy algorithm, the Active Learning Method, which expresses a multi-input-single-output system as a fuzzy combination. The input-output relations, mapping the states into actions in the case of reinforcement learning, are modeled for each input individually and are then combined to get the overall system model. This modeling is done through a fuzzy interpolation method which derives a smooth curve among multiple data points by applying a three-dimensional membership function. The value of this membership function represents the belief for the data points and their neighbors. The system can learn with or without a predefined fuzzy system by leveraging random actions to explore the unknown system.

3.3.6 Inverse reinforcement learning

Inverse reinforcement learning (IRL) tries to extract a reward function from an observed optimal behavior, exactly opposite to other reinforcement learning methods. This approach is especially suitable for modeling human behavior, learning from demonstration, and learning from teleoperation, problems that can be considered apprenticeship (or imitation) learning [96]. When solving a problem with IRL, a key issue is degeneracy, meaning that there are

multiple reward functions for which the observed policy is optimal. The need for inverse reinforcement learning arises from multi-parameter reward functions, where the relative weights of each parameter has to be considered along with the function itself. Algorithms of IRL are also suitable for problems where defining a reward function is not possible due to the ambiguous nature of the desired goal, for example ensuring the "quality" of human–robot interaction through the robot's behavior.

There are two main types of IRL algorithms: first, when the state space is finite, the model is known, and the complete policy has been observed. In such a case, it is easy to find a set of reward functions for which the policy will be optimal (for requirements and proof please refer to [96]), choosing a reward function from this set is not so straightforward, even though it is desirable, since some reward functions can be seen as less meaningful to the problem than others. Reward function selection is possible by adding some requirements not necessarily inherent to the IRL problem, namely, that the observed policy is uniquely optimal for the reward function, and that any deviation from the observed policy is penalized highly. Reward functions with smaller rewards are also often preferable to more robust and complex reward functions. For an infinite state space, another problem arises, since the possible reward functions cannot be checked successfully, or only at a high computational cost. Therefore, a large but finite set of states is sampled, upon which it is possible to evaluate the reward functions [96].

Chapter 4

Cloud computing and cloud robotics

The National Institute of Standards and Technology defines cloud computing as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction" [83]. This definition also lists five essential characteristics, three service models, and four deployment models of cloud computing.

The five characteristics of cloud computing [83] are:

- on-demand self-service – the user can provision computing capabilities (e.g. server time, storage) as needed and does not need human interaction with a cloud service provider to do so
- broad network access – computing capabilities are available over the network (usually the Internet) and are accessible through standard mechanisms from various devices and platforms
- resource pooling – the provider's computing resources are available to multiple users using a multi-tenant model with physical and virtual resources dynamically assigned and reassigned according to consumer demand
- rapid elasticitiy – computing capabilities can be elastically provisioned and released to scale rapidly outward and inward
- measured services – cloud systems automatically control and optimize resource use.

The three cloud service models [83] are:

- Software as a Service (SaaS) – the user is able to use the cloud provider's applications running on a cloud infrastructure

- Platform as a Service (PaaS) – the user is able to deploy onto the cloud infrastructure their own applications using programming languages, libraries, services, and tools supported by the provider
- Infrastructure as a Service (Iaas) – the user can provision processing, storage, networks, and other computer resources where she can run arbitrary software. The user does not manage or control the cloud infrastructure but has control over operating systems, storage, and deployed applications.

Cloud deployment models describe how the cloud is operated and how the user has access to the service resources. [83] defines four deployment models:

- private cloud – the cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers
- community cloud – the cloud infrastructure is provisioned for exclusive use by a community of consumers from different organizations that share some concerns
- public cloud – the cloud infrastructure is accessible by the general public
- hybrid cloud – the cloud infrastructure is a composition of two or more distinct cloud infrastructures that remain unique entities and are deployed according to any of the above-mentioned deployment models.

4.1 Cloud robotics

Since robots have limited computational, storage, and battery capacities, the concept of merging them with more robust computing resources was considered early on. The motivation behind such a system is that, by offloading computationally heavy tasks from the robot to dedicated hardware accessible through some sort of network, it is possible to enhance the robot's capabilities without having an effect on its design (for a general architecture see Figure 4.1). There is only one requirement for the robot, that is, it must be able to communicate with the dedicated hardware.

The idea of a hybrid robotic system with remote dedicated hardware and a robot stems from the mid-90s and originated with the work of Inaba [53], where the robots had a radio-linked remote brain still in the vicinity of the robot. The advent of the world wide web brought about online web robotics. The earliest works focused on teleoperation via the Internet [39], on networked robots with wireless sensor networks, and on networked control systems [68]. The benefits of distributed processing in networked robotics along with new

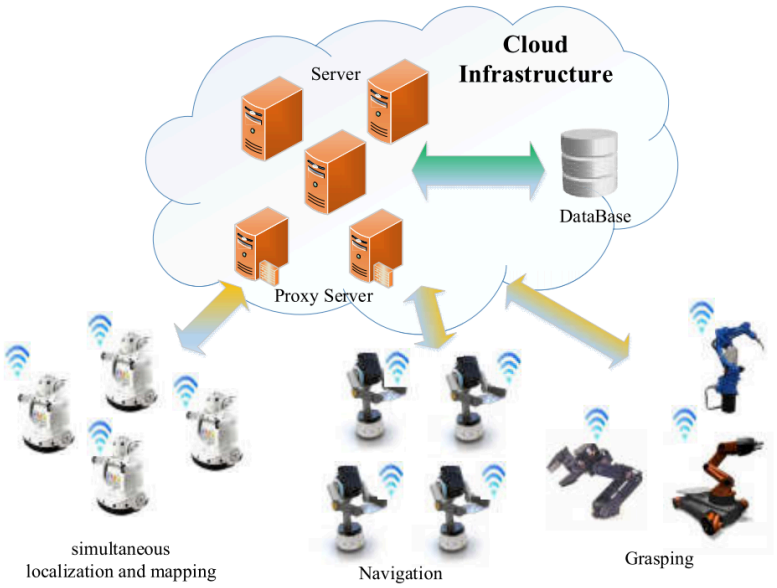

Fig. 4.1 General architecture of cloud robotics systems with the most common applications listed [131]

challenges were first described by McKee in [82]. The term “cloud-enabled robots” or “cloud robotics” was coined by James Kuffner in 2010 [67].

4.2 Advantages of cloud robotics

The growth in the complexity of problems robots had to face in research and application exposed an inherent limitation. Solving computationally heavy problems is only possible with the necessary hardware which results in complex robotic builds that might be counter-productive to the robot’s task. A possible solution is to use dedicated servers which alleviate the load on robots. This realization fueled research on the use of cloud computing in robotics with the goal to lower requirements on physical robots. In the following subsections we discuss some of the advantages of cloud robotics that can be organized into four overlapping categories: offloading computation, collective learning, knowledge-base on the cloud, and public solutions.

4.2.1 Offloading computation

Cloud computing makes it possible to execute complex computations and use robots with high-performance computing applications. This enables simpler robotic system designs and robot bodies which improve the maintainability of the robot. Furthermore, cloud computing

can speed up computationally-intensive robotics and automation systems applications [60][1]. Rapyuta, the RoboEarth cloud engine, allows computationally expensive algorithms to run in the cloud [52]. In [129] cloud computing was used for sample-based statistical motion planning. Cloud-based sampling can also support grasping [61]. The Cloud Enabled Robotics System [76] allows robots to form a local networked system and offload heavy computation tasks to the server. Cloud computing makes mapping, and video and image analysis easier [90][97].

Offloading computational tasks to the cloud also has its new challenges, such as network latency and quality of service [60]. Remote task execution often requires the transfer of real-time sensitive data, what might be hindered by network delays [131]. Therefore, an offloading of tasks more related to the robot's immediate actions, such as motion, is still not plausible. Methods used for cloud robotics must also be designed to degrade gracefully when cloud resources aren't available [60].

4.2.2 Collective learning

Multi-robot systems are an essential field of robotics, and cloud computing provides a suitable platform for networked robots to share data, process information and facilitate machine learning [106]. Collective robot learning can enhance the capabilities of robots with more limited computational resources[41], and improve the overall accuracy of the system [131].

The "Lightning" framework uses cloud computing for parallel planning and trajectory adjustment [13]. The Ubiquitous Networked Robot Platform manages distributed task execution and supervision [57] by abstracting away from the robotic hardware and offering a generic interface. The MyRobots project envisions a social network for robots to support collaboration. A small scale cloud infrastructure for information sharing among networked robots was presented in [128]. Communication protocols facilitating task offloading and information sharing were proposed in [51]. Collective learning environments for ubiquitous robots were described in [23] and [113]. The Kiva automated Materials Handling System supported collective learning for mobile robots moving packages in warehouses [24].

4.2.3 Knowledge-base on the cloud

Most robots are equipped with cameras and sensors that collect data supporting decision making but the volume of these data makes it impossible to manage or share them among multiple robots using only the robots' onboard capacities [106]. Large sets of data such as images, videos, maps, sensor networks, etc. can further facilitate machine learning in

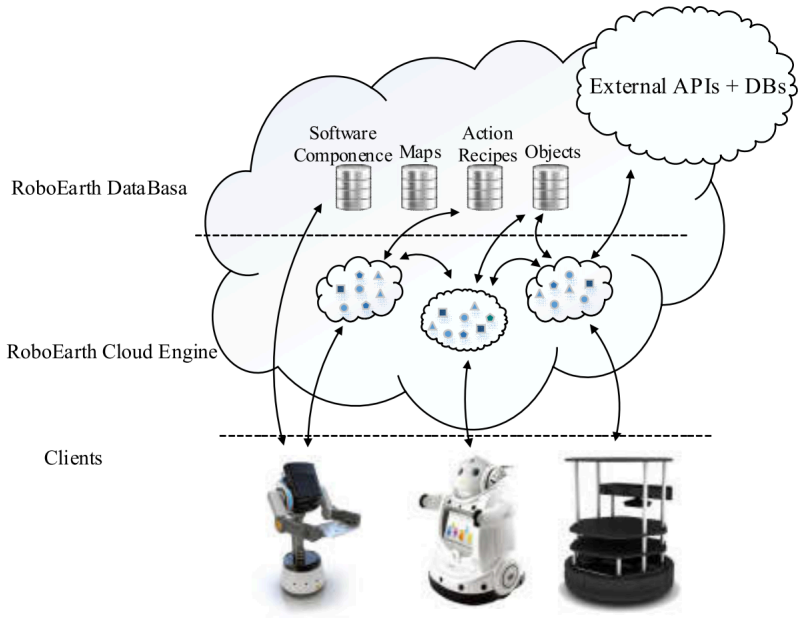

Fig. 4.2 Architecture of RoboEarth [131]

robots. This has been demonstrated mainly in the context of computer vision [60] (object and scene recognition), path planning, and grasping [93]. Google Goggles, a free image recognition service for mobile devices, has been incorporated into a cloud-based robot grasping system [59]. RoboEarth (architecture shown in Figure 4.2) is also used to gather and provide technical support for robotic navigation and grasping [131].

Social robotics can also benefit from data sharing on the cloud. This has advantages primarily in the personalization of interaction, where data gathered about a person might be too big to be stored long-term on the robot's memory without negatively affecting the robot's performance.

Although big data on the cloud can hugely improve robotic applications, it also poses new challenges for guaranteeing the quality of datasets and querying data. Datasets collected from distributed sources often contain erroneous, duplicated, or corrupted, so-called dirty data that can negatively influence data processing and machine learning. Sampling algorithms can provide approximations to keep running time acceptable. Another challenge is caused by the variety in the representation of data [60] due to the lack of standards [104]. Keeping data in the cloud further raises security concerns especially in the case of sensitive data gathered in private or corporate settings [132].

4.2.4 Public solutions

The development of cloud robotics facilitated the use of open-source or public solutions for robotic applications. Open-source licenses allow the improvement of the code reuse rate and development efficiency [131] and they provide access to datasets, benchmarks, and simulation tools [60]. The best-known systems for supporting open source in cloud robotics are the Robot Operating System (ROS) and RoboEarth. ROS's core part was designed at the Willow Garage research laboratory and it provides basic tools for distributed computing [131]. A bigger share of solutions is developed and maintained by the international ROS community and includes algorithms, frameworks and hardware drivers. ROS is slowly becoming a standard for robot developers [60]. RoboEarth is more directed at collaborative solutions for multi-robot systems, providing a network database system that is updated dynamically by robots around the world [131].

Besides open source solutions that can be integrated in the working of robotic systems, there are a number of open source simulation libraries available for speeding up the development of new systems. The overall availability of datasets and cloud services can also lead to the improvement of robotic system intelligence. While the growing rate of sharing ready-to-use solutions can facilitate the development of cloud robotics, open source solutions present the same problems as shared datasets, since the quality of code must be guaranteed.

Cloud services at the level of SaaS can also be used in robotic applications to enhance robotic behavior, especially in social robotics where most of the tasks expected to be executed by the robot are computationally heavy. Social input, such as gestures, emotions, and affect are an integral part of human–human interaction and they provide a communication channel universal across multiple cultures. The observation and processing of these non-verbal cues is natural for humans but are hard to replicate in machines. However, they are essential in HRI since an incorrect assessment of the human's emotional state could lead to an inappropriate response from the robot.

In recent years cloud-based emotion assessment services were made available for inclusion in user applications, such as Google Vision API, Microsoft Face API, Amazon Rekognition, Affectiva Emotion, and Sightcorp F.A.C.E. API. Information about a human's emotional state can also be obtained from speech by using speech-to-text methods and processing the transcribed text independently. A number of speech recognition services are Google Speech API, Bing Speech API, API.AI, Speechmatics, and Vocapia Speech to Text API, with further services focusing on specific languages. Emotion assessment analysis is possible with tools like IBM Watson's Tone Analyzer, Receptiviti Natural Language Personality Analytics API, Alchemy API, and Bitext Text Analysis API. Although the two-step speech analysis method is much more widely used, APIs recognizing emotions directly in

speech are also available, including Good Vibrations, Vokaturi, and Affectiva's Emotion API Speech.

4.3 Cloud robotics for adaptation

Due to its high level of availability, cloud computing provides a suitable framework for the adaptation and personalization of human–robot interaction. In this section we look at the advantages of cloud computing, namely offloading computation, knowledge-base on the cloud, and collective learning, with a specific emphasis on personalization.

Since adaptive human–robot interaction requires the use of machine learning methods, developers of adaptive robotic systems might run into the problem of limited computational capacity on the physical robots. Although some machine learning methods can be executed on the robot's local processor, for a more robust and more accurate solution, usually computationally heavy methods are needed. A further problem arising with these methods is that their execution can be optimized only on specific platforms or using specific frameworks, ones that the physical robot does not necessarily support. Maintenance can also be an issue, since an update of the software frameworks and libraries supporting machine learning must be executed on all robots being used in a robotic system.

In addition to solving these problems, the independence of the adaptation is another motivation for migrating these operations to a cloud environment. Adaptation of HRI takes place with regards to the human user and not the physical robot. Therefore, we expect that the same robotic system will display the same behavior when interacting with a given user, regardless of the actual physical robot being used. It is important to note however, that the physical robot's capabilities might limit the extent of adaptiveness (e.g. when adapting the social behavior of a robot for HRI, an armless robot cannot perform any gestures and hand movements), which must be taken into consideration when developing a cloud-based adaptive robotic system. Differences in robots' capabilities can be bridged by applying a **robotic middleware** that interacts with the adaptive component deployed in the cloud, and translates the selected action into a form that is executable by the physical robot, which enables the development of multi-platform robotic systems (as depicted on Figure 4.3).

The adaptation of HRI requires the gathering and analysis of data about the interaction and the interacting human partner. These data are used to create a model of the interacting human partner, and to evaluate the robot's actions with respect to their acceptance by the human. Once again, we can see that such data are usually not connected to a specific physical robot (although a human might like some gestures or utterances more when performed by a certain type of robot). Therefore, it is more reasonable to move the database containing this

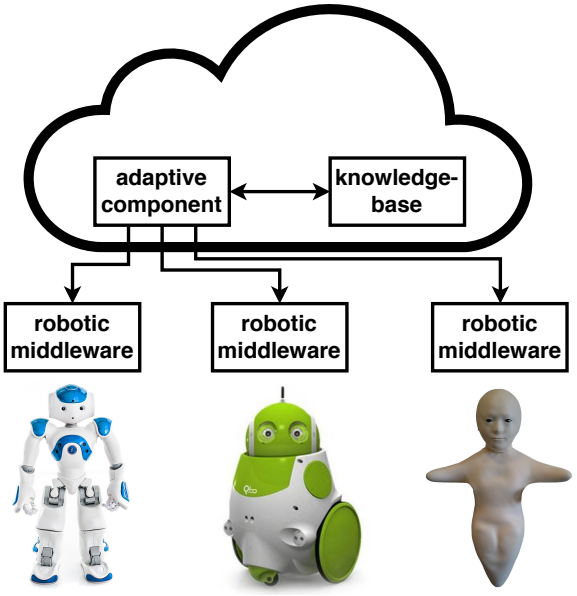

Fig. 4.3 Cloud-based adaptive robotic system architecture with different robot types

information into a cloud environment, especially when considering that the adaptive model must be in a near-constant connection with the database, and the cloud environment offers higher availability. The final motivation for migrating the database to the cloud is the data's nature; since information about a human gathered during an interaction is usually personal and sensitive, it is better to keep these data in the cloud environment that provides a higher overall level of security.

The use of a central database is tightly connected to collective learning, which is enabled exactly by the use of one database for gathering the data. In a robotic system, where each physical robot has its own database, learning is facilitated only by the interactions the physical robot has with different humans. In contrast, robots can rely on the adapted behaviors of other robots in a robotic system with a single central database. This results not only in faster learning for each individual physical robot, but it also ensures that each robot in the robotic system will display the same behavior to an interacting human partner. While this should be the default solution for a cloud-based robotic system, in the case that different behavior from robots is expected, each robot can have its own database representing its adapted behavior in the cloud. The robotic system as a whole still benefits from higher availability of data and greater speed thanks to a direct connection between databases and the adaptive component.

In the previous paragraphs we looked at how cloud computing can aid adaptive robotic systems in particular. We showed that the development and use of robotic systems is easier using a cloud environment. However, we must also elaborate on the advantages the cloud

provides for the maintenance of the system. By deploying the adaptive model on the cloud we ensure that all physical robots display the same behavior and rely on the same version of the robotic system. This characteristic is essential when updating the adaptive component. While in a robotic system where each robot is responsible for adapting its own behavior, the adaptive component must be updated on all physical robots, this update can be carried out centrally in a cloud-based system, and the up-to-dateness of the robotic system is guaranteed. A cloud-based robotic system is also more resistant to faults; if a physical robot is damaged, it can easily be replaced by another robot (not necessarily) of the same type without any need to migrate data and/or losing the adapted behavior.

Chapter 5

PhD thesis proposal

5.1 Scientific goals

5.1.1 Scientific goal 1

Design an on-line reinforcement learning method tailored for the adaptation of robotic tutors for higher learning gains

Measure of success: analyze the learning gains of students using an adaptive robotic tutoring system. In an adaptive tutoring system, learning should take place at a near-constant rate without the student reaching a plateau.

As described in section 2.3, adaptation of human–robot interaction is most often done using the Wizard of Oz methodology with the robot learning behavior from the human Wizard. This presents a number of problems, most notably that the teleoperation might become monotonous to the Wizard very quickly which can have a negative effect on the quality of the interaction. The first scientific goal of this thesis is to address this problem.

The robot participating in a human–robot interaction can learn the appropriate social behavior by directly observing the interaction and deducting rewards for actions from the reaction of the human partner. The human's reaction can be observed in two dimensions: first, the social aspect, by analyzing the human's social input, such as gaze, gestures, facial emotion, mood, etc.; second, by considering the nature of the reaction itself. As in all use cases of reinforcement learning, the success of the learning model depends heavily on the suitability of the reward function. Therefore, in our research, it is important to select an evaluation scenario where designing an appropriate reward function is straightforward.

Due to previous experience with the topic, we decided to test our reinforcement learning method in the context of cognitive stimulation therapy, and in a skill development setting

in addition. These two use cases fit our expectations since the reward can be determined based on the correctness of the human's answer to a question of the robotic tutor. This does not mean however that the robotic tutor's aim is to maximize the number of correct answers the human gives, but to facilitate a learning process at the end of which the human will have achieved the highest learning gain. Therefore, it is necessary for the tutor to select questions to which it expects an incorrect answer from the human learner, so that learning can take place, while also considering the motivation of the human learner to continue in the interaction, since a too high proportion of incorrect answers might be demotivating to the human learner.

From the point of view of implementation, we will create a tutoring system to test and evaluate the reinforcement learning method. Unlike other tutoring systems (see Chapter 1), ours will not require a large set of rules or constraints to evaluate the student's knowledge, to identify gaps in it, and to offer remediations. Instead, we will use reinforcement learning to model the student's knowledge and level of skills, and to adjust the tutor's behavior. Our tutoring system will be implemented using cloud computing and will have the following four components (for an architecture see Figure 5.1):

- **student model** – Using reinforcement learning, this model will predict whether the student's answer to a given question will be correct or incorrect. The component does not try to retrace the student's thinking process and neither does it try to understand why the student's answer might be wrong. Student modeling takes place in a "black box" through reinforcement learning that can nonetheless model the student's knowledge and skills with an increasing accuracy as the learning process progresses.

- **problem pool with corresponding level of difficulty** – A representation of the material to be presented and taught to the student, it is necessary to be able to gradually increase the difficulty of questions.

- **teaching model** – By interacting with the student model and selecting questions from the set of problems, this component is responsible for the pedagogical strategy of the system. Its goal is to select problems in a way that keeps the student constantly just beyond his or her comfort zone. To be able to do this, it relies on the student model to predict the student's response and its correctness.

- **user interface** – As in any tutoring system, the user interface is responsible for communicating with the student and bridging the gap between the system's internal representation of the material and the student's understanding. In our system, we will use the humanoid robot Nao from SoftBank Robotics/Aldebaran.

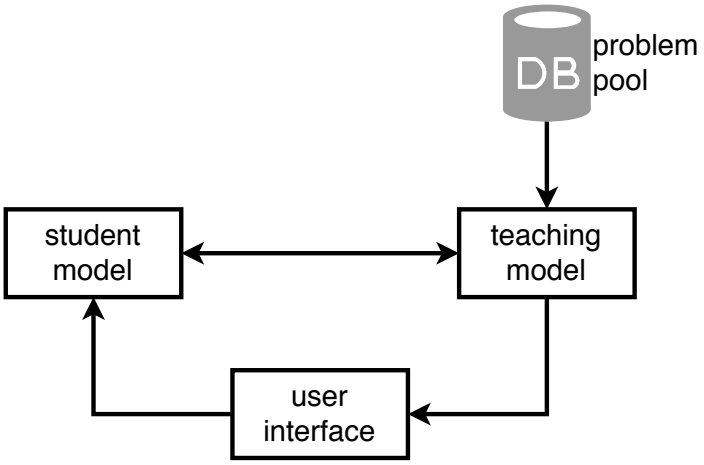

Fig. 5.1 System architecture of the proposed tutoring system

Although the adaptive method will be tested primarily in the context of human learning and cognitive stimulation therapy, it is important to note that it is adjustable for any human–robot interaction where the reward function can be defined in a similar way, that is, reward to the actor (robot) is given based on the reaction and/or performance of the human communicating partner. By adapting the student model we can use it in other human–robot interaction settings for modeling the reaction and/or performance of the human partner.

5.1.2 Scientific goal 2

Test and evaluate a robotic tutor that uses reinforcement learning to personalize the human learning process

Measure of success: compare learning gains of students using a tutoring system with adaptive question selection with learning gains of students using a tutoring system with randomized question selection.

Research on tutoring systems is mostly conducted in the context of knowledge acquisition. Our second scientific goal is to test the system described in section 5.1.1 in the context of skill development. Evaluation will be done in two dimensions: learning gain, and social aspect.

For measuring learning gain, we will compare the human subjects' performance before and after the tutoring session. The tutoring session will be based on the concept of deliberate practice as described in section 1.1.3, where a robot will take the role of the teacher/tutor. The robotic tutor will observe the answers of the human learner and will adjust the difficulty of questions accordingly, identifying problematic aspects that must be targeted during tutoring

to achieve the highest possible learning gain. For the optimal learning process, the tutor should select questions and problems that are just beyond the human learner's comfort zone and current level of understanding. To evaluate whether the questions selected by the robotic tutor indeed lead to higher learning gains, we will deploy two tutoring systems: one that will use our reinforcement learning method from section 5.1.1 for question selection, and one that will select questions randomly. The learning gains of students interacting with the two tutoring systems should show whether or not the learning process is optimized by adaptive question selection.

The social dimension of the human–robot interaction between learner and tutor will be assessed through a questionnaire. The human subjects' answers to the questionnaire should reflect their enjoyment of the interaction, their motivation during the learning process, and their overall evaluation of the robotic tutor.

5.1.3 Scientific goal 3

Conduct a long-term test of robotic tutor

Measure of success: compare learning gains of students interacting with a robotic tutor with the learning gains of students learning through a non-embodied tutoring system. Evaluate the novelty effect of the robotic tutor in a long-term interaction.

As described in section 2.5, the application of robotic tutors in educational settings can further increase the learning gains of a student due to the positive effects of the robotic tutor's embodiment, although some research suggests that the novelty effect of interacting with the robot also contributes to the positive learning gains. Our third scientific goal addresses the effects the robot's embodiment and novelty can have on the learning process.

To examine the embodiment effect, we will compare learning gains of students interacting with a robotic tutor and a simple web-based user interface. It is important to note that the two tutoring systems will have the same functionality and the same question selection strategy. This is possible because the reinforcement learning model does not require information on the human learner other than his or her answer, which can easily be provided in both settings. Since informative feedback is part of tutoring, we believe that the robotic tutor will have an advantage due to its embodiment, since human learners will probably react more positively to feedback given by a robot than a simple message appearing on a computer screen.

Most experiments with robotic tutors are short-term, usually comprising only a few study sessions with a post-learning measurement of learning gains. The results of such experiments might be compromised by the novelty effect of students interacting with the robot for the first time. Therefore, we propose a long-term evaluation of the robotic tutor which would ensure

that this novelty effect wears off. Furthermore, by selecting students who are already familiar with working with the used robotic platform for the evaluation we can further minimize the impact of the novelty effect.

5.2 Technological goals

5.2.1 Technological goal 1

Design and implement a cloud service providing reinforcement learning models
Measure of success: test and evaluate the cloud service in a real-life use case.

Section 4.2 describes the advantages of cloud computing to robotics. In the context of human–robot interaction most cloud-based solutions focus on either offloading computation to a dedicated hardware on the cloud, or knowledge sharing between robots. Commercial cloud services support the processing of social input (e.g. facial emotion assessment, speech recognition) or synthetizing social input (e.g. text-to-speech). With the first technological goal of this thesis, we would like to create a cloud service of a new type.

Reinforcement learning is a general machine learning approach that is used in on-line learning, especially personalization, and it is used more and more often in robotics research. Our goal is to deploy a cloud service that provides its users with the possibility to define and use reinforcement learning models in the cloud. The service will support various methods of reinforcement learning, such as Q-learning, SARSA, deep Q-learning, and actor–critic methods of deep reinforcement learning. Communication with the service will be possible through a RESTful API, whereby the user will send requests to be processed by the service. These requests will contain all information to train the reinforcement learning model. The service will also support action selection, where the user has to send all the information related to the current state of the environment and the service will return the most appropriate action to be executed on the user's side. The user will define and configure the learning model along with hyperparameters when he or she starts using the service.

In accordance with the benefits of cloud computing presented in section 4.2, such a service would fall into the category of offloading computation, knowledge sharing, and collective learning. Additionally, by focusing on the application of reinforcement learning in robotics research, our service would make the development of adaptive human–robot interaction systems simpler and would also support code reuse. Since using our cloud service would mean decreased requirements on the physical robot, the range of robots applicable in human–robot interaction research would also widen. To test and evaluate the service, we will use it for implementing the system outlined in section 5.1.1 and will also incorporate it into the Cloud-based Wizard of Oz teleoperation system developed in our lab to enable on-line learning from the Wizard through reinforcement learning.

5.2.2 Technological goal 2

Design and implement a cloud-based application for personalization of human–robot interaction

Measure of success: test and evaluate the web application in a real-life use case.

In line with the principles of cloud robotics outlined in Chapter 4, our second technological goal aims to create a cloud-based virtual robot that defines all the functionalities and behavior of a physical robot. This division can be viewed as having the "brain" of the robot on the cloud while embodiment is done through the use of a physical robot. Such a system has the following advantages:

- **decreased requirements on the physical robot** – since the bulk of computation is offloaded to a cloud environment, the only requirement on the physical robot is the ability to communicate with the cloud-based virtual robot
- **simpler maintenance** – the functionality is fully defined in the cloud environment, by updating the functionality in the cloud, all physical robots connected to the same virtual robot will change their behavior
- **platform-independence** – the virtual robot works on a higher level of abstraction of robot behavior, it is up to the individual physical robots to execute the selected actions, which makes it possible to connect different types of physical robots to the same virtual robot
- **sharing data and behavior** – from the point of view of personalization, multiple physical robots can share their observations of and experience with the same human subject through the virtual robot, hence enabling quicker and more detailed personalization in real time

The system architecture of a robotic system with a cloud-based virtual robot along with the steps in the human–robot interaction is depicted on Figure 5.2.

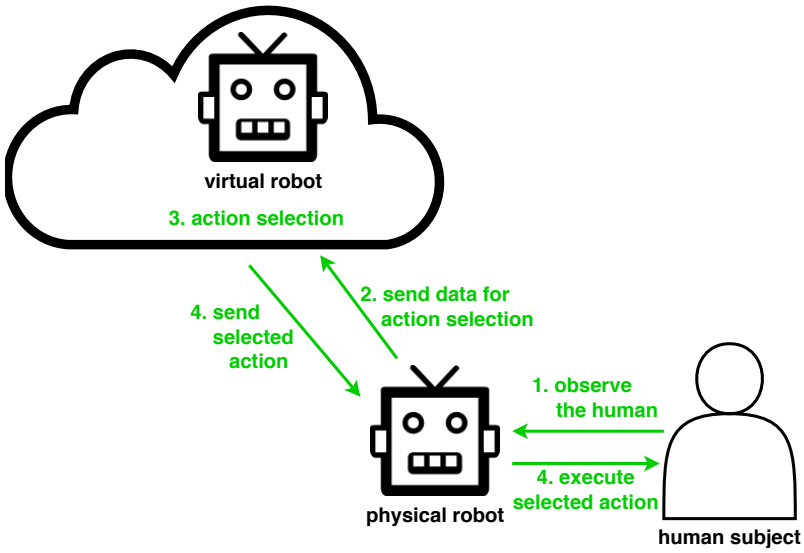

Fig. 5.2 System architecture of a robotic system with a virtual robot

Chapter 6

Scenario for testing

In this chapter we outline our planned research based on the goals presented in Chapter 5. We describe two possible use cases, namely skill development for students, and cognitive stimulation therapy in elderly care. Although the two use cases are similar in nature, their purpose is different: the main measure of success of skill development is the learning gain of students, and the success of cognitive stimulation therapy can be measured by the performance of the humans undergoing therapy.

In accordance with the goals of this thesis, we will implement a tutoring system with two interfaces: a web application with a graphical user interface, and a robotic system with a robotic tutor. In the cognitive stimulation therapy setting, only the robotic tutor will be used, for the skill development scenario we will evaluate both interfaces to compare their effects on the learning experience.

The goals of the tutoring system in cognitive stimulation therapy will be to adapt its behavior to the elderly's capabilities. The therapy will have the form of simple cognitive games. As mentioned above, the primary aim of therapy is not to improve the elderly's cognitive skills, but to aid in their retention. This can be accomplished by adjusting the games to better meet the elderly's current cognitive capabilities. This adaptation can have multiple forms, e.g. difficulty of questions, allotted amount of time for answers, and types of games. One major risk in deploying a learning algorithm is that to ensure the best possible performance of the elderly, the tutoring system might select easy questions, which in turn would diminish the effects of the therapy. Therefore, an appropriate reward function must be devised that takes into consideration not only the number of correct answers, but also the number of incorrect answers, and the types of questions that were incorrectly answered. As additional information, the elderly's emotional state can be supplemented to the learning algorithm to maintain the elderly's motivation and enjoyment of the interaction.

We will test end evaluate the system in the context of cognitive stimulation therapy the following way:

1. If possible, records from previous therapy sessions will be used to pre-train the tutoring system.
2. All participating elderlies will be informed of the goals of the experiment and will have the opportunity to ask any questions they might have about the tutoring system.
3. A small number of therapy sessions will be carried out with each participating elderly using the robotic tutor.
4. The therapy sessions will be evaluated by the participating elderlies (from the aspect of their enjoyment of the therapy and acceptance of the robotic tutor) and an expert (from the aspect of the therapy's nature and success).

In the skill development setting, the performance of the tutoring system will be evaluated based on the learning gains of its users. The exact domain of skill development is yet to be determined, but for research purposes we will choose a domain with a relative simplicity and one where participants can be expected to have previous experience. Two versions of the same tutoring system with different interfaces will be used, a robotic tutor and a web application, in order to compare learning gains when either of them is used. Evaluation of both alternatives will have the same structure:

1. All participating students will be informed of the goals of the experiment and will be introduced to basic functions of the tutoring system to prevent compromised data due to an inability to interact with the system.
2. The students' performance at the skill to be developed will be assessed using the tutoring system.
3. Skill development will take place across multiple study sessions with the tutoring system. During sessions, the system's teaching model will adjust the teaching strategy by spacing out the learning process and selecting problems with the appropriate level of difficulty, and the student model component will be responsible for the personalization of the learning process.
4. The students' skill level will be measured immediately at the end of the learning process to measure the learning gain acquired during the study sessions.

5. The students' skill level will be measured after some period of time to measure the level of long-term retention.

The results of our experiments can contribute to research in multiple domains such as intelligent tutoring systems, reinforcement learning, and robotics.

In the field of intelligent tutoring systems, our system can be used to test whether it is possible to model a student's knowledge using only reinforcement learning and without explicitly defined rules of the material being taught. Reinforcement learning will also be used to adjust the pedagogical strategy, something that has been the subject of some research in tutoring systems, but training was usually done before study sessions took place in contrast with our planned approach of learning on-line and directly from study sessions. We will also explore the effect of spaced learning in skill development, and to our knowledge, our system will be the first attempt in merging the theory of deliberate practice with intelligent tutoring systems.

For the study of reinforcement learning and its application, this thesis can have two main contributions. First, the reinforcement learning as a service concept can be utilized by multiple solutions, independently of use case. Second, our method of applying reinforcement learning to student modeling can be used with some modifications to other settings.

Within robotics and human–robot interaction research in particular we will try to answer questions regarding the novelty effect and the effect of embodiment on the interaction. We will further explore the methods and techniques applicable to social robots in education and our way of modeling student knowledge and skill level could be applied to other types of human–robot interaction since creating the behavioral model of the human interacting partner is an integral problem in human–robot interaction research.

Chapter 7

Conclusion

In this work we presented the theoretical background supporting the aims of the PhD thesis.

First we provided an overview of the human learning process, and skill development, and described various methods connected to it. We also explored the problem of knowledge and skill retention, and discussed the positive effects of spaced repetition. Chapter 1 closed with an overview of intelligent tutoring systems and computer-aided learning, touching on the issues of learning personalization and optimization.

Chapter 2 presented some insights from robotic research relevant to this work. We described the field of social robotics and its specific design issues, and provided an overview of case studies of robotic tutors used in education. We described cognitive stimulation therapy as a useful tool for the retention of cognitive abilities, and slowing down the process of dementia, and we presented some studies where robots were used in therapy. We further described the Wizard of Oz methodology that is often used in human–robot interaction research, and outlined three dimensions of adaptive human–robot interaction: learning from teleoperation, learning social behavior, and learning action selection.

In Chapter 3 we focused on reinforcement learning, a machine learning method most suitable for adaptive human–robot interaction and on-line training. We discussed some design issues connected to the application of reinforcement learning, and described selected reinforcement learning algorithms.

Chapter 4 dealt with cloud computing and it listed some advantages of using cloud computing in robotic applications, with particular emphasis on its benefits to personalization.

In Chapter 5 we presented our research goals for the PhD thesis. We aim to design a self-improving adaptive tutoring system that uses reinforcement learning for student modeling. The system will be an improvement on current intelligent tutoring systems since it will not require an elaborate and detailed representation of the domain being taught in order to be able to adapt its pedagogical strategy to the student's needs, capabilities, and skill level. We

described how we want to use this tutoring system in the context of skill development by applying the theory of deliberate practice, and in cognitive stimulation therapy. We will test and evaluate the system in a long-term study with two interfaces: a web application and a robotic tutor. By using a robotic tutor, our aim is to explore two well-observed effects arising in human–robot interaction, the novelty effect and the effect of embodiment on human engagement.

From a technological perspective, the implementation should result in a cloud service providing reinforcement learning models and methods independently of use case and setting. Such a service will be unique and applicable in further research projects. Another output of the PhD thesis will be a web application connected to the self-improving adaptive tutoring system.

In Chapter 6 we elaborated on our research plans by providing a short overview of our methodology when evaluating the tutoring system in the use cases of skill development, and cognitive stimulation therapy. We closed the chapter by a discussion of possible contributions of the PhD thesis to various research fields.

Bibliography

[1] Agostinho, L., Olivi, L., Feliciano, G., Paolieri, F., Rodrigues, D., Cardozo, E., and Guimaraes, E. (2011). A cloud computing environment for supporting networked robotics applications. In *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, pages 1110–1116. IEEE.

[2] Alemi, M., Meghdari, A., and Ghazisaedy, M. (2014). Employing humanoid robots for teaching english language in iranian junior high-schools. *International Journal of Humanoid Robotics*, 11(03):1450022.

[3] Anderson, J. R. (2013). *The architecture of cognition*. Psychology Press.

[4] Anderson, J. R., Boyle, C. F., and Reiser, B. J. (1985). Intelligent tutoring systems. *Science*, 228(4698):456–462.

[5] Anderson, J. R., Corbett, A. T., Koedinger, K. R., and Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The journal of the learning sciences*, 4(2):167–207.

[6] Atkinson, R. K., Mayer, R. E., and Merrill, M. M. (2005). Fostering social agency in multimedia learning: Examining the impact of an animated agent's voice. *Contemporary Educational Psychology*, 30(1):117–139.

[7] Baird, L. (1995). Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pages 30–37. Elsevier.

[8] Baird III, L. C. and Moore, A. W. (1999). Gradient descent for general reinforcement learning. In *Advances in neural information processing systems*, pages 968–974.

[9] Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846.

[10] Belpaeme, T., Vogt, P., Van den Berghe, R., Bergmann, K., Göksun, T., De Haas, M., Kanero, J., Kennedy, J., Küntay, A. C., Oudgenoeg-Paz, O., et al. (2018). Guidelines for designing social robots as second language tutors. *International Journal of Social Robotics*, pages 1–17.

[11] Benitti, F. B. V. (2012). Exploring the educational potential of robotics in schools: A systematic review. *Computers & Education*, 58(3):978–988.

[12] Berenji, H. R. and Khedkar, P. (1992). Learning and tuning fuzzy logic controllers through reinforcements. *IEEE Transactions on neural networks*, 3(5):724–740.

[13] Berenson, D., Abbeel, P., and Goldberg, K. (2012). A robot path planning framework that learns from experience. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3671–3678. IEEE.

[14] Bhatnagar, S., Precup, D., Silver, D., Sutton, R. S., Maei, H. R., and Szepesvári, C. (2009). Convergent temporal-difference learning with arbitrary smooth function approximation. In *Advances in Neural Information Processing Systems*, pages 1204–1212.

[15] Bloom, B. S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational researcher*, 13(6):4–16.

[16] Boyan, J. A. (2002). Technical update: Least-squares temporal difference learning. *Machine learning*, 49(2-3):233–246.

[17] Broadwell, M. (1969). Teaching for learning. *The Gospel Guardian*, 20(41):2.

[18] Chang, C.-W., Lee, J.-H., Chao, P.-Y., Wang, C.-Y., and Chen, G.-D. (2010). Exploring the possibility of using humanoid robots as instructional tools for teaching a second language in primary school. *Journal of Educational Technology & Society*, 13(2).

[19] Chang, W.-L., Šabanovic, S., and Huber, L. (2013). Use of seal-like robot paro in sensory group therapy for older adults with dementia. In *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 101–102. IEEE.

[20] Chase, W. G. and Ericsson, K. A. (1981). Skilled memory. *Cognitive skills and their acquisition*, pages 141–189.

[21] Chi, M., Jordan, P., VanLehn, K., and Hall, M. (2008). Reinforcement learning-based feature selection for developing pedagogically effective tutorial dialogue tactics. In *Educational Data Mining 2008*.

[22] Chi, M., VanLehn, K., Litman, D., and Jordan, P. (2011). Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. *User Modeling and User-Adapted Interaction*, 21(1-2):137–180.

[23] Chibani, A., Amirat, Y., Mohammed, S., Matson, E., Hagita, N., and Barreto, M. (2013). Ubiquitous robotics: Recent challenges and future trends. *Robotics and Autonomous Systems*, 61(11):1162–1172.

[24] D'Andrea, R. (2012). Guest editorial: A revolution in the warehouse: A retrospective on kiva systems and the grand challenges ahead. *IEEE Transactions on Automation Science and Engineering*, 9(4):638–639.

[25] Ebbinghaus, H. (2013). Memory: A contribution to experimental psychology. *Annals of neurosciences*, 20(4):155.

[26] Ericsson, A. and Pool, R. (2016). *Peak: Secrets from the new science of expertise*. Houghton Mifflin Harcourt.

[27] Ericsson, K. A. (2003). Development of elite performance and deliberate practice. *Expert performance in sports: Advances in research on sport expertise*, pages 49–83.

[28] Ericsson, K. A. (2015). Acquisition and maintenance of medical expertise: a perspective from the expert-performance approach with deliberate practice. *Academic Medicine*, 90(11):1471–1486.

[29] Ericsson, K. A., Krampe, R. T., and Tesch-Römer, C. (1993). The role of deliberate practice in the acquisition of expert performance. *Psychological review*, 100(3):363.

[30] Ericsson, K. A., Tesch-Römer, C., and Krampe, R. T. (1990). The role of practice and motivation in the acquisition of expert-level performance in real life: An empirical evaluation of a theoretical framework. British Psychological Society.

[31] Fasola, J. and Mataric, M. J. (2012). Using socially assistive human–robot interaction to motivate physical exercise for older adults. *Proceedings of the IEEE*, 100(8):2512–2526.

[32] Feil-Seifer, D. and Mataric, M. J. (2005). Defining socially assistive robotics. In *Rehabilitation Robotics, 2005. ICORR 2005. 9th International Conference on*, pages 465–468. IEEE.

[33] Flower, J. (1999). In the mush. *Physician Executive*, 25(1):64–66.

[34] Fong, T., Nourbakhsh, I., and Dautenhahn, K. (2003). A survey of socially interactive robots. *Robotics and autonomous systems*, 42(3-4):143–166.

[35] Fraser, N. M. and Gilbert, G. N. (1991). Simulating speech systems. *Computer Speech & Language*, 5(1):81–99.

[36] Freedman, R., Ali, S. S., and McRoy, S. (2000). Links: what is an intelligent tutoring system? *intelligence*, 11(3):15–16.

[37] Gertner, A. S. and VanLehn, K. (2000). Andes: A coached problem solving environment for physics. In *International conference on intelligent tutoring systems*, pages 133–142. Springer.

[38] Godwin-Jones, R. (2010). Emerging technologies from memory palaces to spacing algorithms: approaches to secondlanguage vocabulary learning. *Language, Learning & Technology*, 14(2):4–11.

[39] Goldberg, K., Chen, B., Solomon, R., Bui, S., Farzin, B., Heitler, J., Poon, D., and Smith, G. (2000). Collaborative teleoperation via the internet. In *Robotics and Automation, 2000. Proceedings. ICRA '00.*, volume 2, pages 2019–2024.

[40] Goldman, R., Eguchi, A., and Sklar, E. (2004). Using educational robotics to engage inner-city students with technology. In *Proceedings of the 6th international conference on Learning sciences*, pages 214–221. International Society of the Learning Sciences.

[41] Gouveia, B. D., Portugal, D., Silva, D. C., and Marques, L. (2015). Computation sharing in distributed robotic systems: A case study on slam. *IEEE Transactions on Automation Science and Engineering*, 12(2):410–422.

[42] Grondman, I., Busoniu, L., Lopes, G. A., and Babuska, R. (2012). A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307.

[43] Han, J., Jo, M., Park, S., and Kim, S. (2005). The educational use of home robots for children. In *Robot and Human Interactive Communication, 2005. ROMAN 2005. IEEE International Workshop on*, pages 378–383. IEEE.

[44] Han, J. and Kim, D. (2009). r-learning services for elementary school students with a teaching assistant robot. In *Human-Robot Interaction (HRI), 2009 4th ACM/IEEE International Conference on*, pages 255–256. IEEE.

[45] Han, J.-H., Jo, M.-H., Jones, V., and Jo, J.-H. (2008). Comparative study on the educational use of home robots for children. *Journal of Information Processing Systems*, 4(4):159–168.

[46] Harmin, M. and Toth, M. (2006). *Inspiring active learning: A complete handbook for today's teachers*. ASCD.

[47] Hasselt, H. V. (2010). Double q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621.

[48] Hausknecht, M. and Stone, P. (2015). Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*.

[49] Henkemans, O. A. B., Bierman, B. P., Janssen, J., Neerincx, M. A., Looije, R., van der Bosch, H., and van der Giessen, J. A. (2013). Using a robot to personalise health education for children with diabetes type 1: A pilot study. *Patient education and counseling*, 92(2):174–181.

[50] Herberg, J. S., Feller, S., Yengin, I., and Saerbeck, M. (2015). Robot watchfulness hinders learning performance. In *Robot and Human Interactive Communication (RO-MAN), 2015 24th IEEE International Symposium on*, pages 153–160. IEEE.

[51] Hu, G., Tay, W. P., and Wen, Y. (2012). Cloud robotics: architecture, challenges and applications. *IEEE network*, 26(3).

[52] Hunziker, D., Gajamohan, M., Waibel, M., and D'Andrea, R. (2013). Rapyuta: The roboearth cloud engine. In *ICRA*, pages 438–444. Citeseer.

[53] Inaba, M. (1993). Remote-brained robotics: interfacing ai with real world behaviors. In *Proceeding 1993 International Symposium on Robotics Research*, pages 335–344, Hidden Valley, Pa.

[54] Johnson, W. L., Rickel, J. W., Lester, J. C., et al. (2000). Animated pedagogical agents: Face-to-face interaction in interactive learning environments. *International Journal of Artificial intelligence in education*, 11(1):47–78.

[55] Jolliffe, I. (2011). Principal component analysis. In *International encyclopedia of statistical science*, pages 1094–1096. Springer.

[56] Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

[57] Kamei, K., Nishio, S., Hagita, N., and Sato, M. (2012). Cloud networked robotics. *IEEE Network*, 26(3).

[58] Kanda, T., Hirano, T., Eaton, D., and Ishiguro, H. (2004). Interactive robots as social partners and peer tutors for children: A field trial. *Human–Computer Interaction*, 19(1-2):61–84.

[59] Kehoe, B., Matsukawa, A., Candido, S., Kuffner, J., and Goldberg, K. (2013). Cloud-based robot grasping with the google object recognition engine. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4263–4270. IEEE.

[60] Kehoe, B., Patil, S., Abbeel, P., and Goldberg, K. (2015a). A survey of research on cloud robotics and automation. *IEEE Transactions on automation science and engineering*, 12(2):398–409.

[61] Kehoe, B., Warrier, D., Patil, S., and Goldberg, K. (2015b). Cloud-based grasp analysis and planning for toleranced parts using parallelized monte carlo sampling. *IEEE Transactions on Automation Science and Engineering*, 12(2):455–470.

[62] Kelley, J. F. (1984). An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems (TOIS)*, 2(1):26–41.

[63] Kennedy, J., Baxter, P., and Belpaeme, T. (2015). The robot who tried too hard: Social behaviour of a robot tutor can negatively affect child learning. In *Proceedings of the tenth annual ACM/IEEE international conference on human-robot interaction*, pages 67–74. ACM.

[64] Knox, W. B., Spaulding, S., and Breazeal, C. (2014). Learning social interaction from the wizard: A proposal. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*.

[65] Kodaganallur, V., Weitz, R. R., and Rosenthal, D. (2005). A comparison of model-tracing and constraint-based intelligent tutoring paradigms. *International Journal of Artificial Intelligence in Education*, 15(2):117–144.

[66] Koedinger, K. R., Anderson, J. R., Hadley, W. H., and Mark, M. A. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education (IJAIED)*, 8:30–43.

[67] Kuffner, J. (2010). Cloud-enabled robots. In *Proc. IEEE-RAS Int. Conf. Humanoid Robot.*, Nashville, TN, USA.

[68] Kumar, V., Rus, D., and Sukhatme, G. (2008). *Springer Handbook of Robotics*, chapter Networked robots, pages 943–958. Springer Berlin Heidelberg.

[69] Landauer, T. K. (1978). Optimum rehearsal patterns and name learning. *Practical aspects of memory*.

[70] Lees, D. and Lepage, P. (1996). Robots in education: the current state of the art. *Journal of Educational Technology Systems*, 24(4):299–320.

[71] Lemaignan, S., Jacq, A., Hood, D., Garcia, F., Paiva, A., and Dillenbourg, P. (2016). Learning by teaching a robot: The case of handwriting. *IEEE Robotics & Automation Magazine*, 1070(9932/16).

[72] Leyzberg, D., Spaulding, S., and Scassellati, B. (2014). Personalizing robot tutors to individuals' learning differences. In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*, pages 423–430. ACM.

[73] Leyzberg, D., Spaulding, S., Toneva, M., and Scassellati, B. (2012). The physical presence of a robot tutor increases cognitive learning gains. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 34.

[74] Li, Y. (2017). Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*.

[75] Lin, L.-J. (1993). Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science.

[76] Liu, B., Chen, Y., Blasch, E., Pham, K., Shen, D., and Chen, G. (2014). A holistic cloud-enabled robotics system for real-time video tracking application. In *Future Information Technology*, pages 455–468. Springer.

[77] Mace, C. A. (1932). The psychology of study.

[78] Maei, H. R., Szepesvári, C., Bhatnagar, S., and Sutton, R. S. (2010). Toward off-policy learning control with function approximation. In *ICML*, pages 719–726.

[79] Magyar, G., Magyar, J., Sinčák, P., and Cavallo, F. (2018). Using social robots as cognitive stimulation therapy coaches in an elderly care facility. In D'Onofrio, G., editor, *Active & healthy aging: novel robotic solutions and internet of things*, chapter 2. Avid Science.

[80] Mayer, R. E., Fennell, S., Farmer, L., and Campbell, J. (2004). A personalization effect in multimedia learning: Students learn better when words are in conversational style rather than formal style. *Journal of educational psychology*, 96(2):389.

[81] Mayo, M. and Mitrovic, A. (2001). Optimising its behaviour with bayesian networks and decision theory.

[82] McKee, G. (2008). *Informatics in Control Automation and Robotics, Vol. 15 of Lecture Notes Electrical Engineering*, chapter What is networked robotics?, pages 35–45. Springer Berlin Heidelberg.

[83] Mell, P., Grance, T., et al. (2011). The nist definition of cloud computing.

[84] Melo, F. S. (2001). Convergence of q-learning: A simple proof. *Institute Of Systems and Robotics, Tech. Rep*, pages 1–4.

[85] Melton, A. W. (1970). The situation with respect to the spacing of repetitions and memory.

[86] Mitrovic, A. (2003). An intelligent sql tutor on the web. *International Journal of Artificial Intelligence in Education*, 13(2-4):173–197.

[87] Mitrovic, A. (2010). Modeling domains and students with constraint-based modeling. In *Advances in intelligent tutoring systems*, pages 63–80. Springer.

[88] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937.

[89] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

[90] Mohanarajah, G., Usenko, V., Singh, M., D'Andrea, R., and Waibel, M. (2015). Cloud-based collaborative 3d mapping in real-time with low-cost robots. *IEEE Transactions on Automation Science and Engineering*, 12(2):423–431.

[91] Moore, A. W. and Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less time. *Machine learning*, 13(1):103–130.

[92] Moreno, R., Mayer, R. E., Spires, H. A., and Lester, J. C. (2001). The case for social agency in computer-based teaching: Do students learn more deeply when they interact with animated pedagogical agents? *Cognition and instruction*, 19(2):177–213.

[93] Moussa, M. A. and Kamel, M. S. (1998). An experimental approach to robotic grasping using a connectionist architecture and generic grasping functions. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 28(2):239–253.

[94] Mubin, O., Bartneck, C., Feijs, L., Hooft van Huysduynen, H., Hu, J., and Muelver, J. (2012). Improving speech recognition with the robot interaction language. *Disruptive Science and Technology*, 1(2):79–88.

[95] Mubin, O., Stevens, C. J., Shahid, S., Al Mahmud, A., and Dong, J.-J. (2013). A review of the applicability of robots in education. *Journal of Technology in Education and Learning*, 1(209-0015):13.

[96] Ng, A. Y., Russell, S. J., et al. (2000). Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2.

[97] Nister, D. and Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 2161–2168. Ieee.

[98] Nwana, H. S. (1990). Intelligent tutoring systems: an overview. *Artificial Intelligence Review*, 4(4):251–277.

[99] Ohlsson, S. (1992). Constraint-based student modelling. *Journal of Interactive Learning Research*, 3(4):429.

[100] Okita, S. Y., Ng-Thow-Hing, V., and Sarvadevabhatla, R. (2009). Learning together: Asimo developing an interactive learning partnership with children. In *Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on*, pages 1125–1130. IEEE.

[101] O'Shea, T., Bornat, R., du Boulay, B., Eisenstadt, M., and Page, I. (1984). Tools for creating intelligent computer tutors. In *Proc. of the international NATO symposium on Artificial and human intelligence*, pages 181–199. Elsevier North-Holland, Inc.

[102] Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.

[103] Piaget, J. and Inhelder, B. (2008). *The psychology of the child*. Basic books.

[104] Prestes, E., Carbonera, J. L., Fiorini, S. R., Jorge, V. A., Abel, M., Madhavan, R., Locoro, A., Goncalves, P., Barreto, M. E., Habib, M., et al. (2013). Towards a core ontology for robotics and automation. *Robotics and Autonomous Systems*, 61(11):1193–1204.

[105] Quick, T. and Dautenhahn, K. (1999). Making embodiment measurable. *Proceedings of '4. Fachtagung der Gesellschaft für Kognitionswissenschaft'. Bielefeld, Germany. http://supergoodtech. com/tomquick/phd/kogwis/webtext. html*.

[106] Qureshi, B. and Koubâa, A. (2014). Five traits of performance enhancement using cloud robotics: A survey. *Procedia Computer Science*, 37:220–227.

[107] Riek, L. D. (2012). Wizard of oz studies in hri: a systematic review and new reporting guidelines. *Journal of Human-Robot Interaction*, 1(1):119–136.

[108] Riek, L. D. and Watson, R. N. (2010). The age of avatar realism. *IEEE Robotics & Automation Magazine*, 17(4):37–42.

[109] Rohrbeck, C. A., Ginsburg-Block, M. D., Fantuzzo, J. W., and Miller, T. R. (2003). Peer-assisted learning interventions with elementary school students: A meta-analytic review. *Journal of Educational Psychology*, 95(2):240.

[110] Rummery, G. A. and Niranjan, M. (1994). *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, England.

[111] Saerbeck, M., Schut, T., Bartneck, C., and Janse, M. D. (2010). Expressive robots in education: varying the degree of social supportive behavior of a robotic tutor. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1613–1622. ACM.

[112] Sagha, H., Shouraki, S. B., Khasteh, H., and Kiaei, A. A. (2008). Reinforcement learning based on active learning method. In *2008 Second International Symposium on Intelligent Information Technology Application*, volume 2, pages 598–602. IEEE.

[113] Sandygulova, A., Swords, D., Abdel-Naby, S., O'Hare, G. M., and Dragone, M. (2013). A study of effective social cues within ubiquitous robotics. In *Human-Robot Interaction (HRI), 2013 8th ACM/IEEE International Conference on*, pages 221–222. IEEE.

[114] Schulze, K. G., Shelby, R. N., Treacy, D. J., Wintersgill, M. C., Vanlehn, K., and Gertner, A. (2000). Andes: An intelligent tutor for classical physics. *Journal of Electronic Publishing*, 6(1).

[115] Self, J. (1988). Student models: what use are they. *Artificial intelligence tools in education*, pages 73–86.

[116] Senft, E., Lemaignan, S., Bartlett, M., Baxter, P., Belpaeme, T., et al. (2018). Robots in the classroom: Learning to be a good tutor.

[117] Shin, N. and Kim, S. (2007). Learning about, from, and with robots: Students' perspectives. In *RO-MAN 2007 - The 16th IEEE International Symposium on Robot and Human Interactive Communication*, pages 1040–1045. IEEE.

[118] Shin, N. and Lee, S. (2008). The effects of appearance and interface design on user perceptions of educational robots. In *Proc. 5th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), Conf*, volume 14372.

[119] Spector, A., Orrell, M., and Woods, B. (2010). Cognitive stimulation therapy (cst): effects on different areas of cognitive function for people with dementia. *International journal of geriatric psychiatry*, 25(12):1253–1258.

[120] Spector, A., Thorgrimsen, L., Woods, B., Royan, L., Davies, S., Butterworth, M., and Orrell, M. (2003). Efficacy of an evidence-based cognitive stimulation therapy programme for people with dementia: randomised controlled trial. *The British Journal of Psychiatry*, 183(3):248–254.

[121] Suraweera, P. and Mitrovic, A. (2002). Kermit: A constraint-based tutor for database modeling. In *International Conference on Intelligent Tutoring Systems*, pages 377–387. Springer.

[122] Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44.

[123] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT press Cambridge.

[124] Szafir, D. and Mutlu, B. (2012). Pay attention!: designing adaptive agents that monitor and improve user engagement. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 11–20. ACM.

[125] Tapus, A. and Vieru, A.-M. (2013). Robot cognitive stimulation for the elderly. In *International Work-Conference on the Interplay Between Natural and Artificial Computation*, pages 94–102. Springer.

[126] Tesauro, G. (1995). Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68.

[127] Tsitsiklis, J. and Van Roy, B. (1996). An analysis of temporal-difference learning with function approximationtechnical. Technical report, Report LIDS-P-2322). Laboratory for Information and Decision Systems ....

[128] Turnbull, L. and Samanta, B. (2013). Cloud robotics: Formation control of a multi robot system utilizing cloud infrastructure. In *Southeastcon, 2013 Proceedings of IEEE*, pages 1–4. IEEE.

[129] Van Den Berg, J., Abbeel, P., and Goldberg, K. (2011). Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information. *The International Journal of Robotics Research*, 30(7):895–913.

[130] Vygotsky, L. S. (1980). *Mind in society: The development of higher psychological processes*. Harvard university press.

[131] Wan, J., Tang, S., Yan, H., Li, D., Wang, S., and Vasilakos, A. V. (2016). Cloud robotics: Current status and open issues. *IEEE Access*, 4:2797–2807.

[132] Weber, R. H. (2010). Internet of things–new security and privacy challenges. *Computer law & security review*, 26(1):23–30.

[133] Weiss, A. (2010). *Validation of an evaluation framework for human-robot interaction: the impact of usability, social acceptance, user experience, and societal impact on collaboration with humanoid robots*. na.

[134] Westlund, J. K. and Breazeal, C. (2015). The interplay of robot language level with children's language learning during storytelling. In *Proceedings of the tenth annual ACM/IEEE international conference on human-robot interaction extended abstracts*, pages 65–66. ACM.

[135] Wozniak, P. and Gorzelanczyk, E. J. (1994). Optimization of repetition spacing in the practice of learning. *Acta neurobiologiae experimentalis*, 54:59–59.

[136] Wu, Y., Mansimov, E., Grosse, R. B., Liao, S., and Ba, J. (2017). Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, pages 5279–5288.

Appendix A

Publications of the author

CoWoOZ — A cloud-based teleoperation platform for social robotics / Gergely Magyar, Peter Sinčák, Ján Magyar, Kaori Yoshida, Alessandro Manzi, Filippo Cavallo – 2017. In: SAMI 2017. – Danvers : IEEE, 2017 S. 00049-00054. – ISBN 978-1-5090-5654-5

Using Social Robots as Cognitive Stimulation Therapy Coaches in an Elderly Care Facility / Gergely Magyar, Ján Magyar, Peter Sinčák, Filippo Cavallo – 2018. In: Active and Healthy Aging: Novel Robotic Solutions and Internet of Things. – Telangana (India) : Avid Science s. 2-38. - ISBN 978-93-86337-71-9

Cloud computing in social robotics / Ján Magyar – 2018. In: SCYR 2018. – Košice : FEI TU, 2018 S. 53-56. – ISBN 978-80-553-2972-7

A cloud-based voting system for emotion recognition in human-computer interaction / Ján Magyar, Gergely Magyar, Peter Sinčák – 2018. In: DISA 2018 : IEEE World Symposium on Digital Intelligence for Systems and Machines : proceedings. – Danver (USA) : Institute of Electrical and Electronics Engineers s. 109-114. – ISBN 978-1-5386-5101-8