

Teaching robots behavior patterns by using reinforcement learning

How to raise pet robots with a remote control

Master's thesis at NADA, Kungliga Tekniska Högskolan, within Computer Science
and Engineering

Måns Ullerstam, モンス ウレスタム, mans@kth.se

March 11, 2004

Assigner was Shibaura Institute of Technology

Examiner and Swedish supervisor was Professor Jan-Olof Eklundh, CVAP,

Kungliga Tekniska Högskolan, Stockholm, Sweden

Japanese supervisor was Professor Makoto Mizukawa, 水川 真, Human-Robot
Interaction Laboratory, Shibaura Institute of Technology, Tokyo, Japan

Teaching robots behavior patterns by using reinforcement learning – How to raise pet robots with a remote control

Abstract

The goal of this project was to show that complex behavior patterns can be learnt by a system based on reinforcement learning. The specific task was to make AIBO, the Sony robot dog, learn complex behavior patterns based on interactions between humans and AIBO. The reinforcement learning system is taught by remote control, used by the human and connected to AIBO. To remember the learnt behavior sequences, a short-term memory of prior actions is used by AIBO. This thesis demonstrates that it is possible to learn behavior sequences and the relationship of cause and effect in complex environments. The thesis also shows that the system works in a natural environment, based on the interaction between humans and AIBO, learning the rewards and the means to reach them in parallel. The thesis presents the methods for implementing such a system.

強化学習を用いたロボット振舞パタンの教示 – 遠隔操作によるペット・ロボットの育成

概要

本研究の目的は、強化学習を用いて複雑な振舞パターンをシステムに教示できることを示すことにある。具体的には、SONY 製ロボット犬 AIBO に対して、人間とのインタラクションを介して複雑な振舞パターンを教示することをタスクとして設定する。本研究の強化学習システムは、AIBO の遠隔操作系を介して教示信号を獲得する。また、AIBO の振舞列を学習する目的のため、学習システムは直前の行動を記憶する短期記憶を備えている。

本論文では、この学習システムの実装、および実験により以下を明らかにした。提案する学習システムは複雑環境下で、振舞列、および因果関係を学習可能である。さらに、本学習システムは、自然環境下で、人と AIBO のインタラクションを介して、報酬と報酬を得るための方策を同時並行的に学習可能である。

Lära robotar beteendemönster med hjälp av kritikerledd inläring - Uppfostra leksaksrobotar med en fjärrkontroll

Sammanfattning

Målet med detta projekt var att visa att det är möjligt att lära sig komplexa beteendemönster med ett system baserat på kritikerledd inläring. Den specifika uppgiften var att få AIBO, Sonys robothund, att lära sig komplexa beteendemönster baserat på interaktion mellan en människa och AIBO. Systemet, baserat på kritikerledd inläring, lär sig av en fjärrkontroll, styrd av människan och kopplad till AIBO. Systemet använder ett korttidsminne bestående av tidigare rörelser, för att komma ihåg inlärd beteendemönster. Denna rapport visar att det är möjligt att lära sig komplexa beteendemönster och kopplingen mellan orsak och verkan i komplexa miljöer. Rapporten visar också att systemet fungerar i en naturlig miljö, baserad på interaktion mellan en människa och AIBO, där systemet parallellt lär sig belöningssystemet och hur man ska agera för att få belöningen. Rapporten presenterar metoderna för att implementera ett sådant system.

Preface

The Master's thesis was written by Måns Ullerstam for NADA, KTH, Sweden, within Computer Science and Engineering. The project was conducted at Shibaura Institute of Technology in Tokyo, Japan, under the guidance of Professor Makoto Mizukawa.

The author would like to thank the Scandinavia-Japan Sasakawa Foundation for their scholarship. The author would also like to thank Gerhard and Stephan Neumann for the use of the Reinforcement Learning Toolbox and their e-mail support. The unknown developers of RCodePlus 2.52A and AiboRemote was helpful and supportive. Without their system, this project would have been very cumbersome.

Individuals who have given invaluable advice during the project are Frank Hoffmann and Rolf Pfeifer. The author would also like to thank Ebba Berggren for the artwork, Winnie Law and Ronnie Johansson for proofreading.

Contents

1	Introduction.....	1
1.1	Background	1
1.1.1	Machine learning.....	2
1.1.2	The thesis.....	2
1.2	Goal	3
1.2.1	AIBO.....	4
1.2.2	Approach	4
1.2.3	Limitations.....	4
1.3	Earlier approaches.....	5
1.3.1	AIBO Explorer	6
1.4	Overview	6
2	Theory.....	7
2.1	Hypothesis.....	7
2.2	Solutions.....	7
2.2.1	Proposed solution	8
2.3	Reinforcement learning.....	9
2.4	Teacher learning	12
2.5	Combined solution	12
3	Method	13
3.1	The Reinforcement Learning Toolbox.....	13
3.2	Learning algorithms	13
3.3	State	14
3.3.1	Sensor Readings	14
3.3.2	Short-term memory	14
3.3.3	Discretization.....	15
3.4	Actions.....	17
3.5	Rewards	17
3.6	Instant learning	18
3.7	Policy.....	19
3.8	Complexity	19
4	Implementation.....	20
4.1	Hardware	20
4.2	Software.....	20
4.2.1	User interface.....	21
4.3	Usage of System.....	21
4.3.1	Teaching mode.....	22

4.3.2	Online learning	23
4.3.3	Offline learning.....	23
4.3.4	Combined mode.....	24
4.4	Communication	24
5	Experimental evaluation	25
5.1	Evaluation method	25
5.2	Learning test.....	25
5.2.1	Process	25
5.2.2	Environment	26
5.2.3	Settings	27
5.2.4	AIBO Explorer	27
5.2.5	Results.....	28
5.2.6	Discussion.....	29
5.3	Behavior sequence test.....	31
5.3.1	Process	32
5.3.2	Settings	32
5.3.3	Results.....	33
5.3.4	AIBO Explorer	34
5.3.5	Discussion.....	34
5.4	Andy's test.....	35
5.4.1	Process	35
5.4.2	Settings	35
5.4.3	Results.....	37
5.4.4	Discussion.....	37
6	Generalization	38
6.1	Other behaviors	38
6.2	Other robots.....	38
6.3	Any learning task	39
7	Conclusions.....	40
7.1	Discussion	40
7.2	Future research	41
8	References	42
9	Appendix.....	43
	Appendix A	43

1 Introduction

In the future there will be robots that live together with humans. Robots will be able to entertain humans and be their “friends”. Simultaneously they will be able to help humans with different tasks. They will live autonomously, learn from experience, be taught by humans and even search for information they require to complete their tasks. That will be the era of consumer robotics.

“I have a dream that robots will one day live in a nation where they will not be judged by the color of their skin but by the content of their character.” – The author’s modified quote by Martin Luther King, Jr.

1.1 Background

There is currently a plethora of robots in the toy industry. In this area, the Sony AIBO ERS-7 (see Figure 1), is the most advanced. As a “friend” there is PARO (see Figure 1), a robot seal developed by AIST (Japan's National Institute of Advanced Industrial Science and Technology) as a companion to elders and young children with mental disabilities. The most helpful robots today are cleaning robots such as Electrolux Trilobite (see Figure 1). The consumer robotics field is slowly maturing into the commercial field. The Sony AIBO has been on the market for several years and is available in different models. The Sony AIBO ERS-7 includes sophisticated software and hardware to handle speech recognition, face recognition, object tracking, behavioral learning, agile movements and more.



Figure 1. (From left to right) Sony AIBO ERS-7 the entertainer, PARO the friend and Electrolux Trilobite the helper (pictures from Sony AIBO official site, Nagoya City Science Museum and Electrolux Trilobite official site).

Many of the hardware and software issues have been given solutions allowing the robots to act autonomously in limited situations. Still, consumer robotics technology is insufficient in many respects. The robots can hardly cope with a normal environment in a consumer’s home, or perform any useful tasks. There are several reasons

for this. One reason is that they cannot fully comprehend the environment they live in. Another is their lack of learning and adaptability.

1.1.1 Machine learning

To address the issues of learning and adaptability there is a research field called ‘machine learning’. The underlying goal of machine learning is to develop systems that learn over time. Machine learning is a broad field covering everything from mathematics, psychology, artificial intelligence to statistics. Moreover, machine learning has come a long way in the last years. Traditional machine learning has solved abstract problems such as playing chess or backgammon. Increasing numbers of projects are however developed and tested in real world environments, especially in the world of robotics. Machine learning is crucial for developing robots capable of more than reacting to instant stimuli or acting according to preprogrammed behaviors. Using machine learning in real environments creates several new issues that still need to be solved. The real world is not programmed or set in any sense, it is changing all the time and it is unpredictable. The stimuli are continuous and the outcome of an action is not fully known beforehand.

1.1.2 The thesis

The background of the thesis derived from earlier work at the Human-Robot Interaction Laboratory at Shibaura Institute of Technology. The laboratory has conducted research of general remote controls for semi-autonomous robots. One of the projects at the laboratory was a remote control for Sony AIBO ERS-210 (called AIBO in this thesis, see Figure 2).

Tsukikawa (月川 剛徳) and Toshinori Hironaka (弘中 利憲) developed the remote control software. It consisted of a joystick connected to a computer and a Wireless LAN between the computer and AIBO. The commands were coded in R-CODE [15]. To handle communication over internet, a standard called Open Robot Interface for the Network [10], ORiN, was used for simplicity. ORiN is an

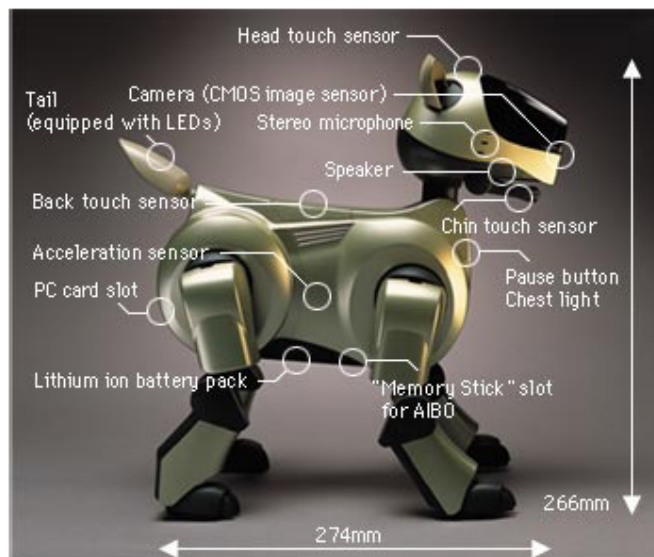


Figure 2. A side view of the AIBO ERS-210 (picture from Sony OPEN-R manual)

industry standard also developed and supported by the laboratory.

When a human (a collective description of any person interacting with AIBO) uses the remote control interface to steer AIBO plenty information is being produced. This information can be used by AIBO to learn behaviors. To accomplish this, AIBO could use the current sensor readings, combined with the previous commands by the remote control to choose an action that is likely to be appreciated by the human.

1.2 Goal

The goal was to develop a learning system that would learn behavior patterns based on context. Context covers both the surrounding environment and the situation leading up to the current state of the environment. The system was aimed at being realizable on a commercial AIBO in a consumer environment. The intention was to make an improvement compared to the learning software by Sony. However, it was considered important to keep the system general and not specific to AIBO. The theories presented should be applicable in other situations as well. Another important goal was to make the learning transparent to the human. The system had to be part of the natural interaction between a human and a robot.

Throughout the whole report the same example will be used to explain different concepts and situations. The example learning task is a dog owner, named Mary who wants to teach her dog, named Andy to sit down when being given food and wait until being told “Go ahead!” before eating (see Figure 3). This example will cover most of the issues in this work, including learning behavior sequences and understanding relationship of cause and effect. In

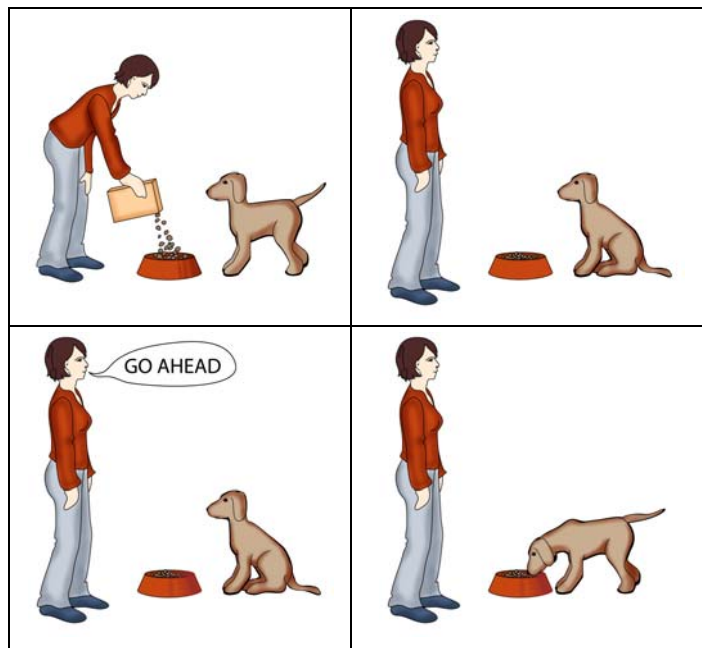


Figure 3. A dog owner, named Mary who wants to teach her dog, named Andy to sit down when being given food and wait until being told “Go ahead!” before eating

this work AIBO will be used for the experiments and for confirmation of the theories presented. AIBO will be tested on similar behavior patterns such as the one described in Figure 3.

1.2.1 AIBO

AIBO (see Figure 2) is a robot dog targeting the consumer market. AIBOs favorite toy is the pink ball (see Figure 4). AIBO can move it's legs, head and tail in many different directions. It can also sing and listen to voice commands from the human. AIBO also has IR distance sensors that can measure the distance to different objects and obstacles. The camera system is specifically geared towards finding the pink ball. AIBO has 20 DOF (Degrees Of Freedom).



Figure 4. AIBO with its favorite toy, the pink ball

1.2.2 Approach

The approach used 'reinforcement learning' (see Section 2.3) as the platform to develop upon. Some specific issues to solve were:

- handle learning in real time, a real environment, and continuous time and space
- handle unpredictable results of actions in the surrounding environment
- incorporate the teaching signal from the remote control into the learning algorithms
- reduce the huge multi-dimensional space of different situations

1.2.3 Limitations

Several issues related to the work were not dealt with for the purpose of clarity and focus. Our work did not deal with:

- Hardware issues related to the joystick, AIBO or computer
- Communication protocols or details of the high level commands of AIBO
- Preprocessing of sensor data (except for translating the sensor readings to discrete values)
- Benchmarking of different learning algorithms

1.3 Earlier approaches

Each subfield of the thesis has been thoroughly investigated in earlier work. The following information is geared toward readers familiar with the field of reinforcement learning. A general background in reinforcement learning is found in an introduction of the field [8]. One of the works similar to the one described in this paper is [17] where a combination of reinforcement learning and user commands is reported. They propose a similar approach as the one in this work for learning intermediate rewards from the user, but with the difference that the overall goal is a predefined task. The importance of having a teacher showing the robot how to act is described in [5]. Learning by social interaction with humans is proposed in [9], which is similar to the approach taken in our work. Research performed on reinforcement learning in continuous time [7] investigated methods of handling the fact that robots live in the real world, not constrained by fixed time steps. To discriminate what stimuli lead to what effect, one can incorporate configural learning [3]. According to [6] traditional greedy policy for reinforcement learning does not work well within robotics control. Instead, the authors propose the use of the Natural Actor-Critic algorithm. Interesting research performed in a simulated world is reported in [2] describing a complex system to handle classical and instrumental conditioning, typical methods to teach real animals different behaviors. Combining reinforcement learning and supervised learning is proposed in [16]. A hierarchical approach was used in [11] to solve complex tasks, with long chains of actions. Their work was based on reinforcement learning, relying on a teacher giving the system subtasks to learn. The authors proposed a new algorithm to use that was supposed to be better than any other algorithm at learning more general and complex tasks, instead of specific simple tasks.

However, most work in the specific field has been performed on lower level learning, such as motor commands. Almost all works have also been performed to solve a specific task, not a general behavior system. Usually the other systems have clearly separated between the autonomous learning phase, teacher phase, etc. Also they have clearly separated different tasks, goals and subgoals. There are currently very few projects that deal with a general behavior system. Current systems built into consumer robots are not able to learn longer behavior patterns or to learn the relationship of cause and effect. The implemented system in this thesis shows that it is possible to learn the goals and the correct actions to achieve the goals simultaneously.

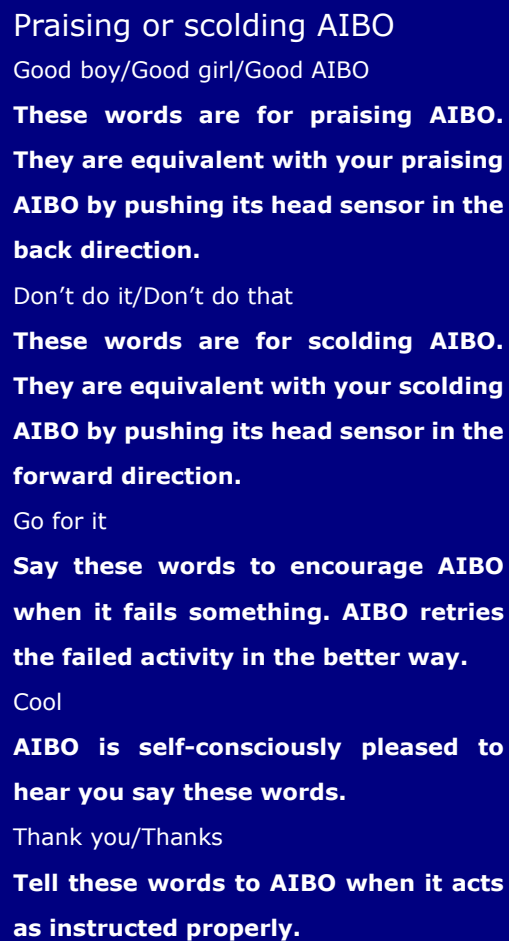
1.3.1 AIBO Explorer

If AIBO is enhanced with commercial software such as AIBO Explorer, it has some learning capability. Sony has made an effort to make AIBO learn based on experience [13] in a similar fashion as reinforcement learning (see Figure 5).

The two main reinforcement signals were “Good AIBO” and “Don’t do that”, giving positive and negative feedback for the current behavior. The other reinforcement signals were similar. The details of the built-in system is not available. According to the observations by the author it seemed like it was a rudimentary system keeping track of the human’s preferences to perform specific actions. Basically a list of all possible actions and a number associated with each action that describes the likelihood of AIBO performing that specific action. When the human encourages or discourages a certain action, the action gets a higher or lower likelihood of being performed again. Therefore it does not seem to be context driven. In that case AIBO could not be encouraged to perform two or more actions in a specific order.

1.4 Overview

In Chapter 2 the theory behind our approach is described. In Chapter 3 the specific methods used to solve the issues are described. In Chapter 4 the actual implementation of the system is described. In Chapter 5 the tests and evaluations are described. In Chapter 6 the possible generalization of the system is discussed. In Chapter 7, finally, conclusions and ideas for future research are presented.



Praising or scolding AIBO
Good boy/Good girl/Good AIBO
These words are for praising AIBO. They are equivalent with your praising AIBO by pushing its head sensor in the back direction.
Don't do it/Don't do that
These words are for scolding AIBO. They are equivalent with your scolding AIBO by pushing its head sensor in the forward direction.
Go for it
Say these words to encourage AIBO when it fails something. AIBO retries the failed activity in the better way.
Cool
AIBO is self-consciously pleased to hear you say these words.
Thank you/Thanks
Tell these words to AIBO when it acts as instructed properly.

Figure 5. Excerpt explaining the possible feedback to AIBO from [13]

2 Theory

This chapter describes the foundations of the work. The work and later experiments were based on the ideas and assumptions presented here.

“If the facts don't fit the theory, change the facts.” – Albert Einstein

2.1 Hypothesis

In this work, the basic idea was that AIBO could learn good behavior patterns by learning chains of actions sent from the remote control. AIBO would also learn during what circumstances a certain behavior was rewarded. At the same time it would adopt behavior patterns that were likely to be similar to the ones that the human would appreciate.

The idea was that the implemented system would use reinforcement learning algorithms combined with teacher instructions from the remote control. The learning algorithms would use both current sensor readings and temporal information (what has happened in the past) to dictate the future actions. The idea behind using a short-term memory recalling earlier actions was for AIBO to learn longer patterns and also understand relationships between cause and effect, not following each other immediately.

The feedback signals sent to the reinforcement system are automatic, plentiful and detailed. This means that the reinforcement signal sent from the human to AIBO is automatically integrated in the process of handling commands from the remote control. Basically any command from the remote control to AIBO is interpreted as positive behavior. If the human makes even more detailed feedback to AIBO by praising or scolding AIBO, it learns faster. Those states are considered goal states. In these states it has completed some behavior or task that is appreciated by the human.

2.2 Solutions

To develop a system that could learn behavior patterns there are several possible solutions. The general problem is to match a vast multidimensional state space into a small number of actions. Therefore any method mapping out the vast space of states into a specific number of regions can be used. There are several statistical approaches and also neural networks, genetic algorithms or reinforcement learning that can be applied to solve the problem. Some special requirements were however present.

It was important that the solution was able to:

- learn in real-time
- learn during normal interaction with the human, similar to a consumer's home environment
- use the existing information for learning, without a need for special handling from the human

2.2.1 Proposed solution

To meet the requirements, the solution was decided to be based on reinforcement learning algorithms (see Section 2.3), also called “learning with a critic”. This was combined with a method of incorporating teacher learning (See Section 2.4). The remote control command was used as a reward, but also as a teacher. The system should be able to handle several cases, listed below:

- **AIBO does not get any signals from human or remote control**
AIBO should walk around and put its learned behavior to use. AIBO will be able to fine tune the system, by applying the learned behavior and study the resulting rewards. Also, AIBO will get more data to learn from by exploring the world, recording all the states it visits. It will also use this phase to find ways to get back on track if lost in an unusual state, not visited earlier.
- **AIBO gets a remote control command**
AIBO should act accordingly. This should give a small positive feedback for the action induced by the remote control. If this occurs, AIBO should instantly learn that performing the same action as the remote control indicates is a good behavior.
- **AIBO gets praised**
AIBO should learn that it has performed some behavior sequence in a manner that the human appreciated. A strong positive feedback should be given to the current state. AIBO should use this to instantly learn the relationship of cause and effect.
- **AIBO gets scolded**
AIBO should learn that it has performed some behavior in a manner that the human disliked. A strong negative feedback should be given to the current state. AIBO should use this to instantly learn the relationship of cause and effect.
- **AIBO is not active**
When AIBO is not active, recharging the battery in the battery station, it can still utilize the time to learn. AIBO can go over its encounters during the active phase and try to understand more general concepts.

2.3 Reinforcement learning

Reinforcement learning is a machine learning approach useful in certain situations. It is considered especially suitable for unpredictable environments, robot learning and goal oriented learning. In these situations the goal is well known, but not how to achieve the goal. Reinforcement learning is also called ‘learning with a critic’. This is because there are no examples of how to complete a task, but rather afterwards it is told (by the critic) how well it completed the task.

Typically there is a high level command given to the robot, i.e. “clean the room”. This command has a high level goal, “the room should be cleaned”. However, exactly how the robot should solve this task is not known and there will be several subgoals that need to be achieved. Every time there will be new situations for the robot to handle to complete the task. In this case the human does not know exactly how the robot should complete the task and it would be tedious for the human to show the robot how to perform in every possible situation. However, it is simple for the human to examine the result after the robot has tried to clean the room and say “Good work” or “Bad work”. In a case like this, reinforcement learning might be suitable.

To get an understanding of the components of reinforcement learning, the different aspects and terminology will be briefly described below. For a more thorough description of reinforcement learning, see [8].



Figure 6. The State-Action-Reward chain. In the current State, the Policy chooses an Action.

Then the Environment gives the Agent the Reward

The basic idea of the reinforcement learning algorithm is to map a state to an action (see Figure 6). That means the input to the algorithm is the current state of the environment. The output should be the “best” action of the agent acting in the environment. The algorithm learns this by getting rewards in some states and/or for some actions, thereby learning this is a special state or action.

Learning algorithms

The learning algorithm of the system decides how to update the stored hypotheses of the world. The basis of all reinforcement learning algorithms is the update rule, which is based on the reward. For each state-action pair, the learning algorithm

stores the expected reward. When exploring the world, these expected rewards get updated based on the algorithm.

There are several different learning algorithms, but most of them are based on Q-Learning. The standard Q-learning rule is described in Figure 7.

$$Q(s_t, a_t) = Q_t(s_t, a_t) + \alpha * \delta_t$$

$$\delta_t = r_{t+1} + \gamma * \max_a Q(s_{t+1}, a) - Q_t(s_t, a_t)$$

Figure 7. Formulas from [17], learning rate – α , future discount – γ , $Q(s,a)$ is the value of a certain action a in a certain state s

Environment

The environment is the representation of the world that the agent acts in. In the example with Andy (see Figure 8) the environment consists of the world surrounding Andy, for example what Andy sees, hears and smells.

State

The state is the unique identifier of a specific situation the agent is in. The state normally consists of the environment. It can also include anything else that is necessary to decide the next action, i.e. a short-term memory.

Action

An action is what the agent can decide to do. It is the means in which the agent has to change the current state. For Andy, an action is to sit down, to eat or to bark (see Figure 8). Performing these actions will normally change the state. The goal of an agent is to perform the action that results in the highest reward.

Reward

Rewards are the means used to teach the agent what actions are considered good or bad depending on the state of the agent. For Andy, a positive reward would be receiving a dog bone; a negative reward would be a scolding.

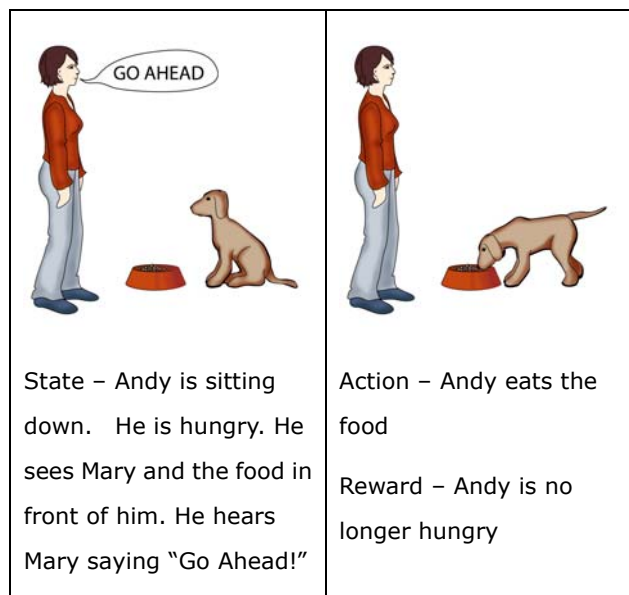


Figure 8. The State, Action and Reward as seen by Andy when he eats food

Agent

The agent is the entity which is in a certain state and takes a certain action. The dog Andy is the agent in the example given in Figure 8.

Policy

The agent follows a specific policy for choosing the action to take. There are possible policies, where the ‘greedy policy’ is the simplest, always choosing the “best” action. The issue with the greedy policy is that the agent performs very little exploration of the state-action space. Other policies, i.e. softmax or epsilon greedy policy, are more stochastic choosing a suboptimal policy sometimes to explore the state-action space.

Eligibility trace

To speed up learning, the history of states and actions are stored until a reward is given, then that reward is propagated back in time along the path taken by the agent. This means that the agent will learn not only that the last action was good or bad depending on the state, but also learn that the state and actions before were also taking part in the resulting reward. This is important since it is not known exactly which action that was the crucial one for getting the reward. The reward distributed along the eligibility trace is reduced with a discount factor to account for the distance in time between action and reward.

Complexity

The complexity of a learning system is normally all possible states multiplied by all possible actions. If the complexity is high, it is possible to understand many different situations and learn many different behaviors, but it is slow and generalization between states is low. However, if the complexity is low, learning will be faster and more general, but it will not be possible to learn nuances between different states. In robotics learning, the state space is normally very large and can easily be infinite, due to continuous state variables.

Discretization

If the state space includes continuous variables the complexity gets very high very fast. Therefore it is important to have a good representation of the state space. This is achieved by discretizing the state space in sensible manner, reducing the size of the space. Discretizing can be made in many different ways. One simple way is to split up a state variable in a specific number of discrete chunks. Another way is to analyze the data from a specific state variable, finding interesting clusters and divide the variable accordingly.

Episode

An episode is a description of one complete session made by the agent in the environment. An episode normally results in either success (with a positive reward) or failure (with a negative reward). It is used to separate different learning sessions and also the eligibility trace has no effect between episodes.

2.4 Teacher learning

In contrast to reinforcement learning, there is another method called “learning with a teacher”. In teacher learning, the learning system gets specific explanations about what to do to solve the task. This means that it gets a number of examples from a teacher of how to solve the task in different situations and from those examples it tries to learn how to solve the task on its own. This method is used with neural networks, where the network is “trained” on the good examples, learning the differences and hopefully generalizes between the examples. The goal is that the system learns to perform the task on its own, even in unknown situations because of the generalization.

In this work, the remote control could be used as the teacher for AIBO. If AIBO got remote control commands, it could store these as good examples of exactly what to do in what situation.

2.5 Combined solution

One of the goals of this work was to combine “learning with a critic” and “learning with a teacher” seamlessly into one system. By doing this, AIBO would utilize the strengths of both systems. Also, it would utilize the data created by the interaction between the human and AIBO. By combining the two learning systems into one, AIBO would also be able to act and learn during many different levels of autonomous operation.

3 Method

This chapter describes the specific methods used to address the issues that arise when implementing the system proposed in this thesis. The specific algorithms chosen are also described in more detail.

"The best way to predict the future is to invent it." - Alan Kay

3.1 The Reinforcement Learning Toolbox

We decided to import existing methods and tools for the implementation of the learning algorithms. The market was surveyed for the best free reinforcement learning implementation available. After some evaluation, the chosen system was The Reinforcement Learning Toolbox [4]. This system covers all standard reinforcement learning algorithms and was easily extendable for new algorithms. It is also open source, which made it possible to follow what happened behind the scenes. However, some of the choices were made because the toolbox supported that feature, and not because it was the best known solution.

3.2 Learning algorithms

Learning can be achieved either online or offline. Online learning means learning is performed live, so the choices made by the learning algorithms decide what the agent does in reality. During offline learning, the learning algorithms are learned from recorded state-action transitions from a live session.

Learning algorithms are may or may not use an eligibility. The algorithms with eligibility trace use the extension (λ) to the name, where λ is the eligibility trace discount factor. The learning algorithms chosen for this work were $Q(\lambda)$ and $SARSA(\lambda)$ (see Figure 9). $SARSA(\lambda)$ is normally considered better than $Q(\lambda)$ for robot situations [4], since $SARSA(\lambda)$ takes the exploration policy into account. This means if there are some high negative rewards, such as running into a wall or down the stairs, $SARSA(\lambda)$ will not try those actions. However, $Q(\lambda)$ was chosen during the online learning sessions and $SARSA(\lambda)$ during offline learning. The reason for choosing $Q(\lambda)$ during online learning was that it allowed the human to interfere during online learning. The human could at any stage give AIBO a remote control

$$\begin{aligned} Q_{t+1}(s,a) &= Q_t(s,a) + \alpha \delta_t e_t(s,a) \\ \delta_t &= r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \\ e_t(s,a) &= \gamma \lambda e_{t-1}(s,a) + 1, \text{ if } s = s_t \text{ and } a = a_t \\ e_t(s,a) &= \gamma \lambda e_{t-1}(s,a), \text{ otherwise} \end{aligned}$$

Figure 9. Formulas from [17], Learning rate – α , Eligibility trace discount – λ , Future discount – γ , $Q(s,a)$ is the estimated reward of a certain action a in a certain state s

command, praise or scold AIBO for its behavior. This would not be possible with SARSA(λ), since in SARSA(λ) requires the preceding action to be known beforehand. The standard SARSA(λ) learning rule is described in Figure 9.

3.3 State

The state was represented as a combination of the sensor readings from AIBO and a short-term memory of its own actions.

3.3.1 Sensor Readings

The sensor readings that were used to describe the model state are described in Table 1.

Table 1. All sensor variables and their value ranges

Sensors	Values
Distance to object/obstacle	100 – 900 (mm)
Hour	0 – 23 (hours)
Brightness	0 – 255 (dark to bright)
Voice ID	0 – 69 (each phrase has unique ID)
Voice horizontal angle	0,1,2,4,8,16,32,64,128 (different directions)
Pink ball distance	33 -1094 (mm)
Pink ball horizontal angle	-20 – 21 (degrees)
Pink ball vertical angle	-26 – 26 (degrees)
Pink ball	0 – 1 (No pink ball, or pink ball in vision)
Head tilt angle	-90 – 90 (degrees)
Head pan angle	-82 – 43 (degrees)

All sensor variables are given in integer values from AIBO. There is no information about the accuracy of the sensor readings. Most sensor readings are values from physical sensors, but some sensor readings are software processed data, i.e. Voice ID and Pink ball.

3.3.2 Short-term memory

A short-term memory was used as part of the state. The short-term memory remembered the last actions that AIBO had performed. The short term memory was a FIFO list, remembering the last ten actions. The idea behind the short-term memory is that AIBO can learn what action to perform based on the earlier actions in time. This makes the learning not limited to the current sensor readings, but includes a temporal aspect of the state.

For example, Andy can use his short-term memory to learn when his owner Mary allows him to eat his food (see Figure 10). This makes it possible to learn when served food he should sit down and wait for Mary to say “Go ahead!” before starting to eat. The complexity of the short-term memory equals Actions^{Memory}.



Figure 10. Andy using his short-term memory to remember what happened earlier in time to decide how to behave

3.3.3 Discretization

One very important part of any learning system is how to discretize continuous data. In this case, the continuous data were the sensor readings. To be able to generalize between different states, even discrete data can be discretized in new parts. The goal of discretization is to simplify the learning process, reducing the size of the state space.

Table 2 gives an example of the discretization of the Distance sensor of AIBO. The whole state space from 100-900 mm is discretized in three regions in this example.

Table 2. Example of discretization of the distance sensor of AIBO, dividing the sensor into three regions

Distance sensor			
Value	100-365	366-633	634-900
Discretization	0	1	2

There is a trade-off with discretization of state variables. If each state is divided into many small steps, learning is slow and the system cannot generalize between states very well. But it is also possible to learn very detailed behavior, and to be sensitive to different state values. If each state is divided into large chunks, the learning is fast and it can generalize better. On the other hand it makes it difficult to learn fine nuances in state variables.

Teaching robots behavior patterns by using reinforcement learning

In this project it was chosen to use a method compensating for this effect. It is a combination of dividing the states into smaller chunks and generalizing between states. These algorithms are called RBF features (Radial Basis Function). An example of a RBF function would be a discretization of the two dimensional space of a Vertical angle and a Horizontal angle sensor (see Figure 11).

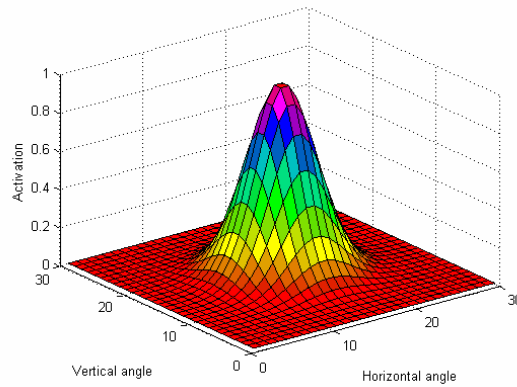


Figure 11. An example of a two dimensional RBF feature where the activation of two state variables is described

This means that from the “exact” state where AIBO currently is (the peak in Figure 11), all surrounding states in all dimensions are also activated, but less. The activations of surrounding states follow a bell curve, where the width of the curve can be different in each dimension. However this process is very complicated, which makes online learning slow with the current system.

Discretization of the data is performed before the data is presented to the learning algorithms. Discretization is the representation of the data upon which the learning system makes its decisions. The way the data is discretized therefore affects what the learning system can learn.

3.4 Actions

The actions the learning system could make AIBO perform are described in Table 3. Through different tests of the system different action were enabled to the learning algorithm to choose from. The complete list of possible actions [12] is more extensive and has a range of low level and high level actions such as “bend left front knee 20 degrees backward” and “beg for food”. These actions were chosen since they were high level actions but still made it possible for AIBO to move from point A to point B in different ways, looking in any direction and also perform a few specific visible and audible cues such as “lie down” and “sing a happy song”.

Table 3. All actions that the learning algorithm could make AIBO perform

AIBO Actions	
Do nothing	Look up left
Sit down	Look up right
Stand up	Look down left
Lie down	Look down right
Walk forward	Walk forward left
Walk backward	Walk forward right
Look down	Walk backward left
Look up	Walk backward right
Turn right	Search for pink ball
Turn left	Sing a happy song
Look right	Sing a sad song
Look left	Look ahead

3.5 Rewards

Rewards were given by ways of either be remote control rewards based on remote control commands made by the human, positive final rewards based on praising by the human, or negative final rewards based on scolding by the human.

What makes this project different from many other efforts in this field is the reward is not known beforehand. There is no predefined clear goal, such as winning a game of chess. The rewards are learned by feedback from the remote control and the human in parallel with the behaviors. The reward function is a list of special states. These states are either remote control states, positive final states or negative final states. Each time the system receives a remote control command it associates the state after the command with a positive value (+0.2). Since the history of actions is stored in the state, the history of actions will be part of the reward. If the human gives additional reinforcement signals, using praise or scolding, these states will be stored as strong negative or positive final states (+1.0 or -1.0). The final states differ from remote control states in the sense that they end the current episode and start a new episode. For AIBO this is equivalent to have completed a behavior pattern and either succeeded or failed.

During the operation of AIBO, each state was compared to the final states to see if a final reward should be given. This system compared the negative final rewards first,

which meant that if a state was considered negative, AIBO would get a negative reward and that episode would be completed. This meant that the system was geared towards avoiding negative rewards first, then achieving positive rewards. If a state had been stored as both negative and positive, the negative state had precedence.

3.6 Instant learning

To make the learning process very fast, a method we chose to call ‘instant learning’ was used. Instant learning means the agent will learn what it is taught during teaching mode instantly. The system replays the teaching command instantly through the learning algorithm, making the connection between State-Action-Reward. This means whenever the human gives a command from the remote control, praising or scolding, AIBO instantly learns the relationship of cause and effect of that reward signal.

To make an analogy with the real world, Andy is taught by Mary to sit. This is performed by Mary saying “Sit!”, and then pushing down Andy’s back to force him to sit. Afterwards Andy is given a dog bone as a reward (see Figure 12). This sequence will instantly be replayed in Andy’s mind, where he learns that the combination of the word “Sit!” and sitting down is rewarded with a dog bone. Therefore Andy learns the relationship of cause and effect instantly.



Figure 12. Given a dog bone, Andy replays the event in his mind, figuring out what caused the reward

3.7 Policy

The softmax formula (see Figure 13) will be used as exploration policy. The parameter β determines how sharp the distinction between a good and a bad action is. For $\beta = 0$ the distribution will be uniform and for $\beta \rightarrow \infty$ the policy is similar to the greedy policy, always choosing the action with the highest value. $Q(s,a)$ is the reward for action a in state s . The basic idea of the softmax policy is if one action results in a significantly higher reward, that action will be chosen most times, but if many actions results in similar rewards the policy will be distributed more equally among those actions. Therefore, the exploration is high if there is high uncertainty, but low if it is clear which action is the most favorable. According to [17] it is not clear if the epsilon-greedy policy or the softmax policy is the better and they have found “no careful comparative studies of these two simple action-selection rules”. However, according to [6] traditional greedy policy for reinforcement learning does not work well within robotics. Therefore softmax policy was chosen in this work.

$$P(a_i) = \frac{e^{\beta * Q(s,a_i)}}{\sum_j e^{\beta * Q(s,a_{ji})}}$$

Figure 13. The softmax policy equation, according to Gibbs distribution. $Q(s,a)$ is the reward for action a in state s . β is the temperature, deciding the sharpness of the distinction between a good and a bad action

3.8 Complexity

The theoretical complexity of the problem is the matching between all possible states to all possible actions. If every parameter is kept in its original

$$\begin{aligned} & \text{NumberOfValues}^{\text{Sensors}} * \text{Actions}^{\text{Memory}} * \text{Actions} \\ & = 801 * 24 * 256 * 8 * 1060 * 181 * 43 * 53 * 126 \\ & * 2 * 70 * 24^{10} * 24 \approx 1.0 * 10^{35} \end{aligned}$$

Figure 14. The theoretical complexity of the state space

form the complexity of the problem is about 10^{35} according to the calculations in Figure 14. It can be compared with some other complexities such as:

- number of humans in the world, less than 10^{10}
- number of neuron connections in the human brain, about 10^{14}
- mass of the sun in grams, 10^{33}

The complexity of the problem (10^{35}) was about hundred times as big as the mass of the sun measured in grams. If each neuron connection in each human brain in every human living on earth today could store one State-Action pair, it would still be far less than enough to store this data. Therefore, one of the major issues with any learning algorithm is the issue of discretizing the data into manageable chunks. Discretization has to be made differently based on what to learn and is described in detail in Chapter 5.

4 Implementation

This chapter describes the implementation of the ideas presented in the previous chapter. It covers the hardware and software architecture and design.

4.1 Hardware

The hardware used for the project was standard commercial hardware. All components are available for purchase at retail outlets. The hardware consisted of:

- Microsoft Sidewinder Force Feedback 2 Joystick
- 2 AIBO ERS-210, one older and one with the new SuperCore (with twice the processor speed)
- AIBO Wireless LAN cards (11 Mb/s, IEEE 802.11b)
- Buffalo AirStation g54, Wireless LAN access point (11/54 Mb/s, IEEE 802.11b/g)
- hp Compaq d330 PC, 2.2 GHz processor, 512 MB RAM memory

"In theory, there is no difference between theory and practice. But, in practice, there is." - Jan L.A. van de Snepscheut



Figure 15. (Top to bottom) The joystick used for remote controlling and the two AIBOs with their pink ball

Some of the specific hardware are shown in Figure 15. The remaining hardware was interchangeable with any other standard components.

4.2 Software

The implemented software is based on C++, developed on Windows OS with Visual Studio. An open source remote control for AIBO, AiboRemote [1], is used. The basic architecture is modular, making the remote control software run on its own, with or without the learning system. For the learning algorithms, another open source project, The Reinforcement Learning Toolbox [4], is used.

4.2.1 User interface

The user interface of the implemented software is mainly the remote control interface. From that interface you can reach all functionality of the system, except offline learning. A screenshot of the remote control interface is provided in Figure 16 showing the basic interface of the software.

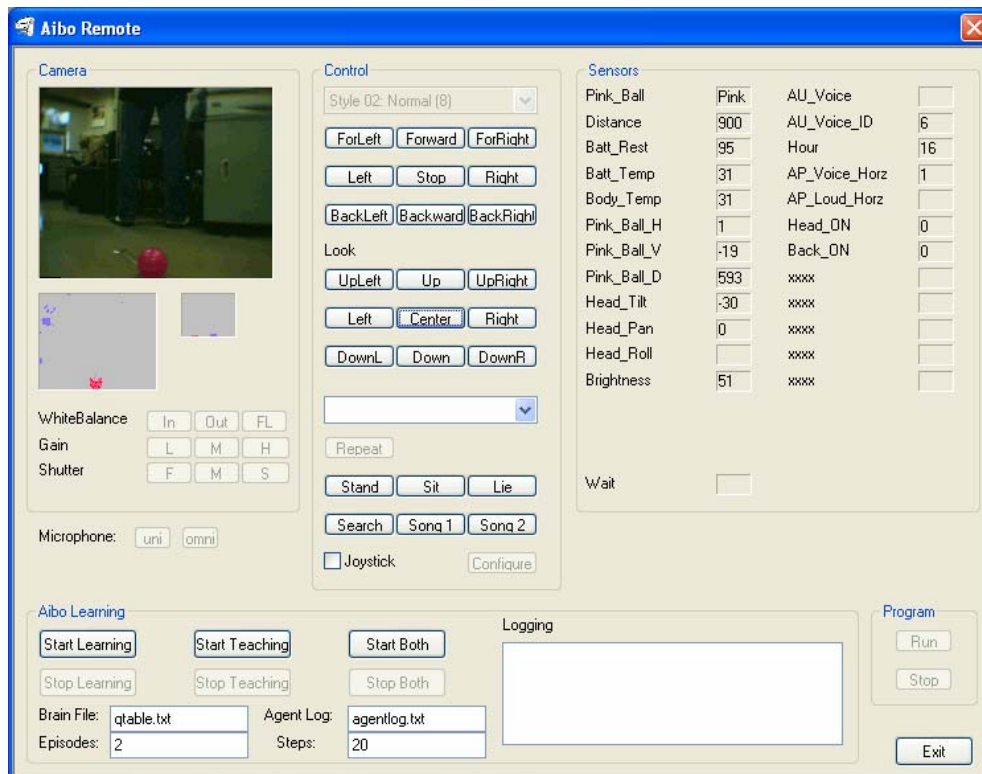


Figure 16. A screenshot of the implemented software

In the figure you can see the live camera from AIBO and also the color filter image. The color filter is used by AIBO to identify the pink ball. All sensor readings used for learning are captured in real-time and shown in the interface. In the bottom half of the interface, functions such as learning and teaching were controlled.

4.3 Usage of System

The implemented software consisted of several components. Each software component is described below and shown in Figure 17. The *Remote Control* is the user interface with a graphical user interface and a joystick. In this interface all sensor readings from the camera are also shown. *AIBO* is both the physical AIBO and the abstract representation of AIBO, keeping track of short-term memory and current state. *Rewards* are the collections of reward states used by the reward function to reward AIBO in different states. The *Real World* is where the sensor readings come from and where the agent performs its actions. *Learning* is the learning algorithms used to

Teaching robots behavior patterns by using reinforcement learning

learn correct behavior. The *Brain* is the storage space remembering the value of all state-action pairs.

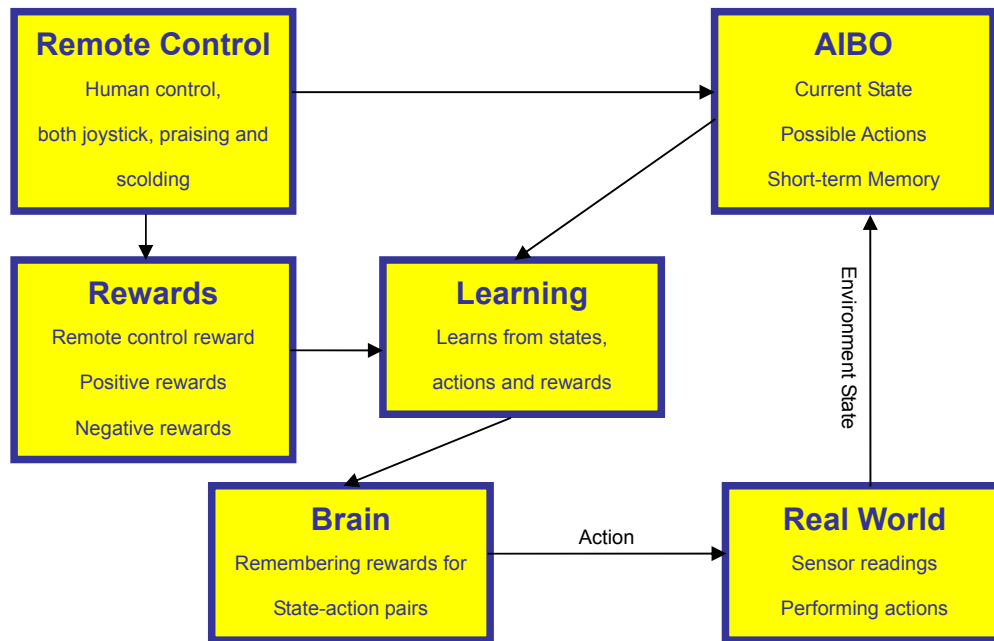


Figure 17. A simplified overview of the software architecture

There are several different ways to interact with the system. The goal of the system was to make learning transparent to the human. This meant that the human should not need to know there was an underlying learning system learning behavior patterns. It is therefore possible to run all subsystems in parallel except offline learning. Below is a description of each subsystem.

4.3.1 Teaching mode

This interface consists of the remote control of AIBO. During this phase AIBO acts according to the remote control signals sent from the human. AIBO also learns that the states visited are positive states. The human can use this to teach AIBO how to perform an action or behavior.

While AIBO is in teaching mode it will also respond to reward and punishment. If AIBO gets a reward or a punishment it will learn that these states are very positive or very negative. These states are also identified as final states, where AIBO believes it has completed a behavior, either as a success or a failure. After AIBO has reached a final state, it will restart and look for a new goal.

The working of the system is described below:

- AIBO gets a reward by
 - patting AIBO on head or
 - saying “Good AIBO!” or “That’s Right!”AIBO learns that this is a positive final state. This concludes the episode, starting a new one
- AIBO gets a punishment by
 - pressing AIBO on back or
 - saying “Don’t do that!” or “That’s wrong!”AIBO learns that this is a negative final state. This concludes the episode, starting a new one

4.3.2 Online learning

This is when AIBO is autonomous. During this phase the system works as follows:

- AIBO tries to behave as good as possible
- AIBO is learning to behave according to the different reward signals
- AIBO is testing different strategies to achieve positive rewards

During online learning the learning algorithms are active. AIBO is acting according to the stored state-action pairs, deciding the next action based on the current state of AIBO and the exploration policy chosen by the learning system. After each action is performed, the new state of AIBO is calculated. This state is then compared with all reward states, giving AIBO a reward if the state is one of the reward states.

4.3.3 Offline learning

The offline mode is AIBO’s “dreaming” mode. In this phase, AIBO is not physically doing anything. What happens during this phase is:

- AIBO replays the online learning episode in its mind
- AIBO fine tunes the behaviors by thinking through the actions and the rewards again
- AIBO learns more complex relations about cause and effect
- AIBO generalizes between similar states

This phase is used by the learning system to make calculations that are impossible to make in real-time. During this phase the RBF features are turned on, making learning better but slower. The RBF features generalize between similar states, making AIBO able to handle unknown states as long as they are similar to known states.

4.3.4 Combined mode

During the combined mode, both the teaching and the online modes are active. This means that AIBO will act according to the learning system, unless the human interferes and gives AIBO a remote control command. If so, AIBO uses instant learning to learn that this specific action in this specific state is positive. AIBO also acts according to the remote control command.

4.4 Communication

Since the system resided on a PC, it had to communicate with AIBO over Wireless LAN. Sony has developed a communication protocol called R-CODE [12] for that purpose. R-CODE is a proprietary protocol for communication between AIBO and a computer. R-CODE can be used both to program AIBO remotely from a computer and to send commands to AIBO in real time. It can also be used to receive feedback from AIBO, such as sensor values and internal status. The possibilities of R-CODE set the limitations of both the input data from the environment and the output in the form of actions. In R-CODE there are 45 possible action commands and 67 different variable or sensor readings [15]. Of these, many sensors and actions are specific joint angles and tail movements, which were not used for the project. In this project, a special version of R-CODE was used, called RCodePlus 2.52A (210) [1]. The extension made it possible to fetch camera image and sound from AIBO among other things.

5 Experimental evaluation

This chapter covers all tests and evaluations that were performed on the implemented system. It also covers the results of the tests and a discussion of possible improvements.

*"Do, or do not. There is no 'try'." -
Yoda ('The Empire Strikes Back')*

5.1 Evaluation method

There were several different experiments carried out on the developed system. Since the aim of the work was to make AIBO learn behavior patterns, it was somewhat fuzzy and subjective by nature. It was chosen that the presented solution would be compared with commercial AIBO with the software AIBO Explorer [13] installed. According to Sony this software is the most enhanced, where AIBO understands all voice commands, can perform all actions, but still can learn by praising and scolding from the human.

5.2 Learning test

The learning test was used to confirm the presented solution had some credibility. One of the goals was to see if AIBO could learn a behavior in a complex state space. Also, the performance of the system was evaluated and it was used to optimize learning parameters. The behavior tested was to find the pink ball. This choice of behavior was made due to simplicity and the fact that AIBO Explorer was supposed to have this behavior. It was therefore possible to compare the performance between the solution presented in this thesis and the AIBO Explorer software.

5.2.1 Process

To be able to evaluate the performance and to study improvement rate an orderly process was decided upon. The process went through each subsystem, starting in teaching mode and ending up in an evaluation of the performance.

The test was performed in the following steps:

1. Teach AIBO how to look around and walk around to find the ball by using the remote control, while giving AIBO about 30 good examples. Teach AIBO it has succeeded with the behavior as soon as AIBO saw the pink ball.
2. Let AIBO perform ten episodes of online autonomous learning, all resulting in a success. During this phase only positive and negative final rewards were allowed to be given to AIBO.
3. Start offline learning. This learning phase was made on all stored episodes, so the number of episodes grew between each test.
4. Evaluate the performance. The evaluation of the performance was made by starting another online learning session of five episodes, watching how well AIBO behaved and how well he found the ball. In this stage, no interferences were allowed.

This process was repeated from either Step 1 or Step 2 for five rounds.

5.2.2 Environment

The environment was the Human-Robot Interaction Laboratory. A small 'dog yard' was set up for AIBO (see Figure 18), to keep the experiments somewhat similar for each test round and for both systems. It was also used to keep the dog contained in roughly the same area as the pink ball. The dog yard had three walls and one open end. The reason for this was that the distance sensors are able to register any values from 100 to 900 mm. The other reason was to make it possible for AIBO to walk away from the experiment. Otherwise it would inevitably see the pink ball sooner or later.

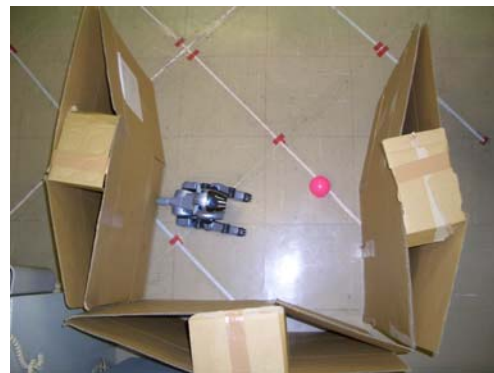


Figure 18. A side view and a top view of the dog yard for AIBO

The surrounding was kept noisy instead of a quiet environment on purpose. This made the learning algorithms perform in an environment more like a consumer environment, where

AIBO normally lives. There were people moving around, speaking and sometimes moving AIBO or the ball during the tests. The tests were run with two different AIBOs, making different hardware an extra error source. During the test some incorrect final states were added due to noise of sensors, etc. About 5% of the states were incorrect. Both incorrect positive and negative states were added.

5.2.3 Settings

There are several parameters for the learning system. These were the parameters used during testing:

- 0.10 as RBF β , deciding the shape of the RBF features
- 0.40 as α , deciding the learning rate
- 0.95 as γ , deciding the discount-rate parameter for future rewards
- 0.90 as λ , deciding the decay-rate parameter for eligibility traces
- 9.0 as the softmax β , deciding how greedy the policy was

For the learning test, a subset of the full system was used to reduce the state space. The sensor variables used were:

- Distance to object/obstacle
- Pink ball distance
- Pink ball horizontal angle
- Pink ball vertical angle
- Pink ball
- Head tilt angle
- Head pan angle

A subset of the actions was used, enabling only 13 of the actions. The short-term memory was set to remember only the last two actions, instead of the original ten. The continuous sensor readings were all divided into three discrete values. RBF-features were only used during offline learning, where each of the continuous sensor readings was smoothed out. The discrete parameters were left untouched.

The complexity of the learning algorithms in the first trials was therefore roughly three

$$\text{Complexity} = \text{ContinuousValues}^{\text{Sensors}} * \text{DiscreteValues} * \text{Actions}^{\text{Memory}}$$

$$* \text{Actions} = 6^3 * 2 * 13^2 * 13 \approx 3000000$$

Figure 19. The complexity of the system during the Learning test

million states, according to the calculations in Figure 19. This meant that the system had to learn roughly three million state-action pairs. Compared to the full complexity (10^{35}) the reduction was still very large.

5.2.4 AIBO Explorer

An AIBO equipped with AIBO Explorer was tested to perform according to the

Teaching robots behavior patterns by using reinforcement learning

same behavior. The test was also run for five rounds. The evaluation of the performance was made by starting five episodes, watching how well AIBO behaved and how well it found the ball.

AIBO was also given positive or negative rewards if it succeeded or failed. A test episode was considered a success if AIBO found the pink ball. It was considered a failure if:

- moving into obstacles or
- leaving the dog yard too far for too long time or
- if AIBO did nothing productive for a very long time

The learning was made by patting AIBO on the head when it found the pink ball and hitting AIBO on the head if it failed (according to the instruction manual for AIBO Explorer). The number of test rounds was five for this test.

AIBO Explorer performed quite well from the beginning by succeeding in 2 of 5 episodes during the first round. The performance slowly rose to 3 of 5 episodes in round 3 (see Appendix A for details of the results). The first thing to notice was that AIBO Explorer has many preprogrammed behavior goals. This means it was not possible to contain AIBO within the specific behavior tested. Also, AIBO Explorer seems to be very sensitive to constant interaction with humans. If it does not get a lot of feedback it gets bored and does not do much at all.

5.2.5 Results

The test was run five times. After the tests, AIBO was able to find the ball 5 times out of 5 (see Appendix A for details of the results). At that time the reward states consisted of 881 remote control rewards, 185 positive final rewards and 25 final negative rewards.

A comparison graph between the implemented learning algorithms and AIBO Explorer is provided in Figure 20. The conclusion from this graph is the system was able to learn the behavior. For AIBO Explorer it was however hard to draw the conclusion it learned the behavior better over time. From the way AIBO Explorer acted it seemed that AIBO Explorer had learned that it was positive to see the pink ball, but it did not learn it was positive to look for the ball.

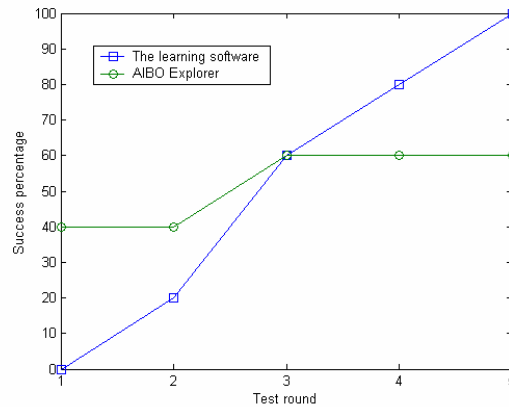


Figure 20. The results of the learning test, comparing AIBO Explorer software with the implemented system

5.2.6 Discussion

The conclusion of the learning test is the implemented system seemed to work. It is however important to understand some of the issues with this test. The main issue is the learning algorithms are meant to learn behaviors, which are basically action sequences depending on the environment, not to solve tasks. However, this test was evaluating the possibility for AIBO to find the pink ball and to understand it had completed a task, not a behavior. This test was also used to calibrate the parameters, therefore it was not expected that the parameters were optimally chosen during the experiment.

As usual, experiments in the real world are not without its problems. The main issue was the real time performance of the learning algorithms. For AIBO to learn the more complex issues, the problem space was huge. This was not possible to do in real time over a Wireless LAN, using a 2.4 GHz computer for processing. Therefore the learning had to be done offline as well, making the experiments more tedious to carry out. Offline learning was very slow, since the RBF features are very complex to compute.

Possible enhancements

Even though the performance was better than expected on the implemented system compared to AIBO Explorer there was room for many enhancements. The division of the distance sensors into only three different regions was too crude. AIBO had a hard time understanding the difference between negative states when it was very close to a wall and states when it should just turn around or walk backwards away

from a wall, since any distance between 0 and 366 mm were considered the same state. The distance sensors could for example be sampled in a logarithmic scale, making closer distances more detailed than farther distances.

Generalization of the short-term memory seemed to be an interesting improvement, since two states were not considered similar if any of the actions leading to the state was different. The offline learning was time intensive. Some simplifications of the RBF features would speed up the process immensely.

One apparent problem with the learning system was the issue of learning what to do to end up in a certain reward state. Only the reward states were recorded, not the state transitions from state S_1 , via action A, to reward state S_2 . This meant that the action to take in State S_1 was not learnt the first time, only after trying it again, hopefully ending up in state S_2 . This meant that the learning process was unnecessary slow. The information existed during learning, but was not used optimally.

The action space could also be described in another way with something similar to vector summation. It could be possible to combine for example move forward, move backward, turn left and turn right to a direction vector, giving AIBO a walking direction instead. Combining many discrete actions into one continuous would also make the movements less rigid. This was not supported by the standard reinforcement learning system in the Reinforcement Learning Toolkit [4].

It would also be interesting to discretize each parameter based on the actual values each parameter take during normal operations of AIBO. This could be used to discretize the dense regions into small chunks and sparse regions into large chunks.

The high level action, where AIBO would search for the pink ball, was not used in this test. If used, AIBO would find the ball much easier and the learning would be less complex. Adding many more actions to AIBO would also make the behaviors more diverse, making the movements of AIBO less rigid.

Other issues were that the state space was not optimized, i.e. if the pink ball was not seen, AIBO still gave information about the vertical and horizontal angle of the pink ball. This data had no relevance, since the pink ball was not seen at all. This could be optimized and discretized in a different way, taking into account the relevance of the input sensors.

Another major area of enhancement would be to add hierarchical learning from simpler subspaces of the state space. Hierarchical learning is a method for learning simpler behaviors first, then combining the simpler behaviors to more complex behaviors.

The discrete states of the state space were not generalized using RBF features. This meant that the discrete state space was large. Generalizing the short-term memory would be an interesting enhancement.

Implemented enhancements

Before moving to the next test there were several enhancements made to the system to account for the issues in the previous test. These were:

- implementing instant learning, making AIBO learn commands during teaching mode instantly
- optimizing the state space to account for the actual sensor ranges possible
- calibrating some of the parameters in the learning system

5.3 Behavior sequence test

The Behavior sequence test was chosen to reflect the changes to the system. It was also chosen to be a more complex behavior. It was also not known if it was possible to teach AIBO Explorer this behavior. The behavior chosen was to make AIBO move in two different patterns. The trigger for the behavior patterns was the pink ball. AIBO was supposed to behave according to the description below:

- If no ball is seen, move according to the first pattern
 1. Stand
 2. Search Pink Ball
 3. Turn left
 4. Turn left
 5. Turn left
 6. Loop forever from Step 2This pattern has length of five, with a loop of length 4.
- As soon as a pink ball is seen, move according to the second pattern
 1. Move backward
 2. Turn right
 3. Move backward
 4. Turn right
 5. Sing happy songThis pattern has the length of five, with two actions repeating themselves.

Teaching robots behavior patterns by using reinforcement learning

The reason for having loop and repeated actions was to test if learning of longer behavior sequences was possible. Both patterns were also longer than the short-term memory, to see if it could still learn these patterns, without being able to remember the whole sequences.

This behavior did not need any more input than the pink ball sensor. Therefore the learning was measured on the ability to learn the correct behavior sequence. This learning task is also without any continuous sensor data. Everything is discrete, so there were no RBF features used and no generalization was made.

5.3.1 Process

The test was performed in the following steps:

1. Teach AIBO the whole pattern, by using the remote control, giving AIBO five good examples. Teach AIBO that it has succeeded with the behavior as soon as AIBO had completed the whole pattern.
2. Let AIBO perform ten episodes of online autonomous learning, all ending in a success or failure of the behavior. During this phase only positive and negative final rewards were allowed to be given to AIBO.
3. Start offline learning. This learning phase was made on all stored episodes, so the number of episodes grew for each new test round.
4. Evaluate the performance. The evaluation of the performance was made by starting another online learning session of five episodes, watching how well AIBO succeeded. In this stage, no interferences were allowed.
5. If AIBO was not able to behave correctly, the test was repeated from either Step 1 or Step 2 continuously until AIBO performed well.

5.3.2 Settings

There are several parameters for the learning system. Below those variables that were changed from the previous test are described:

- 0.30 as α , deciding the learning rate, compared to 0.40 in the previous test. Due to the implementation of instant learning, the learning rate was higher
- 50.0 as the softmax β , compared to 9.0 in previous tests, making the exploration more similar to the greedy policy. There seemed to be a lot of exploration in earlier experiments

For the Behavior sequence test the only sensor variable needed by the learning was the Pink ball sensor. All 24 actions were enabled during this test. The short-term memory was set to remember the last three actions, instead of the two in the previous experiment.

The complexity of the learning algorithms in the second test was therefore roughly 700 000 states, according to the calculations in Figure 21.

$$\begin{aligned} & \textit{DiscreteValues} * \textit{Actions}^{\textit{Memory}} * \textit{Actions} \\ & = 2 * 24^3 * 24 \approx 700000 \end{aligned}$$

Figure 21. The complexity of the system during the behavior sequence test

The environment was decided to be the laboratory environment without any restricting dog yard, since there was no need for any restrictions to perform the tests on the behavior.

5.3.3 Results

AIBO was able to learn the Behavior test very accurately. In the first round AIBO succeeded 2 out of 5 times. The success rose to 5 out of 5 times after 4 rounds (see Figure 22). Details of the results can be found in Appendix A.

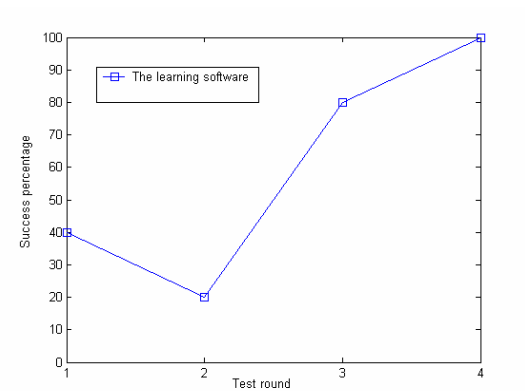


Figure 22. The results of the behavior sequence test

5.3.4 AIBO Explorer

Unfortunately it was not possible to teach an AIBO equipped with AIBO Explorer the same behavior. The closest thing to learning a specific behavior sequence with AIBO Explorer was to use a functionality called “Teaching actions”. An explanation of that functionality is provided in Figure 23. This makes it possible to record a certain action. This action is up to ten seconds and records the joint movements made by the human during teaching. It is however not possible to make it perform these actions based on the surroundings. It only performs the actions when told so by the human. The actions learned are also low level joint angles, not high level such as “Sit”.

Teaching actions

You can teach *AIBO Explorer* various actions by actually moving its legs to show how AIBO is supposed to move. You can also teach *AIBO Explorer* the number corresponding to the action you have taught it. After *AIBO Explorer* has learnt the action and its number, it acts as instructed when you call out the number.

***AIBO Explorer* learns an action you teach it while you are pressing either paw sensor of its front legs. *AIBO Explorer* can learn an action that takes up to 10 seconds. You can move the front legs, rear legs and tail of *AIBO* during action registration.**

Figure 23. An excerpt explaining the functionality called “Teaching actions” from [13]

5.3.5 Discussion

One noted problem was that the remote control rewards are only weak positive, +0.1, but negative rewards are always strong negative, -1.0. Therefore if AIBO made a bad action and got punished, that punishment erased many good moves made before that, because it overshadowed the weak positive signals. It was not possible to compensate this by giving strong positive feedback while AIBO was on its way to perform a correct behavior, since any strong positive or negative feedback was considered to be the end of a behavior sequence, and a new episode started.

The reason for this problem is if AIBO performs an incorrect action based on the state, the subsequent state is stored as a negative final reward state. It is not always the case that the reward state is bad, only the action taken in a specific state. If a state normally visited got stored as a negative final state, AIBO got stuck in that state. Therefore there was a reason to enhance the reward function. For example it could make sure that discrepancies such as states recorded as both positive and negative were deleted from the reward function.

5.4 Andy's test

The final evaluation, Andy's test, was a combination of behavior sequences and a complex representation of the state with continuous variables. Andy's test was made similar to the learning situation with Mary and Andy. The behavior was supposed to work according to the following:

- If the pink ball is seen (Andy's food)
 1. Walk close to the ball if not already close
 2. Sit down (Andy sitting down)
 3. Keep sitting down until voice command. (Andy waiting for Mary's command)
- When "Go Ahead!" is heard (Mary's command)
Sing a happy song (Andy's eating)
- If no pink ball is seen
Wander around, hoping to see the pink ball

5.4.1 Process

The test was performed in the following steps:

1. Teach AIBO the whole pattern, by using the remote control, giving AIBO 15 good examples. Teach AIBO that it has succeeded with the behavior as soon as AIBO had completed the whole pattern.
2. Let AIBO perform 15 episodes of online autonomous learning, all ending in a success or failure of the behavior. During this phase remote control commands are allowed to make AIBO perform the correct action. This step was not present in the other tests. It was needed due to the complexity of the behavior.
3. Let AIBO perform 15 episodes of online autonomous learning, all ending in a success or failure of the behavior. During this phase only positive and negative final rewards were allowed to be given to AIBO.
4. Start offline learning. This learning phase was made on all stored episodes, so the number of episodes grew for each new test round.
5. Evaluate the performance. The evaluation of the performance was made by starting another online learning session of five episodes, watching how well AIBO succeeded. In this stage, no interferences were allowed.
6. If AIBO was not able to behave correctly, the test was repeated from either Step 1 or Step 2 continuously until AIBO performed well.

5.4.2 Settings

The only setting that was changed in this experiment was the remote control reward .

Teaching robots behavior patterns by using reinforcement learning

This was set to +0.2, compared to +0.1 in previous test, to account for negative final rewards overshadowing positive remote control rewards.

The sensors used were:

- Pink ball
- Pink ball distance
- Pink ball horizontal angle
- Voice ID
- Distance

The new discretization the distance sensors were split in four slots compared to three in earlier tests, according to the following:

- Distance: 100-299, 300-499, 500-699, 700-900
- Pink ball distance: 33-299, 300-564, 565-829, 830-1094

Also the variables *Pink ball distance* and *Pink ball horizontal angle* were discretized differently. The reason was that if no pink ball was seen, these parameters held the latest known data of the pink ball. This was often incorrect data. Therefore the data were set to a preset value when pink ball was not seen during this test, and only updated when the ball was seen. This made the state space less complex when the ball was not seen. However, if the pink ball was lost AIBO did not remember roughly where it was, which made it harder to find it again. The Voice ID was discretized in a simpler state variable with only two values:

- 1 if Voice ID was 37 which was the command “Go Ahead!”
- 0 otherwise

The short-term memory was set to remember the last three actions. The

$$\begin{aligned} & \textit{ContinuousValues} * \textit{DiscreteValues} * \textit{Actions}^{\textit{Memory}} * \textit{Actions} \\ & = 48 * 4 * 24^2 * 24 \approx 3000000 \end{aligned}$$

Figure 24. The complexity of the system during Andy's test

complexity of the learning algorithms in the second test was therefore roughly three million states, according to the calculations in Figure 24.

The environment was decided to be the laboratory environment without any restricting dog yard, since that would be an environment more similar to any open home environment that AIBO would normally be in.

5.4.3 Results

In the first round AIBO succeeded 1 out of 5 times. The success then was irregular, ending up after 5 rounds of succeeding 3 out of 5 times (see Figure 25). Details of the results can be found in Appendix A.

5.4.4 Discussion

The environment was very noisy. It was hard to use voice commands as the trigger, since there were many error sources. AIBOs voice recognition has weaknesses as well, its own noise triggers voice commands and the recognition is not stable.

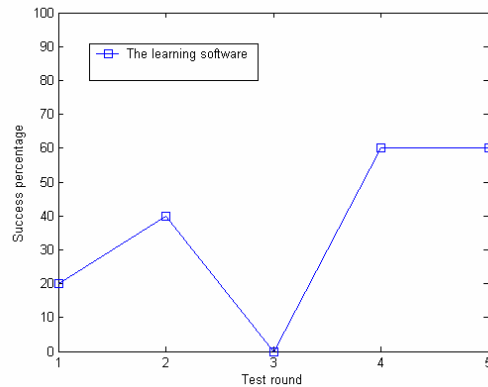


Figure 25. The results of Andy's test

In this test it became obvious that the model for positive and negative final rewards was too simple. If AIBO got scolded when it sat down without the ball present, it could not separate that from when sitting down with the ball present. This was due to the fact that the final negative state did not include any information about the ball being seen in the previous state. A more robust usage and recording of the final states was necessary.

6 Generalization

The theories presented in this thesis have been proven to work for the specific case of learn AIBO complex behavior patterns. An important aspect of the work was however to be able to use the same system in other situations.

This chapter will present a discussion of

how general the solution is and in what other situations the same solution might be applicable.

“When I’m working on a problem, I never think about beauty. I think only how to solve the problem. But when I have finished, if the solution is not beautiful, I know it is wrong.”
- R. Buckminster Fuller

6.1 Other behaviors

The system seemed to be robust and possible to use for different learning tasks. With this system AIBO can learn many different behaviors. Any behavior that has these properties:

- Where the actions do not repeat themselves in loops longer than the length of the short-term memory
- Where the complexity of the system is about 3 million states-action pairs or less (it was tested on 10 million states which made the process slow)
- Where the final positive state is clearly identifiable
- Where the number of actions between a cause and effect relationship is shorter or equal to the length of the short-term memory

Since the complexity of the short-term memory equals $\text{NumberOfActions}^{\text{LengthOfMemory}}$, the short-term memory could be no longer than four in the tests described in this thesis ($2^4 = 331776$). With a longer memory, the complexity would be higher than 3 million state-action pairs.

6.2 Other robots

By using a standard toolbox for reinforcement learning [4], the design was forced to separate the learning task from the specific hardware and therefore became more general. There were no specifics concerning AIBO in the learning system, which means that it should work well on other robots too. This system is geared towards high level decisions. The higher the level of the actions is, the better the system performs.

However the discretization of the state space has to be made specifically for each robot system. Unfortunately it is normally necessary to discretize differently based

on the different tasks. The reason for this is to keep the complexity within reasonable bounds.

6.3 Any learning task

The system should work for other learning tasks as well, but it is geared towards uncertain systems, where sensor readings are instable and continuous of its nature. It is also geared towards cognitive systems such as robots or agents that move around in an open world.

The system should probably not be used for abstract and deterministic learning tasks such as playing chess. If the current environment state contains all the information needed to make the correct action, there is no point in having the short-term memory. But even for abstract tasks such as dialing a phone number it is very important to remember the last digits dialed.

7 Conclusions

This thesis shows that it is possible to have an AIBO that learns both the goals, subgoals and the means to achieve them during real-time interaction with humans. Some of the unique features described in this thesis are:

“The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka!' (I found it!) but 'That's funny ...'” - Isaac Asimov

- The short-term memory - the history of actions is part of the state from which AIBO chooses the next action. This makes it possible to learn complex action sequences correctly
- Parallel learning of reward function and policy - the reward function is not known beforehand and is taught while learning how to achieve the rewards
- Unified system - the learning algorithms are seamlessly integrated in the natural interaction between a human and AIBO. The human does not need any computer or robot knowledge to teach AIBO behaviors
- Instant learning - makes the system fast at picking up new behaviors and understanding what actions and state that caused the rewards.

7.1 Discussion

The first test of the implemented system was satisfactory. It seems like the system can learn simple behaviors better than the AIBO Explorer software. The more complex behaviors were impossible for AIBO Explorer to learn, but worked well on the implemented system. Part of this functionality would therefore be interesting to have in the commercial AIBO.

Many traditional issues with artificial intelligence became obvious to the author by performing this work. The idea was to make a robust learning system that could be used in any learning situation, no matter the properties of the state or the action space. However, it became apparent that to make the system run in real-time, the state space had to be geared specifically to each behavior to learn. If all sensor variables would be used as input and the short-term memory kept long enough to learn any behavior sequence, the complexity would explode, making the system impossible to run in real-time. The underlying system generalized well, but in practice, the state description did not generalize between behaviors.

Another idea is not to predefine the parameters of the learning algorithm. These parameters could instead automatically tune themselves during usage of the imple-

mented system. The idea was for example that the discretization would automatically divide the sensors variables and that the free parameters of the algorithms would change automatically. However, it became apparent to the author that it is very difficult to evaluate the performance of a system if many parameters are automatically changing constantly. Another issue with allowing variables to change is that there is no mathematical proof that such a system will ever converge towards a solution. There were several ideas for how to enhance the system, but with the time constraints on the project it was not possible to pursue all tracks.

7.2 Future research

Future research can be spun off in different directions. It is possible to continue the work with the specific application. An idea is to move the learnt behavior to AIBO's memory, so AIBO can act according to the learnt behavior autonomously. To extend that even further, the whole learning algorithms can be built into AIBO. In that case, the major task is to handle the real time issues in the limited hardware and software in AIBO. This would be necessary for commercializing the ideas presented in this thesis.

The second possibility is to make several of the enhancements proposed in this thesis. Working on a generalization of the short-term memory and an automatic system for variable discretization would be interesting. Transforming the short-term memory to a fuzzy discrete state was not done in this work. Some ideas of how it could be realized are:

- Using a discount factor of actions made earlier in time. This would result in the short-term memory fading out back in time, making it less and less important that the old actions were exactly the same as in the memory for the state to be activated.
- Making all actions part of a hierarchy. Deciding the importance of different actions. This would make it more important that the essential actions were made exactly at the same time in history, but the other actions could be different, still activating the state.
- Comparing short-term memory states based on loose chains of actions, allowing for other actions to be added in between and for some actions to be taken away.

To add hierarchical learning, making it possible to learn a combination of high and low level behaviors would also be very interesting.

8 References

- [1] AIBOPET@AIBOHACK.COM. 2003. AiboPet's AiboHack Site. <http://aibohack.com/rcode/index.html>. Last visited February 15, 2004.
- [2] B. M. BLUMBERG, PETER M. TODD AND PATTIE MAES. 1996. No Bad Dogs: Ethological Lessons for Learning in Hamsterdam.
- [3] D. S. TOURETZKY, N. D. DAW AND E. J. TIRA-THOMPSON. 2002. Combining Configural and TD Learning on a Robot. *Proceedings of the Second International Conference on Development and Learning*.
- [4] G. NEUMANN AND S. NEUMANN. 2003. The Reinforcement Learning Toolbox. <http://www.igi.tugraz.at/ril-toolbox/manual/RILToolboxDocu.pdf>. Last visited February 15, 2004.
- [5] I. H. WITTEN. 1995. PBD Systems: When will they ever learn?
- [6] J. PETERS, S. VIJAYAKUMAR AND S. SCHAAAL. 2003. Reinforcement Learning for Humanoid Robots. *Third IEEE-RAS International Conference on Humanoid Robotics*.
- [7] L. C. BAIRD III. 1994. Reinforcement Learning in Continuous Time: Advantage Updating. *Proceedings of the International Conference on Neural Networks*.
- [8] L. P. KAELBLING, M. L. LITTMAN AND A. W. MOORE. 1996. Reinforcement Learning: A Survey.
- [9] L. STEELS AND F. KAPLAN. 2001. AIBO's first words. The social learning of language and meaning.
- [10] M. MIZUKAWA, H. MATSUKA, T. KOYAMA AND A. MATSUMOTO. 2000. ORiN: Open Robot Interface for the Network, a proposed standard. *Industrial Robot: An International Journal*. Volume 27, Number 5, 2000, pp.344-350.
- [11] M. MARTIN AND U. CORTÉS. 1995. Learning to Solve Complex Tasks by Reinforcement: A New Algorithm. *European Conference on Machine Learning*.
- [12] R. S. SUTTON AND A. G. BARTO. 1998. *Reinforcement Learning. An Introduction*. MIT Press. Massachusetts. ISBN 0-262-19398-1.
- [13] SONY CORPORATION. 2002. AIBO Explorer User Guide.
- [14] SONY CORPORATION. 2002. R-CODE Ver1.2 – ACTION.AMS file.
- [15] SONY CORPORATION. 2002. R-CODE Ver1.2 – Reference Manual.
- [16] S. SCHAAAL. 1997. Learning from Demonstration. *Advances in Neural Information Processing Systems 9*. pp. 1040-1046.
- [17] Y. WANG, M. HUBER, V. N. PAPUDESU AND D.J. COOK. 2003. User-Guided Reinforcement Learning of Robot Assistive Tasks for an Intelligent Environment. *Proceedings of the 2003 IEEE/RSJ IROS*.

9 Appendix

Appendix A

Detailed results from the Learning test:

Round	Evaluation	Comment
1	0 of 5	Did not understand it had succeeded when the pink ball was found.
2	1 of 5	Still had to learn it that it was positive to find the pink ball. Also recognized that it ended up in a negative final state once.
3	3 of 5	Only half of the time it needed information about end state. It thought a state was a positive end state even if there was no ball.
4	4 of 5	It sits down and lies down often, even though that has seldom being encouraged by the human. It thought a state was a positive end state even if there was no ball.
5	5 of 5	Now the tests were repeated from Step 2 instead of 1. It still does not recognizes all the pink ball states as positive, but that is actually not what AIBO has been taught.

Detailed results from AIBO Explorer performing the Learning test:

Round	Evaluation	Comment
1	2 of 5	Seems bored and does not do much. Failed twice, succeeded when ball was put in front of AIBO.
2	2 of 5	Succeeded only when ball was in front of AIBO.
3	3 of 5	Succeeded mostly when ball was in AIBO's visual front area. Walked around once and found it.
4	3 of 5	Walked around and found it twice.
5	3 of 5	Walked around and found it twice.

Teaching robots behavior patterns by using reinforcement learning

Detailed results from the Behavior test:

Round	Evaluation	Comment
1	2 of 5	Since there was no generalization, it could not perform the correct sequence if the situation was different than during learning.
2	1 of 5	For some reason it seems like the offline learning phase made it learn that it was bad to make the correct move in one behavior. The system is sensitive to negative feedback, as well as sensitive if taught the wrong behavior.
3	4 of 5	Very good performance. It seems like offline learning degrades the level of development.
4	5 of 5	AIBO performed excellent. There were some incorrect negative final states that were deleted. After that offline learning increased the performance.

Detailed results from Andy's test:

Round	Evaluation	Comment
1	1 of 5	This is a very complex test. Especially difficult since two distance sensors are in use. As soon as the distance to an object is different it does not recognize the state. It managed to complete one round because the situation was exactly the same.
2	2 of 5	Walked into a wall once, has not learnt that it was negative to walk into walls. Sits down sometimes without seeing the ball or with the ball too far away. Succeeded only when ball was close to it. Never walked to the ball.
3	0 of 5	Two negative situations when lying down. One case where he sat without a ball. Two perfect walks to the ball and then sitting down, both these cases failed due to the stochastic exploration.
4	3 of 5	This was pretty good performance. It had one false positive case however. Sometimes AIBO sits down without any ball close. This performance was good enough, but since it failed completely in the last round, one extra round was decided upon.
5	3 of 5	Same level of performance again. The system was very sensitive to erroneous states. These states made the offline learning phase strengthen the errors. The test was completed at this stage.