Escola Politècnica Superior de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

MASTER THESIS

Titel: Smart Home – Opportunity to make life easier

Author: Martin Hasaj

Director: MESEGUER PALLARÈS, Roc

Date: September 3, 2008

Titel: Smart Home – Opportunity to make life easier

Author: Martin Hasaj

Director: MESEGUER PALLARÈS, Roc

Date: September 3, 2008

Overview

A great deal of contemporary research is showing that it is not work that goes home but home that goes to work. I would like to write my thesis about smart home possibilities (theoretically), which are available, or proposed to the market. Then I would like to make a business plan for hypothetical company which wants to arrive to the market with smart homes development. Finally, I would like to design (practically) part of smart home according to available technologies. I am more considered in the SW part of the problem, but I want to use real hardware if it would be possible rather then the simulator.

The SW should be portable written in JAVA, developed on LINUX (better hardware support). It should be able to manage all smart home interfaces and extensible. Brain of the system should be Neural Network, which should be able to learn automatically the inhabitants behaviour and help them in their everyday routine. Make them disappear and you have succeeded.

For example, if system finds out that some switch is turning on every time after doors are open, it will turn it on automatically. But it should consult his decisions in the beginning t avoid collisions with human decision (according to Asimov rules :) These shall avoid starting mixer while nothing in there, or even turning lights on when nobody is at home (only if it is necessary according to security policies).

The system shall communicate throw the network, which could be wired, or wireless. I search the technologies available and found some of them but I will need to decide which one to use. I prefer the "over power lines" transmitting because it is less expensive in real houses, but may be it's possible to use bluetooth or wi-fi as well.

Titel: Smart Home – Opportunity to make life easier

Author: Martin Hasaj

Director: MESEGUER PALLARÈS, Roc

Date: September 3, 2008

Dedication and acknowledgement

I would like to thank to all developers without whom consideration I wouldn't be able to develop this application:

To Sun Microsystems for **JAVA** and **Netbeans**

To Paolo Marrone for **Joone**

To J. A. Bayer and Fachhochschule Deggendorf for **KNX@HOME**

To Roc Messeguer, for his dedication towards his students and my work.

And to all my friends and colleagues for being patient while writing this project.

INDEX

INTRODUCTION

A great deal of contemporary research is showing that it is not work that goes home but home that goes to work. Basic Idea of all this research is to make life as easy for inhabitants as possible. “Make control process disappear and you have succeed”.

For example, if system gather that whenever somebody opens doors to the bathroom, he turns the lights on. It can automatically turn on lights whenever doors are opened. But it should consult its actions to avoid collisions with human decision (according to Asimov rules). These shall avoid starting empty devices if it is not necessary, or even turning lights on when nobody is at home and wasting of energy (except necessary rules according to security policies).

In 2001, Orange, a UK mobile network operator, announced the “Orange at Home” project, a smart house incorporating the latest technology wizardry built some 20 miles north of London. It was intended to be more than a mere showcase, with plans for real families to move in and live with the smart home.

In 2002, Intel launched the proactive health research project, while in 2000, Sun Microsystems and Invensys announced the development of "smart home" infrastructure to accelerate the delivery of services to wired appliances in the home. Also, C.P. Technology has used IBM's web sphere and developed the high tech "smart home" for marketing from the year 2004.

More applications are discussed at Chapter 2.7 with concern on Artificial Neural Network in Smart Homes environment.

My goal is to develop autonomous smart home system, which will be portable, robust, extensible and smart enough to help inhabitants make life easier in everyday situations without disturbing them. I want to prepare system, which will be easily reconfigured for new sensors and controllers set configuration without necessary longterm time consuming configuration and debugging process. This can be achieved by selecting technologies which can learn and adapt to changing environment of smart homes autonomously.

More specifically, I would like to concentrate on development of implementation of neural network which will serve particular needs of virtual home inhabitants. These needs shall be defined as subset of real problems set. After proving ability to learn I would like to test behavior on real hardware. Furthermore, test possibility of solving more complex situations as well.

CHAPTER 1. SMART HOMES

1.1 Background of Smart Homes

To effectively address the needs within the realms of the 'Smart Homes'[1] environment requires a multidisciplinary collaboration. Such developments will usually involve specialists from Architecture, Computer Science, Electrical and Electronic Engineering and some applications related to the health sector may also require participation of professionals from Social Sciences, Medicine and Occupational Therapy. Within this text I will try to address only part of this huge discipline. I will discuss possibility of smart homes to help inhabitants in everyday routine without necessary system maintain which is common for smart homes in general.

The use of Smart Homes is focused on support of independent living needs. There is possibility of designing an intelligent monitoring system that can detect when an undesirable situation may be developing (hazard, security threat, etc). Although all people can be involved in such undesirable situations, elderly people and people with health problems require more exhaustive monitoring when they are not accompanied by a health care professional.

Smart Homes are usually equipped with embedded devices that can enhance the functionality of conventional domotics appliances. ‘Sensor’ is a keyword in this area. By this we understand a coupling of hardware and software that is installed in one or more locations of the house and can provide information about what it is happening inside the house or in its vicinity. Examples of these can be a movement sensor, a smoke alarm and a timer fitted in a microwave. There are also enriched devices which besides providing information can also accept it, e.g., a tap can provide information detailing how long it has been turned on for and conversely the water flow to the tap can be remotely stopped.

Most of the work focused in the literature is devoted to individual sensors/devices that can bring new services. My goal is to reach a second, and more evolved stage it has to introduce more AI-related problem solving strategies so that all the sensors available can be used to form : (a) a more robust system combining different sources of information in order to provide a more global insight into the activities of a patient and (b) a more flexible executing tool as the circumstances in which the semi-autonomous devices have to act can be better identified having a global view of the activities inside the home.

The essential point here is that, according to my experience, Smart Homes can be much more intelligent with less effort for maintenance than they currently are. Problem of nowadays systems is not their usability, but their user-unfriendly (or maintenance-unfriendly nature) which comparing to their high price inhibit their wide spreading between end users. On one side is nice to have perfect system prepared in lab, but if you have to spend few months to debug it before possibility to apply it commercial, it is not that much useful as it could seems.

1.2 *Smart Home Issues*

From recently developed Smart Homes system we can learn lot of essential data, which can help develop system which is accurate, easy to maintain an robust. Problems which exist are usually devoted from lack of contextual informations of systems. There are documented some issues when owners of such “SMART HOMES” were complaining about their houses controlling them instead of vice versa desired behavior. For instance there were incidents of doors opening automatically in front of possible thief while owner were walking toward doors and otherwise had been closed and locked after owner went out to check his mailbox and kept doors open, because he left keys at home. This undesired behavior must be avoided under any circumstances, that's why is important to implement security policies, which prevents this kind of behavior of AI Smart Homes.

1.3 *Area of Possible AI applications*

AI techniques can be used to contribute in many ways to the development of Smart Homes. The Sections below exemplify this further, however, this should not be considered as an exhaustive set of possibilities but more an opportunity to offer some concepts of how the various levels of applications waiting to be explored may be deployed.

1.3.1 Spatio-temporal Reasoning

Analysis of the activities developed in the Smart Home should not be based on visual aids due to privacy issues. Instead it should be based on the information obtained from the sensors. It is vital in this process for the system to be aware of the places visited by the inhabitant and the order of events generated as well as their duration.

For example, it is fundamental to the understanding of a situation to be able to consider in which order the places where visited and realizing that by being in bed last the resident is now in a passive position and hence is more likely to forget about taking care of what has been left on the cooker. Detection of position can be achieved via the movement sensors in each room; to disambiguate some situations RFID detectors are needed which helps to distinguish between more residents of same flat or even visitors and pets.

Spatio-temporal reasoning is fundamental here to reason about where the resident can be and according to the time elapsed to diagnose if action is required. The whole situation of leaving the cooker on followed by a period of time where the resident goes outside the kitchen leaving the cooker unattended is a complex event, depending on the condition that it takes more time than it is considered safe, this may or may not constitute a situation of interest for the monitoring system. Here *qualitative* as well as *quantitative* temporal reasoning is required to differentiate between these situations.

1.3.2 Temporal Granularity

Activities can be monitored at many different levels in terms of information granularity. Choosing the right level of information granularity is not trivial and can have important consequences in terms of what the system can achieve and how well it can perform in terms of efficiency.

There are reasoning activities that may demand to analyze only one, two or three successive activities and for that a few seconds are enough but to infer a trend a much more global view spanning for several minutes or days may be needed. For some diagnostic tasks low levels of time granularity are needed to allow the system to infer that the inhabitants are in regular movement because of the speed at which they are moving through the successive rooms. But for other diagnostic tasks a way of abstracting different views of the system in terms of the different granularity levels of temporal information is very important.

1.3.3 Causal Reasoning

Certainly the relation between the actions performed by the inhabitants and the resulting states is a fundamental part of an automated diagnosis system however this issue has not yet been considered from the perspective of information analysis in Smart Homes and is not clear, for example, what system will be most effective in predicting possible outcomes given a trend of behavior of residents.
Suppose the cooker has no timer, it will stay on and if it persists in that state for a sufficiently long time the food may be burnt with the consequent possibility of causing a fire and subsequently generating a call to the fire brigade. Persistency coupled with causal reasoning is fundamental here to predict possible outcomes and help the system to react in order to avoid possible problems. Also of relevance here is the detection of complex events.

1.3.4 Planning

Planning can be applied to the use of Smart Homes for health care. For example, according to the patient’s demands for medicines, a plan can be constructed regulating medication intake hence its application or realization by the patient can be monitored by the system reporting instances of non-compliance or significant deviations from the prescribed medication regimen.
Other scenario is shopping planing. Before your fridge go out of food system can plan shopping event, enter order into e-shop and let it deliver on time. Which means that resident wouldn't realize it was running out of stock.

1.3.5 Learning

Learning about the behavior of the inhabitants is essential in order to self-tune the system to further support the personalized independence the Smart Home can offer. Even when a system is prescribed for a Smart Home environment, different homes would have different conditions and habits which have to be taken into account for the system to be useful. For example, the normal time for a resident to be in bed may be from 11PM to 7AM and s/he may be expected every day to be up and active at 7:30 AM. Detecting this is not the case will be a cause for concern, while for other at 9AM will be the normal time and being waken up at 7:30 AM by automatic coffee machine would be inappropriate.

Data Mining is useful here in order to learn new information from the statistics of the system, which can then be focused on a single person, group of residents sharing a specific condition, or in a more heterogeneous group of people using system of Smart Homes. These learning capabilities are essential for correct predictions.

1.3.6 Case-Based Reasoning

CBR can be used to identify patterns of person behavior. This can be used for diagnosis of situations which need urgent reaction on behalf of the system.
As shifts in habits are detected and learned by the system CBR-based techniques can be applied, for example, to diagnose when these shifts are similar or dissimilar to patterns of behavior which constitutes a reason for concern.

Sometimes deviations from the expected time for taking meals or going to bed can be expected as daily activities can be changed by visits or exceptional events which are not related to long term change of habits and don't need any change of learned patterns. Differentiating these issues on the basis of contextual and historical information poses a real challenge for a diagnostic system to give useful advice whilst minimizing the chances of making false decisions.

1.3.7 Decision Trees

Decision Trees may be used to monitor the sequence of events within the home and allow the monitoring system to react accordingly, especially in instances of concern where the person requires a form of intervention. Within the home environment it is common that sequences of events will be repeated, some of which will lead to hazardous situations requiring intervention and others can be considered as normal behavior. These events, although occurring at different instances in time, should result in similar resulting supporting actions by the system. Although finite, it is possible to identify a large set of repeating actions with moderately varying degrees of support required.

For example, the system suggesting a possibility that the TV has been left on has different risks involved than when it is suggesting the possibility that the cooker has been left on. Although the aforementioned are at the most simplistic levels they may be supported by such an approach. More intricate sequences and outcomes can be detected by longer term monitoring and profiling of activities which can be used as the basis to form the decision model and where appropriate introduce the notion of time.

1.3.8 Neural Networks

Neural Networks have long been established as powerful tools for classification and prediction. The vast array of topologies and supervised and unsupervised methods of learning offer the potential for a number of solutions to the management and processing of information within the Smart Home. For example, from a monitoring perspective Time Delay Neural Networks can be used to represent temporal relationships and hence within the given context of Smart Homes be used to predict likely activities of the person based on a preceding set of actions over time. Considering the scenario described in beginning of this chapter, the activities of the person and their time dependencies can be used as continuous inputs to the network and the output of the network can be trained to either automatically ’turn off the cooker’ as in the scenario described above or for a more generic sequence of time related events, automatically raise an alarm to the authorities.

1.3.9 Multi-agent Systems

MAS can be used as a development paradigm where a house is monitored for a group of agents with different skills and duties. The application of agent-oriented technology to smart homes environment is an area which is receiving increasing attention in the research community especially in medical care.

The mentioned scenario can be assisted at a design level by including MAS-related developing methodologies where different parts of the system are seen as interacting agents with specific skills and monitoring duties. For example, an agent can be designed to monitor activities in the kitchen and another in the living and another in the bedroom or there can be agents specialized in different types of sensors. Hence, while one is dealing with devices another is in charge of movement monitoring sensors. Conclusions in the system can be reached after deliberation of different agents and using different techniques related to the representation of consensus and disagreement.

CHAPTER 2. ARTIFICIAL NEURAL NETWORKS

2.1 Overview

An Artificial Neural Network (ANN)[2] or alternatively known as Neural Network is a computational tool, which follows the activities of a biological brain. The basic processing unit of ANNs is a neuron [3], which has similarity with a biological neuron, first described by Cajal in 1911. ANNs learn relationships between the inputs and outputs from examples presented to the networks without any prior knowledge about the underlying relationships that exist within the data sets. This important 'adaptive' attribute of the ANNs makes it suitable for many applications where there is very little or no information available regarding the input-output relationships or the underlying transfer function among the data [4].

There are many standard texts [5][6] that describe the history of ANNs and the links that exist between the operations of the biological and artificial neurons. An ANN is usually characterized by:

- the way the neurons are connected to each other,
- the method that is used for the determination of the connection strengths or weights
  the activation function that is used to process the data.

Some basic features of ANNs are briefly described below:

2.2 Model of Neuron

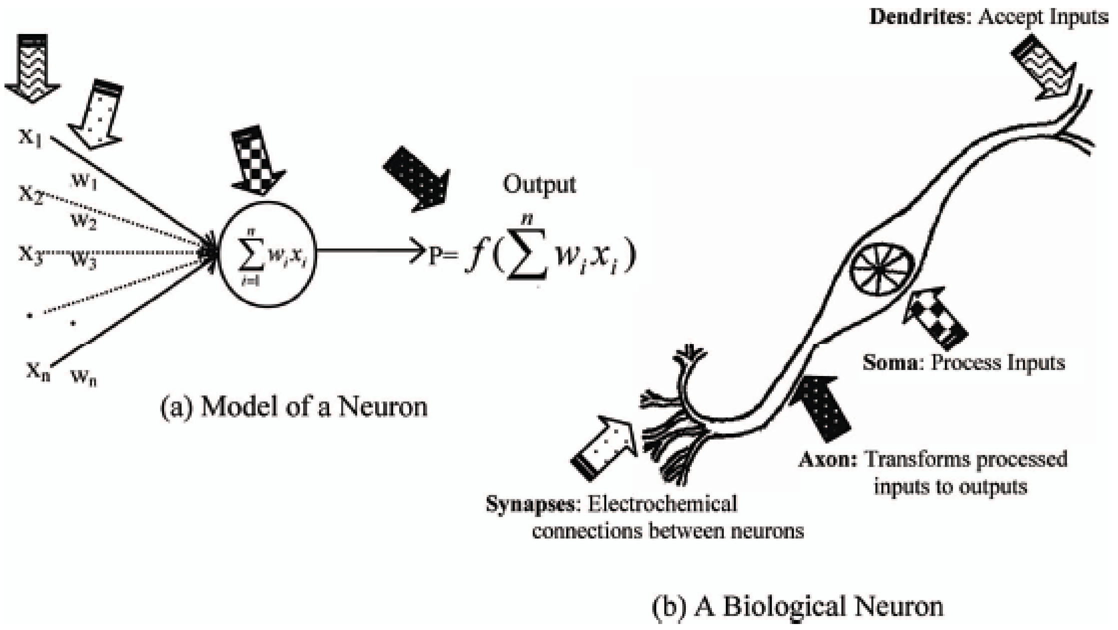


Fig. 2.1*:* Activity similarities between (a) a computing neuron and (b) a biological neuron

As already stated a neuron is the fundamental unit of the ANN. A neuron takes a number of inputs and after some mathematical processing it produces an output. Each neuron is connected to the other neurons by means of links that have weights, which represent strengths of the connections. Figure 1 compares a computational (artificial) neuron and a biological neuron. The inputs to the neuron signify activities of the *dendrites* of a biological neuron. The weights on the edges represent the activities of *synapses,* the mathematical equation shown in the circle is a representation of the *soma,* and the calculation of the output of each neuron signifies the *axon*. Generally, a neuron in an ANN has more than one input nodes but only one output. The weights shown on each connection line represent information to be used in processing the input signals to build the ANN model of the problem. The mathematical function used by each neuron to calculate the output is called the activation function. There are different types of activation functions used for processing the incoming signals, some of which are briefly discussed in section 2.5.

2.3 Network Architecture

The way the neurons are connected to each other can be visualized using the network architecture. Depending on the position of the neurons in an ANN three layers are defined: *input layer*, *hidden layer* and the *output layer*. There may be none or more than one hidden layer in an ANN which primarily depends on the complexity of the problem. Researchers have proposed different types of network architectures that are suitable for solving their own particular problem. However, the most popular and widely used network architectures include: feed-forward fully connected neural network, recurrent neural network and time-delay neural networks.

2.3.1 Feed-Forward Architecture

In a feed-forward neural network (Figure 2(a)), the neurons of each layer are connected to the next layer only in the forward direction, i.e., neurons are connected in unidirectional way and there is no backward connection among the neurons. Figure 2(a) illustrates the structure of a three layer feed-forward fully connected neural network. Feed-Forward ANN has a relatively simple architecture to map the input-output relationship to automate devices (e.g., face recognition, temperature control, fire alarm, etc), which can be applied in a smart home environment.

2.3.2 Recurrent Neural Network

The structure of a recurrent neural network (RNN) is the same as the feed-forward ANN except that, there is connection between neurons that feeds back the output of that neuron to the input of the previous neuron. Figure 2(b) illustrates the structure of a RNN. Here the outputs are fed back into the hidden layer, thus the structure representing a RNN. A RNN is normally efficient for dealing with sequential problem where the current state depends on past states of the same variable (e.g., human behavior prediction in a smart home).

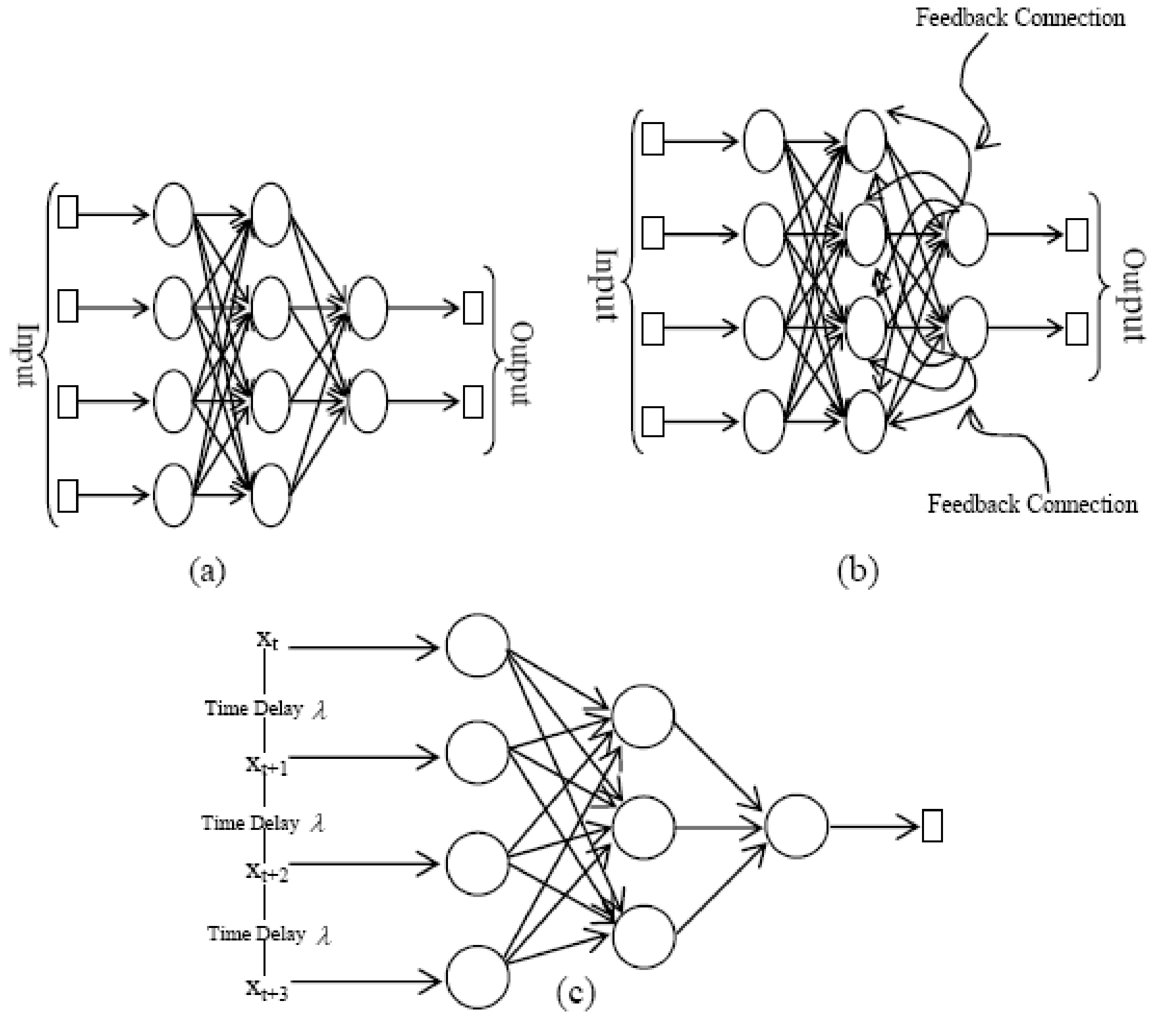


Fig. 2.2: Structure of a (a) Feed-Forward neural network (b) Recurrent neural network, RNN (c) Time-Delay neural network, TDNN

2.3.3 Time-Delay Neural Network

A time delay neural network (TDNN) or also known as Tapped Delay Line (TDL) is a temporal network where the input pattern is fed into the network by introducing a delay in time. The successive representation of input data sequences facilitates the network to adapt to the temporal behavior of the problem (e.g., daily activities of smart home inhabitant). TDNN can be quite useful for those problems where both the temporal information and the relationship between the input/output data sequences are important. Figure 2(c) illustrates the architecture of a TDNN, here the successive inputs have been delayed by □.

2.4 Learning process

One of the main characteristics that make ANNs unique from other mathematical processing methods is that it can learn from the characteristics of the data and adapt parameters of the network according to the underlying structures in the training dataset. The process is known as learning. As defined by Haykin [3]:

"*Learning is a process by which the free parameters of a neural network are adapted through a continuing process of stimulation by the environment in which the networks are embedded. The type of learning is determined by the manner in which the parameter changes take place*".

Depending on the learning environment there can be fundamentally two different learning paradigms: supervised learning and unsupervised or self-organized learning. As the name suggests, in a supervised learning paradigm the ANN is trained under some guided supervision, i.e., there is target output(s) that the ANN aims to achieve during the learning/training process. On the contrary, in unsupervised learning there is no external supervisor to facilitate ANN's learning: the distribution of the data and the structural similarities amongst the dataset are used as the basis for learning.

Over the years researchers have proposed many rules for training the ANNs that follow the learning technique. Some of the widely used supervised learning rules include:

1. Perceptron Learning rule
2. Error-Correction/ Delta rule
3. Hebbian Rule [7]

Among these rules, the Delta rule is the most widely used one.

2.5 Activation Function

The output of a neuron in the ANN architecture is generated using a mathematical function, which is known as activation function. Most often a nonlinear function is used as activation function so that the ANN structure becomes suitable to adapt to nonlinear problems. Two of the most popular nonlinear activation functions are:

- Sigmoidal function
- Signum function

Details of these functions can be found in [3].

2.6 Learning Function

There are several learning algorithms proposed in the literature that can be used to determine the appropriate weights of the ANNs. Some well-known learning algorithms include:

- Back propagation
- Radial Basis Function
- Hopfield Algorithm

Among the many learning algorithms, the back-propagation algorithm is widely used involving majority of the applications in a smart home environment (according to [2]). In a back propagation algorithm, the weights are modified first by calculating the error via some suitable error function. If a differentiable function is chosen as an error function, gradient descent on such a function will naturally lead to a learning rule. Historically, several researchers have proposed this idea e.g.[8]. Eventually, Rumelhart et al. [9] developed practical techniques that gave us a powerful engineering tool. The back-propagation technique follows the delta rule while modifying the network weights.

2.7 Application of Neural Networks in Smart Home Environment

In the above sections we have briefly discussed the operation and algorithms of ANNs. In the context of smart homes, ANN has been used for classification, control and automated home appliances, next step/action prediction, etc. In the following, lets review some of the significant applications involving ANNs approaches in smart home environment.

Chan et al. [10] appear to be the first to use ANN to help develop a smart home automation system for use by the elderly and disabled people. Altogether twelve rooms were included as part of the system where sensors were deployed in the rooms to collect information representative of the behaviors of the inhabitants. ANNs were trained using the acquired data to classify normal and abnormal behaviors.

The first comprehensive application involving ANNs in smart home environment was undertaken by Mozer [11] in 1998. It was named ACHE which stands for Adaptive Control of Home Environments. The system was able to alter the environmental system (e.g., air heating, lighting, ventilation, water heating) in line with the residential needs and comfort levels. Thus the home environment was in tune with the lifestyle and requirements of the residents. ACHE had two main goals. It was able to anticipate the inhabitants' needs such as lighting, air temperature, hot water, and ventilation by studying the patterns of adjustments made by the inhabitants. These settings acted as training signals for ACHE so that after some time it was able to anticipate the needs of the resident. The second objective was to conserve energy, i.e., meeting the inhabitant's demand with less energy consumption. To satisfy these two objectives the author introduced a cost function to optimize the comfort level of the resident with least possible amount of energy being consumed. Feed-forward ANNs trained with back propagation algorithm were reported for data analysis and controlling the home appliances.

Jorge and Goncalves [12] focused on automated health monitoring of elder citizens' everyday life using artificial intelligence tool. They used ubiquitous computing devices to collect data from the elderly relating to neurological disorders (e.g., loss of motor, sensorial and cognitive problems). The authors emphasized the usage of artificial intelligence to learn patterns from data produced by ubiquitous computing devices and then to use them to predict next activities. For instance, for patients requiring constant medical monitoring should be considered differently as they might need special needs and care. So, the challenge for the system is to take a decision whether someone should be classified under this group or not based on the available information. For such a problem, the rules and patterns that are inferred by the system should be unique and individual-specific, and in this regard an artificial intelligence tool like the ANN technique could potentially play a significant role in decision-making.

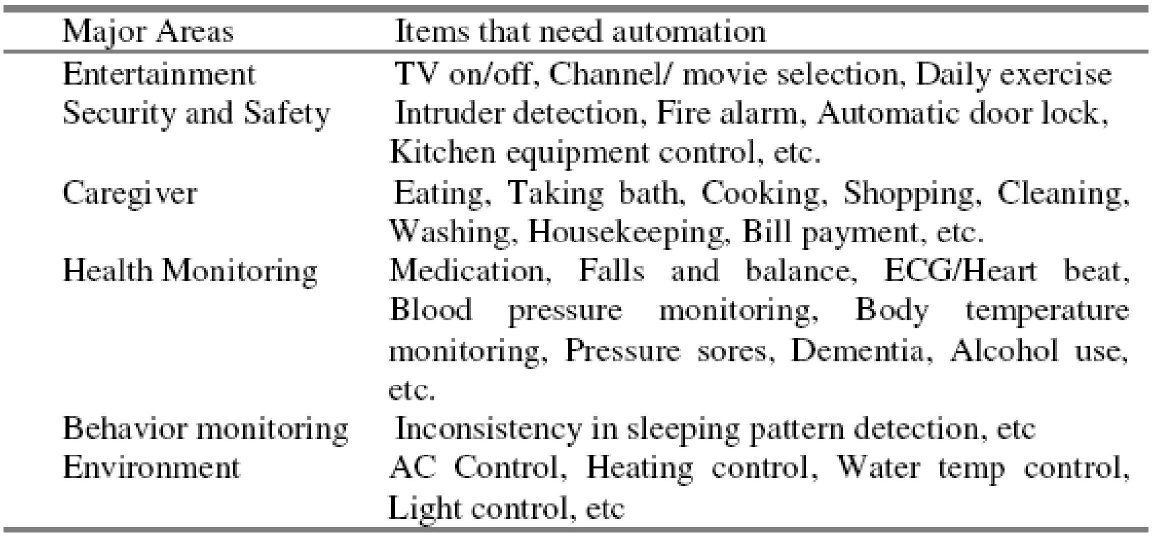
Pigot et al. [13] proposed a theoretical as well a practical framework with a view to minimizing risks that may arise due to the actions undertaken by the elderly in a physical environment. In this framework, the computer infrastructure of the smart home system consisted of three layers: The application layer, the middleware layer and the hardware layer. The hardware layer included electrical equipment, e.g. sensors [14]. The middleware connected the devices in the hardware layer with the application layer. The application layer was the decision making layer which took decision about the action to be undertaken in the next step. Tele-monitoring module was one of the application modules in the decision-making process to detect abnormal behaviors of the monitored person. The authors emphasized the advantage of applying ANNs along with other mathematical models to help in the detection of patterns associated with a risk.

Cook et al. [15] developed a project called MavHome. This project focuses on the creation of an environment that acted as an intelligent agent. The main operations of MavHome included the control of temperature, electric appliances, TV, video, robot, etc. The MavHome was organized into an architecture, which connected the agent and the technologies within each agent. The decision layer was one of these four layers. This layer selected actions for the agent to be executed based on information supplied from the other layers. MavHome has been defined as a unique agent, which combined technologies from artificial intelligence, machine learning, databases, mobile computing, robotics, and multimedia computing. To acquire and recognize the activities of the inhabitants within the MavHome a number of prediction algorithms were employed in parallel: the SHIP prediction algorithm, Active LeZi (ALZ) online algorithm, and a data-mining algorithm known as Episode Discovery. In the final decision phase the MavHome developed a metapredictor called Predict to select one of the predicted actions obtained from the aforementioned predictors. In predict algorithm, a back propagation ANN was applied to learn the confidence value for each of the prediction algorithms based on the gathered data, the accuracy strength of each of the predictors and the meta features (i.e., amount of data, number of devices, etc).

R-Illengworth et al. [16] presented novel connectionist embedded agent architecture to recognize daily life behaviors. A number of unobtrusive and relatively simple sensors were used in this architecture. Different high-level daily routine activities (e.g., sleeping, eating) were recognized with the help of a constructive algorithm with temporal capabilities. Consequently, abnormal behaviors were also identified using this algorithm. The focus of the task was the development of some connectionist approach which would overcome the difficulties of adapting neural network to the continuous changes in the smart home, and eliminate the requirement of retraining the ANN to take into consideration the newly added input/output nodes (if any). The authors used adaptive neural architecture, first proposed by Kasabov [17], known as EcoS (Evolving Connectionist System), which was found suitable for the environmental data. A memory layer was added to the model to handle temporal component i.e., to detect abnormalities in the sequence, frequency and duration of the activities. Being RNN in nature, there were feedback connections between the two layers (memory and hidden layers) that followed the architecture proposed in [18]. The authors applied this model, in conjunction with 18 sensors, for activity recognition and to identify abnormalities inside a dormitory room at the University of Essex.

2.8 ANNs for Control of Smart Home Devices and Appliances

People have defined smart homes from different contexts. Smart homes have been designed with the primary aim of helping people with special needs, adapt to meet the user's limitation and also to improve the general lifestyle of the inhabitants. Accordingly, the use of devices within the home environment and the associated models vary. Some application examples of smart homes might be for people with movement disabilities that may require assistance with mobility, for elderly people to help with balance problem, and for people with low hearing and vision requiring specialized communication and lighting devices. Furthermore, smart homes are designed with a focus towards minimizing caregiver support for the people who would like to live independently at home. Table 1 lists some of the main items that might require automation and control in a smart home.



Major Areas	Items that need automation
Entertainment	TV on/off, Channel/ movie selection, Daily exercise
Security and Safety	Intruder detection, Fire alarm, Automatic door lock, Kitchen equipment control, etc.
Caregiver	Eating, Taking bath, Cooking, Shopping, Cleaning, Washing, Housekeeping, Bill payment, etc.
Health Monitoring	Medication, Falls and balance, ECG/Heart beat, Blood pressure monitoring, Body temperature monitoring, Pressure sores, Dementia, Alcohol use, etc.
Behavior monitoring	Inconsistency in sleeping pattern detection, etc
Environment	AC Control, Heating control, Water temp control, Light control, etc

Table 1: List of some of the main devices that require automation in smart home

2.8.1 Entertainment

There are different kinds of equipment that may need automation to provide entertainment to the inhabitants in a smart home. Examples include the provision of an automated TV control that may need turning on and off according to the daily routine of the inhabitant. A trained ANN can be utilized to make this action automated via information collected from various sources (e.g., time/day and activity of the inhabitant such as morning exercise, etc). Chan et al. [10] has already demonstrated the usefulness of ANNs in the control of TV in their smart home. ANNs can also be programmed to expand the controls of TV/ Video/DVD during any time of the day by feeding the requisite information relating to the behavior and activities of the inhabitant. Similarly, ANNs can be employed for automated control of daily exercises. For example, it can be used to activate reminder notes for exercise and to regulate the air temperature of the exercise room (e.g., using a TDNN).

2.8.2 Safety and Security

One of the major aims of smart homes is to provide a secure and safe environment for their inhabitants. This may include intruder or unwanted person detection and automatic activation of, for example, fire alarm for 24 hours. With regard to the previous task, computerized technique involving face detection, person and object identification can play a vital role. Recently, many research projects have been undertaken pertaining to the use of ANNs for face detection and recognition, see for example [19]. In some researches, it has been shown that ANNs are able to recognize face with very high success rate (82% to 100%), which suggests their suitability for application in smart home environment for intruder detection, and subsequent activation of appropriate security services such as the police or security guard.

ANNs may also be used to assess fire hazards in the home by studying smoke alarm signals from various parts in the house. In [20], the researchers have developed satellite-based remote sensing techniques for identifying smoke from forest fires. It seems that with little modification, this concept could be applied in smart homes to detect and assess fire hazards using a dedicated ANN system for intelligent decision making as to whether to activate or not the fire alarm system [20].

2.8.3 Caregiver

In this category, issues are discussed to help the inhabitants of smart homes to do his or her daily routine activities, such as eating, house keeping, cleaning, etc. Already significant progress has been made in this area, especially with the help of indoor robots. In [21], the authors have employed ANNs to control a robot within the indoor environment as a personal assistant for indoor service applications such as surface cleaning, distribution of meals, etc. Such type of controlled robot has the potential to be employed in smart home environment to carry out various tasks, e.g. to deliver meal according to the daily routine of the inhabitant. ANNs have played a substantial role in many such control applications, for example, helping the robots to navigate using sensor signals taken from the surrounding environment or monitoring the time of taking meal.

Besides robots, ANNs have been valuable for doing many other household duties and tasks. For example, for automated control of bathtub water temperature within a comfortable level, both back propagation ANNs [22] and hybrid systems (neuro-fuzzy inference network) have been proposed [23]. ANNs, with the help of speech recognition techniques, could also potentially provide assistance with the online shopping activities undertaken from within the home environment.

2.8.4 Health Monitoring

Continuous health monitoring capability offers enormous benefits to individuals living in smart homes, especially for older persons and individuals with chronic diseases (heart, neurological disorders, etc) to have a better quality of life. Here the idea is to have sensor devices that would constantly deliver information about the health status of the inhabitants. ANNs or other artificial intelligence tools might be used for processing medical and physiological data, and to take intelligent decision about the overall health condition of the subject. ANNs could also facilitate information transfer to medical and health professionals in case of detected abnormality in recorded signals that would warrant urgent medical attention. Some of the examples include routine medication surveillance, falls event detection in older adults, heartbeat and ECG monitoring, blood pressure monitoring, EEG monitoring in dementia patients, etc. Already a substantial body of literature is available describing various issues related to health monitoring and using sensors placed in smart homes (e.g., bed, bath, toilet, etc) to obtain various physiological and health information data. ANNs can play a vital role in this context for processing various biomedical signals and assisting to diagnose diseases from the acquired biosignals [24] and also, for example, to develop ANN-based medical decision support system [25].

CHAPTER 3. FUZZY SYSTEMS

Since human speech and communication use fuzzy variables that have inexact bounds, fuzzy systems present an easy way to communicate between humans and computers. [26] proposed that by using fuzzy systems a more natural and realistic environment can be established.

3.1 Fuzzy logic

Fuzzy systems theory is based on uncertainty and on imprecision. Uncertainty is very natural to humans and people usually make decisions based on indiscreet observations. [27] A man can reason that he will not need an umbrella when going outdoors, if it does not rain much. Here the meaning of the fuzzy word ‘much’ is dependent on context and from the point of comparison.

The presentation of uncertainty is complicated to a computer using traditional methods. Among scientist uncertainty has been considered to be an inevitable part of speech and a precise expression has been an objective of research for a long period of time. [27] For example, for a computer it is very hard to understand what dark and bright means. Let us define a border between dark and bright to 100 luxes. If the sun shines outdoors, the illumination can be measured to be 50 000 luxes. Here it is quite easy to say that it is bright outside. However, if we measure a workspace to have 110 luxes, the room is still brightly illuminated.

That doesn’t sound very meaningful to a man. Hence, the borderline cases are difficult to handle. The problem exists in the binary logic of the computer. Outdoor light level can be either dark or bright. Fuzzy logic presents a solution called multi-value logic. Using multi-value logic illumination level can be dark and bright at the same time, that is more than two truth-values exist. Here, it possible for illumination level to be at the same time much bright but little dark. There is no need to define exact bounds for continuous values and the different regions of truth can overlap each other. [27]

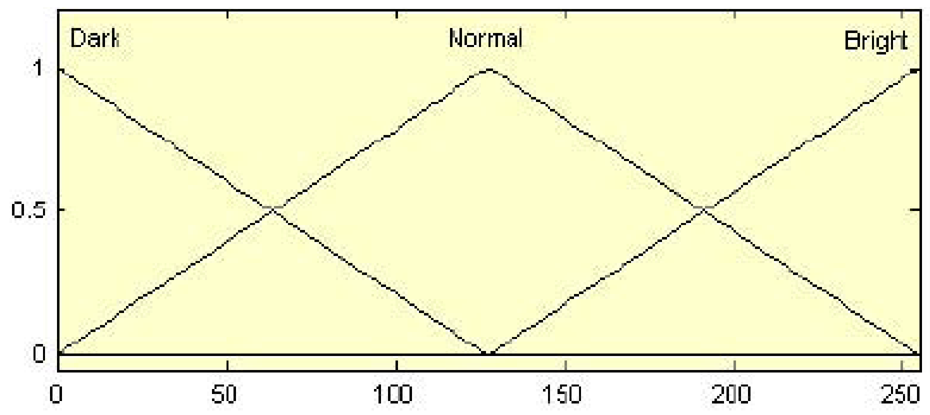


Fig. 3.1: Linguistic variable 'luminance' defined with three membership functions

Linguistic variables are used in fuzzy systems as a method for performing calculations. The values of linguistic variables can be presented using membership functions that define the degree of membership in real input or output spaces. Figure 3 demonstrates how the linguistic variable ‘luminance’ is defined with values ‘dark’, ‘normal’ and ‘bright’. For example, with a real input value of 150 from a light sensor luminance is characterized to be dark in degree of 0, normal in degree of 0.8 and bright in degree of 0.2.

3.2 Fuzzy control

Fuzzy control systems have many abilities that conventional control systems don’t possess because of their multi-value logic. Fuzzy control includes three major steps: fuzzification, fuzzy inference and defuzzification. [27]

Fuzzification In order to be able to do fuzzy inference the real measurement values must be fuzzified. Fuzzification is performed using membership functions that define the input spaces and the values’ degrees of membership for each variable separately. During fuzzification, every value of a linguistic variable receives a degree of membership based on the definition of the membership function and on the real measurement. An example of this was already presented with the linguistic variable ‘luminance’ in the last chapter.

Fuzzy inference Fuzzy systems utilize a relatively simple rule table that represents the behavior of the whole system. The rules are of ‘if – then’ type. In the ‘if’ part the inputs’ and in the ‘then’ part the outputs’ linguistic variables’ values are defined. ‘If’ part determines whether the rule is or is not valid in the current situation. ‘Then’ part is used to define the states of the outputs.

All the fuzzified input values must be aggregated to achieve a complete degree of truth to a rule. The aggregation is usually done using a minimum operator that takes the smallest degree of membership of the values of the input variables.

Composition follows next from the input aggregation. In the composition step the aggregated degree of truth of the ‘if’ part is multiplied by a weighing factor to get a degree of support (DoS) for the whole rule. Weighing factor or weight is associated with each rule and describes the significance of the rule. The weighing factor is usually defined to be in the interval [0, 1].

Since many of the rules can be true at the same time to a different degree, the result must be aggregated for each output variable’s value separately. This is usually done by selecting the biggest result after the composition step controlling the output’s value. This value determines the linguistic variables’ output’s degree of membership.

Defuzzification After result aggregation the linguistic variables are defuzzified to real output signals to actuators. Defuzzification can be done in several ways.

One common method is the center of gravity defuzzification. All the values of the output linguistic variables are filled from zero to the degree of the membership of the value. Then the center of the filled area is calculated and the real result is read from that point.

3.3 Fuzzy ART Neural Network

Fuzzy ART has been introduced in the neural network literature by Carpenter[28], et al., 1991, and since then it has been established as one of the premier neural network architectures in solving classification problems. To deal with supervised learning tasks, the so-called Fuzzy ARTMAP classifier was developed around the combination of two Fuzzy ART modules by Carpenter, et al., 1992.

To reduce the sensitivity of Fuzzy ARTMAP to the order of training samples, output combinations of independently trained Fuzzy ARTMAP systems were proposed, such as Williamson[29] proposed a Fuzzy ARTMAP based on the Gaussian radix in 1996, Verzi improve Fuzzy ARTMAP in 1998,and proposed BARTMAP and Eduardo proposed a improve neural network in 2002.

But all these algorithms proposed are sensitive to the vigilance, and with the increase of the vigilance the output patterns increases saliently, and the output patterns are so similar to each other, the false segment rate increases. To solve this problem, Zhang[30] proposed a new Fuzzy ART neural network model based on dual competition and resonance technique. This new model takes the competition and resonance method of the input nodes into the output nodes, and combines the output pattern according to the vigilance to avoid high similarity between the output patterns.

CHAPTER 4. GENETIC ALGORITHMS

Genetic algorithms[31] are a part of evolutionary computing, which is a rapidly growing area of artificial intelligence. Obviously, they are inspired by Darwin's theory of evolution. Simply said, solution to a problem solved by genetic algorithms is evolved.

4.1 History

Idea of evolutionary computing was introduced in the 1960s by I. Rechenberg in his work "Evolution strategies" [32]. His idea was then developed by other researchers. Genetic Algorithms (GAs) were invented by John Holland and developed by him and his students and colleagues. This lead to Holland's book published in 1975.[33]

In 1992 John Koza has used genetic algorithm to evolve programs to perform certain tasks. He called his method "genetic programming" (GP). LISP programs were used, because programs in this language can expressed in the form of a "parse tree", which is the object the GA works on.

4.2 Biological background

4.2.1 Chromosome

All living organisms consist of cells. In each cell there is the same set of chromosomes. Chromosomes are strings of DNA and serves as a model for the whole organism. A chromosome consist of genes, blocks of DNA. Each gene encodes a particular protein. Basically can be said, that each gene encodes a trait, for example color of eyes. Possible settings for a trait (e.g. blue, brown) are called alleles. Each gene has its own position in the chromosome. This position is called locus.

Complete set of genetic material (all chromosomes) is called genome. Particular set of genes in genome is called genotype. The genotype is with later development after birth base for the organism's phenotype, its physical and mental characteristics, such as eye color, intelligence etc.

4.2.2 Reproduction

During reproduction, first occurs recombination (or crossover). Genes from parents form in some way the whole new chromosome. The new created offspring can then be mutated. Mutation means, that the elements of DNA are a bit changed. This changes are mainly caused by errors in copying genes from parents.

The fitness of an organism is measured by success of the organism in its life.

4.3 Genetic Algorithm basics

Genetic algorithms are inspired by Darwin's theory about evolution. Solution to a problem solved by genetic algorithms is evolved.

Algorithm is started with a set of solutions (represented by chromosomes) called population. Solutions from one population are taken and used to form a new population. This is motivated by a hope, that the new population will be better than the old one. Solutions which are selected to form new solutions (offspring) are selected according to their fitness - the more suitable they are the more chances they have to reproduce.

This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied.

Basic Routine:

1. **[Start]** Generate random population of *n* chromosomes (suitable solutions for the problem)
2. **[Fitness]** Evaluate the fitness *f(x)* of each chromosome *x* in the population
3. **[New population]** Create a new population by repeating following steps until the new population is complete
    1. **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
    2. **[Crossover]** With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
    3. **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).
    4. **[Accepting]** Place new offspring in a new population
4. **[Replace]** Use new generated population for a further run of algorithm
5. **[Test]** If the end condition is satisfied, **stop**, and return the best solution in current population
6. **[Loop]** Go to step **2**

As you can see, the outline of Basic GA is very general. There are many things that can be implemented differently in various problems.
First question is how to create chromosomes, what type of encoding choose. With this is connected crossover and mutation, the two basic operators of GA. Encoding, crossover and mutation are introduced in next chapter.
Next question is how to select parents for crossover. This can be done in many ways, but the main idea is to select the better parents (in hope that the better parents will produce better offspring). Also you may think, that making new population only by new offspring can cause lost of the best chromosome from the last population. This is true, so called elitism is often used. This means, that at least one best solution is copied without changes to a new population, so the best solution found can survive to end of run.

4.4 *Operators of Genetic Algorithms*

As you can see from the genetic algorithm outline, the crossover and mutation are the most important part of the genetic algorithm. The performance is influenced mainly by these two operators. Before we can explain more about crossover and mutation, some information about chromosomes will be given.

4.4.1 Encoding of a Chromosome

The chromosome should in some way contain information about solution which it represents. The most used way of encoding is a binary string. The chromosome then could look like this:

Chromosome 1	1101100100110110
Chromosome 2	1101111000011110

Each chromosome has one binary string. Each bit in this string can represent some characteristic of the solution. Or the whole string can represent a number. Of course, there are many other ways of encoding. This depends mainly on the solved problem. For example, one can encode directly integer or real numbers, sometimes it is useful to encode some permutations and so on.

4.4.2 Crossover

After we have decided what encoding we will use, we can make a step to crossover. Crossover selects genes from parent chromosomes and creates a new offspring. The simplest way how to do this is to choose randomly some crossover point and everything before this point point copy from a first parent and then everything after a crossover point copy from the second parent.
Crossover can then look like this (| is the crossover point):

Chromosome 1	11011 | 00100110110
Chromosome 2	11011 | 11000011110
Offspring 1	11011 | 11000011110
Offspring 2	11011 | 00100110110

There are other ways how to make crossover, for example we can choose more crossover points. Crossover can be rather complicated and very depends on encoding of the encoding of chromosome. Specific crossover made for a specific problem can improve performance of the genetic algorithm.

4.4.3 Mutation

After a crossover is performed, mutation takes place. This is to prevent falling all solutions in population into a local optimum of solved problem. Mutation changes randomly the new offspring. For binary encoding we can switch a few randomly chosen bits from 1 to 0 or from 0 to 1.

Mutation can then be following:

Original offspring 1	1101111000011110
Original offspring 2	1101100100110110
Mutated offspring 1	1100111000011110
Mutated offspring 2	1101101100110110

The mutation depends on the encoding as well as the crossover. For example when we are encoding permutations, mutation could be exchanging two genes.

4.5 Parameters of Genetic Algorithms

There are two basic parameters of GA - crossover probability and mutation probability:

Crossover probability says how often will be crossover performed. If there is no crossover, offspring is exact copy of parents. If there is a crossover, offspring is made from parts of parents' chromosome. If crossover probability is **100%**, then all offspring is made by crossover. If it is **0%**, whole new generation is made from exact copies of chromosomes from old population (but this does not mean that the new generation is the same!).

Crossover is made in hope that new chromosomes will have good parts of old chromosomes and maybe the new chromosomes will be better. However it is good to leave some part of population survive to next generation.

Mutation probability says how often will be parts of chromosome mutated. If there is no mutation, offspring is taken after crossover (or copy) without any change. If mutation is performed, part of chromosome is changed. If mutation probability is **100%**, whole chromosome is changed, if it is **0%**, nothing is changed.

Mutation is made to prevent falling GA into local extreme, but it should not occur very often, because then GA will in fact change to **random search**.

There are also some other parameters of GA. One also important parameter is population size:

Population size says how many chromosomes are in population (in one generation). If there are too few chromosomes, GA have a few possibilities to perform crossover and only a small part of search space is explored. On the other hand, if there are too many chromosomes, GA slows down. Research shows that after some limit (which depends mainly on encoding and the problem) it is not useful to increase population size, because it does not make solving the problem faster.

4.6 Selection

As you already know, chromosomes are selected from the population to be parents to crossover. The problem is how to select these chromosomes.

According to Darwin's evolution theory the best ones should survive and create new offspring. There are many methods how to select the best chromosomes, for example roulette wheel selection, Boltzman selection, tournament selection, rank selection, steady state selection and some others. Some of them will be described in this chapter.

4.6.1 Roulette Wheel Selection

Parents are selected according to their fitness. The better the chromosomes are, the more chances to be selected they have. Imagine a **roulette wheel** where are placed all chromosomes in the population, every has its place big accordingly to its fitness function, like on the following picture.

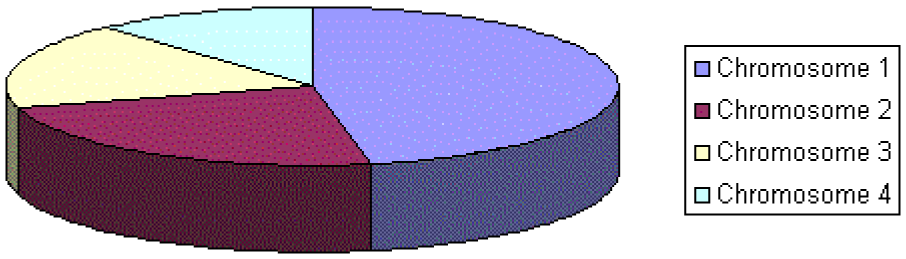


Fig. 4.1: Roulette Wheel Selection

Then a marble is thrown there and selects the chromosome. Chromosome with bigger fitness will be selected more times.

This can be simulated by following algorithm.

1. **[Sum]** Calculate sum of all chromosome fitness in population - sum ***S***.
2. **[Select]** Generate random number from interval ***(0,S)*** - ***r***.
3. **[Loop]** Go through the population and sum fitness from ***0*** - sum ***s***. When the sum ***s*** is greater then ***r***, stop and return the chromosome where you are.

Of course, step **1** is performed only once for each population.

4.6.2 Rank Selection

The previous selection will have problems when the fitness differs very much. For example, if the best chromosome fitness is 90% of the entire roulette wheel then the other chromosomes will have very few chances to be selected.

Rank selection first ranks the population and then every chromosome receives fitness from this ranking. The worst will have fitness ***1***, second worst ***2*** etc. and the best will have fitness ***N*** (number of chromosomes in population).

You can see in following picture, how the situation changes after changing fitness to order number.

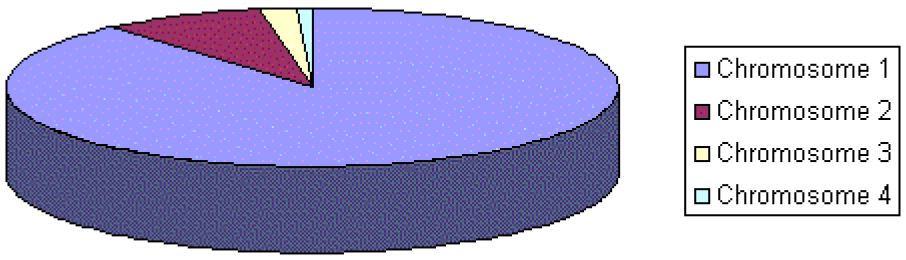


Fig. 4.2: Situation before ranking (graph of fitness)

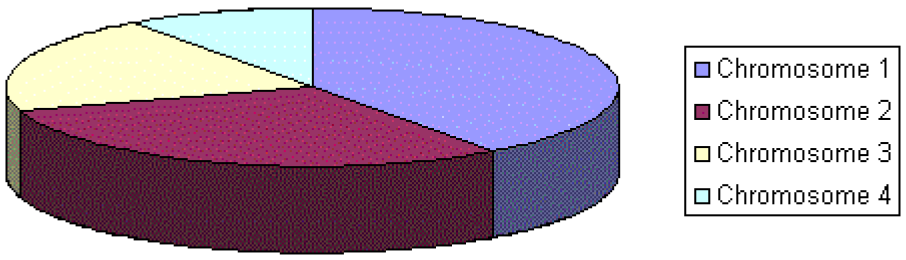


Fig. 4.3: Situation after ranking (graph of order numbers)

After this all the chromosomes have a chance to be selected. But this method can lead to slower convergence, because the best chromosomes do not differ so much from other ones.

4.6.3 Steady-State Selection

This is not particular method of selecting parents. Main idea of this selection is that big part of chromosomes should survive to next generation.

GA then works in a following way. In every generation are selected a few (good - with high fitness) chromosomes for creating a new offspring. Then some (bad - with low fitness) chromosomes are removed and the new offspring is placed in their place. The rest of population survives to new generation.

4.6.4 Elitism

Idea of elitism has been already introduced. When creating new population by crossover and mutation, we have a big chance, that we will loose the best chromosome.

Elitism is name of method, which first copies the best chromosome (or a few best chromosomes) to new population. The rest is done in classical way. Elitism can very rapidly increase performance of GA, because it prevents losing the best found solution.

CHAPTER 5. KNX@HOME

KNX@Home[34] is an open source free building control server developed at the University of Applied Sciences in Deggendorf, Germany[35]. The project is managed by Professor Dr.-Ing. Andreas Grzemba from the Institute of Electrical Engineering. Goal is to offer an easy-to-use web based graphical user interface to everybody whose house has got an integrated KNX/EIB-Bus.

The main goal of the application is to allow the user to control his KNX-BUS installation with his home desktop computer. This goal is to be achieved by using the KNXLive! from the Vienna University of Technology, which is an operating system that starts directly from CD. There is also a Teach-In-Module that allows the user to simply add elements from the KNX-Bus-System to the KNX@HOME Application.

The application should:

- Be able to control a KNX installation of arbitrary size
- Be Controlled and presented web based
- Be flexible and adapted to the users' needs
- provide full functionality to the user for manual configuration
- be simple

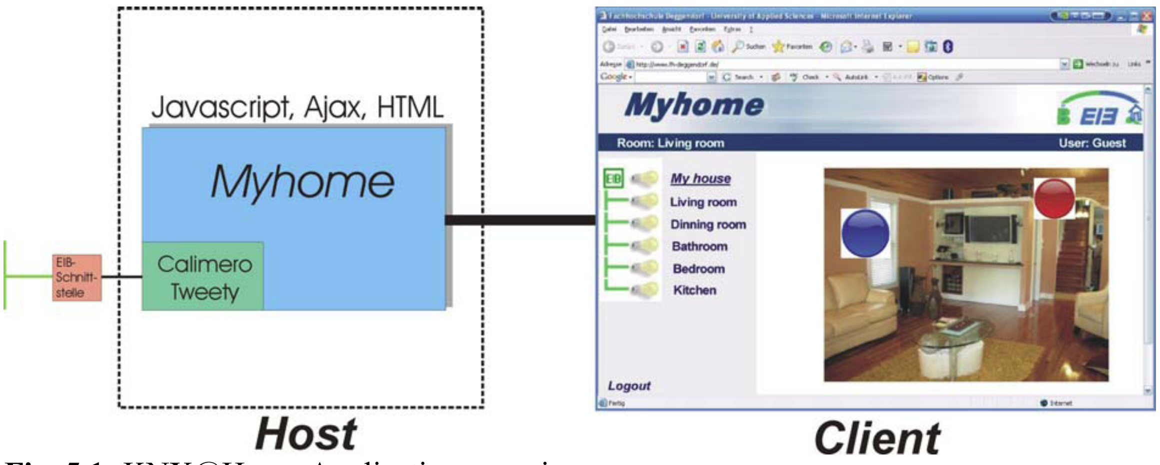


Fig. 5.1: KNX@Home Application overview

5.1 Basic conditions for the live CD

The whole applications (Calimero, Persister, web server) should be integrated into a GNU Linux Live CD, like Knoppix.

Because of this, the programs have to get by without a hard disc and with the RAM of an average computer. In fact this is not a big problem, except the database, which requires the access on a hard disc, USB-stick, SD-card, etc.

Several Live distributions can handle an available empty partition on a data medium to store the database.

Once the Live CD has started a complete Linux with a graphical user interface like KDE, the user will be able to use the main application immediately by opening a browser like Konqueror or Firefox. As home page, the browser will show the welcome screen of the web interface.

Due to this, it will be very easy for the user to launch our application-bundle, without installing any programs permanently to his system.

5.2 Architecture

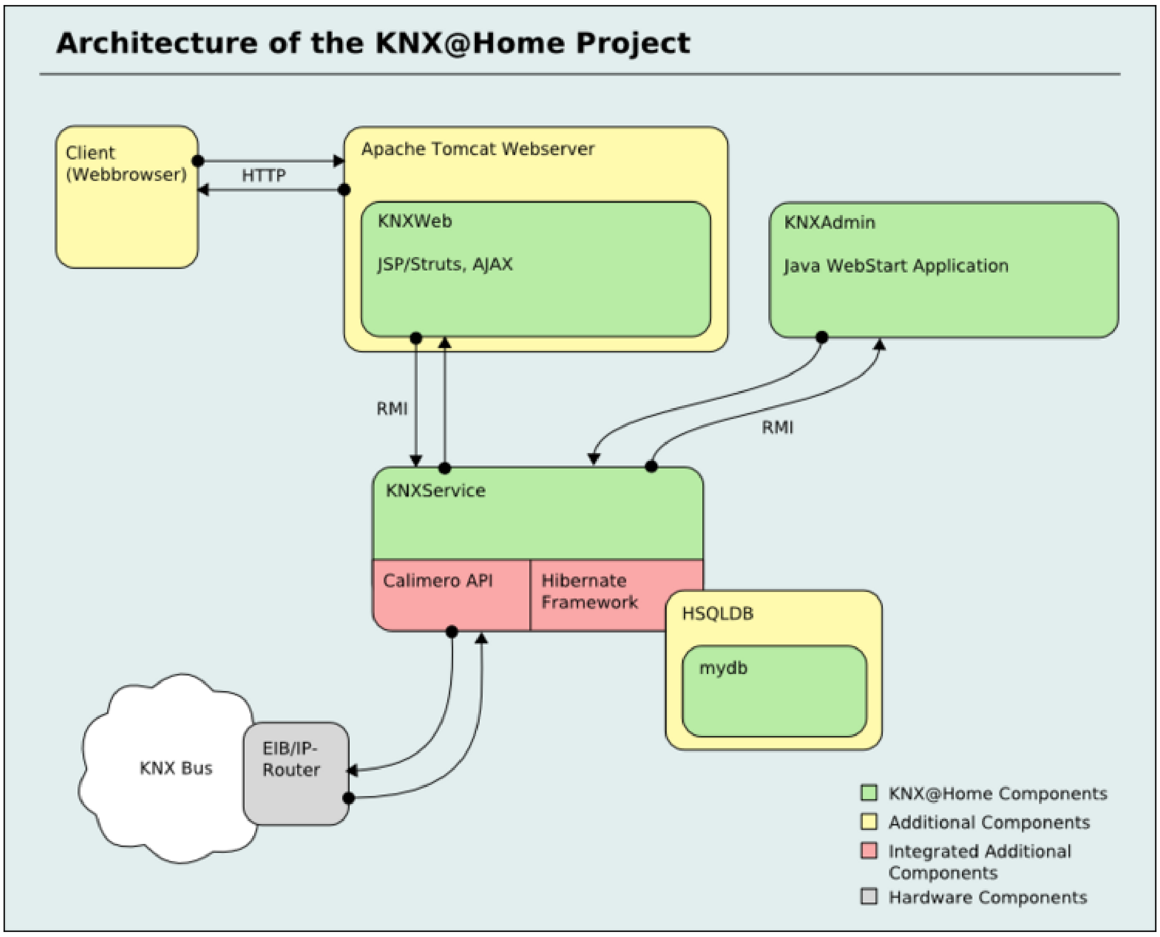


Fig. 5.2: Architecture of the KNX@Home Project

On Figure 8 is shown architecture of KNX@HOME Project. The following technologies are used within KNX@Home:

- Calimero API
- Prototype JS Framework
- Struts Framework
- Hibernate Framework

CHAPTER 6. JOONE

Joone (http://www.joone.org/) is a Java framework to build and run AI applications based on neural networks. Joone applications can be built on a local machine, be trained on a distributed environment and run on whatever device. Joone consists of a modular architecture based on linkable components that can be extended to build new learning algorithms and neural networks architectures. All the components have some basic specific features, like persistence, multi threading, serialization and parametrization. These features guarantee scalability, reliability and extensibility, all mandatory features to reach the final goal to represent the future standard on the AI world. Joone applications are built out of components. Components are pluggable, reusable, and persistent code modules. Components are written by developers. AI experts and designers can build applications by gluing together components with graphical editors, and controlling the logic with scripts. Around the components will be based all the modules and applications written with Joone.

6.1 Basic Concepts

Each neural network (NN) is composed of a number of components (layers) connected together by connections (synapses). Depending on how these components are connected, several neural network architectures can be created (feed forward NN, recurrent NN, etc).

This section deals with feed forward neural networks (FFNN) for simplicity's sake, but it is possible to build whatever neural network architecture is required with Joone. A FFNN is composed of a number of consecutive layers, each one connected to the next by a synapse. In a FFNN recurrent connections from a layer to a previous one are not permitted. Consider the following Figure 9. This is a sample FFNN with two layers fully connected with synapses. Each layer is composed of a certain number of neurons, each of which have the same characteristics (transfer function, learning rate, etc). A neural net built with Joone can be composed of whatever number of layers belonging to different typologies (linear, sigmoid, etc.). Each layer processes its input signal by applying a transfer function and sending the resulting pattern to the synapses that connect it to the next layer. So a neural network can process an input pattern, transferring it from its input layer to the output layer. This is the basic concept upon which the entire engine is based.

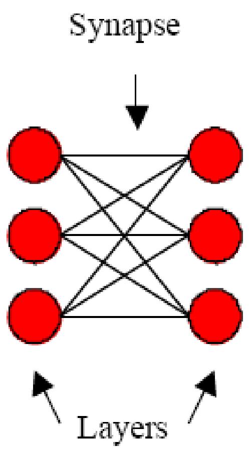


Fig. 6.1: Sample FFNN

6.2 Transport Mechanism

In order to ensure that it is possible to build whatever neural network architecture is required with Joone, a method to transfer the patterns through the net is required without the need of a central point of control. To accomplish this goal, each layer of Joone is implemented as a Runnable object, so each layer runs independently from the other layers (getting the input pattern, applying the transfer function to it and putting the resulting pattern on the output synapses so that the next layers can receive it, processing it and so on) as depicted by the following basic illustration at figure 10:

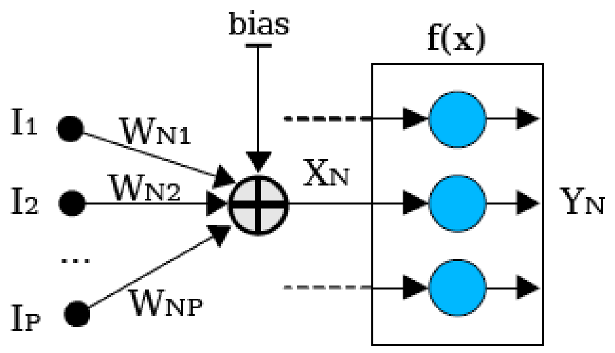


Fig. 6.2: Bias calculation

Where for each neuron N:
XN – The weighted net input of each neuron = (I1 * WN1) + … + (IP * WNP) + bias
YN – The output value of each neuron = f(XN)
f(X) – The transfer function (depending on the kind of layer used)

This transport mechanism is also used to bring the error from the output layers to the input layers during the training phases, allowing the weights and biases to be changed according to the chosen learning algorithm (for example the backprop algorithm). In other words, the Layer object alternately 'pumps' the input signal from the input synapses to the output synapses, and the error pattern from the output synapses to the input synapses. To accomplish this, each layer has two opposing transport mechanisms, one from the input to the output to transfer the input pattern during the recall phase, and another from the output to the input to transfer the learning error during the training phase, as depicted in the following figure 11:

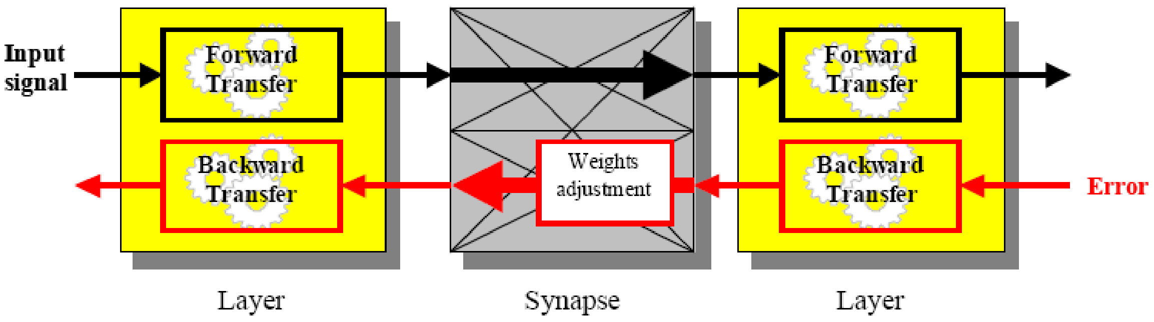


Fig. 6.3: Error learning process

Each Joone component (both layers and synapses) has its own pre-built mechanisms to adjust the weights and biases according to the chosen learning algorithm. Complex neural network architectures can be easily built, either linear or recursive, because there is no necessity for a global controller of the net. Imagine each layer acts as a pump that 'pushes' the signal (the pattern) from its input to its output, where one or more synapses connect it to the next layers, regardless of the number, the sequence or the nature of the layers connected. This is the main characteristic of Joone, guaranteed by the fact that each layer runs on its own thread, representing the unique active element of a neural network based on the Joone's core engine. Look at the following figure 12 (the arrows represent the synapses):

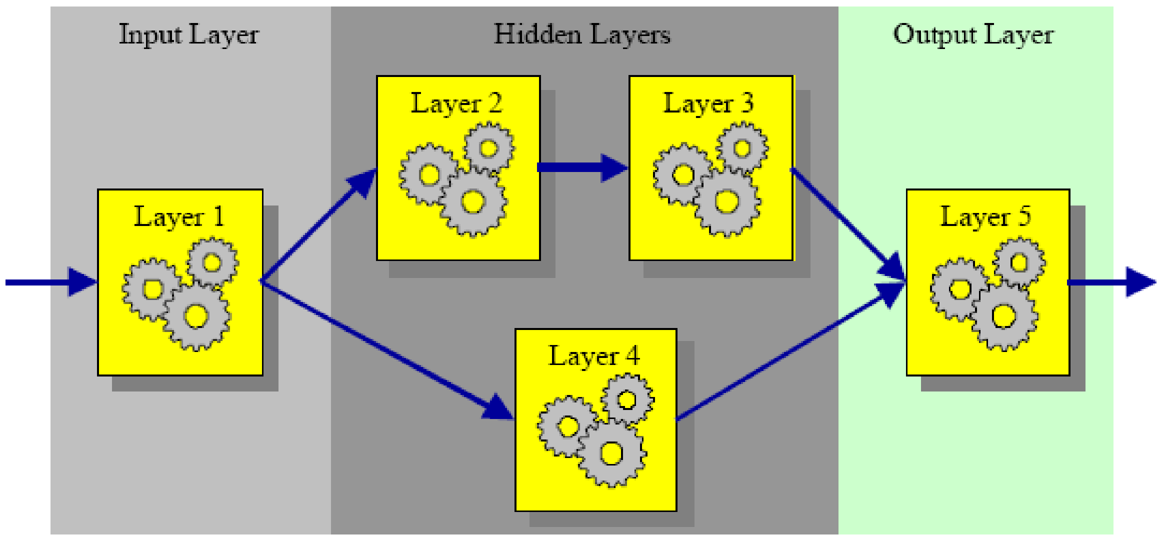


Fig. 6.4: Layer joining

In this manner any kind of neural networks architecture can be built. To build a neural network, simply connect each layer to another as required using a synapse, and the net will run without problems. Each layer (running in its own thread) will read its input, apply the transfer function, and write the result to its output synapses, to which there are other layers connected and running on separate threads, and so on.

Joone allows any kind of neural network to be built thanks to its modular architecture, like a LEGO® bricks system!

By this means:

- The engine is flexible: you can build any architecture you want simply by connecting each layer to another with a synapse, without being concerned about the architecture. Each layer will run independently, processing the signal on its input and writing the results to its output, where the connected synapses will transfer the signal to the next layers, and so on.
- The engine is scalable: if you need more computation power, simply add more CPU to the system. Each layer, running on a separated thread, will be processed by a different CPU, enhancing the speed of the computation.
- The engine closely mirrors reality: conceptually, the net is not far from a real system (the brain), where each neuron works independently from each other without a global control system.

More characteristics of system, layers and synapses can be found in documentation [36]

CHAPTER 7. SELECTED ARCHITECTURE

From available architectures I have chosen KNX bus in combination with LINUX, as free operating system available for all platforms, and JAVA, portable system, possible to port on all platforms. Later I have decided to use already implemented solution KNX@HOME (discussed in chapter 5). This is basic layer responsible for hardware management and communication dispatching. All data will be held in SQL database, from where could be fast used for learning process as well as for evaluations. For neural network development I used framework Joone for its scalability and extensibility. Over Joone I have developed my own Neural Network which is able to solve problem of following scenario.

7.1 Scenario

I have tried to select some simple scenario which would be able to demonstrate computing power of neural network in cooperation with fuzzy logic theory and genetic evolution. Scenario simulates normal behavior of family living in any home. I have identify everyday routine as following. Whenever you enter the bathroom you turn on lights, in case that bathroom doesn't have window. In Figure 13 you can see Decision Flowchart.

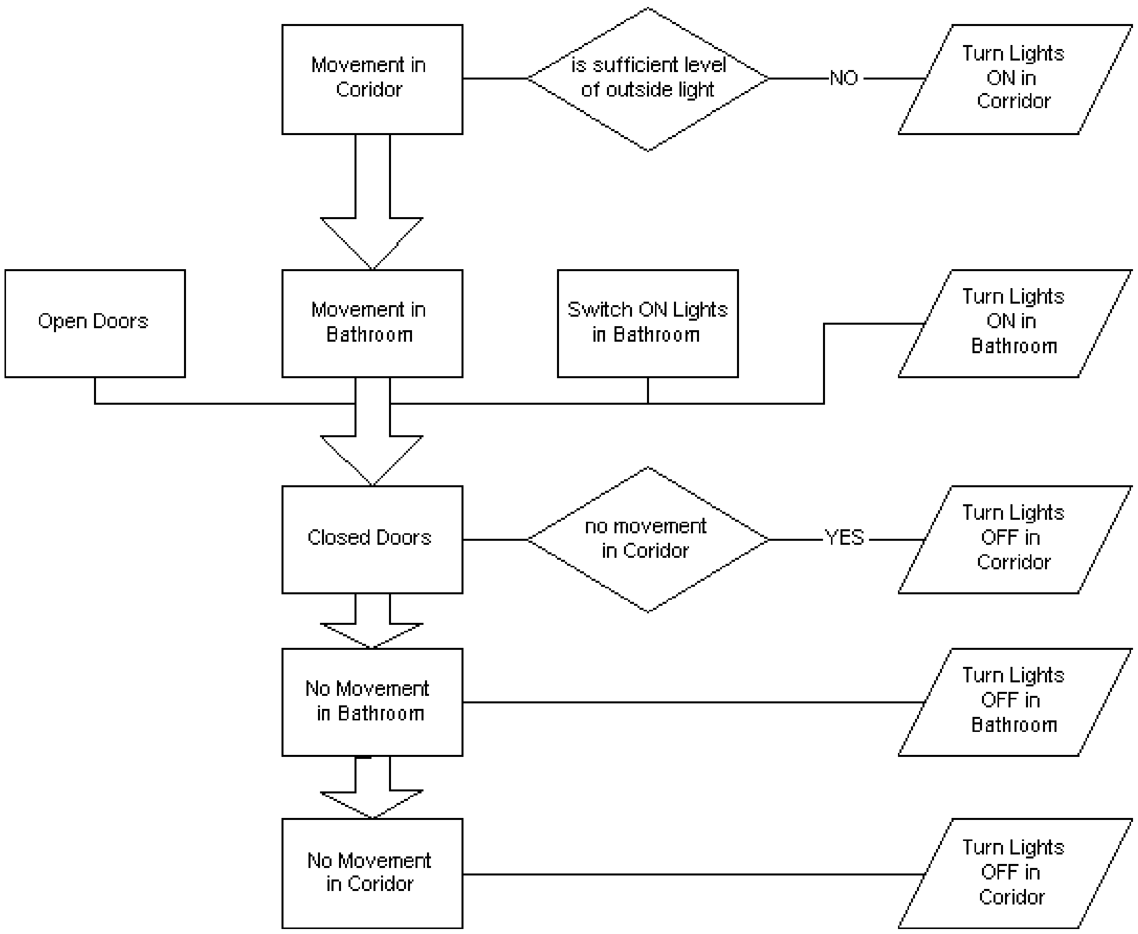

Movement in Coridor
is sufficient level of outside light
NO
Turn Lights ON in Corridor
Open Doors
Movement in Bathroom
Switch ON Lights in Bathroom
Turn Lights ON in Bathroom
Closed Doors
no movement in Coridor
YES
Turn Lights OFF in Corridor
No Movement in Bathroom
Turn Lights OFF in Bathroom
No Movement in Coridor
Turn Lights OFF in Coridor

Fig. 7.1: Decision Flowchart

7.2 Proof of Concept

I want to prove that neural network is able to learn behavioral patterns alone and turns light on without necessary . I have decide to use motion detectors in bathroom and in corridor in front of bathroom. Then I add one sensor which controls doors. Another controllers take care of lightning switch in bathroom and corridor (they can turn on or off lights and return status of switch) and last is sensor which measure light intensity outside. I will observe behavior of system to research whether system is able to learn simple pattern to turn on light after somebody enter bathroom or corridor. And don't turn lights on in corridor if outside light intensity is sufficient.

For this reason I will propose fuzzed variables of outside light intensity. This will contain 3 membership functions Dark, Normal and Bright according to Chapter 3.

7.3 Architecture

Architecture according to Figure 14 will consist of real hardware controller module Calimero, database module, neural network computing module, GUI KNX@HOME Web front end and Simulator to evaluate test without real hardware devices.

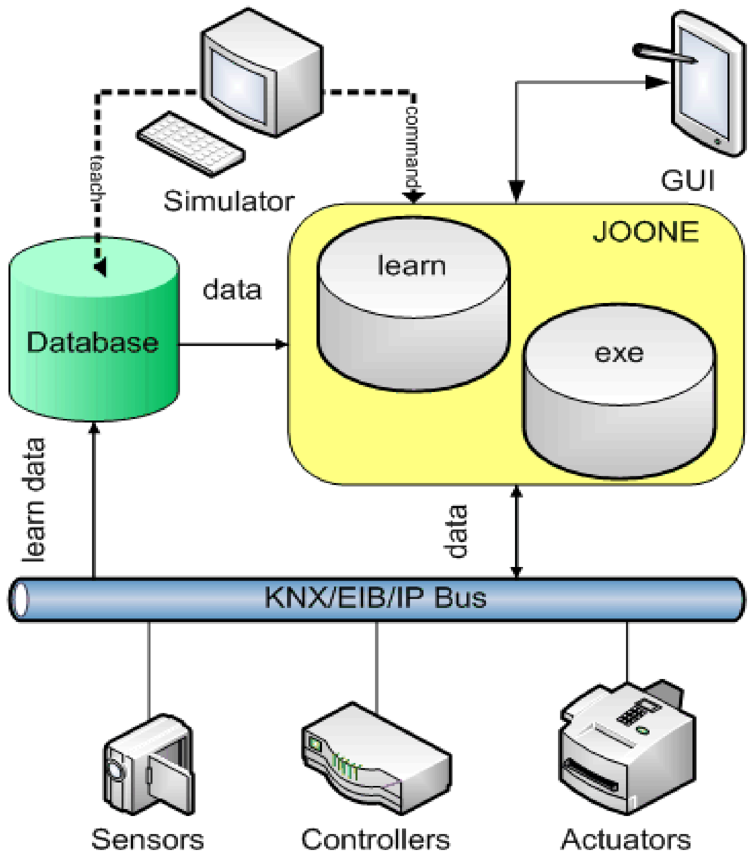


Fig. 7.2: Architecture

CHAPTER 8. IMPLEMENTATION

I have started development with Joone.

8.1 Joone

Trying to find best layers configuration has brought me through plenty of layers and synapses configuration towards one which you can see on Figure 8.1. This could be done inside of Joone IDE without necessary development which save time and let me more space for testing more configurations.

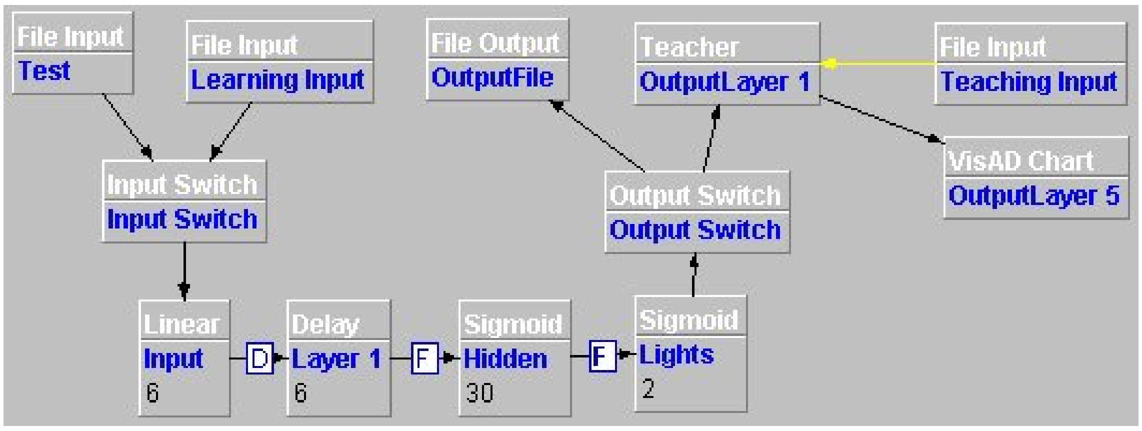


Fig. 8.1: Implementation of Neural Network

As you can see I have chosen to use Delay Layer to be able to recognize some time related patterns inside my smart home neural network system. Input linear layer has 6 neurons and serve only to distribute input values to rest of network.

Hidden layer was chosen to be Sigmoid function Layer with 30 neurons. This number must be greater then Delay layer size multiply by delay window size plus one. I have chosen delay window to be 2. This means network will be able to recognize relations between patterns in time 3 events in row.

Last layer is Output Sigmoid Layer with 2 neurons which values represents lights states according to mentioned scenario from Chapter 7.1. You can find following project saved as “smart.ser” in files folder which is possible to open with Joone IDE and evaluate testing.

Configuration is base on premise that neural network with 3 layers, one hidden, one input and one output, can be trained for almost every problem. I have just add one more layer for delays. Then I was just looking for best configuration of layers, their dimensions and synopses between them. In this search I have used algorithm based on genetic selection.

After I found desirable configuration I have started to implement selected neural network into JAVA with help of Joone API.

8.2 SQL

After deciding suitable neural network configuration was necessary to develop data store for application. I decided to use MySQL, because I have already installed server in my PC and this database is light weighted meanwhile powerfull. I have projected following architecture as shown in Figure 8.2.

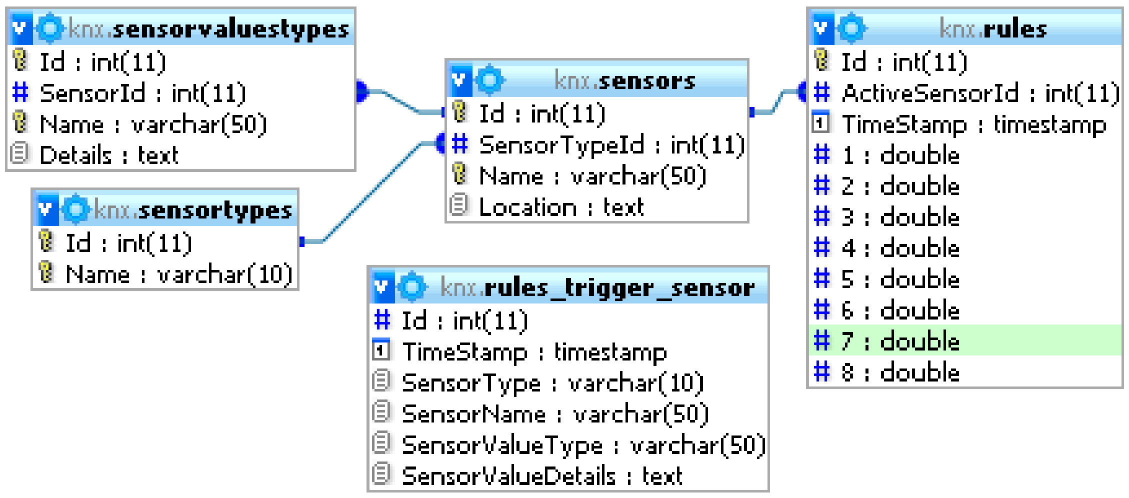


Fig. 8.2: SQL database structure

Basic element is table “rules” which concentrate all sensors from data for each rule. It is linked on sensor throw ActiveSensorId so it is possible to log which sensor change fired logging action.

Sensors have few (3) sensor types, input, output, and input/output. Values sensor gives have types as well. This allows fuzzification of sensor data.

In 'rules' table columns indexed 1-8 represent sensors values from 'sensorvaluestypes' table with same Id respectively.

Rules trigger sensor is function which shows data of sensor which has fired event logging.

In case of adding new sensor or any change is necessary to change data in tables 'sensors', 'sensortypesvalues' and add or change column in table 'rules'.

8.3 Application

Application consist of 3 projects written in netbeans with suport of SVN version control.

8.3.1 DataStore

DataStore is responsible for communication with Database. It uses JDBC driver to connect to MySQL and release insert and select queries.

8.3.2 Predictor

Predictor supposed to be brain of all system. It lies on the top of Joone framework. Use Calimero to communicate with hardware and DataStore to store data. Meanwhile problems with hardware communication it was only tested in cooperation with Simulator.

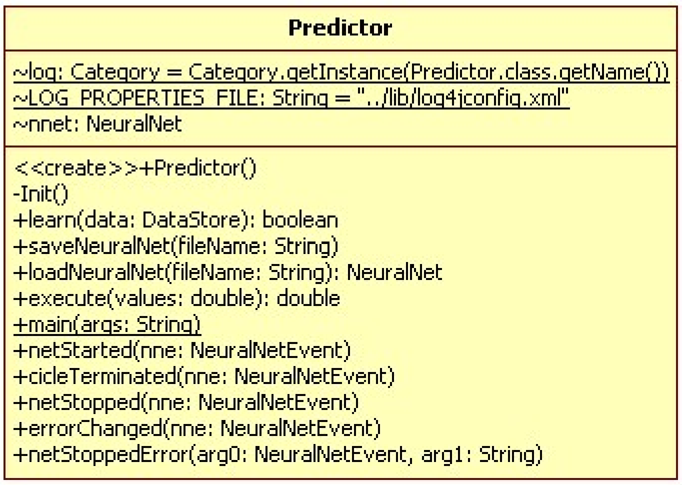

Predictor
~log: Category = Category.getInstance(Predictor.class.getName())
~LOG_PROPERTIES_FILE: String = "../lib/log4jconfig.xml"
~nnet: NeuralNet
<<create>>+Predictor()
-Init()
+learn(data: DataStore): boolean
+saveNeuralNet(fileName: String)
+loadNeuralNet(fileName: String): NeuralNet
+execute(values: double): double
+main(args: String)
+netStarted(nne: NeuralNetEvent)
+cicleTerminated(nne: NeuralNetEvent)
+netStopped(nne: NeuralNetEvent)
+errorChanged(nne: NeuralNetEvent)
+netStoppedError(arg0: NeuralNetEvent, arg1: String)

Fig. 8.3: Predictor structure

8.3.3 Simulator

Simulator is classic JAVA window form application. It is based on SWING and AWT components. Event driven is able to send data to DataStore as well as receiving and visualized decision from Predictor.

More information you can find in following Chapter 9.2.

CHAPTER 9. EVALUATION

Evaluation has started with proof of concept.

9.1 Proof of Concept

I have let chosen neural network to teach itself selected patterns according to mentioned scenario from Chapter 7.1. This teaching data set you can find in files folder as “learning.in”. After 5000 epochs the RMSE value was smaller then 0.1% which means neural network was ready for test. I have prepare test evaluation input file “test.in” which you can find as well in same folder files. Result of test is in followed Figure 9.1.

	Outside light			Movement		Doors	Lights	
Id	**dark**	**normal**	**bright**	**bathroom**	**corridor**	**opened**	**bathroom**	**corridor**
1	1	0	0	0	0	0	0.012330192	1.77E-06
2	1	0	0	0	0	0	0.011618775	0.003081578
3	0	1	0	0	0	0	0.001111757	0.001839245
4	0	1	0	0	0	0	1.16E-04	5.42E-04
5	0	1	0	0	0	0	2.79E-04	0.001487011
6	0	1	0	0	0.8	0	5.76E-04	0.976076142
7	0	1	0	0	0.9	1	0.993362794	0.999983419
8	0	1	0	0.8	0.9	1	0.996040459	0.998979509
9	0	1	0	1	0	0	0.99997847	1.72E-04
10	0	1	0	1	0	0	0.999994219	1.43E-04
11	0	1	0	1	0.2	1	0.998900632	0.528744726
12	0	1	0	0	0.8	1	2.02E-04	0.99983363
13	0	1	0	0	0.9	0	1.06E-05	0.995422608
14	0	0	1	0	0.9	0	0.270383426	0.434955202
15	0	0	1	0	0	0	0.761534343	3.78E-07
16	0	0	1	0	0	0	0.138293602	4.73E-10
17	0	0	1	0	0	0	9.31E-04	1.70E-09
18	0	0	1	0	0	0	9.31E-04	1.70E-09
19	0	0	1	0	0	0	9.31E-04	1.70E-09

Fig. 9.1: Result of test, red values are strange values

As you can see except of 2 cases network was able to predict lights condition quite well. Mentioned strange values (highlighted red in figure) are result of combination of two factors: small learning set of input patterns and delay layer, which needs time to inhibit excited neurons. That's why I decided to write new project of Simulator to be able to test more this system.

9.2 Simulation

Simulator is prepared to test more complex set of problems, thats why it has whole flat floor plan equipped with lights, movement sensors in every room, door sensors and outside light intensity meter. You can set this values according to your wish, or let watch neural network decisions or real life actions from KNX BUS.

Because of problems with communication interface which shows up while testing on real hardware equipment in laboratory we decide to change status of simulator from middleware between KNX BUS and Calimero engine into front end lightweight listener. This means it doesn't interact directly with hardware. Instead of that it listen to the states of neural network in database and send commands directly to neural network.

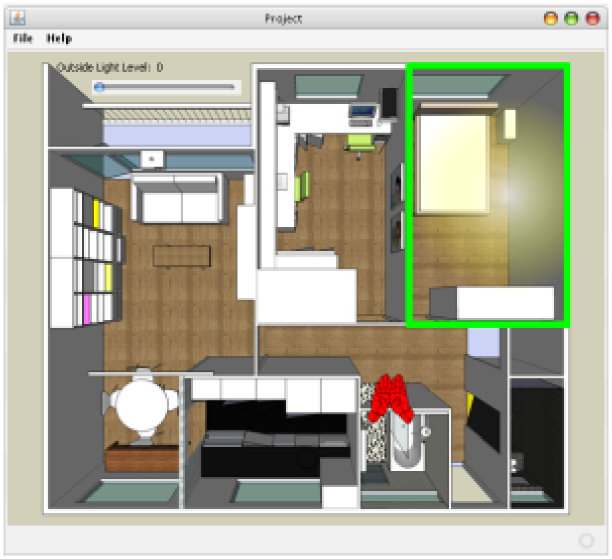


Fig. 9.2: Simulator with selected Bedroom

In Figure 9.2 you can see green rectangle which specify actually selected room. Inside green rectangle, which shows selected bedroom you can see status of bedroom light is turn to ON. You can change this with left click on room while selected (green). Selection is done by moving mouse over object. You can see status of bathroom doors to be closed (red hands sign).

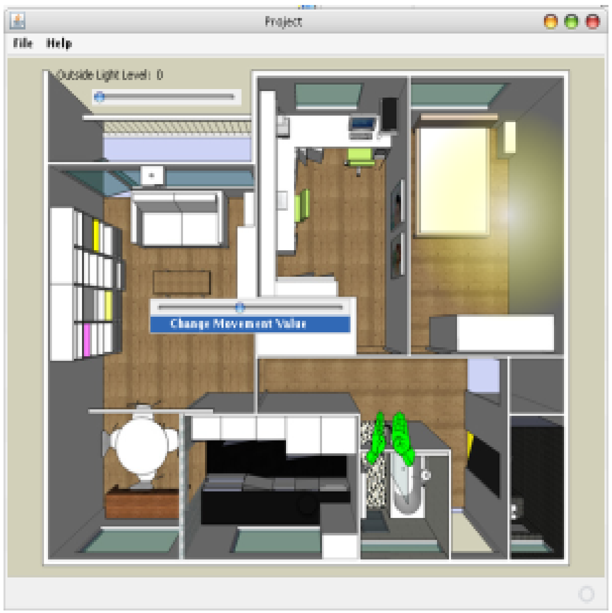


Fig. 9.3: Simulator with movement in Living room

In Figure 9.3 you can notice change of icon signalizing open doors into bathroom. In upper left corner is slide bar which helps you control outside light intensity value. In the middle you can see pop-up menu with slide bar for movement control in particular rooms.

CHAPTER 10. CONCLUSIONS

According to difficulties which occurs while testing connection of computer to KNX bus it was impossible to test project on real hardware.

KNX@HOME uses JAVA virtual machine and framework Calimero developed on this platform to communicate with KNX BUS. Meanwhile Sun Microsystems, developer of JAVA has suspended support of serial ports in JAVA virtual machine. There are few works around to make it work, however they work only with serial port, not USB. That's why KNX@HOME doesn't support USB as connection port to the KNX BUS. Problem arise when we realized that available equipment in laboratory has only USB connectivity available. Which means we have to find some work around to redirect Serial port to USB with reasonable addressing and protocol control.

Test of software engine in simulator has proven ability of Neural Network trained with patterns input from real life situations to predict future stages.

Problem was in amount of patterns. Too much means long time necessary for training process. Too little has conclude into weak ability of prediction. It is necessary to find some reasonable compromise between strength of prediction ability and computing time. But this problem can be solved with Genetic Algorithm employed to search for better weights of neurons and synapses.

CHAPTER 11. FUTURE WORKS

First I would like to point on wide spectrum of possible application of neural networks, which is related to possible future works. First of all it is necessary to solve connection difficulties to test software on real hardware. For this reason is necessary to examine usb-to-serial drivers, emulators and virtual ports. Then rewrite simulator to be able to simulate serial port stub.

It is necessary to customize KNX@HOME linux distribution and develop live CD which will be able to run independently on any PC with installed Joone and Neural Network.

Engine alone, can be improved with Joone DTE (distributed training environment) technology, which is kind of farm of parallel computers independently teaching neural networks in different configurations. After training process is finished they compare results of RMSE and one with smallest value is winner. This configuration is later used for evaluation of neural network. This technology can be really interesting according to increasing amount of chips in every possible electrical devices in smart home environment. These devices would be able to serve as parallel farm for DTE.

There is possibility to implement genetic algorithm to solve problem of neural network learning. This can be done via new module into Joone, Genetic Layer or Synapse. This can help to find better solution in shorter time.

List of Figures

List of Shortcuts

ACHE – Adaptive Control of Home Environments
AI – Artificial Intelligence
AJAX – Asynchronous JavaScript and XML
ALZ – Active LeZi
ANN – Artificial Neural Network
API – Application Interface
ART – Adaptive Resonance Theory
ARTMAP – Predictive ART
AWT - Abstract Window Toolkit, part of the Java programming language
CBR – Case Based Reasoning
CPU – Central Processing Unit
DNA – Deoxyribonucleic Acid
ECG – Electro Cardiogram
EcoS – Evolving Connectionist System
EEG – Electro Encephalogram
EIB – European Installation Bus
FFNN – Feed Forward Neural Network
GA – Genetic Algorithms
GP – Genetic Programming
GUI – Graphic User Interface
HSQLDB – Hyperthreaded Structured Query Language Database
HTTP – Hypertext Transfer Protocol
JAVA – programming language originally developed by Sun Microsystems
JDBC – Java DataBase Controller
Joone – Java based Neural Network Development Environment
Joone DTE – Joone Distributed Training Environment
JSP – Java Server Pages
KDE – Linux K Desktop Environment
KNX – standardized (EN 50090,ISO/IEC 14543), OSI-based network communications protocol for intelligent buildings administered by the Konnex.
LINUX – Unix-like computer operating system family
LISP – high-level programming language with fully parenthesized syntax
MAS – Multi Agent Systems
OS – Operating System
RAM – Random Access Memory
RFID – Radio Frequency Identification
RMI – Remote Method Invocation
RMSE – Root Mean Square Error
RNN – Recurrent Neural Network
SHIP – Prediction Algorithm
SD – Secure Digital
SQL – Structured Query Language
SVN – Subversion
SWING - Sun's lightweight GUI library for the Java language
UNIX – operating system originally developed by AT&T employees at Bell Labs
TDL – Tapped Delay Line
TDNN – Time Delay Neural Network
USB – Universal Serial Bus

Bibliography

1: Juan C. Augusto and Chris D. Nugent, Smart Homes Can Be Smarter, 2006
2: Rezaul Begg and Rafiul Hassan, Artificial Neural Networks in Smart Homes, 2006
3: Haykin, S., Neural Networks: A Comprehensive Foundation, 1994
4: Hassoun, M. H., Fundamentals of Artificial Neural Networks, 1995
5: Hecht-Nielson, R., Neuro Computing
6: Eliasmith, C. and Charles H. Anderson, Neural Engineering Computation And Dynamics in Neurological Systems, 2003
7: Hebb, D. O., The Organisation of Behavior: A Neurophysiological Theory, 1949
8: Takagi, H., Introduction to Fuzzy Systems, Neural Networks, and Genetic Algorithms, 1997
9: Rumelhart, D. E., J.L. McClelland, and the PDP Research Group, Parallel distributed processing: explorations in the microstructure of cognition, 1986
10: Chan, M., C. Hariton, P. Ringeard, E. Campo, Smart House Automation System for the Elderly and Disabled
11: Mozer, M. C., The Neural Network House: An Environment that adapts to its Inhabitants, 1998
12: Jorge, D. and Goncalves, V., Ubiquitous Computing and AI Towards an Inclusive Society, 2001
13: Pigot, H., B. Lefebvre, J. Meunier, B. Kerherve, A. Mayers, S. Giroux, The role of intelligent habitats in upholding elders in residence, 2003
14: Rialle. V, N. Noury, T. Herve, An experimental Health Smart Home and its distributed Internet based Information and Communication System: first steps of a research project, 2001
15: Cook, D. J., M. Youngblood, E. O. Heierman, III , K. Gopalratnam, S., Rao, A. Litvin, and F. Khawaja, MavHome: An Agent-Based Smart Home, 2003
16: Illingworth, F.R., V. Callaghan, and Hagras H., A Neural Network Agent Based Approach to Activity Detection in AmI Environments, 2005
17: N. Kasabov, Evolving connectionist systems: Methods and applications inbioinformatics,brain study and intelligent machines, 2002
18: Elman, J. L., Finding structure in time, 1990
19: Prasanna, C. S. S., N. Sudha, Kamakoti, V., A Principal Component Neural Network based Face Recognition System and Its ASIC Implementation, 2005
20: Zhanqing, L., A. Khannanian, R. H. Fraser, and J. Cihlar, Automatic Detection of Fire Smoke using Artificial Neural Networks and Threshold Approaches Applied to AVHRR Imagery, 2001
21: Hanebeck, U.D., Fischer, C. Schmidt G., ROMAN: a mobile robotic assistant for indoor service applications, 1997
22: Khalid, M. and S. Omatu, A Neural Network Controller for a Temperature Control System, 1992
23: Lin, C. T., C-F Juang, and C-P Li, Temperature Control with a Neural Fuzzy Inference Network, 1999
24: Nazeran, H. and Behbehani, K, Neural Networks in Processing and Analysis of Biomedical Signals, 2001
25: Silva, R. and Silva A.C.R., Medical Diagnosis as a Neural Networks Pattern Classification Problem, 1998

26: Vainio A.-M., Valtonen M., Vanhala J., Learning and adaptive fuzzy control system for smart home, 2004
27: Wang, L. X., A course in fuzzy systems and control, 1997
28: Carpenter, G. A., Grossberg, S., Rosen, D. B., Fuzzy ART: Fast Stable Learning and Categoization of Analog Patterns by an Adaptive Resonance System, 1991
29: Williamson, J.R., Gaussian ARTMAP: A Neural Network for Fast Incremental Learning of Noisy Multidimensional Maps, 1996
30: Zhang, L., Wang, G., Wang, W., A New Fuzzy ART Neural Network Based on Dual Competition and Resonance Technique, 2006
31: Obtiko M., Introduction to Genetic Algorithms, 2008, http://www.obitko.com/tutorials/genetic-algorithms/
32: Rechenberg I., Evolutionsstrategie, 1960
33: Holland J., Adaption in Natural and Artificial Systems, 1975
34: Wurm, R., Reichert, A., Bayer, J., KNX@HOME, 2006
35: KNX@HOME, http://knxathome.fh-deggendorf.de/knxathome/wiki/
36: Marrone P., Java Object Oriented Neural Engine, 2007