Hierarchical Self-organizing Learning Systems for Embodied Intelligence

A dissertation presented to

the faculty of

the Russ College of Engineering and Technology of Ohio University

In partial fulfillment

of the requirements for the degree

Doctor of Philosophy

Yinyin Liu

March 2009

This dissertation titled

Hierarchical Self-organizing Learning Systems for Embodied Intelligence

by

YINYIN LIU

has been approved for

the School of Electrical Engineering Computer Science

and the Russ College of Engineering and Technology by

Janusz A. Starzyk

Professor of Electrical Engineering and Computer Science

Dennis Irwin

Dean, Russ College of Engineering and Technology

ABSTRACT

LIU, YINYIN, Ph.D., March 2009, Electrical Engineering

Hierarchical Self-organizing Learning Systems for Embodied Intelligence (271 pp.)

Director of Dissertation: Janusz A. Starzyk

In this work, a framework of designing embodied intelligence (EI), along with the essential elements and their design principles, is proposed. This work intends to deploy the following design principles.

Firstly, hierarchical self-organizing learning systems in the form of network made of neurons are the essential elements for building machine intelligence. The supervised, unsupervised and reinforcement learning are all necessary aspects of learning and are studied for machine intelligence building. In supervised learning, an efficient learning method for hierarchical multi-layered network structure is proposed and studied. In addition, a quantitative measure is proposed to quantify overfitting of a network in a given learning problem to determine proper network structure or proper learning period. In unsupervised learning, a sparsely-connected hierarchical network is developed to build the neural representations effectively and efficiently for densely-coded sensory inputs, and to enable the memory with large memory capacity and great fault tolerance.

Secondly, the memory-based intelligence is not only for passive information processing and pattern storage. One of the critical capabilities of intelligence is continuous and intentional learning. Therefore, a goal creation system (GCS), also as a type of hierarchical self-organizing learning system based on simple and uniform structure, is presented that acts as the trigger for the agent's goal creation, memory management, active interaction and goal-oriented learning. As a self-organizing structure, it is responsible for evaluating actions according to goals, stimulating the learning of useful associations and representations for sensory inputs and motor outputs. It enables the more powerful hierarchical reinforcement learning, finds the ontology among sensory objects, creates the needs, and affects the agent's attention and perception.

Biologically inspired structural design concept and the framework of EI proposed in this dissertation create a promising direction in the field of EI. It enables the desired capabilities for an intelligent machine to have, including the efficient, continuous and intentional learning, large representative memory capacity, and goal-oriented perception and action. The hierarchical self-organizing learning systems include all ingredients necessary to develop intelligence, and to motivate a machine to act on its own in its environment. Having the framework defined and design principles prepared, the future work will be done more consistently.

Approved: ____

Janusz A. Starzyk

Professor of Electrical Engineering and Computer Science

ACKNOWLEDGMENTS

One of the pleasures of doing doctoral research and writing a dissertation is to find the topic you work on more and more fascinating. Several years ago, I was someone who found machine intelligence vaguely interesting, while now I truly believe that it will shape our future technology in an amazing way that many people wouldn't imagine. I wouldn't have had this exciting and rewarding experience without the help of my advisor, Dr. Janusz A. Starzyk. I would like to gratefully and sincerely thank him for his guidance, understanding, patience, and most importantly, his care and friendship during my graduate studies at Ohio University. He led me into this wonderful field of machine intelligence, taught me how to conduct research and helped me to be creative, thoughtful, committed. His mentorship provided me a well rounded experience consistent with my long-term career goals. His constant inspiration and encouragement made me believe in myself and made this difficult and stressful doctoral program a lot easier.

I would also like to thank the faculty members in Ohio University and my dissertation committee members, especially Dr. Jeffrey Dill, Dr. Robert Curtis, Dr. Savas Kaya, Dr. Jundong Liu, Dr. Jeffrey Vancouver, Dr. Sergiu Aizicovici for their inputs, valuable discussions and accessibility. I would like to thank Dr. Constantinos Vassiliadis and Mrs. Zofia Starzyk for their precious intellectual and emotional help.

Additionally, I am very grateful for the friendship from my friends in Ohio University, especially Zhen Zhu, Ning Zhou, Xinming Yu, Nathan Frey, James Graham, Yiming Huang, Haibo He, Mingwei Ding, Yue Li, Beibei Wang, Luyi Chen, Ting Zhou for the much needed humor and entertainment over the years.

My greatest gratitude goes to my parents, Qingsu Liu and Lijia Men for being the most wonderful and fun parents, for their faith in me and unending love and support. It was under their watchful eye that I gained so much drive and strength to tackle the challenges head on.

Finally, and most importantly, I thank my husband, Yu Hou for his support, encouragement, tremendous patience and sweetest love that were undeniably the rock upon which the past nine years of my life have been built. Among all the wonderful things that happen in my life, having him with me is and will always be the best one.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AC:	Actor Critic
ACT-R:	Adaptive Control of Thought – Rational
AIC:	Akaike’s information criterion
AN:	Action Network
BDI:	Beliefs, Desires and Intentions
BP:	Back-propagation
CI:	Clustering Index
CN:	Critic Network
CPD:	Conditional probability distribution
CPU:	Central Processing Unit
EI:	Embodied Intelligence
EUA:	End of Unsuccessful Action
fMRI:	Functional Magnetic Resonance Imaging
FPGA:	Field Programmable Gate Array
FSM:	Finite State Machine
GCS:	Goal Creation System
GOMS:	Goals, Operators, Methods and Selection Rules
HRL:	Hierarchical Reinforcement Learning
HSOM:	Hierarchical Self-organizing Memory
HTM:	Hierarchical Temporal Memory
kWTA:	k-Winner-Take-ALL

LSF:	Least-squares fitting
LSM:	Least-squared Learning Method
LSMLP	LS-based MLP training algorithm
LSMLP-ITF	LS-based MLP training with iterating fitting
LTM:	Long Term Memory
MDL:	Minimum Description Length
MLP:	Multi-layer Perceptron
MPS:	Motive Processing System
MSE:	Mean Squared Error
NN:	Neural Networks
OAA:	Optimized Approximation Algorithm
OTA:	Oligarchy-take-all
PET:	Positron Emission Tomography
PRS:	Procedural Reasoning System
RL:	Reinforcement Learning
SNRF:	Signal-to-noise Ratio Figure
SOAR:	State, Operator, Application, Result
SOM:	Self-organizing Map
SVD:	Singular Value Decomposition
SWTA:	Sparse Winner-Take-All
UI:	Unexpected Input
VLSI:	Very Large Scale Integrated Circuit

WGN:	White Gaussian Noise
WLSF:	Weighted Least-Squares Fitting
WTA:	Winner-Take-All

CHAPTER 1: INTRODUCTION

Even though artificial intelligent machines have been portrayed in fiction and motion pictures for so many years and they have been a research topic for decades, they are still one of the great science problems not fully tackled. Machine intelligence has become a scientific discipline, focused on providing solutions to real life problems. Examples include natural language processing, search engines, planning, prediction, decision making, and handwriting, speech, and facial recognition etc. Although the existing technologies enable the machines with great computational power, many of the problems, which are easily dealt with by human intelligence, are still difficult, expensive or even impossible to handle by existing machines. Machines are still far from reaching the level of human intelligence. Therefore, it would be promising to build the machines using approaches insipired by brain intelligence.

Without complete understanding of the mechanism and architecture of brain intelligence, it is hard or impossible to build the machine wshich can achieve what human can do. Discoveries in the fields of neuroscience, medical scanning and imaging, anatomy, physiology and psychology on the human brain can all provide beneficial information and important insights for building the intelligent machine. For example, using functional Magnetic resonance imaging (fMRI) technique, it is possible to make repeated measures of brain responses in different areas. It can provide information on the temporal sequence of information processing and activation sites in brain areas, which can support or be interpreted by anatomical, neurobiological, or psychological studies. Present and future research in related areas, with interdisciplinary studies can begin to provide a functional model of the human brain and motivate the design of machine intelligence. And hopefully, researchers in engineering and computer science will find very efficient ways to build the intelligent machines using structural self-organization, modern electronics, and nano-technology. The power of machine's intelligence may be extended into new levels with many successful applications in areas such as pattern recognition, language recognition, and intelligent mobile devices, etc. In addition, this new area, where modern technology can be applied, may create a big industry through exponentially growing demand for electronic hardware and may create new job markets for generations to come. We will move from information technology of today to knowledge technology tomorrow, where machines we use and interact with will be intelligent.

1.1. Motivations

Although there is no uniform definition of intelligence, after decades of research on machine intelligence we can list capabilities or the features which the intelligence machines are desired to have. They are desired to have large associative and representative memory capacity, be able to conduct efficient, continuous and goal-oriented learning and action, predict future events and make optimal plans, while building a model of the environment in which they are situated and with which they interact.

It has been widely accepted that behavior should not be a major measure of intelligence or the objective for building intelligent machines [Haw 04]. Instead, the significance of being situated and ability to interact with the environment motivated the concept of embodied intelligence (EI) [Bro 91] [Pfe 99]. Embodied intelligence (EI) has developed into a multidisciplinary field, including biology, neuroscience, electrical engineering, robotics, biomechanics, material science, and dynamic systems. It focuses on understanding biological intelligent systems, extracting general principles of intelligent behavior and applying this knowledge to design robots and intelligent devices.

Learning is a critical element of intelligence. Human learning is an active process. Human beings accumulate knowledge through interaction with the environment. Even in supervised learning, we cannot learn without conducting trial-and-errors and likely motivated by certain goal-oriented behavior. The actions we take affect the environment and the responses from the environment help us to build perceptions and learn skills. Unsupervised learning, which builds representations for the input information from the environment, is not an involuntary process and should be only conducted when the perceptual objects are meaningful for human. In addition, reinforcement learning that occurs when responses from the environment are the only directional information available is another major aspect of the human learning. The agent explores different actions and learns which actions are desirable and which are not guided by the reinforcement signal from external environment. In general, the machine interacts with its environment and gradually builds the model of the environment and gains and knowledge and skills through supervised, unsupervised and reinforcement learning.

In this work, a framework of designing embodied intelligence (EI), along with the essential elements of EI and their design principles, is proposed. Hierarchical self-organizing systems in the form of networks made of neurons are proposed as the essential elements for building machine intelligence. Considering all three types of learning involved, the design concepts and principles are proposed, and various issues involved are presented and studied.

The framework of EI proposed in this work enables the desired capabilities for an intelligent machine to have, including efficient and continuous learning, large memory capacity, goal-oriented perception, optimal action selection, and future events prediction.

1.2 Related works and background

During the research on intelligence, scientists normally use various tests and measures to compare the levels of intelligence and differentiate between intelligence of humans and other species. In fact, scores on various tests for (human level) intelligence was used as a substitute for its definition. Complex skills and behaviors were used to define how intelligence manifests itself. This was a result of poor understanding of what is needed to create intelligence. Such an approach was inconsistent, because a machine that was obviously not intelligent could still satisfied certain tests, while failed in others.

R. Brooks [Bro 91] [Bro 02], the father of embodied intelligence, proposed to design an EI system through layers of simple sensory-motor coordinations built on finite state machines (FSM). In his subsumption architecture, higher levels are built upon the lower levels, subsuming the lower levels functionality. In subsumption architecture, each layer consists of asynchronous modules that send messages to each other. Each module is an augmented FSM. Inputs to such modules can be suppressed and outputs inhibited by signals from other modules.

Although subsumption architecture may be an efficient design approach to building robots capable of complex behavior, it cannot lead to intelligence. A designer must be involved in developing each FSM. These FSMs do not know how to modify their own structures to handle new tasks. There is no self-organization and no learning. Moreover, since new tasks may not be compatible with old ones, modification of the machine behavior to incorporate new tasks may become extremely difficult. Very quickly complexity exceeds understanding of the machine's operation by the human designer, who no longer understands how to add a new layer of functionality.

R. Pfeifer [Pfe 99] modified the subsumption architecture approach to include self-organization and the emergence of necessary links between lower-level processes that control sensory-motor coordination. He also added a value principle to his design approach, requiring a mechanism for self-supervised, perpetual learning that employs the principle of self-organization. The value system acts as a teacher telling an agent what actions are good for its objectives. Memory of the recent history is necessary to implement this value system. This memory is accomplished by time-averaging neuron activities. But as Sporns and Edelman pointed out [Spo 93] "the issue of value constraints and their number present one of the greatest future challenges to selectional theories of brain function."

Earlier attempts to design working models of intelligence include, for instance, GOMS [Car 83], SOAR [New 90] [Nas 04], and ACT-R [And 93] software systems. GOMS (Goals, Operators, Methods, and Selection Rules) is a software system for modeling and describing human performance that provides a framework to analyze human computer interactions. Goals, that a user is trying to accomplish, are organized hierarchically. Methods describe sequences of basic operations used to accomplish a goal. Selection rules describe which method should be used in a particular situation. It uses a production-system representation of human procedural knowledge required to accomplish production goals, and it gives good quantitative predictions of performance time and learning.

SOAR (State, Operator, Application, Result) is a cognitive goal-oriented architecture that develops a minimal set of rules to support intelligent behavior in a specified environment. It uses symbolic knowledge and knowledge-based symbolic reasoning to solve problems. It creates subgoals even with incomplete or inconsistent knowledge. SOAR can also generate rules for the implementation of goals using a process called chunking. The SOAR program learns using explanation-based learning, macro-operator learning, strategy acquisition, and learning by instruction.

ACT-R (Adaptive Control of Thought - Rational) is a model of the human cognitive process focusing on learning and problem solving. Cognitive tasks are performed using if-then production rules, with working memory (declarative or procedural). ACT-R uses pattern matching to match conditions for its production rules and conflict resolution to decide which rule applies. Using ACT-R requires developing a domain-specific knowledge model of the cognitive task for a specific application. None of these three systems use self-organization, unsupervised learning, or create a knowledge base for its actions with the environment.

A practical effort to design structural and algorithmic properties of the neocortex was undertaken by J. Hawkins from Numenta Inc. [Haw 04]. Numenta develops software code for Hierarchical Temporal Memory (HTM). HTM uses a hierarchy of spatio-temporal associations and learns complex goal-oriented behaviors. The information, in the form of probability distributions, passes up and down the hierarchy to represent the sensory inputs and to make predictions. It uses a combination of unsupervised and supervised learning to make associations. In the authors' opinion, future HTM may yield machines that exceed human level of performance in cognitive tasks.

Human perception building and cognition are very efficient processes. Typically, it takes less than one second for sensory information to be obtained, perceived, processed and acted upon. Since neurons in the human brain take several milliseconds to fire, a typical cognition task takes less than 100 sequential pattern-processing steps. This recognition performed within such a short period indicates the high efficiency of human perception. Such recognition, along with the corresponding action, is a very fundamental task for a human; nonetheless, it is already quite complicated for existing machines.

Furthermore, the way that the brain stores a pattern in its hierarchical memory of the neocortex is very different from the way a computer does. Neurons in the human brain self-organize to store the patterns which the brain receives. Humans utilize their senses to build a perception of the environment and activate appropriate motor neurons to apply actions. This enables a human to build a model of the world in a fascinating way. They use this model to quickly recognize patterns in order to respond to the external stimuli and interact within the environment. They also use this model to build expectations of future events, accomplish efficient planning, and do logical thinking. Up to date, there are several research findings on building a hierarchical system for intelligent machines [Lee 03] [Geo 05a] [Geo 05b]. They are mainly focused on information processing from sensory input (visual cortex) through hierarchical memory to upper levels of abstract cognition. In this way, an image coming from the visual cortex is stored in the form of its characteristic features instead of full image details.

The network made of processing units, or called neurons, in the form of feed-forward multi-layer perceptrons (MLP) have appeared and have been utilized as a powerful function approximation and adaptive filtering tool as early as 1950s [Ros 58]. A lot of research has been done using it as the adaptive component in a learning system [Mil 90]. Basically, an MLP consists of the input and the output layer and several hidden layers, and the hidden layers consist of a number of hidden neurons. All the layers are connected through trainable, one-directional interconnection weights. The neurons in hidden layers apply a nonlinear transformation to input data. The nonlinear function implemented in the neurons act as basis functions, and the interconnection weights combine the nonlinear basis functions in a complex way. The network weights can be adjusted during supervised learning based on the training data. So, in this sense, the weights contain information stored in a neural network. Even though using only MLP can’t realize any intelligence itself, a network with nonlinear basis functions, like MLP, is a powerful tool in function approximation. Therefore, it will be very useful for building hierarchical memories in supervised learning scenarios.

Reinforcement learning (RL) [Sut 98] enables the agent to behave in a goal-oriented way and to adapt to the changing environment using externally administrated and measurable reward system. Provided with the specific goal, the agent ought to find an effective way to achieve it following the reward/punishment signals received from the environment.

Typically, RL algorithms treat all the experiences equally in the knowledge building. However, during the exploration, the agent experiences a lot of random states and executes random actions which do not affect its goal achievement. In human intelligence, if the states or actions are not helpful for the goal achievement, corresponding representations and skills should not be built. In addition, the goal-relevant states should be more salient to the agent. The perceptions and actions are activated selectively by the brain with attention focused on those observations and actions that are related to human objectives.

RL may require long learning periods and equally long periods of time to adopt to new requirements (new goals), since all the skills that agent developed in learning one task are specialized to this task only. It is obvious that learning complex tasks can be facilitated if simpler tasks are learned first, and the acquired knowledge is reused to advance both understanding and skill levels. However, there is no clear mechanism of reusing prior knowledge to learn new skills. With each new externally formulated and monitored goal, the machine experiences new learning phase that does not guarantee protecting prior knowledge or skills.

In the existing models for designing the intelligent machines, the goal is defined by designers and is given to the learning agent. Setting a goal makes the machine specialize to perform specific types of operation. While this may be a useful limitation from a utilitarian point of view, such a limitation will make the machine less intelligent. In such cases, the agent is not able to create its own goals or find the sub-goals in order to accomplish a complicated task. In fact, we would argue that an agent who only follows externally set goals or subgoals would not be able to develop some higher level cognitive abilities such as intentional learning, deliberate thought, creativity, and consciousness.

One may ask, why is it important that intelligent machine defines its own goals rather than relays only on the goals defined by a designer? If a machine only follows an externally set and monitored goal, it is not equipped to represent complex goals, since proper interpretations of such goals require complex representations of the environment. A machine needs a goal to build useful representations, but it also needs those representations to define goals.

A new learning paradigm is desired in EI in which learning is organized bottom up from the simplest to more complex representations, skills, and internally created goals. In such systems, simple external rewards can only be directly applied to satisfy the most primitive goals. Since humans and animals create their own goals, it is desirable for the intelligent machines to be able to do so as well. The goal creation mechanism may be one of the most important elements of machine intelligence. Thus it will be one of the fundamental requirements for EI.

1.3 Research objectives

Motivated by the above insight, this work intends to deploy the following design principles. Firstly, the memory-based intelligence is assumed to be one promising framework to implement intelligence in human brain so that hierarchical self-organizing memory systems are essential for deploying the machine intelligence. Supervised, unsupervised and reinforcement learning are all essential aspects of learning and should all be considered and utilized in building machine intelligence. During supervised learning when the outputs of the network are given, the network is trained as in model approximation, thus the efficiency and optimality of the solutions are the major concerns. However, determination of the learning accuracy and the proper parameters of the learning network, including its structure or learning period, are the major challenges, which make the design and use of the network more of an art than a science [Xia 05]. Thus, they should be justified quantitatively to avoid overfitting defined as building too complex models to approximate the input data. In the unsupervised learning, where the neuronal representations are not defined, we need a mechanism to enable a large capacity self-organizing memory to build the representations with great fault tolerance.

Secondly, it should be noted that the hierarchical memories, referred in the memory-based intelligence, are not only for passive information processing and pattern storage, but for continuous and active learning. Thus an agent that knows how to handle one problem in a set environment but cannot learn new skills or adapt to new environment should not be considered intelligent. Continuous and active learning requires not only information storage but also the memory resources management, so that only the important memories are retained. Therefore, a mechanism is required that will act as the trigger for the agent's memory management, active interaction and continuous learning.

Thirdly, we require that an intelligent machine has a built-in mechanism to create goals for its behavior, and we desire to build a goal creation system based on hierarchical self-organizing learning memory. It uses a simple and uniform structure and interacts with the sensory and motor functions of the memory. In a sense, goal creation should result from the machine's active interaction with its environment, by perceiving successes or failures of its actions.

Following the EI framework proposed in this work, the objective of this dissertation will include several aspects. Firstly, we will propose and study efficient learning methods for hierarchical multi-layered network structure in the supervised learning applications. When multi-layered network structure, for instance, in the form of MLP, is used for supervised learning, a measure is proposed to quantify underfitting or overfitting of a network in a given learning problem. An algorithm based on such measure should be able to recognize the occurrence of overfitting by examining the training error without using a validation set and show where the process can be safely stopped so that the optimal structure of the MLP for a given problem is found.

To build working models of intelligent machines, an arbitrary and utilitarian definition of intelligence is adopted in this work and is related to the concept of embodied intelligence suggested but not defined in [Bro 91] [Pfe 99]. The definition is general enough to characterize agents of various levels of intelligence including human. It is our aim to base the design concepts of the embodied intelligence on a minimum set of requirements and mechanisms from which all traits of intelligence can be derived.

In the proposed learning paradigm, the embodied intelligent (EI) machine learns predominantly in a deliberate, continual, and goal-oriented manner. Basic requirements for EI are proposed with the detailed models, including hierarchical self-organizing memory (HSOM) and a goal creation system (GCS). According to different functions for different parts, the HSOM can be divided into three pathways: sensory pathway, motor pathway, and goal creation pathway. We will show how these elements of EI can be implemented in a hierarchical, self-organizing memory made of neurons.

The mechanisms to build object representations on HSOM in unsupervised learning are proposed and are studied to ensure the large memory storage and robust recognition capability. Other major types of memories (sequential long term and short term memories) related to this concept of embodied intelligence are studied elsewhere [Sta 07b].

The machine needs to organize its learning to acquire useful knowledge based on its goals and create goals for its behavior using a built-in mechanism. In GCS, also as the goal creation pathway in HSOM, external rewards can only be directly applied to satisfy the most primitive goals. The creation of goals on various abstract levels is stimulated by the primitive goals using the GCS' self-evolving structure. GCS is responsible for evaluating actions in relation to its goals, stimulating the learning of useful associations and representations for sensory inputs and motor outputs. The goal creation pathway finds the ontology among sensory objects, makes connections among the actions and objects, and creates the needs and affects the agent's attention. Therefore, the goal creation pathway stimulates the interaction among the three pathways (sensory, motor, and goal creation) on various abstract levels. It facilitates the growth of the hierarchy involving sensory representations, motor abilities and abstract goals. Such hierarchy includes all ingredients necessary to develop intelligence, and to motivate a machine to act on its own in its environment.

1.4 Research significance

By understanding the way the human brain remembers and builds the model of the world, and by implementing a similar mechanism in machine learning using modern technology, the power of the machine memory can be significantly improved. An intelligent machine with hierarchical self-organizing learning systems will extend the ability of machines into a brand new stage, and the artificial intelligence will have many more successful applications ahead, which can greatly affect and change our life. The significance of this work is stated as follows.

It is required to design a hierarchical network that is able to perform efficient supervised learning when the desired outputs of the network are given. In addition, it is required to have a scientific way to adjust the desired learning accuracy and to determine proper network structure, so that useful information is extracted using minimum cost without overfitting.

It is also required that the designed hierarchical network is able to perform unsupervised learning when desired outputs are not given. The object representations should be built in a predictable way and densely-coded input information should be effectively and efficiently processed leading to invariant representations of the observed objects. The memory should be able to store a large number of input patterns and have robust recognition capabilities.

It is useful and promising to equip the hierarchical memory with the appropriate mechanisms of reinforcement learning (RL). On one hand, the machines need information from the environment to build the ontological relations among perceived objects and their relations to various actions. On the other hand, it is very difficult to deliver enough information using the typical RL scalar reward without a hierarchical memory organization to deal with the representation and implementation detail. Combining them will facilitate hierarchical reinforcement learning so that a complex task can be divided into subtasks or sequential steps and execution as well as supervision of these subtasks will be accomplished by the agent itself.

It is important and meaningful to define the uniform framework of building EI and detailed design requirements and principles. Building structural and functional organization of EI has a long way to go in order to achieve human level intelligence and to benefit from the developing technologies. Having the promising and uniform framework and design principles prepared, the future work will be done more consistently.

1.5 Dissertation organization

The rest of the dissertation is organized as follows.

In Chapter 2, efficient and optimal training algorithms are presented for multi-layered network in the supervised learning. The training methods can not only save the training time and improve the training accuracy, but also they can be expanded into a network structure with various types of connections. To avoid overfitting and determine proper network structure, an optimized approximation algorithm (OAA) with a novel quantified criterion is proposed in Chapter 3. The OAA procedure utilizing a novel stopping criterion based on signal-to-noise ratio figure is demonstrated and validated using both simulated and benchmark data on optimizing the network structure and learning periods.

The framework of building EI, including the definition of EI, and the needed elements, are proposed in Chapter 4. Principles of designing an embodied intelligence with hierarchical self-organizing memory, sensory, motor and goal creation pathways are discussed.

Based on the characteristics presented for hierarchical self-organizing memory in Chapter 4, Chapter 5 presents the model for building object representations in a hierarchical self-organizing memory (HSOM) with sparse connectivity in unsupervised learning. The effective learning methods and learning abilities are presented. The memory capacities and fault tolerance of such memory are studied. In addition, the mechanism of attention-aided perception on HSOM is presented. Chapter 6 discusses how goals are created based on a hierarchical self-organizing structure for machines' goal-oriented behavior, how the relations among objects are learned, how the action is selected according to the goal, how the value system required in a typical RL is embedded in such goal-creation system and how anticipation is generated. The dissertation is concluded in Chapter 7 with future works. Figure 1.1 shows the organization of the dissertation and relations among different chapters.

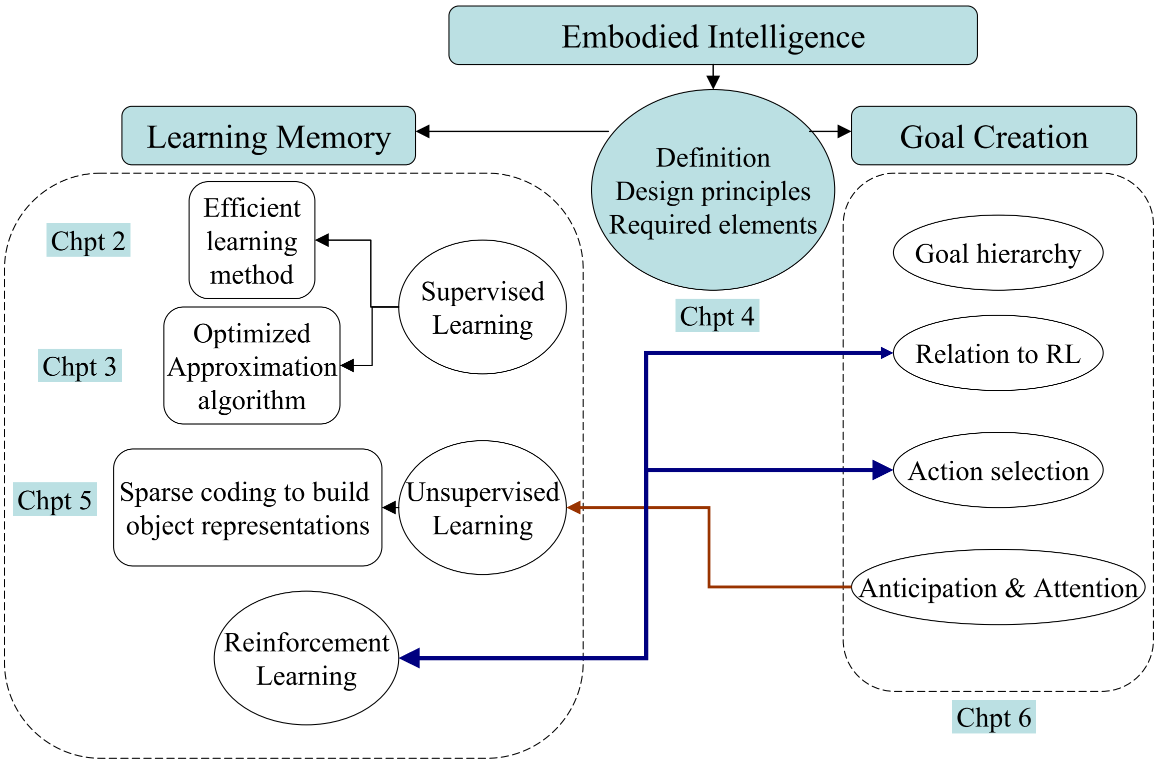


Figure 1.1. Organization of the dissertation.

CHAPTER 2: EFFICIENT SUPERVISED LEARNING OF MULTI-LAYERED NEURAL NETWORKS FOR MACHINE LEARNING

2.1 Introduction

In supervised learning, the input and the output for a hierarchical network structure are given. To achieve a proper mapping between the given set of input and output, the interconnection structure and the connection weights should be adjusted. The multi-layered networks in the form of feed-forward multi-layer perceptrons (MLP) have appeared and have been utilized as a powerful learning model for supervised learning problems [Ros 58]. The well-known and widely appreciated error back-propagation (BP) algorithm was proposed by Rumelhart [Rum 86a]. But the slow convergence speed due to its gradient-descent nature was a concern of many researchers for the last decade [Erd 05]. It takes many iterations to optimize, and it is very easily trapped in local minima. A global search procedure and a fast training method are highly desired for efficient and effective training of MLP, or networks with a similar multilayer structure.

Hebbian learning [Heb 49] as a biologically plausible learning algorithm is widely used in various neural networks. Instead of the supervised learning, Hebbian learning is more suitable for unsupervised learning when the desired network output is unknown and self-organizing is needed. Each new pattern leads to a training, which produces a change of the connection weights. In order to obtain the optimal weights to achieve a desired training performance, a large number of epochs and training data are usually needed. In addition, since Hebbian learning is a type of unsupervised learning, the change of weights may go unbounded. Variant versions of Hebbian learning were been proposed for different considerations and to make it more practical, for example, Oja's rule [Oja 82] and Generalized Hebbian Algorithm [San 89].

Least-square fitting (LSF) is a method to determine the values of unknown parameters in a statistical model by minimizing the sum of squared fitting residues (the difference between the predicted and given values). In the linear LSF, the optimal values of unknown parameters can be obtained efficiently. When the samples in the given dataset have different importance in determining the solution, the weighted version of least-squares fitting (WLSF) can be used. In the learning of multi-layered networks, the calculation used in LSF or WLSF can be applied in the optimization of the connections weights between any two layers or desired hidden signals. To introduce a uniform notation for the later discussion the LSF and WLSF are stated in the following section.

2.2. Least-squares method and weighted least-squares method

For a one-dimensional problem, with a given dataset *{xi, yi} (i=1, 2,...n)*, the unknown function *f* can be approximated, for example, using polynomials of order *B* via LSF, so that *y=f(x)*, as shown in (2.1).

$$\begin{Bmatrix} y_1 \\ \cdots \\ y_n \end{Bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^B \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_n & x_n^2 & \cdots & x_n^B \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \\ \cdots \\ a_B \end{Bmatrix} \tag{2.1}$$

as $Y = XA$

In linear regression, the unknown coefficients a_0 through a_B can be obtained using QR decomposition or singular value decomposition (SVD) as in (2.2).

$$A = (X)^{-1} Y \tag{2.2}$$

The function value y is estimated for any input x as,

$$y = \sum_{j=0}^{B} a_j x^j \ . \tag{2.3}$$

Using polynomials for function approximations could be easy and straightforward. In a typical function approximation, the basis functions could be, for instance, a set of orthogonal functions, Walsh functions, sinusoidal waves or sigmoid functions.

In certain cases, some samples from the training dataset are relatively more important and should be weighted more heavily than other ones in the function approximation. In such cases, weighted least-squares fitting (WLSF) is applied and the weighting terms can be arbitrarily determined by the user considering the characteristics of the problem that they are facing. Assuming $\lambda_i \ (i = 1, 2, \ldots n)$ represent the weighting terms applied to n samples, the WLSF is expressed as follows.

$$\begin{bmatrix} 1\cdot\lambda_1 & x_1\lambda_1 & x_1^2\lambda_1 & \ldots & x_1^B\lambda_1 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1\cdot\lambda_n & x_n\lambda_n & x_n^2\lambda_n & \ldots & x_n^B\lambda_n \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \\ \cdots \\ a_B \end{Bmatrix} = \begin{Bmatrix} y_1\lambda_1 \\ \ldots \\ y_n\lambda_n \end{Bmatrix}$$

$$\Rightarrow \Lambda \cdot XA = \Lambda \cdot Y \tag{2.4}$$

where $\Lambda = \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \cdots & \\ 0 & & & \lambda_n \end{bmatrix}$. Due to the weights applied to the given samples, the approximated function obtained from WLSF will fit into given data to different degrees of accuracy according to the weighting terms.

The following example in Figure 2.1 illustrates WLSF and compares it with the LSF.

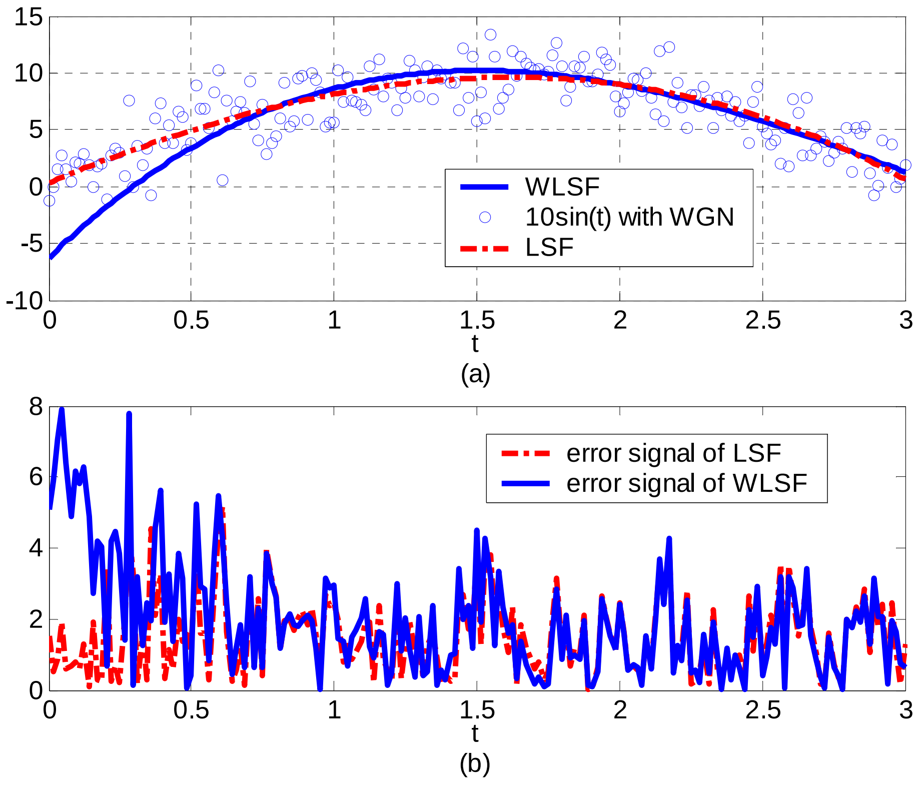


Figure 2.1. Weighted least-squared fit compared with least-squared fit.

(a). Approximated functions from WLSF and LSF.
(b). Absolute error signals from WLSF and LSF.

In this example, the training data set *{X, Y}* are obtained from a sine wave added with white Gaussian noise. The weighting terms adopted in this example are

$$\lambda_t = \frac{\alpha^t}{n} \quad (t = 1, 2, \dots n), \tag{2.5}$$

where we can define $\alpha^n = n$. A 2nd order polynomial is used to approximate the function. The function is obtained using both WLSF and LSF, shown in Figure 2.1 along with the corresponding absolute error.

It is noted from Figure 2.1 that WLSF fits the function more to the samples with larger value of *t*. And, by observing the absolute error signals from LSF and WLSF, we can find that absolute errors from WLSF are smaller for those samples than the corresponding absolute errors from LSF.

2.3 Least-squares-based multi-layered perceptron training with weighted adaptation

The training of MLP can be performed using the concept of LSF to find the global optimal solution for a given problem. The algorithm finding MLP weights using linear LSF was presented in [Erd 05] as an initialization method, based on which further back-propagation (BP) training can be applied. In fact, we can demonstrate that a properly applied algorithm can be an effective training method for MLP even without the refinement of weights by BP algorithm.

Given the inputs and desired outputs in the training samples, the training procedure for MLP can be carried out in two fashions: **back-propagation starting with the desired output signal and forward-propagation starting with the given input signal**.

In this work, the least-square-based MLP training method is introduced. Considering the distribution of the training samples and the type of nonlinearity of a MLP transfer function, weighted adaptation can be applied at various steps of the LS-based MLP training to improve the training performance.

2.3.1 Training from the desired output back-propagation

2.3.1.1 Training algorithm

The LS-based MLP training algorithm in [Erd 05] consists of two major steps: propagation of the given desired outputs in the training data backwards through layers and then optimization of the weights between layers. To back-propagate the desired outputs, random weights and biases are initially used to calculate the signals on each layer. Once the signals on the second layer are obtained based on the desired output information, the weights and biases between the first and the second layer can be optimized using the given input signals on first layer and signals on the second layer by LSF. Then the signals on the second layer are re-evaluated based on the given inputs and the optimized weights and biases. After that, signals on the second and third layers are used to optimize biases and weights between the second and the third layer and the signals of the third layer are re-evaluated. The optimization of weights and biases and recalculation of the signals on the hidden layers are conducted in a feed forward fashion (moving from lower to higher layers) until the output layer is reached.

Specifically, in a 3-layered MLP shown in Figure 2.2, the inputs x ($x \in \Re^{n_{in} \times N}$) and desired outputs d ($d \in \Re^{n_{out} \times N}$) are given in the training data. The weights and biases between the input and hidden layer are denoted as W^1 ($W^1 \in \Re^{n_{hidden} \times n_{in}}$) and

b^1 $(b^1 \in \Re^{n_{hidden} \times 1})$. The weights and biases between the hidden and output layer are denoted as W^2 $(W^2 \in \Re^{n_{out} \times n_{hidden}})$ and b^2 $(b^2 \in \Re^{n_{out} \times 1})$. The signals on the hidden layers before and after the nonlinearity are denoted as y^1, z^1 $(y^1, z^1 \in \Re^{n_{hidden} \times 1})$. The output signals before and after the output layer are y^2, z^2 $(y^2, z^2 \in \Re^{n_{out} \times 1})$. They are calculated as

$$\begin{matrix} y^1 = W^1 x + b^1 \\ z^1 = f^1(y^1) \end{matrix} \quad (y^1, z^1 \in \Re^{n_{hidden} \times N}), \tag{2.6}$$

$$\begin{matrix} y^2 = W^2 z^1 + b^2 \\ z^2 = f^2(y^2) \end{matrix} \quad (y^1, z^1 \in \Re^{n_{out} \times N}). \tag{2.7}$$

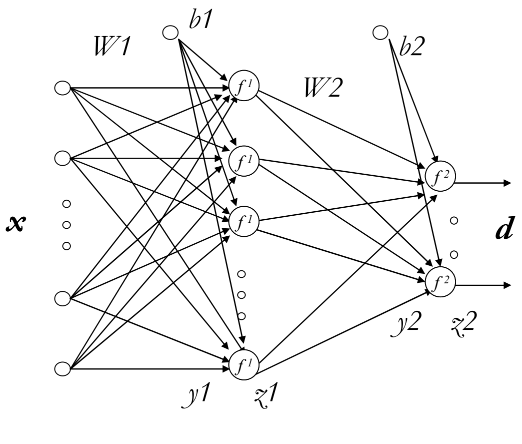


Figure 2.2.. A typical 3-layered MLP.

In (2.6) and (2.7), f^1 and f^2 are transfer functions of neurons on the hidden and output layers, whose inverse functions are denoted as $(f^1)^{-1}$ and $(f^2)^{-1}$, respectively. The transfer functions can be either linear or nonlinear, for example, the hyperbolic tangent sigmoid transfer function defined as

$$z = \frac{1 - e^{-2y}}{1 + e^{-2y}} \tag{2.8}$$

The basic LS-based MLP training algorithm (LSMLP-1) is described based on a 3-layered MLP as follows. The procedure was discussed in [Erd 05] and is restated here for clarity and further discussion.

Note that this algorithm can be easily extended into MLP with more hidden layers.

LSMLP-1 Algorithm:

Step 1). Initialize the weights (W^1 and W^2) and biases (b^1 and b^2) in the network.

Step 2). Back-propagate the desired output signals d through the output layer. Calculate the signal on the output layer before its transfer functions, $\hat{d}$, as

$$\hat{d} = (f^2)^{-1}(d)\,. \tag{2.9}$$

Step 3). Based on the weights W^2 and bias b^2 between the hidden and the output layer, calculate the output signal z^1 on the hidden layer using LSF to satisfy $W^2 z^1 + b^2 = \hat{d}$, as in (2.10), where $\begin{bmatrix} W^2 & b^2 \end{bmatrix}^{-1}$ represents the pseudo-inverse of the matrix.

$$\begin{Bmatrix} z^1 \\ 1 \end{Bmatrix} = \begin{bmatrix} W^2 & b^2 \end{bmatrix}^{-1} \hat{d} \tag{2.10}$$

Step 4). If f^1's output is limited within a certain range, the signal z^1 should be linearly scaled into that range. Back-propagate the signals on the hidden layer through the neurons' transfer function to obtain the hidden neurons activation signals, as

$$y^1 = f^{-1}\left(z^1\right). \tag{2.11}$$

Step 5). Optimize W^1 and b^1 using LSF to satisfy $W^1 x + b^1 = y^1$, as

$$\left[W^1 \quad b^1\right] = y^1 \begin{Bmatrix} x \\ 1 \end{Bmatrix}^{-1}. \tag{2.12}$$

Step 6). Evaluate y^1, z^1using the new values of the first layer weight W^1 and bias b^1, as in (2.6).

Step 7). Optimize W^2, b^2 using the LSF to satisfy $W^2 z^1 + b^2 = \hat{d}$, based on z^1 from Step 6) and $\hat{d}$ from Step 2).

$$\left[W^2 \quad b^2\right] = \hat{d} \begin{Bmatrix} z^1 \\ 1 \end{Bmatrix}^{-1}. \tag{2.13}$$

Step 8). Evaluate y^2, z^2 using the new values of the weight W^2 and bias b^2, as in (2.7). Then calculate the normalized value of mean squared error (MSE) between obtained output z^2 and desired output d, as in (2.14).

$$J = \frac{E\left[\left(d - z^2\right)^T \left(d - z^2\right)\right]}{E[d^T d]}. \tag{2.14}$$

This is the end of one iteration of the algorithm. The procedure from Step 2) through Step 8) can go through several iterations until the resulting MSE does not vary much anymore.

The set of the resulting weights and biases obtained during these iterations that can achieve the best results will be recorded as the final result. Since in this procedure, the desired output is propagated back through layers and uses the LSF to obtain the optimal weights and biases, it yields a single global optimum solution for all stages of the applied procedure. It has been tested that this algorithm usually takes 2 or 3 iterations to get stable results. It enables the fast learning of MLP, and avoids local minima characteristics of BP. As a result, it can handle several learning benchmarks, while BP training cannot.

2.3.1.2 Weights optimization with weighted least-squared fitting

In LSMLP, the signals are propagated backwards through the nonlinear transfer functions for weights optimization, as discussed in Step 5) and Step 7). If a sample's operating point (neuron's input signal value) on the nonlinear function is close to the saturation point, the variance caused by an error on this sample before the nonlinearity results in a small variance after the nonlinearity. As shown in Figure 2.3, the error before the nonlinearity *Δy* of sample A (whose operating point is at A) results in a larger error *Δz* after the nonlinearity than sample B's with the same amount of error *Δy*.

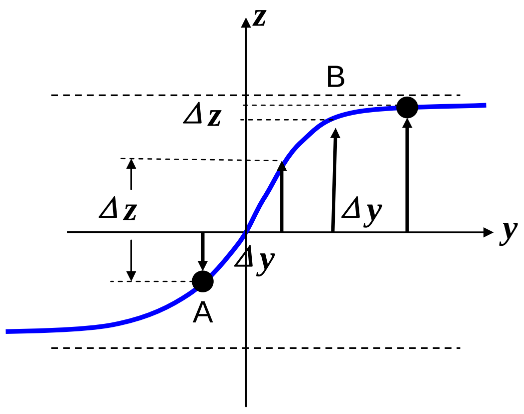


Figure 2.3. Nonlinear transfer function and training samples' operating point.

Therefore, the samples far from the saturation will have larger effect on the MSE after the nonlinearity, and should have larger weighting terms when the weights and biases are optimized to satisfy $W^1 x + b^1 = y^1$, as in Step 5). The weighting terms should reflect the nonlinear function sensitivity to changes in the sample's value. For example, a sample weighting term λ_{ji} can be set to derivative value of the neuron's transfer function computed at the input signal value y_{ji}, as in (2.15), where $f^{1'}$ denotes the derivative of f^1.

$$\lambda_{ji} = f^{1'}\left(y_{ji}\right) \; (j = 1, 2, \ldots n_{hidden}, i = 1, 2, \ldots N), \tag{2.15}$$

In such case, the weights and biases between input and the hidden layer are calculated as,

$$\begin{gathered} \begin{bmatrix} W^1_{j:} & b^1_j \end{bmatrix} \begin{Bmatrix} x \\ 1 \end{Bmatrix} \Lambda = y_{j:}^{\;1} \; \Lambda, \\ j = 1, 2, \ldots n_{hidden}, \; \Lambda = \begin{bmatrix} \lambda_{j1} & & & 0 \\ & \lambda_{j2} & & \\ & & \cdots & \\ 0 & & & \lambda_{jN} \end{bmatrix} . \end{gathered} \tag{2.16}$$

For example, if the neurons use the hyperbolic tangent transfer function, the weighting terms expressed through derivatives are

$$\lambda(y) = \frac{dz}{dy} = 1 - \left(\frac{1 - e^{-2y}}{1 + e^{-2y}} \right)^2 = (1 + z)(1 - z). \tag{2.17}$$

Notice that such defined weights are always positive and when z approaches ± 1 (that corresponds to large $|y|$ and saturation region of nonlinear function), this weight approaches 0. The results from LSF and WLSF for weights optimization are compared in Figure 2.4. Before the nonlinearity, approximations from LSF in Figure 2.4 (a) show closer match between the given value and the approximated values over all the samples than that from WLSF in Figure 2.4(c). However, after the nonlinearity, norm of error from WLSF is 6.4717 and the error norm from LSF is 7.1324, which demonstrates better quality of the fit due to minimization of error in the sensitive region.

Therefore, WLSF can be used in Step 5) and Step 7) to optimize the weights and biases to reduce the output error norm after nonlinearity transformation.

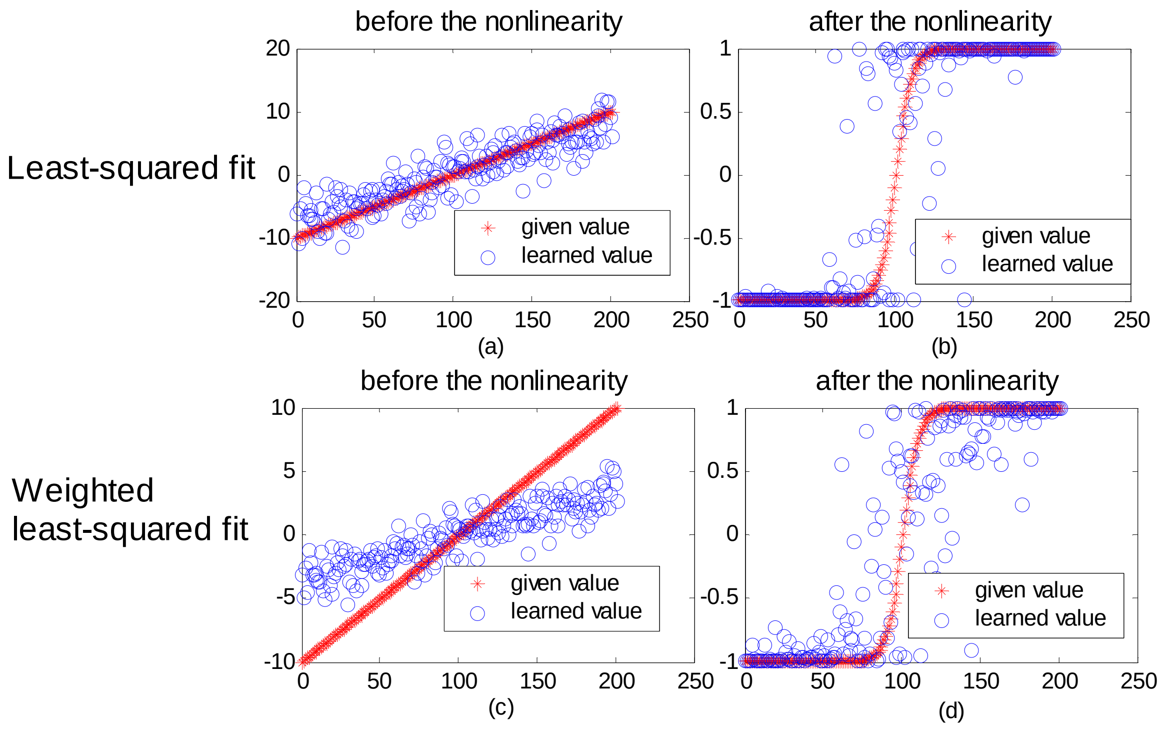


Figure 2.4. Comparison between LSF and WLSF.

2.3.1.3 Weights optimization with iterative fitting

After a weight matrix between certain two layers is calculated, the signals can be propagated to the output layer of the MLP to find out what the final error signal will be. For a 3-layered MLP, when W^1 and b^1 are optimized, the mismatch between the desired signal y^1 on the next layer and the actual one, denoted as e^1, is obtained as,

$$e^1 = y^1 - (W^1 x + b^1). \tag{2.18}$$

Due to e^1, the error signal on the output layer before the transfer function will be

$$e_{out} = \hat{d} - y^2 = \hat{d} - \left[W^2 f^1(W^1 x + b^1) + b^2\right]. \tag{2.19}$$

In order to reduce e_{out} at the current step, W^1 and b^1 are further adjusted after the WLSF. The signals on the output layer y^2 and e_{out} can be expressed as

$$\begin{aligned} e_{out} &= \hat{d} - y^2 = \hat{d} - W^2 f^1(W^1 x) \\ &= \hat{d} - \left[W^2_{:1} f^1(W_{1:}{}^1 x) + W^2_{:2} f^1(W_{2:}{}^1 x) + \ldots + W^2_{:n_{hidden}} f^1(W_{n_{hidden}:}{}^1 x)\right], \end{aligned} \tag{2.20}$$

where $W_{i:}{}^1$ denotes the i^{th} row of W^1, and $W^2_{:j}$ denotes the j^{th} column of W^2.

We can consider the hidden neurons as the basis functions for the output, in the form as $W^2_{:j} f^1(W_{j:}{}^1 x)$ for the j^{th} basis function. Then, e_{out} can be reduced by adjusting each of its basis functions. This process starts the weights going to the 1st hidden neuron, $W_{1:}{}^1$. By adjusting $W_{1:}{}^1$ into $W_{1:}{}^1 + \Delta W_{1:}{}^1$, we attempt to have (2.21) so that the e_{out} will be zero.

$$W^2_{:1} f^1[(W_{1:}{}^1 + \Delta W_{1:}{}^1)x] + W^2_{:2} f^1(W_{2:}{}^1 x) + \ldots + W^2_{:n_{hidden}} f^1(W_{n_{hidden}:}{}^1 x) = \hat{d} \tag{2.21}$$

Using Taylor expansion for nonlinear function f^1 around the operating point $W_{1:}^{1}x$, we can have,

$$
\begin{gathered}
W_{:1}^{2} f^{1}(W_{1:}^{1}x) + W_{:1}^{2}\frac{df^{1}}{dx}(W_{1:}^{1}x)\Delta W_{1:}^{1}x + W_{:2}^{2} f^{1}(W_{2:}^{1}x) + \ldots + W_{:n_{hidden}}^{2} f^{1}(W_{n_{hidden}:}^{1}x) = \hat{d} \\
W_{:1}^{2}\frac{df^{1}}{dx}(W_{1:}^{1}x)\Delta W_{1:}^{1}x = \hat{d} - y^{2} = e_{out} \\
\Delta W_{1:}^{1} = \left(W_{:1}^{2}\frac{df^{1}}{dx}(W_{1:}^{1}x) \right)^{-1} \cdot e_{out} \cdot (x)^{-1}
\end{gathered}
\quad . (2.22)
$$

The procedure continues until e_{out} is zero or it updates all the hidden neurons. The process iteratively reduces e_{out} by further adjusting W_1 and b_1, so will be called Iterative Fitting (ITF) for weights optimization or least-squared based MLP learning with iterating fitting (LSMLP-ITF).

To show the effect of ITF, the training performance of MLP will be presented on a popular classification benchmark problem, the two-spiral classification problem. The two-spiral problem is known as difficult to achieve the perfect classification due to high nonlinearity. The spiral dataset contains 50 samples with 2 features, from two classes. Using the given classes IDs ($d \in \{-1, 1\}$) as the desired MLP output, a 3-layered MLP, with size 2-50-1, is trained using LSMLP-1 algorithm. Adding the ITF as an extra step, the training performance is greatly improved over direct LSMLP fitting, as shown in Figure 2.5.

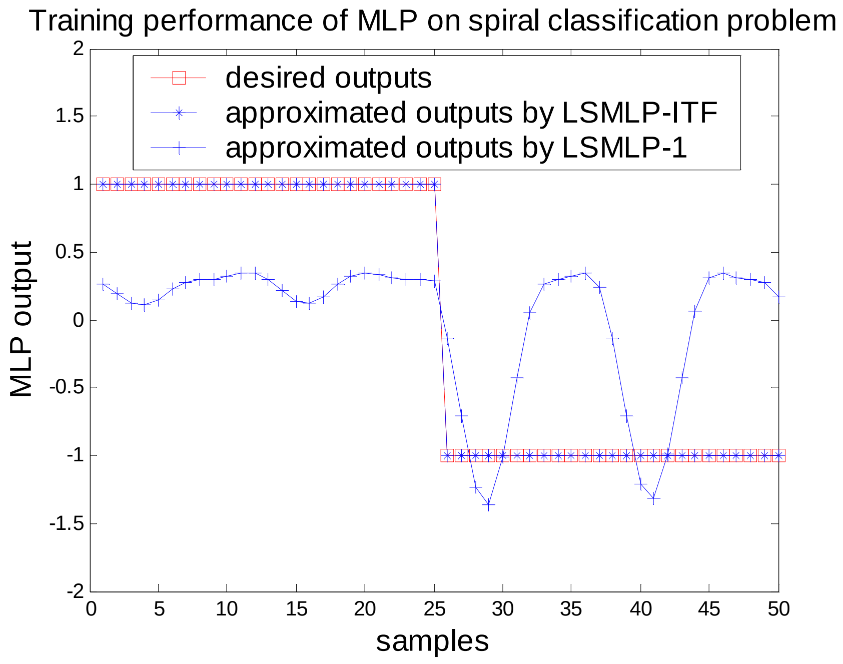


Figure 2.5. Training performances of LSMLP-ITF and LSMLP-1

2.3.2 Training from input signal forward-propagation

2.3.2.1 Training algorithm

The alternative fashion of the LS-based MLP training algorithm starts with the input signal forward-propagation. It consists of two major steps: propagation of the given inputs in the training data forwardly through layers and then optimization of the weights between layers. To propagate the input signals, random weights and biases are initially used to calculate the signals on each layer. Once the signals on the last hidden layer are obtained, the weights and biases between the last hidden layer and the output layer can be optimized. Then the signals on the last hidden layer are re-calculated based on the given desired outputs and the optimized weights and biases. Subsequently, the weights and biases of previous layer will be updated, so will the signals on previous layers. The optimization of weights and recalculation of the signals on the hidden layers are conducted until the input layer is reached. The algorithm (LSMLP-2) is described based on a 3-layered MLP as follows.

LSMLP-2 Algorithm:

Step 1). Initialize the weights (W^1 and W^2) and biases (b^1 and b^2) in the network.

Step 2). Evaluate y^1, z^1using W^1 and b^1, as in (2.6).

Step 3). Back-propagate the desired output signals d through the output layer. Calculate the signal before the transfer functions on the output layer, $\hat{d}$, as in (2.9).

Step 4). Optimize W^2, b^2 to satisfy $W^2 z^1 + b^2 = \hat{d}$, based on z^1 from (ii) and $\hat{d}$ from Step 3).

Step 5). Based on the new W^2 and b^2, optimize z^1 using LSF to satisfy $W^2 z^1 + b^2 = \hat{d}$, as in (2.10).

Step 6). Scale z^1 into the $f^{\,1}$'s output range. Then back-propagate z^1 through the neurons' transfer function.

Step 7). Optimize W^1 and b^1 using LSF to satisfy $W^1 x + b^1 = y^1$, as in (2.12).

Step 8). Evaluate y^1, z^1, y^2, z^2 using the new W^1,W^2 and b^1,b^2. And calculate the mean squared error (MSE) between obtained output z^2 and desired output d.

This is the end of one iteration of the algorithm. The procedure from Step 2) to Step 8) can go through several iterations until the resulting MSE does not vary much anymore.

The set of the resulting weights and biases, which can achieve the best results, will be recorded as the final result.

2.3.2.2 Signal optimization with weighted adaptation

In the steps of LSMLP-2 algorithm in which the desired signals are scaled down to the limit of the nonlinear outputs before being back-propagated through the layer, the weights of the following layer should be modified accordingly, since the scaling process would produce mismatch of the signals on the next layer. It is different than in the procedure of LSMLP-1 in which this modification is not necessary. An alternative is that in Step 5) of LSMLP-2 algorithm, after W^2 and b^2 are updated, z^1 is optimized considering the nonlinearity limit. After Step 4), the obtained signal y^2 will be different from $\hat{d}$, as

$$\hat{e}^2 = \hat{d} - (W^2 z^1 + b^2). \tag{2.23}$$

To reduce the error $\hat{e}^2$, z^1 needs to be optimized, so that

$$\begin{aligned} W^2(z^1 + \Delta z^1) &= \hat{d} - b^2 \\ W^2 \Delta z^1 &= \hat{e}^2 \end{aligned}. \tag{2.24}$$

z^1's values are limited to a certain interval, typically, a (-1, 1) interval. Distances of z^1 to the maximum and the minimum value of the interval (shown in Figure 2.6) are determined as

$$\begin{aligned} \delta_{z^1 \max} &= 1 - z^1 \\ \delta_{z^1 \min} &= -1 - z^1 \end{aligned}. \tag{2.25}$$

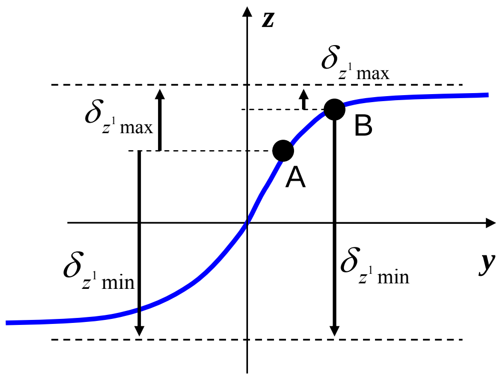


Figure 2.6. z^1's distances to the nonlinearity output limit

The allowable amount of change of z^1 is

$$\delta_H = \begin{cases} \delta_{z^1 \max}, & \textit{if } \operatorname{sgn}(W^2 \cdot \hat{e}^2) > 0 \\ \delta_{z^1 \min}, & \textit{if } \operatorname{sgn}(W^2 \cdot \hat{e}^2) < 0 \end{cases}. \tag{2.26}$$

The allowable amount of change δ_H of z^1 shows how much the signal can be changed and affects how significant it is to update this signal. We can combine δ_H, as the other type of weighting terms, with the weighting terms determined by the signals' operating points on the nonlinearity (as discussed in Section 2.2.2.2) so that each sample of Δz^1, denoted as $\Delta z^1_{(:i)}$ *(i=1, 2,…N)*, is computed using WLSF in (2.27).

$$\begin{gathered} W^2 \Delta z^1_{(:i)} \cdot \Lambda = \hat{e}^2 \cdot \Lambda \\ \Lambda = \begin{bmatrix} \delta_{H(1,i)} f^{1'}(y^1_{1,i}) & & & 0 \\ & \delta_{H(2,i)} f^{1'}(y^1_{2,i}) & & \\ & & \cdots & \\ 0 & & & \delta_{H(n_{hidden},i)} f^{1'}(y^1_{n_{hidden},i}) \end{bmatrix} \\ \Delta z^1_{(:i)} = \left(W^2\right)^{-1} \hat{e}^2 \cdot \Lambda \cdot (\Lambda)^{-1} \end{gathered} \tag{2.27}$$

2.4 Simulation and discussion

The 3 variant versions of least-squared based MLP learning algorithms, including LSMLP-1, LSMLP-ITF, LSMLP-2, are tested and compared using the following examples.

Example 1

Firstly, the spiral classification problem, used in Section 2.3.1.3, is tested using these algorithms and the performances are compared in Table 2.1. It is noted that LSMLP-1 is not able to handle spiral problem very well, while the other two types make very close approximation of the training data.

Example 2

The identification problem of the engine dynamics [Pow 98] is tested for MLP learning. The dataset contains 999 samples. A 3-layered MLP, with size 4-5-1, is trained using these 3 algorithms. The performances are compared in Table 2.2.

On this problem, all algorithms can handle the identification problem, while LSMLP-2 generates the best performance. Based on these two experiments, the LSMLP-2 algorithm seems to have better learning capability.

Table 2.1

Performance Comparison on Spiral Classification Problem

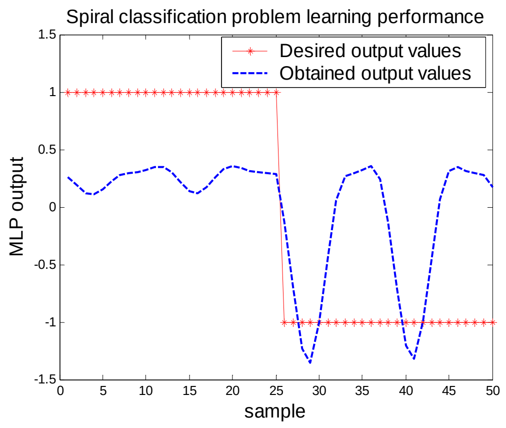
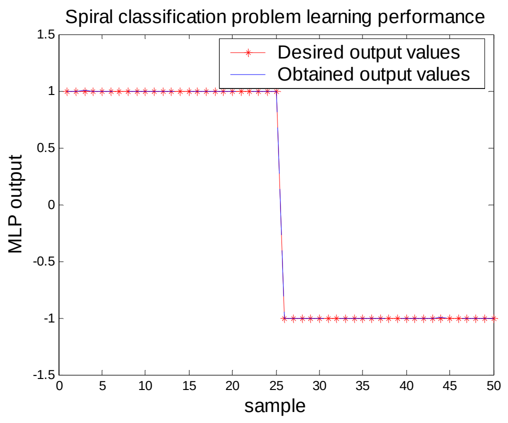
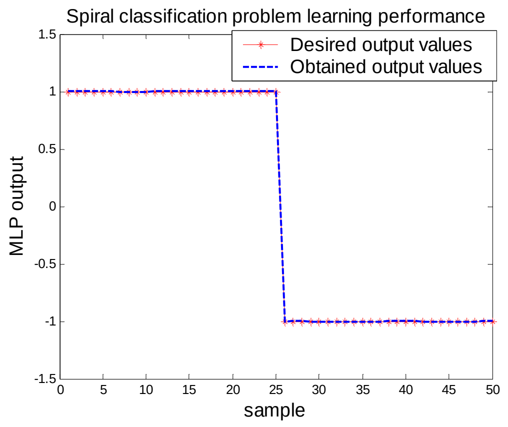
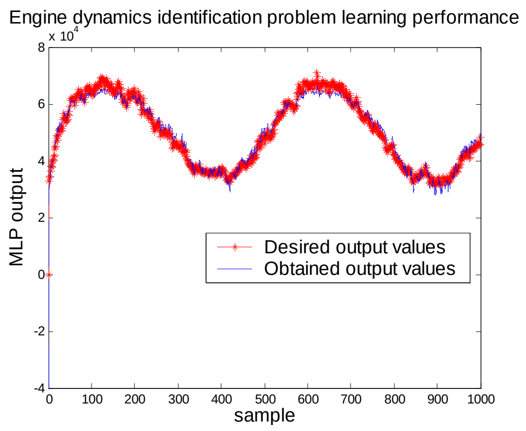
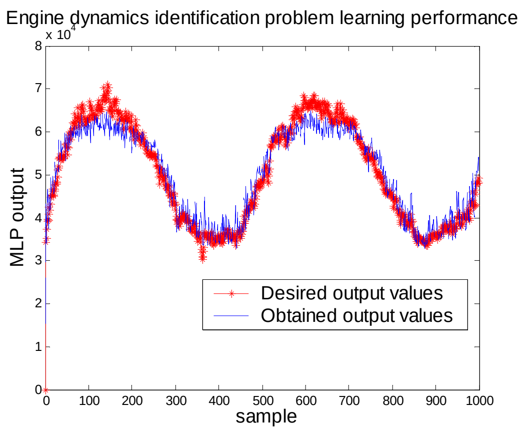
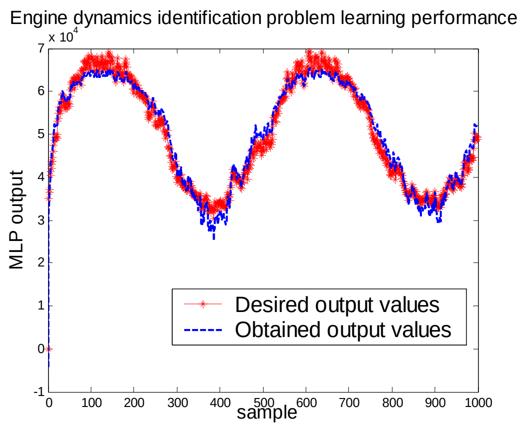
	Learning results	Normalized MSE
LSMLP-1		0.74622
LSMLP-ITF		0
LSMLP-2		0

Table 2.2

Performance Comparison on Engine Dynamics Identification Problem

	Learning results	Normalized MSE
LSMLP-1		0.0026
LSMLP-ITF		0.0082
LSMLP-2		0.0017

2.5 Conclusions

The weights of a MLP with a preset structure can be efficient and globally optimized using these least-squares based learning algorithms. They can be applied not only to classical feed-forward network structure, but networks with other types of connections (feedback or lateral connections) as well. According to the signal flows in different networks, these types of algorithms can be tailored, while the concept of LSF or WLSF can always apply. It makes MLP or similar network structures more suitable for building complex systems in supervised learning problems. In Chapter 3, we will discuss how to determine the network structure using optimized approximation algorithm.

CHAPTER 3: OPTIMIZED APPROXIMATION ALGORITHM FOR SUPERVISED LEARNING

3.1 Introduction

Supervised learning problem using multi-layered networks can be looked as and handled as unknown function approximation or a model approximation problem. Neural networks (NN) in the form of feed-forward multi-layer perceptrons (MLP) are often utilized in such problems [Gal 90]. In neural network learning, adding more hidden neurons is equivalent to adding more basis functions in function approximation and affects the training accuracy. In addition to the number of hidden neurons, the training accuracy could also be affected by several other parameters, including the number of layers, the number of training samples, the length of learning period, the choice of neuron activation functions, and the training algorithm. Previous work has shown that neural networks can be used as universal approximators [Lor 66] [Hor 89] [Zur 92]. For universal approximators, how to determine the proper parameters to use in the model without a pre-set target for training accuracy is one of the major challenges.

In order to optimize the number of hidden neurons, several techniques have been developed in the MLP-related literature, which correlate it with the number of training samples or the input and output layer sizes [Swi 96][Ber 97] [Bog 97]. Other work estimates the complexity of the desired function and relates it to the number of hidden neurons [Cam 01]. If the neural network training uses back-propagation (BP) algorithm, it has been shown that increasing the number of hidden neurons and the number of weights makes it easier to find the global minimum [Law 96] [Law 97]. However, without examining the goodness-of-fit or considering the statistical characteristics of the training data, these approaches are less theoretically sound. Geometric interpretation given in [Xia 05] provides some insight into the problem of determining the number of neurons. It helps to find the minimum structure of MLP necessary for a satisfactory approximation of a given problem. However, such a method can be only applied to problems with the input space's dimensionality up to two and in most of the applications, it is difficult to estimate the order of the problem by observing the available noisy data. Some work [Hua 98][Hua 03] [Sar 15] [Tam 97] on estimating the number of hidden neurons focused on the learning capabilities of the MLP on a training dataset without considering the possibility of overfitting.

Using an excessive number of basis functions will cause overfitting, which means that the approximator over-estimates the complexity of the target problem. This is usually referred to as the bias/variance dilemma [Gem 92]. The major purpose of developing function approximation is to interpolate in a meaningful way between the training samples [Hol 92], in order to generalize a model from existing training data and make predictions for novel data. Such generalization capability, usually measured by the generalization error [Kar 00], is degraded by overfitting, which leads to a significant deviation in prediction. It was addressed in [Xia 05] that finding the minimum structure of MLP in most cases results in the least cost of computation, least requirements on implementation resources and best generalization. In this sense, determining the optimum number of neurons or finding the minimum structure to prevent overfitting are critical in function approximation.

During back-propagation (BP) training in MLP, the weights are adjusted incrementally. Therefore, besides the network size, training accuracy also depends on the number of training epochs. Too many epochs used in BP training will lead to overtraining, which is a concept similar to overfitting.

To find the optimal network structure with an optimal size of the hidden layer or optimal value of a certain network parameter, constructive/destructive algorithms were adopted to incrementally increase or decrease the parameter to be optimized [Alp 91] [Kwo 97] [Ree 93] [Fre 90]. During the constructive/destructive process, cross-validation is commonly used to check the network quality [Set 01] and the design parameter is chosen using early-stopping [Ama 97] [Pre 98] [Wan 93]. In these approaches, the available data are divided usually into two independent sets: a training set and a validation or testing set. Only the training set participates in the neural network learning, and the validation set is used to compute validation error, which approximates the generalization error. The performance of a function approximation during training and validation is measured respectively by training error ε_{train} and validation error ε_{valid} presented, for instance, in the form of mean-squared-error (MSE). Once the validation performance stops improving as the target parameter continues to increase, it is possible that the training has begun to fit the noise in the training data, and overfitting occurs. Therefore, the stopping criterion is set so that, when ε_{valid} starts to increase, or equivalently when ε_{train} and ε_{valid} start to diverge, it is assumed that the optimal value of the target parameter has been reached.

Singular value decomposition (SVD) approach was also used to quantify the significance of increasing the number of neurons in the hidden layer in the constructive/destructive process [Teo 06]. The number of neurons is considered sufficient when contributory effect of each additional neuron is lower than an arbitrary threshold. There are several other model selection criteria, such as Akaike's information criterion (AIC) [Aka 74] and the minimum description length (MDL) [Ris 86], as a function of the model complexity, the training performance and the number of training samples. Some work applied such information criteria in the problem of finding optimal neural network structures [For 91] [Mur 94]. AIC was introduced in order to maximize the mean log-likelihood of a model while avoiding unnecessary complexity. A penalty term was applied to make model with excessive number of independent parameters less desirable. The algorithm using AIC as stopping criterion will choose the model with the minimum AIC. The bias/variance decomposition [Hua 98] is a method to decompose the bias and variance term from MSE and measure the sensitivity of a learning model to the training data. Fitting into the available data will reduce the bias while overfitting may induce large variance. In practice, the bias and variance components for a certain learning model are estimated statistically over several training sets samples from the same function. Among several model choices, the one with least bias and variance is chosen as the optimum. Overall, cross-validation and early-stopping are still the common techniques used in finding optimal network structure up to date.

Nevertheless, in cross-validation and early stopping, the use of the stopping criterion based on ε_{valid} is not straightforward and requires definite answers to several issues. For example, users have to find out the distribution of data so that training and validation sets can be properly divided and to assure that each of them have good coverage of the input space. In addition, as demonstrated in [Law 96], the validation data have to be representative enough with regard to its size and data distribution, so that ε_{valid} can provide an unbiased estimate of the actual network performance and the real generalization error ε_{gen}. As validation data are statistically sampled, ε_{valid} has only a statistical chance to correlate with the generalization error, thus it is not a reliable measure. ε_{valid}, as a function of target parameter, may have many local minima during the training process. It is not definite which one indicates the occurrence of overfitting [Pre 98] [Wan 93] and it is even more difficult to find out how likely it is that overfitting actually happened. Therefore, during the constructive/destructive process, users have to go through the process of adjusting the target parameter and observing the variation of ε_{valid} to vaguely determine a good place to stop, which is a somewhat empirical and a not well-quantified process. Three classes of better-defined stopping criteria based on the concept of early-stopping were proposed in [Pre 98], from which users can choose based on different concerns on efficiency, effectiveness or robustness. The first class of stopping criteria (GL) proposes to stop training as soon as the generalization loss, measured by the increase of ε_{valid}, exceeds a certain threshold. The second class (PQ) evaluates the quotient of generalization loss and training progress so that even if generalization error increases, the rapid decrease of training error will suggest continuation of the process. The third class (UP) suggested stopping the process when the generalization error kept increasing in several successive steps. It helped the users to choose stopping criterion in a systematic and automatic way to avoid the ad-hoc process. However, as long as cross-validation is used, the methods require omission of the validation set in the training stage, which is a significant waste of the precious data available for training in some real-life cases, eg. plant dataset [Sug 93].

In general, overfitting occurs when excessive numbers of neurons are used in the network. In these cases, although the ε_{valid} may not be severely degraded, the network does overestimate the complexity of the problem and it cost more resources to train and implement. The case of severe overfitting that went undetected using the validation set can be easily illustrated with an example of a synthetic data set obtained from a noisy sine wave signal approximated by polynomial functions. Figure 3.1(a) shows training and validation data sets. Figure 3.1(b) shows the values of training and validation errors as a function of the orders of approximating polynomials. Also shown in Figure 3.1(b) is (usually unknown) generalization error, which measures the deviation of the approximating result from the original sine wave. As illustrated in Figure 3.1(b), the validation error did not increase significantly enough to indicate severe overfitting that occurs when the order of approximating polynomial was higher than 18.

Thus, it is desirable to have a measure that can quantify underfitting or overfitting of a network in a given learning problem. An algorithm based on such a measure should be able to recognize the occurrence of overfitting by examining the training error without using a validation set and show where the process can be safely stopped so that the optimal structure of the MLP for a given problem is found. In this paper, a signal-to-noise ratio figure (SNRF) is defined to measure the goodness of fit using the training error. Based on the SNRF measurement, an Optimized Approximation Algorithm (OAA) is proposed to avoid overfitting in function approximation and neural network design applications.

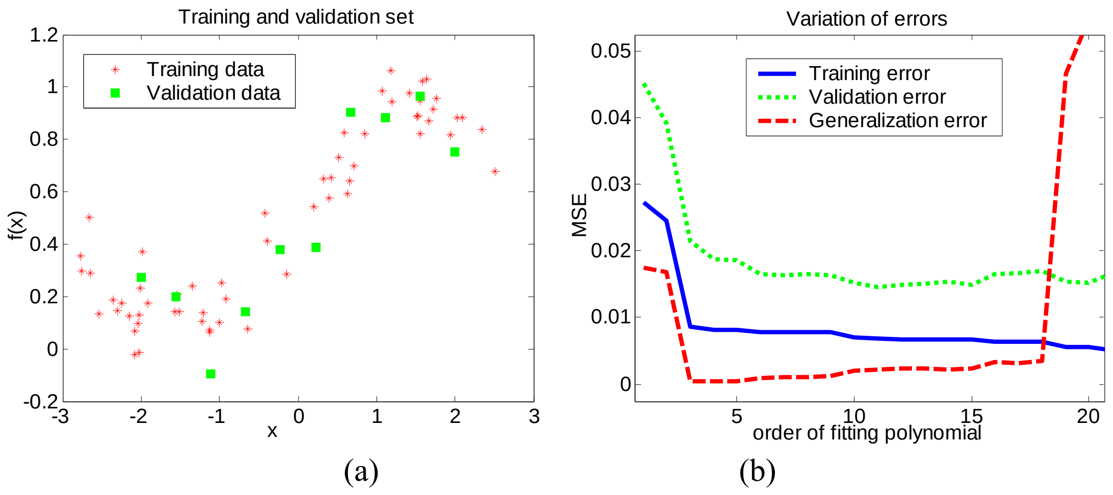


Figure 3.1. Validation error and generalization error on overfitting detection.

(a) Training and validation set
(b) Variation of errors in function approximation

3.2 Estimation of signal-to-noise ratio figure

3.2.1 SNRF of the error signal

In order to have a clear indication of overfitting, we need to examine the difference between the approximated function and the training data. This difference, which is defined as the error signal in this work, comes from two possible sources: the approximation error due to the limited training accuracy in approximation with the given set of basis functions, and an unknown level of noise in the training data. The noise can be the result of multiple causes, such as input noise, output noise or system disturbance which will all be treated as the output noise. In function approximation, without any knowledge of the noise sources and based on central limit theorem, we can assume the White Gaussian Noise (WGN) without losing generality. A critical question is whether there is still useful signal information left to be learned in the error signal. If there is, based on the assumption that the target function we try to approximate is continuous and that the noise is White Gaussian Noise (WGN), we can estimate the level of signal and noise in the error signal. The ratio of the estimated signal level over the noise level in the error signal is defined as SNRF, and it is used to measure the amount of information left unlearned in the error signal. The SNRF can be pre-calculated for a signal that contains solely WGN. The comparison of SNRF of the error signal with that of WGN determines whether WGN dominates in the error signal. If the noise dominates, there is little useful information left in the error signal, and there is no point to reduce it anymore as this will lead to overfitting. The estimation of SNRF will be first illustrated using a one-dimensional function approximation problem, followed by the discussion for multi-dimensional problems.

3.2.2. SNRF estimation for a one-dimensional function approximation

Assume that in a one-dimensional function approximation problem, training data are uniformly sampled from the input space $X \in \Re^1$ with additive noise at an unknown level.

An approximation is obtained using a certain set of basis functions. The error signal *e* contains a noise component denoted by *n*, and an approximation error signal component, which is the useful signal left unlearned and therefore denoted by *s*.

$$e_i = s_i + n_i = s_i + \beta m_i \;\; (i = 1,2,...N)\,, \tag{3.1}$$

where *N* represents the number of samples. Without losing generality, *n* can be modeled as a WGN process with standard deviation *β*, and *m* stands for a WGN process with unit standard deviation. The energy of the error signal *e* is also composed of signal and noise components.

$$E_{s+n} = E_s + E_n \tag{3.2}$$

The energy of *e* can be calculated using the autocorrelation function:

$$E_{s+n} = C(e_i, e_i) = \sum_{i=1}^{N} e_i^{\,2}\,, \tag{3.3}$$

where *C* represents the correlation calculation. Notice that a presumption is made that the target function needs to be continuous, and the approximation $\hat{F}$ is usually a continuous function. Practically, the useful signal left unlearned, *s*, is also a continuous function. We could further assume that, if treated as time signals, the target function and $\hat{F}$ both have relatively small bandwidth compared to the sampling rate or to the noise bandwidth. As a result, there is a high level of correlation between two neighboring samples of *s*. Consequently,

$$C(s_i, s_{i-1}) \approx C(s_i, s_i)\,, \tag{3.4}$$

where s_{i-1} represents the (circular) shifted version of the *s*. Due to the nature of WGN, noise of a sample is independent on noise of neighboring samples:

$$C(n_i, n_{i-1}) = C(\beta m_i, \beta m_{i-1}) = 0\,, \tag{3.5}$$

where n_{i-1} represents a replica of n_i shifted by one-sample. Since the noise component is independent of the signal component, the correlation of e_i with its shifted copy e_{i-1} approximates the signal energy, as shown in (3.6).

$$C(e_i, e_{i-1}) = C(s_i, s_{i-1}) \approx E_s \tag{3.6}$$

The difference between the autocorrelation with no time shift defined in (3.3) and $C(e_i, e_{i-1})$ gives the noise energy in the error signal.

$$E_n = E_{s+n} - E_s = C(e_i, e_i) - C(e_i, e_{i-1}) \tag{3.7}$$

The ratio of the signal level to the noise level, defined as the SNRF of the error signal, is obtained as:

$$SNRF_e = \frac{E_s}{E_n} = \frac{C(e_i, e_{i-1})}{C(e_i, e_i) - C(e_i, e_{i-1})}\,. \tag{3.8}$$

Notice that in SNRF, the signal component and noise component are decomposed by using the correlation between neighboring samples. In the bias/variance decomposition, similar estimations of the signal or noise level are obtained from bias and variance components, which are calculated statistically in common practice.

When learning of the target function improves, it is expected that the useful signal left unlearned in the error signal is reduced, while the noise component does not change so that $SNRF_e$ will decrease. In order to detect the existence of useful signal in *e,* the $SNRF_e$ has to be compared with the SNRF estimated for WGN using the same number of samples. When there is no signal in *e*, we have,

$$e = s + n = n = \beta m\,. \tag{3.9}$$

The SNRF for WGN is calculated as,

$$SNRF_{WGN} = \frac{C(n_i, n_{i-1})}{C(n_i, n_i) - C(n_i, n_{i-1})} = \frac{C(\beta m_i, \beta m_{i-1})}{C(\beta m_i, \beta m_i) - C(\beta m_i, \beta m_{i-1})} = \frac{C(m_i, m_{i-1})}{C(m_i, m_i) - C(m_i, m_{i-1})} \quad (3.10)$$

It is observed that the *SNRF_{WGN}* is independent of the noise level *β,* which means that $SNRF_{WGN}$ only needs to be estimated with unit standard deviation in order to obtain the general characterization for any level of WGN. The expected value of the correlation $C(m_i, m_{i-1})$ is zero, which would intuitively indicate a zero $SNRF_{WGN}$. However, $SNRF_{WGN}$ is estimated using a limited number of samples, thus it is a random value related to the number of samples *N*. The average value and the standard deviation of $SNRF_{WGN}$ can be derived for a given *N*.

$$\mu_{SNRF\_WGN}(N) = \mu\left[\frac{C(m_i, m_{i-1})}{C(m_i, m_i) - C(m_i, m_{i-1})}\right] \quad (3.11)$$

Since $C(m_i, m_i) >> C(m_i, m_{i-1})$,

$$\mu_{SNRF\_WGN}(N) \approx \mu\left[\frac{C(m_i, m_{i-1})}{C(m_i, m_i)}\right] \approx \frac{\mu[C(m_i, m_{i-1})]}{N} = 0, \quad (3.12)$$

and,

$$\begin{aligned}\sigma_{SNRF\_WGN}(N) &= \sigma\left[\frac{C(m_i, m_{i-1})}{C(m_i, m_i) - C(m_i, m_{i-1})}\right] \approx \sigma\left[\frac{C(m_i, m_{i-1})}{C(m_i, m_i)}\right] \\ &\approx \frac{\sigma[C(m_i, m_{i-1})]}{N} = \frac{\sqrt{\sum_{i=1}^{N}(\sigma(m_i m_{i-1}))^2}}{N} = \frac{\sqrt{N}}{N} = \frac{1}{\sqrt{N}}\end{aligned} \quad . \quad (3.13)$$

Note that the samples of $SNRF_{WGN}$ are statistically independent. According to the central limit theorem, if *N* is large enough, the samples of $SNRF_{WGN}$ tends to follow

Gaussian distribution with mean μ_{SNRF_WGN} and standard deviation σ_{SNRF_WGN}. In Figure 3.2 (a), $\sigma_{SNRF\_WGN}(N)$ from a 10000-run Monte-Carlo simulation is shown in the logarithmic scale as a function of the number of samples. The estimated $\sigma_{SNRF\_WGN}(N)$ in (3.13) agrees with the simulation results, especially for the N values larger than 64. Such estimation is expected to work well for the sample numbers available in real-world training datasets.

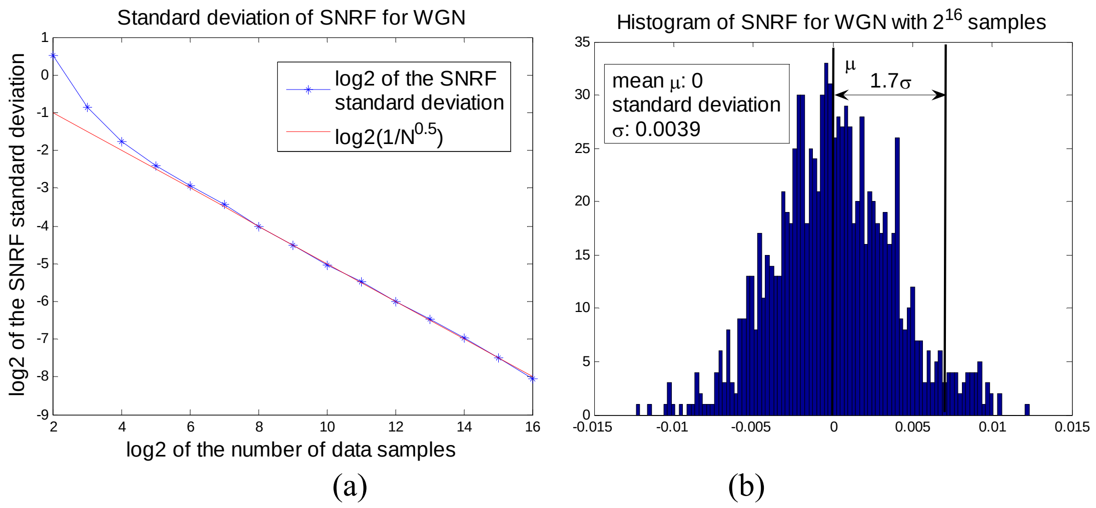


Figure 3.2. SNRF estimation for WGN in one-dimensional case

(a) Standard deviation of SNRF for WGN
(b) Histogram of SNRF of WGN for 2^{16} samples

3.2.3 One-dimensional SNRF-based stopping criterion

The stopping criterion in OAA can now be determined by testing the hypothesis that $SNRF_e$ and $SNRF_{WGN}$ are from the same population. The value of $SNRF_e$ at which the hypothesis is rejected constitutes a threshold below which training OAA is stopped. Figure 3.2 (b) illustrates the histogram of $SNRF_{WGN}$ with 2^{16} samples, as an example. It is observed that the p = 5% significance level [Leh 97] can be approximated by the average value plus 1.7 times standard deviations for an arbitrary N. As shown in Figure 3.2(b), the threshold can be calculated using $\mu + 1.7\sigma = 0 + 1.7 \times 0.0039 = 0.006$ for 2^{16} samples. Notice it agrees with the threshold of 5% significance level calculated using Gaussian distribution with mean μ_{SNRF_WGN} and standard deviation σ_{SNRF_WGN} .

SNRF-based stopping criterion in OAA can be defined as a $SNRF_e$ smaller than the threshold determined by (3.14), in which case, there is at least 95% probability that error signal represents a WGN and learning must stop to avoid overfitting.

$$th_{SNRF\_WGN}(N) = \mu_{SNRF\_WGN}(N) + 1.7\sigma_{SNRF\_WGN}(N)\,. \tag{3.14}$$

The threshold can be re-calculated for different significance levels if needed, and it is also based on the mean μ_{SNRF_WGN} and standard deviation σ_{SNRF_WGN} derived in (3.12) (3.13).

In above discussion, (3.6) and (3.7) have been developed based on the assumption that e could be treated as a signal with evenly-spaced samples. In a general one-dimensional function approximation problem, the input samples may be unevenly spaced. Yet, E_{s+n} , E_s and E_n can still be approximated using (3.3), (3.6) and (3.7) respectively. In addition, in the cases when only sparse data samples are available, the data set can be expanded using the approaches in [Hol 92][Kar 00][Wan 99]. Thus, the $SNRF_e$ can be estimated using (3.8) and the overfitting is determined by comparison of $SNRF_e$ with the threshold in (3.14).

3.2.4. SNRF estimation for multi-dimensional function approximation

In a general multi-dimensional function approximation problem, the training data are usually randomly sampled from the input space $X \in \Re^D$. The method used to estimate SNRF in the one-dimensional case cannot be directly applied to such multi-dimensional problem. However, we could still assume that variation of s along each of the dimensions is slow compared to the average sampling distance. Thus the same principle of signal and noise level estimation using correlation may be utilized. Since s changes slowly in all directions, the continuous function can be locally approximated around e_p using weighted average of a set of $M+1$ points, which includes e_p and its M neighbors with the shortest distances. These points are expected to have correlated values, whereas the noise on these points is assumed to be WGN and has independent samples. As a result, the signal and noise levels at each sample e_p $(p=1, 2, \ldots N)$ can be estimated through the correlation with its M nearest neighbors and computed using a weighted combination of the products of e_p values with each of its neighbors, e_{pi} $(i=1, 2, \ldots M)$. Since the samples of e_i are assumed to be spatially correlated, the distances between samples can be used to calculate the weight values. In a D-dimensional space, the weights are obtained based on the scaled distance d_{pi} between e_p and e_{pi} to the power of D, and normalized, as given by (3.15),

$$w_{pi} = \frac{\frac{1}{d_{pi}^D}}{\sum_{i=1}^{M} \frac{1}{d_{pi}^D}}, \text{ where } d_{pi} = \left\| e_p - e_{pi} \right\| \quad \begin{pmatrix} p = 1,2,\ldots N \\ i = 1,2,\ldots N \end{pmatrix}. \tag{3.15}$$

Thus, the overall signal level of e can be calculated as,

$$E_s = \sum_{p=1}^{N} E_{sp} = \sum_{p=1}^{N}\sum_{i=1}^{M} w_{pi} \cdot e_p \cdot e_{pi} \quad . \tag{3.16}$$

As in (3.3), the autocorrelation of e_i estimates signal plus noise level:

$$E_{s+n} = \sum_{i=1}^{N} e_i^{\ 2} \ . \tag{3.17}$$

Finally, the $SNRF_e$ for M neighbors approach in a multi-dimensional input space is computed as

$$SNRF_e = \frac{E_s}{E_n} = \frac{\sum_{p=1}^{N}\sum_{i=1}^{M} w_{pi} \cdot e_p \cdot e_{pi}}{\sum_{i=1}^{N} e_i^2 - \sum_{p=1}^{N}\sum_{i=1}^{M} w_{pi} \cdot e_p \cdot e_{pi}} \ . \tag{3.18}$$

Notice that when applied to one-dimensional cases with $M=1$, (3.18) is identical to (3.8).

The same calculation is done for WGN with unit standard deviation to characterize the $SNRF_{WGN}$ in multi-dimensional space. When there is no signal, $SNRF_{WGN}$ is estimated using (3.18) with $e = n$. In the calculation of E_{sp} of WGN, $e_p \cdot e_{pi}$ is an independent random process with respect to p or i. Since

$$\sum_{i=1}^{M} w_{pi} = 1, \tag{3.19}$$

we can have,

$$\sigma\left[e_p \cdot e_{pi}\right] \geq \sigma\left[\sum_{i=1}^{M} w_{pi} \cdot e_p \cdot e_{pi}\right] \geq \sigma\left[\sum_{i=1}^{M} \frac{1}{M} \cdot e_p \cdot e_{pi}\right], \tag{3.20}$$

where $\sigma\left[\sum_{i=1}^{M} w_{pi} \cdot e_p \cdot e_{pi}\right] = \sigma_{p,WGN}$ is the standard deviation of the perceived signal energy at sample e_p in WGN. It has the minimum value when the w_{pi} have equal values (i.e., with uniform sampling distance), which sets the lower bound. Notice that the upper and lower bound in (3.20) are equal for *M=1*, independently of the input space dimensionality. For *M>1*, the standard deviation gets closer to the upper bound in problems with large dimensionality *D*, since the closest neighbor dominates the weight calculation.

In the estimation of $\sigma\left[\sum_{p=1}^{N}\sum_{i=1}^{M} w_{pi}\cdot e_p\cdot e_{pi}\right]$, it has to be considered that not all the $e_p\cdot e_{pi}$ items are independent of each other with respect to *p* and *i*. For instance, when points *p* and *p₁* are the closest neighbors to each other, $w_{p1}e_p e_{p1}$ is calculated twice in $\sum_{p=1}^{N}\sum_{i=1}^{M} w_{pi}\cdot e_p\cdot e_{pi}$. In the worst case, all the terms in $\sum_{p=1}^{N}\sum_{i=1}^{M} w_{pi}\cdot e_p\cdot e_{pi}$ may appear twice, therefore we have

$$\sigma\left[\sum_{p=1}^{N}\sum_{i=1}^{M} w_{pi}\cdot e_p\cdot e_{pi}\right]\le\sqrt{2}\cdot\sqrt{N}\cdot\sigma\left[\sum_{i=1}^{M} w_{pi}\cdot e_p\cdot e_{pi}\right]\le\sqrt{2}\cdot\sqrt{N}\cdot\sigma\left[e_p\cdot e_{pi}\right]=\sqrt{2N} \quad (3.21)$$

Then we have the estimate for the standard deviation of $SNRF_{WGN}$ as follows,

$$\sigma_{SNRF\_WGN}(N)=\sigma\left[\frac{\sum_{p=1}^{N}\sum_{i=1}^{M} w_{pi}\cdot e_p\cdot e_{pi}}{\sum_{i=1}^{N}e_i^2-\sum_{p=1}^{N}\sum_{i=1}^{M} w_{pi}\cdot e_p\cdot e_{pi}}\right]\approx\frac{\sigma\left[\sum_{p=1}^{N}\sum_{i=1}^{M} w_{pi}\cdot e_p\cdot e_{pi}\right]}{N}\le\frac{\sqrt{2}}{\sqrt{N}}. \quad (3.22)$$

Also, the average of $SNRF_{WGN}$ is estimated as,

$$\mu_{SNRF\_WGN}(N)=\mu\left[\frac{\sum_{p=1}^{N}\sum_{i=1}^{M} w_{pi}\cdot e_p\cdot e_{pi}}{\sum_{i=1}^{N}e_i^2-\sum_{p=1}^{N}\sum_{i=1}^{M} w_{pi}\cdot e_p\cdot e_{pi}}\right]\approx 0 \quad (3.23)$$

3.2.5 Multidimensional SNRF-based stopping criterion

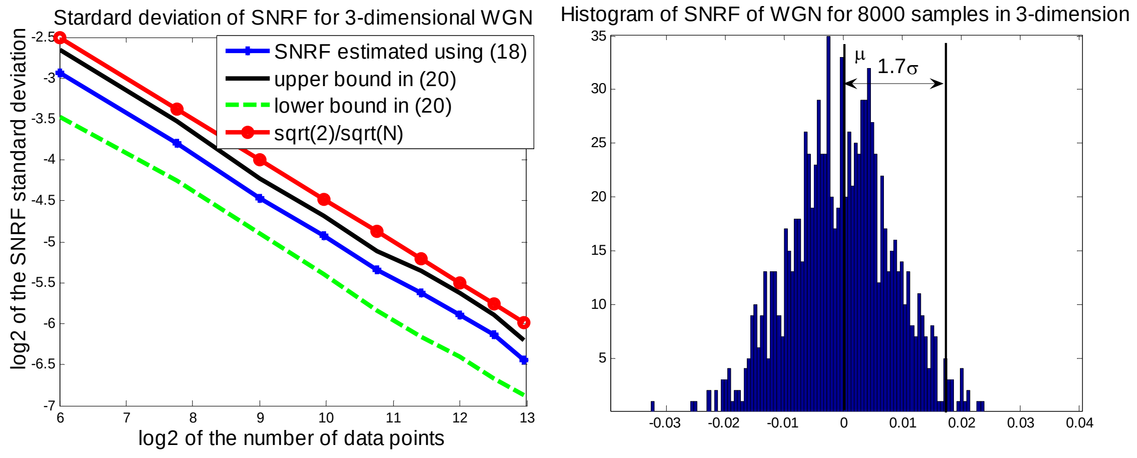
Notice that the estimation of $\sigma_{SNRF\_WGN}(N)$ and $\mu_{SNRF\_WGN}(N)$ using (3.22) and (3.23) is no longer a function of the number of samples in the neighborhood or problem dimensionality. Such simplification yields a universal detection threshold. $\sigma_{SNRF\_WGN}(N)$ for *M=3* in a three-dimensional case from a 1000-run Monte-Carlo simulation is shown in the logarithmic scale in Figure 3.3(a). The distances among WGN samples are randomly generated. The estimated $\sigma_{SNRF\_WGN}(N)$ in (3.22) is consistent with an upper bound of $\frac{\sqrt{2}}{\sqrt{N}}$, and the bounds developed in (3.20) are validated.

Figure 3.3(b) shows the histogram of *SNRF_{WGN}* for 8000 samples in the three-dimensional case with *M=3*. We note that the threshold of the significance level $p = 5\%$ can be approximated by the average value plus 1.7 times the standard deviations. With $N = 8000$ the threshold is calculated as $\mu + 1.7\sigma = 0 + 1.7 \times 0.0115 = 0.0195$. If not all the samples are independent, central limit theorem does not apply and the distribution of $SNRF_{WGN}$ is not Gaussian. In such case, the upper estimate of the standard deviation in (3.22) is used. The threshold can be experimentally established as the average value plus 1.2 times upper estimate of the standard deviation, to achieve the 5% significance level. Note that this result coincides with (3.14) for *M=1*.

$$\begin{aligned} &th_{SNRF\_WGN}(N) = \mu_{SNRF\_WGN}(N) + 1.2 \cdot \sigma_{SNRF\_WGN}(N) \\ &\sigma_{SNRF\_WGN}(N) \text{ as the approximated upper limit in (22)} \end{aligned} \tag{3.24}$$

While using *M>1* can improve estimation of the signal level by greater noise filtering when a large number of training samples is available, we did not observe a significant change in the detection threshold levels, comparing to *M=1*. Thus using *M=1* is preferred for computing efficiency even in multidimensional cases, when the number of training data is small.

In summary, a method for estimating the SNRF of the error signal has been demonstrated. By comparing $SNRF_e$ with $SNRF_{WGN}$, we are able to develop the optimized approximation algorithm (OAA) as discussed in the next section. The threshold for the OAA stopping criterion is determined from the estimate of $SNRF_{WGN}$, and can be applied to problems of an arbitrary number of samples and dimensions.



(a) (b)

Figure 3.3. SNRF estimation for WGN in three-dimensional case

(a) Standard deviation of SNRF of WGN in a three-dimensional case with $M=3$
(b) Histogram of SNRF of WGN for 8000 samples in a three-dimensional case

3.3 Optimized approximation algorithm

Using SNRF, we can estimate the signal level and the noise level for the error signal and then determine the amount of useful signal information left unlearned. When there is no information left, the learning process must be stopped, and the optimal approximation has been obtained without overfitting. Otherwise, the target parameter has to be increased to improve the learning and reduce the approximation error. The following procedure describes the basic steps of the OAA for the optimization of a given parameter of the NNs.

Optimized Approximation Algorithm:

(a). Assume that an unknown function *F*, with input space $X \subset \Re^D$ is described by *N* training samples as $F(x_i) = u_i,\ (i = 1,2,...N)$.

(b). The signal detection threshold is pre-calculated for the given number of samples *N* based on $th_{SNRF\_WGN}(N) = \frac{1.7}{\sqrt{N}}$.

(c). Select *B* as the initial value for the target parameter, for example, number of hidden neurons or number of BP training iterations.

(d). Use the MLP (or other learning models) to obtain the approximated function $\hat{F}(x_i) = a_i\ (i = 1, 2, ...N)$.

(e). Calculate the error signal $e_i = u_i - a_i$, (i=1, 2, ...N).

(f). Determine SNRF of the error signal e_i, $SNRF_e$. For a one-dimensional problem, use (3.8); for a multi-dimensional problem, use (3.18).

(g). Stop if the $SNRF_e$ is less than th_{SNRF_WGN}, or if *B* exceeds its maximum value. Otherwise, increment *B* and repeat (d)-(g).

(h). If $SNRF_e$ is equal to or less than th_{SNRF_WGN}, $\hat{F}$ is the optimized approximation.

3.4 Simulation and discussion

A multi-layer perceptron (MLP) is used as an example learning system to demonstrate the use of the proposed OAA. The MLP contains the input layer and the output layer with linear transfer functions and hidden layers with nonlinear transfer functions in the middle. OAA with SNRF-based stopping criterion will be tested in two aspects, optimization of the number of hidden neurons and optimization of the number of learning epochs, using synthetic datasets and benchmark datasets. First, the synthetic datasets are studied since we know the true target function so that real generalization error ε_{gen} can be calculated and the results provide a visual insight to the problem and its proposed solution. Subsequently, the benchmark data sets provide justification for the use of OAA in practical applications.

In all the simulation examples, when OAA is tested in optimization of the number of hidden neurons, the Least-Squared learning Method (LSM) proposed in [Erd 05] as initialization method will be used as training method in this paper. In LSM, the adaptation of weights in MLP is based on the least-squared calculation so that the learning performance is only affected by the number of hidden neurons representing the number of basis functions without concerning the number of iterations.

In addition, in all the simulation examples, when the number of learning epochs is optimized, a MLP with preset structure is trained using the back-propagation (BP) method, implemented using the MATLAB neural networks toolbox. The SNRF-based criterion in OAA will determine when to stop the learning to avoid overtraining (overfitting).

It is expected that when the SNRF-based criterion recognizes overfitting, either ε_{train} and ε_{valid} will start to diverge from each other, or ε_{valid} will reach a minimum. Such observation will help to prove the effectiveness of the OAA with the SNRF-based stopping criterion. The results, including the stopping points and corresponding ε_{train}, ε_{valid} and ε_{gen} (for synthetic data) from OAA will be compared with those from 4 other classes of stopping criteria described in [Pre 98] and [Aka 74]. Specific criteria used in the comparison are denoted as follows: AIC (Akaike's information criteria [Aka 74]), GL_1~GL_5 (generalization loss with thresholds 1~5 [Pre 98]), $PQ_{0.5}$~PQ_3 (generalization loss over training progress with thresholds 0.5~3 [Pre 98]) and UP_2~UP_8 (the number of successive increases in the generalization error [Pre 98]). To calculate the AIC for MLP, the number of free parameters is equal to the overall number of weights and the bias.

3.4.1. Simulation I: one-dimensional function approximation

First, the desired function to be approximated is $y = 0.4\sin x + 0.5$, same as the target function used in [Hol 92]. A 4-layered MLP is used as learning prototype with the number of hidden neurons to be optimized. The number of hidden neurons in these 2 hidden layers is set equal in the following simulation. The training and validation datasets, containing 200 samples each, are randomly sampled from the input space, and the outputs are subjected to WGN with a standard deviation of 0.2.

Simulation results show that $SNRF_e$ goes below the threshold when the number of hidden neurons on each layer is more than 3 for the 4-layered MLP, as can be seen from Figure 3.4(a). As shown in Figure 3.4(b), the approximated function obtained from the MLP with size 1-3-3-1 approximates the target function well. At the same time, it makes reasonable predictions on the unseen validation data. Although the ε_{valid} produced by MLP with 20 neurons is only 6% higher than that by 3 neurons, but it can be found from comparison in Figure 3.5(a) that MLP with 20 neurons seriously overestimates the complexity of the problem and the overfitting definitely shows up.

The results from different kinds of stopping criteria are compared in Table 3.1. Among all the stopping criteria, SNRF-based stopping criterion suggests the minimum structure that can efficiently handle the target problem and yield the minimum generalization error, which corresponds to possibly the best generalization ability.

In [Hol 92], the same target function is approximated using a MLP with size 1-13-1. It was demonstrated that the overfitting problem can be mitigated to some degree by using additive noise to expand the sparse dataset [Hol 92]. However, without optimizing the network structure, the approximated function still deviates from the desired function. Using the proposed OAA, the SNRF-based stopping criterion shows that the optimal number of hidden neurons for this 3-layer MLP is 5.

With such 1-5-1 MLP, the number of learning epochs of the BP algorithm can be optimized using SNRF-based stopping criterion in OAA. It suggests stopping the training after 10 epochs. The approximated function after 10 epochs is compared with that after 200 epochs in Figure 3.5(b), which shows that large number of learning epochs induces overfitting and the SNRF-based stopping criterion is able to stop the learning process at the optimum point.

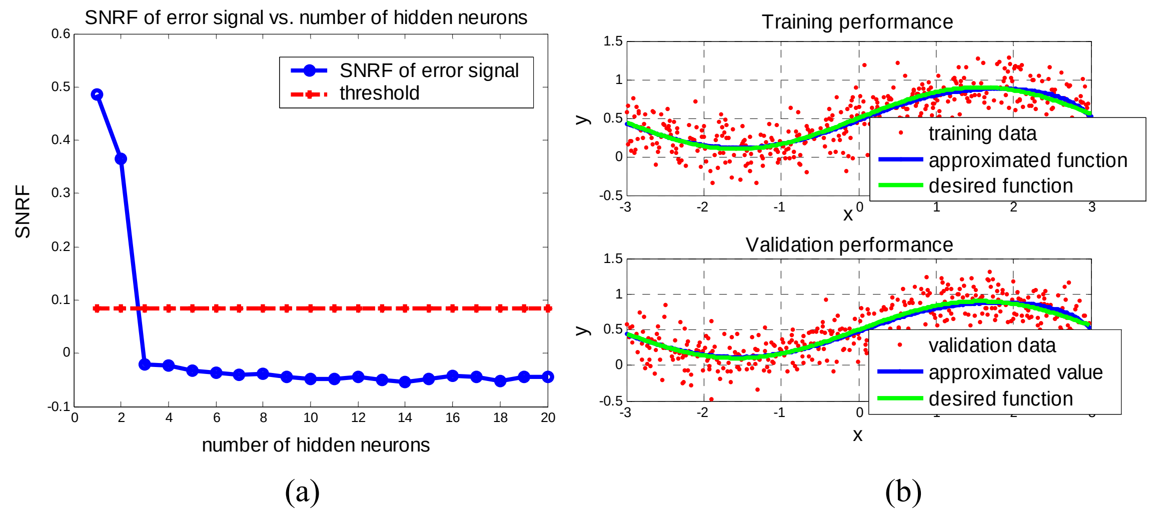


Figure 3.4. Simulation I: optimization of number of hidden neurons

(a). SNRF of the error signal and threshold
(b). The approximated function and desired function

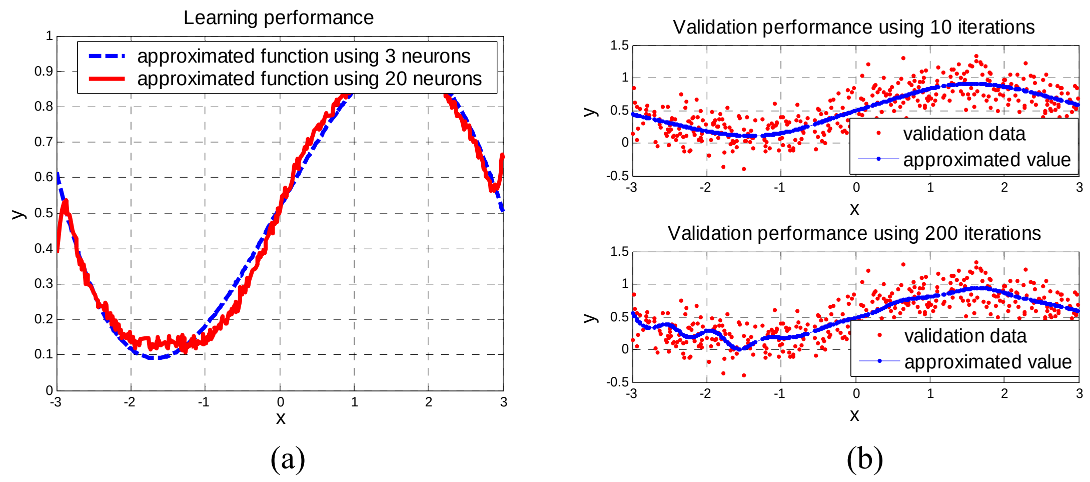


Figure 3.5. Comparison of approximated function

(a). Comparison of approximated function using 3 and 20 neurons
(b). Comparison of approximated function using 100 and 200 learning epochs

Table 3.1

Simulation I: Results Comparison for Optimizing Number of Neurons

Stopping Criteria	Optimum number of hidden neurons	Training error ε_{train}	Validation error ε_{valid}	Generalization error ε_{gen}
SNRF	3	0.11278	0.10559	0.0022411
AIC	3	0.11278	0.10559	0.0022411
GL_1	10	0.11084	0.10623	0.0024636
GL_2	15	0.10895	0.10779	0.0038922
GL_3	17	0.10802	0.10994	0.0057212
GL_5	18	0.10701	0.11148	0.0091781
$PQ_{0.5}$	10	0.11084	0.10623	0.0024636
$PQ_{0.75}$	10	0.11084	0.10623	0.0024636
PQ_1	11	0.11075	0.10549	0.0025474
PQ_2	14	0.1095	0.10677	0.0042547
PQ_3	14	0.1095	0.10677	0.0042547
UP_2	7	0.11123	0.10528	0.0027053
UP_3	7	0.11123	0.10528	0.0027053
UP_4	9	0.11112	0.10566	0.002306
UP_6	13	0.10955	0.10594	0.0032101
UP_8	17	0.10802	0.10994	0.0057212

The results of optimizing the number of learning epochs from different kinds of stopping criteria are compared in Table 3.2. SNRF-based stopping criterion suggests stopping the training with minimum number of learning epochs in this case and shows minimum generalization error. Notice that since the network structure does not change during the process, the AIC stopping criterion cannot be applied and will be denoted as "N/A" in the result tables. Some of the stopping criteria, including GL_2, GL_3 and GL_5, have not been met even with the maximum number of learning epochs and will be denoted as "Incomplete" in the result tables.

3.4.2. Simulation II: two-dimensional function approximation

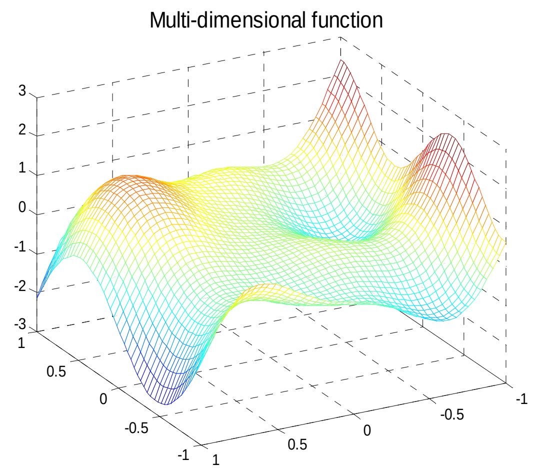
A function $y = x_2{}^2 + \sin(3x_2) + 2x_1^2 \sin(4x_1) + x_1 \sin(4x_2)$ is used as the target function to illustrate a multidimensional case, as shown in Figure 3.6(a). Data points are randomly sampled adding WGN with a standard deviation of 0.1 to produce training and validation data sets, each containing 100 samples.

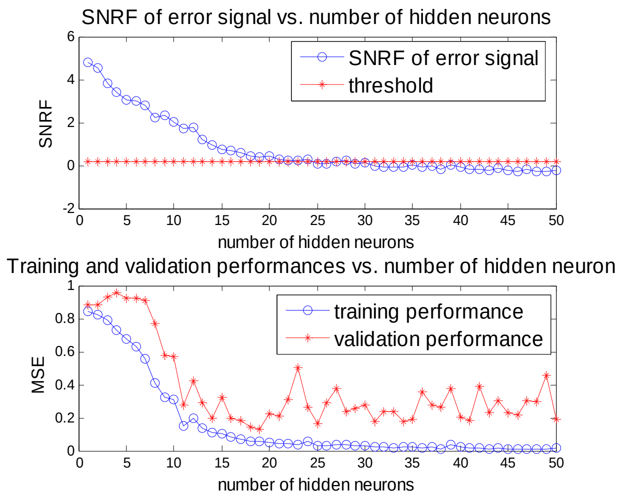
The OAA is applied to optimize the number of hidden neurons of a 4-layered MLP and it is discovered that $SNRF_e$ falls below the threshold when the number of hidden neurons exceeds 25 as shown in Figure 3.6(b). It may be seen that the validation error has many local minima located in the range from 25 to 35 neurons. In this case, it would be difficult to exactly determine where overfitting begins by using ε_{valid}.

Table 3.2

Simulation I: Results Comparison for Optimizing Number of Learning Epochs

Stopping Criteria	Optimum number of learning epochs	Training error ε_{train}	Validation error ε_{valid}	Generalization error ε_{gen}
SNRF	10	0.1086	0.0987	0.00053
AIC	N/A	N/A	N/A	N/A
GL_1	180	0.1064	0.1003	0.0073
GL_2	Incomplete	Incomplete	Incomplete	Incomplete
GL_3	Incomplete	Incomplete	Incomplete	Incomplete
GL_5	Incomplete	Incomplete	Incomplete	Incomplete
$PQ_{0.5}$	30	0.1076	0.0989	0.0026
$PQ_{0.75}$	30	0.1076	0.0989	0.0026
PQ_1	30	0.1076	0.0989	0.0026
PQ_2	130	0.1076	0.0989	0.0026
PQ_3	130	0.1076	0.0989	0.0026
UP_2	50	0.1066	0.0993	0.0065
UP_3	60	0.1074	0.0991	0.0037
UP_4	90	0.1078	0.0988	0.0012
UP_6	150	0.1075	0.0992	0.0029
UP_8	160	0.1073	0.0990	0.0034





(a) (b)

Figure 3.6. Simualtion II: optimization of number of hidden neurons

(a). Multi-dimensional function to be approximated
(b). SNRF of the error signal vs. threshold and training and validation performances

Using such 2-25-25-1 MLP as a function approximator, the approximated function in the given input space replicates the desired function well, as shown in Figure 3.7(a). However, using 35 hidden neurons, the approximated function has significant deviations from the target function at the unseen data, which is illustrated in Figure 3.7(b). The function surface obviously indicates that overfitting already occurs. The optimal network size with 25 neuron optimum is correctly predicted by the OAA. The optimization results based on different stopping criteria are compared in Table 3.3. In this case, other methods stop too early resulting in larger generalization errors.

Subsequently, OAA was used in a 3-layered MLP with size 2-25-1 to find proper number of learning epochs and the results are compared with others methods in Table

3.4. Again, we can see that the proposed SNRF criterion yields an optimum number of the training epochs with the smallest validation and generalization errors.

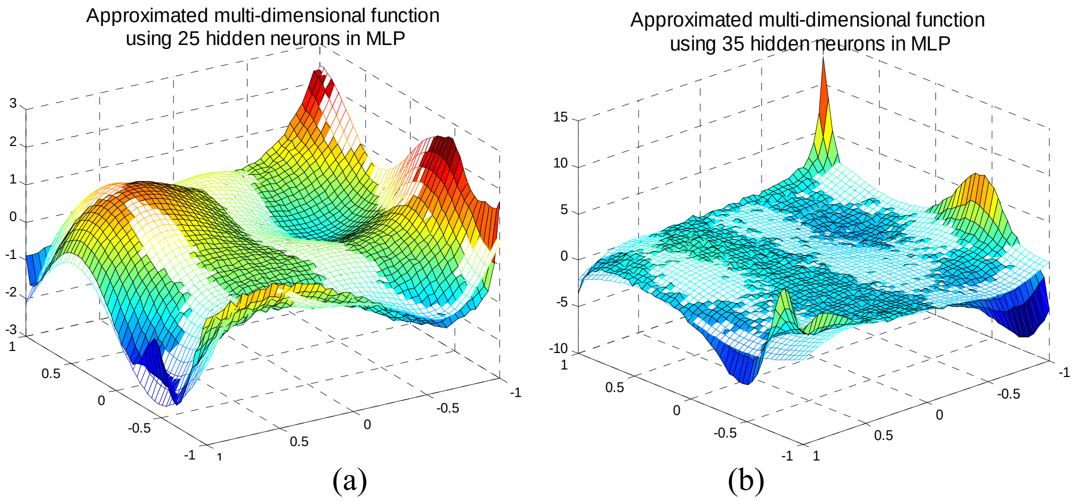


Figure 3.7. Approximated function using (a) 2-25-25-1 and (b) 2-35-35-1MLPs

3.4.3. Simulation III: Puma robot arm dynamics dataset

The OAA has also been applied to a benchmark dataset to illustrate another multidimensional case generated from the dynamics of a Unimation Puma 560 robot arm [Del]. The dataset is subject to an unknown level of noise. The task is to predict the angular acceleration of the robot arm's links from 8 inputs include angular positions of 3 joints, angular velocities of 3 joints and torques of 2 joints of the robot arm. Various numbers of neurons (from 1 to 100 with a step size of 3) are used in the MLP to find the optimum number of hidden neurons using OAA.

Table 3.3

Simulation II: Results Comparison for Optimizing Number of Hidden Neurons

Stopping Criteria	Optimum number of hidden neurons	Training error ε_{train}	Validation error ε_{valid}	Generalization error ε_{gen}
SNRF	25	0.023222	0.22471	0.065502
AIC	1	0.85216	0.87783	0.89908
GL_1	6	0.64278	0.82437	0.64931
GL_2	6	0.64278	0.82437	0.64931
GL_3	6	0.64278	0.82437	0.64931
GL_5	16	0.072757	0.41385	0.15182
$PQ_{0.5}$	6	0.64278	0.82437	0.64931
$PQ_{0.75}$	16	0.072757	0.41385	0.15182
PQ_1	16	0.072757	0.41385	0.15182
PQ_2	16	0.072757	0.41385	0.15182
PQ_3	16	0.072757	0.41385	0.15182
UP_2	6	0.64278	0.82437	0.64931
UP_3	16	0.072757	0.41385	0.15182
UP_4	16	0.072757	0.41385	0.15182
UP_6	16	0.072757	0.41385	0.15182
UP_8	17	0.055236	0.19659	0.11702

Table 3.4

Simulation II: Results Comparison for Optimizing Number of Learning Epochs

Stopping Criteria	Optimum number of learning epochs	Training error ε_{train}	Validation error ε_{valid}	Generalization error ε_{gen}
SNRF	21	0.017228	0.072691	0.02101
AIC	N/A	N/A	N/A	N/A
GL_1	41	0.015332	0.07962	0.024886
GL_2	41	0.015332	0.07962	0.024886
GL_3	41	0.015332	0.07962	0.024886
GL_5	41	0.015332	0.07962	0.024886
$PQ_{0.5}$	41	0.015332	0.07962	0.024886
$PQ_{0.75}$	41	0.015332	0.07962	0.024886
PQ_1	41	0.015332	0.07962	0.024886
PQ_2	61	0.014128	0.13411	0.031172
PQ_3	101	0.012752	0.16083	0.039461
UP_2	41	0.015332	0.07962	0.024886
UP_3	61	0.014128	0.13411	0.031172
UP_4	101	0.012752	0.16083	0.039461
UP_6	131	0.0097924	0.42093	0.095002
UP_8	171	0.0041367	0.78034	0.17466

The $SNRF_e$ is compared with threshold, as shown in Figure 3.8(a), and indicates that overfitting starts to occur when the number of neurons is 46. Note that ε_{valid} has many local minima, as seen in Figure 3.8(b), and using a local minimum of ε_{valid} as a stopping criterion would be ambiguous. The optimization results based on different stopping criterion are compared in Table 3.5. With a MLP of size 8-46-1, OAA can be used to find proper number of learning epochs and the results are compared with others in Table 3.6.

In summary, for all tested datasets, the SNRF quantitatively identified overfitting and helped to find the proper structure or the number of training epochs for effective neural network learning for a given problem.

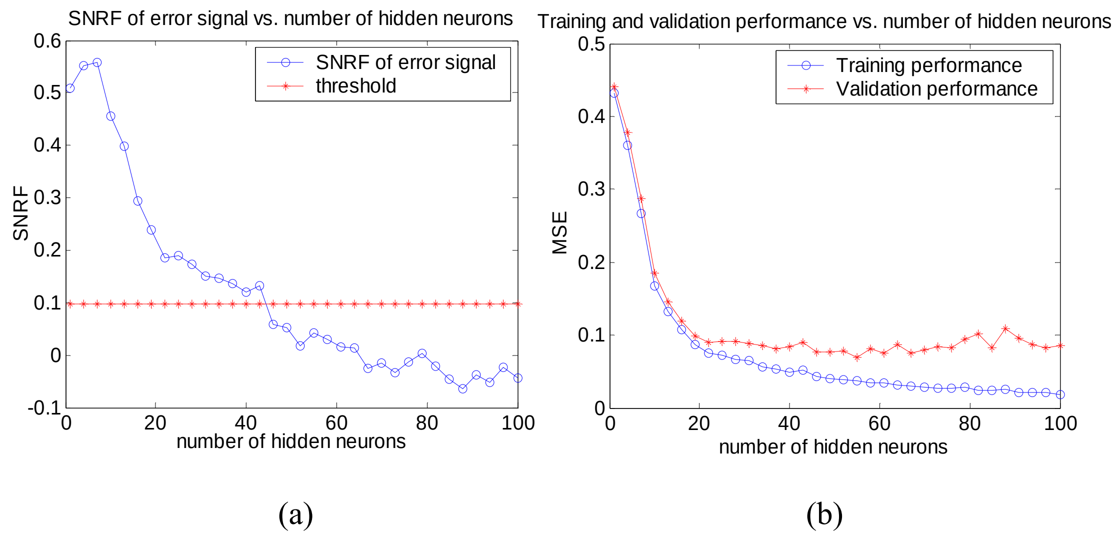


Figure 3.8. Simulation III: Optimizing number of hidden neurons.

(a) SNRF of the error signal and threshold
(b) Training and validation performance

Table 3.5

Simulation III: Results Comparison for Optimizing Number of Hidden Neurons

Stopping Criteria	Optimum number of hidden neurons	Training error ε_{train}	Validation error ε_{valid}
SNRF	46	0.043562	0.077474
AIC	1	0.43152	0.44099
GL_1	28	0.066768	0.091784
GL_2	40	0.049517	0.083847
GL_3	40	0.049517	0.083847
GL_5	43	0.052393	0.089354
$PQ_{0.5}$	61	0.034858	0.07574
$PQ_{0.75}$	73	0.027735	0.084617
PQ_1	73	0.027735	0.084617
PQ_2	79	0.027958	0.094146
PQ_3	85	0.024118	0.082011
UP_2	28	0.066768	0.091784
UP_3	28	0.066768	0.091784
UP_4	28	0.066768	0.091784
UP_6	43	0.052393	0.089354
UP_8	52	0.038588	0.078167

Table 3.6

Simulation III: Results Comparison for Optimizing Number of Learning Epochs

Stopping Criteria	Optimum number of learning epochs	Training error ε_{train}	Validation error ε_{valid}
SNRF	4	0.039814	0.077506
AIC	N/A	N/A	N/A
GL1	7	0.032243	0.10845
GL2	7	0.032243	0.10845
GL3	7	0.032243	0.10845
GL5	7	0.032243	0.10845
PQ0.5	11	0.0027257	0.14612
PQ0.75	26	1.0016e-011	0.16009
PQ1	101	1.4087e-018	0.18978
PQ2	101	1.4087e-018	0.18978
PQ3	101	1.4087e-018	0.18978
UP2	7	0.032243	0.10845
UP3	7	0.032243	0.10845
UP4	10	0.0038718	0.11284
UP6	26	1.0016e-011	0.16009
UP8	51	1.3981e-016	0.21248

In most simulation cases, OAA suggests the minimum structure or minimum length of the training period unlike other stopping criteria. In the few cases it does not, OAA still delivers better generalization in the sense of the smallest ε_{valid}. In many stopping criteria variation of ε_{valid} is one of the measures used to determine possibility of overfitting rather than providing quantified evaluation of the goodness-of-fit accomplished by SNRF. To meet the quantified stopping criterion, it may take slightly more hidden neurons or learning epochs for the SNRF to fall below the threshold than in some other criteria. However, in all the cases, the network optimized with OAA outperforms all the other stopping criteria by providing optimized generalization ability.

3.5. Conclusions

In this chapter, an optimized approximation algorithm is proposed to solve the problem of overfitting in function approximation applications using neural networks. The OAA utilizes a quantitative stopping criterion based on the signal-to-noise-ratio figure (SNRF). This algorithm can automatically detect overfitting based on the training errors only. The algorithm has been validated for optimization of the number of hidden neurons for MLP and the number of iterations for the BP training. It can be applied to parametric optimization of any learning model or model selection for other function approximation problems. Therefore, in a given supervised learning problem, the structure of the multi-layered network can be determined by the OAA so that useful information is effectively extracted while minimum computational resources are consumed.

CHAPTER 4: A FRAMEWORK OF BUILDING EMBODIED INTELLIGENCE

4.1 Introduction

Intelligence cannot develop without an embodiment or active interaction with the environment. The embodiment is the extension and interface of the intelligence for its interaction with the environment. Through embodiment, intelligent agents carry out motor actions and affect the environment. The response of the environment is registered through sensors implanted in the embodiment. At the same time the embodiment is a part of the environment that can be perceived, modeled and learned by intelligence. Properties of the motors and sensors, their status and limitations can be studied, modeled and understood by intelligent agents, so that the agents model and understand limitations of their embodiment and their motor abilities, or effectively their abilities to effect the environment.

The intelligence core interacts with its environment through its embodiment, as shown in Figure 4.1. This interaction can be viewed as closed-loop sensory-motor coordination. The embodiment does not have to be constant nor physically attached to the EI body. The boundaries between embodiment and the environment change during the interaction which modifies the intelligent agents' self-determination. Because of the dynamically changing boundaries, the definition of embodiment has to reflect this fact and contain elements of indetermination.

In this chapter, a framework of building an embodied intelligence (EI) is proposed. The definition of embodiment is presented along with its characteristics. And a general and uniform definition of EI is given for the framework. Based on the definition, we propose essential elements for building an intelligent machine and spell out its general design principles. As one of the essential elements of intelligence, hierarchical memories are further discussed in later chapters.

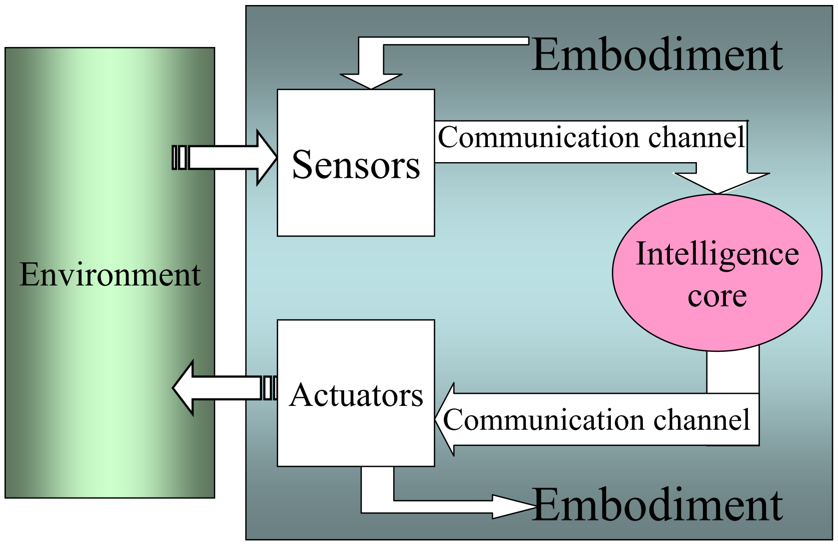


Figure 4.1. Intelligence core with its embodiment and environment.

4.1.1 Definition of embodiment of embodied intelligence

Definition:

Embodiment of embodied intelligence is a mechanism under the control of the intelligence core that contains sensors and actuators connected to the core through communication channels.

A first consequence of this definition is that the mechanism under control may change. For instance, when the embodiment changes, the way that the embodiment works and the intelligent agent interacts with the environment will be affected. The efficiency of the EI sensory perception may be altered by external effects, and its motor ability may depend on skills and strength of its body, etc.

Secondly, embodiment does not have to be permanently attached to the embodied intelligence in order to play its role of sensory- motor interaction with the rest of the environment. For instance, if we operate a machine (drive a car, use keyboard, play tennis), our embodiment dynamics can be learned and associated with our action to the extent that reduces the distinction between the dynamics of our own body and the dynamics of our body operating in tandem with the machine. Likewise, artificially enhanced senses can be perceived and characterized as our own senses (e.g. glasses that improve our vision, or a hearing aid that improve our hearing). Another example of sensory extension could be an electronic implant stimulating the brain of a blind person to provide visual information or an ultrasound device warning a blind person of incoming obstacles, etc.

Extended embodiment does not have to be of a physical (mechanical) nature. It could be in the form of remote control of tools in a distant surgery procedure or monitoring Martian landscape through mobile, remotely controlled cameras. It could also be our distant presence at the soccer game through received TV images or our voice message delivered through a speakerphone to a group of people at a teleconference.

Finally, extended embodiment of intelligence comes in the form of organizations and their internal working mechanisms and procedures. A general directing troop on a battle field feels a similar directive power of moving armies as a crane operator that feels the mechanical power of the machine that he operates. The president also feels the political power of his address to the nation and the large impact it makes on people's lives.

This extended embodiment enhances EI's ability to interact with the environment and thus its ability to grow in complexity, skills and effectiveness. If the President learns how to address the nation, his ability to affect the environment grows differently than that of a woman in Darfur trying to save her child from violence and hunger.

Our knowledge of embodiment properties and its limitations is a key to its proper use in interaction with the world. We rely on this knowledge to plan our actions and predict the responses from the environment. A change in the way that our embodiment implements desired actions or perceives response from the environment introduces uncertainty in our behavior and may lead to confusion and less than optimum decision making. If a car's controls were suddenly reversed during operation a user would require some adaptation time to adjust to the new situation and probably would not be able to effectively control the machine that may result in a crash. Therefore, what we learn about our environment and our ability to change this environment is affected not only by our intelligence (ability to learn, understand, represent, analyze and plan) but by correct perception of our embodiment as well. This perception leads to the self awareness and conscious behavior.

4.1.2 Definition of embodied intelligence

In order to build working models of intelligent machines, an arbitrary and utilitarian definition of intelligence is adopted in this work [Sta 06]. It is our aim to base the design concepts of embodied intelligence on a minimum set of requirements and mechanisms from which all traits of intelligence can be derived. We will demonstrate that the definition is general enough to characterize agents of various levels of intelligence including human. To differentiate it from enigmatic meaning of intelligence, we will limit it to embodied intelligence suggested by Brooks [Bro 91] and described in more detail by Pfeifer [Pfe 99].

Definition:

Embodied intelligence (EI) is defined as a mechanism that learns how to survive in a hostile environment.

A mechanism in this definition applies to all forms of embodied intelligence, including biological, mechanical or virtual agents with fixed or variable embodiment, and fixed or variable sensors and actuators. Implied in this definition is that EI interacts with an environment (real or virtual) and that the results of actions are perceived by its sensors. Also implied is that the environment is hostile to EI so that EI has to learn how to survive. This hostility of environment symbolizes all forms of pains that EI may suffer – whether it is an act of open hostility or simply scarcity of resources needed for the survival of the EI. The important fact is that the hostility is persistent as it stimulates the mechanism to act and learn. For example, low battery power is a persistent threat for an agent requiring it. Gradually the energy level goes down, and unless the EI replenish its energy, a perceived discomfort from its energy level sensor will increase.

Hostile stimulation from the environment towards EI is necessary for it to acquire necessary knowledge, develop environment related skills, build models of the environment and its embodiment, explore and learn successful actions, create its value system and goals, and grow in sophistication.

We will show in more details how the perpetual hostility will be the foundation and motivations for learning, goal creation, planning, thinking, and problem solving. In more advanced forms of EI it will also lead to intuition, consciousness, and emotions. Thus all forms and levels of intelligence can be considered under the proposed definition of EI.

Notice that this definition of EI clearly differentiates knowledge from intelligence. Knowledge is the acquired set of skills and information about the environment, while intelligence requires the ability to acquire knowledge.

4.2 Designing the embodied intelligence

Learning whether certain actions are desirable makes the learning agent more capable of surviving in the hostile environment. There are several means of adapting to the environment: evolutionary - by using the natural selection of those agents that are most fit; cognitive - by using learning memory, pattern recognition, and associations; and group behavior – by using the individual member skills, specialization, and communication for the group survival. Here we address only the second, and the most critical from the machine intelligence point of view, form of adaptation.

4.2.1 Basic requirements for EI

The spatio-temporal patterns that we experience during a lifetime underlie our knowledge, and produce our internal models of the environment. The perceptual objects that we can recognize, the relations among the objects, and the skills that we have are all stored in our memory. The perceived patterns have features at various levels of abstraction, so that they will be remembered accordingly.

Another critical aspect of human brain development is self-organization. By self-organizing their interconnections, our brains allow us to quickly create representations of these patterns, interact with the environment, and build expectations regarding future events. A six year old child has many redundant and plastic connections ready to learn almost anything. After years of learning, the connection density among neurons is reduced, as only the most useful information is retained, and related memories and skills are refined. At the same time, the learning ability is gradually reduced. It is not to say that an adult cannot learn more efficiently than a child. The knowledge that he accumulated makes him an efficient learner, however the memory resource that he can use to learn new facts are gradually depleted since brain's ability to create new neurons is very limited.

Although most of the existing neural network models assume full or almost full connectivity among neurons, human cerebral cortex is a sparsely connected network of neurons. For example, it has been estimated that synapses of neurons projecting through the mossy pathway (of the rat) from the dentate gyrus to subregion CA3 of the hippocampus reach 0.0078% of CA3 pyramidal cells [Rol 89]. Sparse connections can, at the same time, improve the storage capacity per synapse and reduce the energy consumption of a network working with the stored patterns.

For the purpose of building intelligent machines, it seems instructive to develop a neural network structure that allows the machine to perceive in a manner similar to how humans do. Since human cortex is rather uniform in its organization and its able to learn almost anything in any of its location, the EI memory should be built based on a simple, uniform, hierarchical, and sparsely-connected structure with capability to self-organize. EI learns predominantly in an unsupervised manner by responding to stimuli from the environment. However, learning, as a pattern processing process, is not involuntary. Learning is deliberate, perpetual, and should be closely related to the machine's situation in the environment, and related to how the machine can survive in it.

Having the purpose of surviving and certain more specific goals, the machine can efficiently organize its resources to process the useful incoming information and learn the important skills. The creation of goals should result from the machine's interaction with its environment. Therefore, we require that an intelligent machine must have a built-in mechanism to create goals for its behavior and such mechanism will be called the goal creation system (GCS). As we like to develop the machine's memory based on simple, uniform hierarchical and self-organizing structure, we also desire to build a GCS based on hierarchical structure. The memory structure grows and evolves as the goal hierarchy develops. Better perceptions and skills facilitate understanding and creation of more advanced goals. Meanwhile, the creation of more advance goals stimulates the growth of the hierarchy representing sensory inputs, perception and understanding of the environment and the hierarchy representing actions and skills.

Therefore, it is proposed that in order to build intelligent machines, the following elements are essential:

1. Hierarchical self-organizing memory (HSOM) with sparse connectivity to perceive and act according to the machine's objectives.

2. Goal creation system (GCS) to develop sensory-motor coordination, goal-oriented learning and goal-oriented action, and to act as stimuli for interaction with the environment.

In the proposed model of EI, a HSOM will use three basic pathways – a sensory pathway responsible for perception, a motor pathway responsible for actions, and a goal creation pathway responsible for goal creation, planning, evaluation of actions in relation to its goals, learning of useful associations, and stimulation of the machine to perform useful actions. These three pathways interact and associate on various levels of the memory hierarchy.

4.2.2 Hierarchical self-organizing memory

HSOM is made of multi-layered processing units (neurons) and their connections. Neurons on different levels handle the recognition tasks with different levels of abstraction. Lower-level neurons are either activated directly by the sensory inputs or indirectly by certain detailed features. Subsequent level neurons combine the extracted features and represent elements of more complex entities by creating necessary associations between the lower-level features. The information is gathered, associated and abstracted (in an invariant form) as it flows upwards in the hierarchy. Finally, top-level neurons represent perceived entities, ideas, and relations in the observed environment.

In human brains, the neurons in the sensory pathway at the input stage usually make divergent connections with later processing stages [Kan 00]. Then the information reaches and activates various parts of the neocortex. In human processing of visual information, 1 million, densely firing neurons in the optic nerves provide information to 200 million neurons in the primary visual cortex V1 [And 05] that, in turn, activate significant parts of the neocortex which contains as many as 10^{11} neurons [Kan 00]. The work presented in [Ste 01] suggested that not only the number of V1 neurons increases as the 3/2 power of the number of LGN neurons ($N_{V1} = N_{LGN}^{3/2}$), but also the entire volume of neocortex and thalamus follow a similar expansion rate. The work raised a possibility that similar scaling rates exist for other cortical areas.

In addition, increasing the number of neurons on the top of the hierarchy increases the number of abstract symbols that can be represented. This increases the memory capacity of the network which is a desired feature of the proposed memory organization. Therefore, the proposed HSOM may accommodate a larger number of neurons at higher hierarchical levels to represent a large variety of abstract symbols. The feature recognized in the lower-level neurons may be related to multiple objects represented on the higher-level, so the lower-level neurons' activity may be potentially spread to several neurons on the higher level. It is structured to potentially reach and activate groups of neurons on the top layers of its hierarchy. However, the number of higher level neurons can be reduced if needed for smaller memories to lower the simulation cost.

4.2.2.1 Sensory pathway

The primary objective of neurons in the sensory pathway is to register the input information received from the environment and to build intentional representations to either be acted upon right away or stored in the long-term memory for later use. The intentional representation is an internal representation of objects, symbols, abstractions, relations, actions, etc. (jointly called perceptions) related to EI's interaction with the environment. Typically, in an HSOM, many neurons are activated on the lower levels to represent the detailed features of the sensorium while few neurons need be activated at higher levels. The reduction in neuronal activities at higher levels builds the "sparse codes" [Bar 72] [Wil 69] [Ama 93] [Fie 94] that represent information. The active neurons at all layers of the hierarchy form the activation pathways for various stimuli. Simple features recognized by neurons on the lower levels may be combined to generate representations of many objects and their relations at higher levels.

The idea of "sparse coding" emerged in an earlier work as one of the propositions of human visual perception [Bar 72]. The principle has been elaborated and advanced by several other authors [Wil 69] [Ama 93] [Fie 94] [Day 01] [Fol 02] [Ols 04]. In recent years, various experimental studies have supported the notion that visual information in human primary visual cortex is represented by a relatively small number of active neurons out of a large population of neurons [Vin 00] [Vin 02]. Sparse coding in the visual system also exists as a consequence of metabolic demands [Len 03] [Bad 96] [Lau 03], and as a result of adaptation to special statistical properties of the natural visual environment [Vin 00] [Ols 96] [Sim 01]. Sparse coding is expected in auditory cortex as well [Lew 02] [Lew 02], and is observed in the spiking rates of neurons in primary auditory cortex [DeW 03]. The advantages of using a learning network with sparse connections and sparse data representations were recently discussed conceptually in [And 05]. With the information flowing up the hierarchy, the responses of the neurons on the higher levels can become very selective and only a few neurons are active for certain stimuli.

In the design of machine intelligence, power dissipation is one of the most critical design factors, especially in large, parallel computing systems. Thus, a learning and processing model that involves only a small subset of active neurons will save the power consumed by the learning memory. This containment of neuronal activities in representation building in sensory pathways is even more critical in the computational model we assume in this work in which it is possible that higher layers have many more neurons than the lower layers. Such assumption depends on memory capacity and is biologically plausible.

If the lower layers correspond to the sensory input stimuli, their size is determined by the number of sensory inputs. Higher layers use these sensory inputs to build abstract representations, and finally contribute to representation memories at the top layers in the hierarchy. In large memories these top layers contain many neurons. For instance it is estimated that in the human brain the number of cortical neurons is on the order of 10^{11}, while the number of neurons stimulated by the sensory inputs is on the order of 10^{6}. Thus a majority of neurons reside farther away from the sensory inputs, leading to the topologically expanding hierarchical structures. Therefore, computational models to build representations used in this work assume such expanding sensory and motor structures.

Memory structures are developed through the modification of the interconnection weights between active neurons. Weight adjustment has an important impact on the machines ability to learn. Our system restricts the magnitude of weight adjustment based on the amount of times a neuron's weights have been previously modified. As a result of the expanding layers and reduced probability of firing, weights on the lower levels are adjusted more frequently then upper level weights. As sensory information is presented to the network, lower level connections quickly become rigid and can no longer be modified (learn). In contrast, upper level neurons are infrequently modified and therefore retain their ability to be modified even after extensive exposure to the input level stimuli. This results in a system which can be trained to form representations on each increasing level with an increased ability to represent new objects.

The structure of the HSOM sensory pathway is proposed as shown in Figure 4.2. The neurons in the sensory pathway are organized in a layered structure. The neurons on each layer have sparse connections with neurons of the next higher level. In the figure, two sensory inputs activate two pathways, shown as the shaded areas. In each activation pathway, the neuronal activity decreases for higher levels. Features on a particular level may be reused in two or more different activation pathways during a recognition process. Such an organization of sensory pathways in which various streams either specialize in building localized features or represent completely different sensory inputs are well supported in neurobiological studies [Ung 82] [Van 83]. Such paths interact with each other on various levels of the hierarchy [Sam 97].

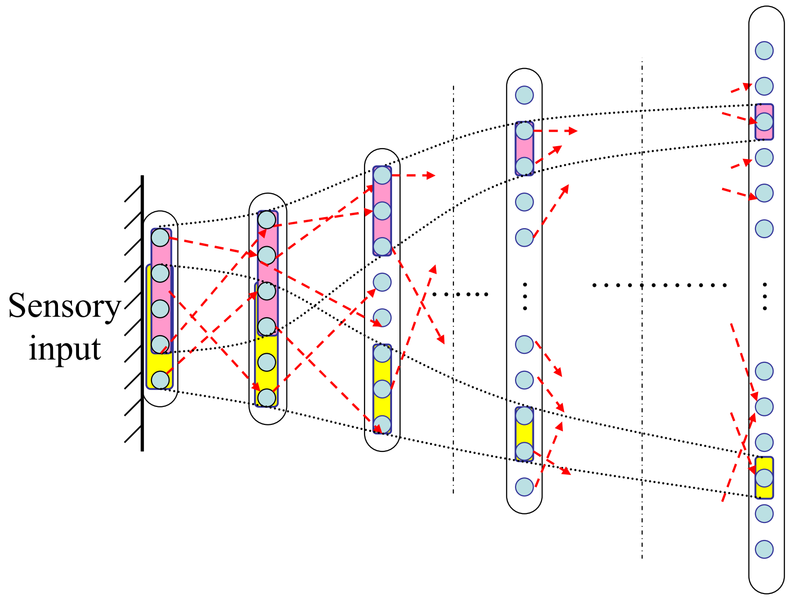


Figure 4.2. Structure of sensory pathways in the HSOM and exemplar activation pathways.

EI uses two mechanisms to store the information in its memories and to build the representation invariance. Firstly, only new and useful information is stored in the form of intentional representations. To accomplish this, the machine continuously predicts what information will be coming in. This prediction manifests its understanding of the perceived signals and an assumption of the continuity and self-similarity (sameness) of the observed scene over a short observation time. In short time intervals, the sensory input changes, are usually the result of minor changes in the point of view, motion, lighting conditions, or gradual modification of the object shape, color or form. When EI interacts with the environment, it is situated in a specific location, observing specific objects, and performing a specific task. Thus the assumed continuity of the observed sensory input is used for invariance building. Although we look at an object from different angles, we know that this is the same object and thus these various inputs must trigger the same representation on the higher level of the sensory pathway that forms the representation memory.

Secondly, the intentional representations have to be related to the machine's objectives. When the EI realizes that a specific action resulted in a desirable effect, it stores the representation of the perceived entity and learns associations between the activated sensory neurons and the motor neurons. If the effect is not desirable, it learns not to perform such an action by reducing the strength of the connections. Finally, when no goal is affected, no learning takes place, the machine does not create intentional representation nor does it remember the action it took. Such organization of the learning process protects the machine's memory from overloading by unimportant information.

4.2.2.2 Motor pathway

The primary objective of neurons in the motor pathway is to represent and control execution of actions. The motor pathway represents skills learned by the EI.

Neurons in the motor pathway are organized in a hierarchical way to be able to store a large number of skills and actions. At the bottom of the hierarchy are row motor outputs. Memories are developed through registering these activities by modifications of the interconnection weights between neurons on the lower levels and activated neurons on the higher level.

Hierarchies of action representations are built bottom-up, from the simplest actions that require little sensory-motor coordination or sequential memories, to the most complex ones, that may last for a long period of time and require lots of memories. An example of a simple action may be a reactive response to a painful shock, while driving home may be an example of a more complex action. Higher level actions can only be obtained after lower level skills are learned.

To have large learning capacity for various skills, the number of motor neurons on the higher levels is much greater than that of the lower levels in the motor pathway hierarchy. However, the number of activated neurons that represent skills and actions on the higher levels is less than that on the lower levels. Lower levels may no longer be capable of storing any new information since they were involved in learning many action patterns. This results in lower plasticity of the interconnections on the lower level than on the higher levels.

EI activates motor neurons in response to the request from the value system. If an action was taken that resulted in a positive value (or a reward), an association between the sensory and the motor neuron's activity is learned. This makes it more likely that a similar action will be executed again when the EI is exposed to a similar environmental situation. The same complex operation may be executed using various simpler operations, which leads to a similar concept of invariance building in the motor pathway as that in the sensory pathway. Continuity of a higher level action is used for invariance building between an action represented on the higher level and its lower level implementations.

A prediction mechanism is also used in the motor pathway. At every step of a motor action, a prediction is made regarding expected inputs from the sensory pathway and the value system. If the prediction is correct there is no need to learn any new associations.

Figure 4.3 shows a schematic representation of interactions between sensory and motor units on different levels of HSOM hierarchy.

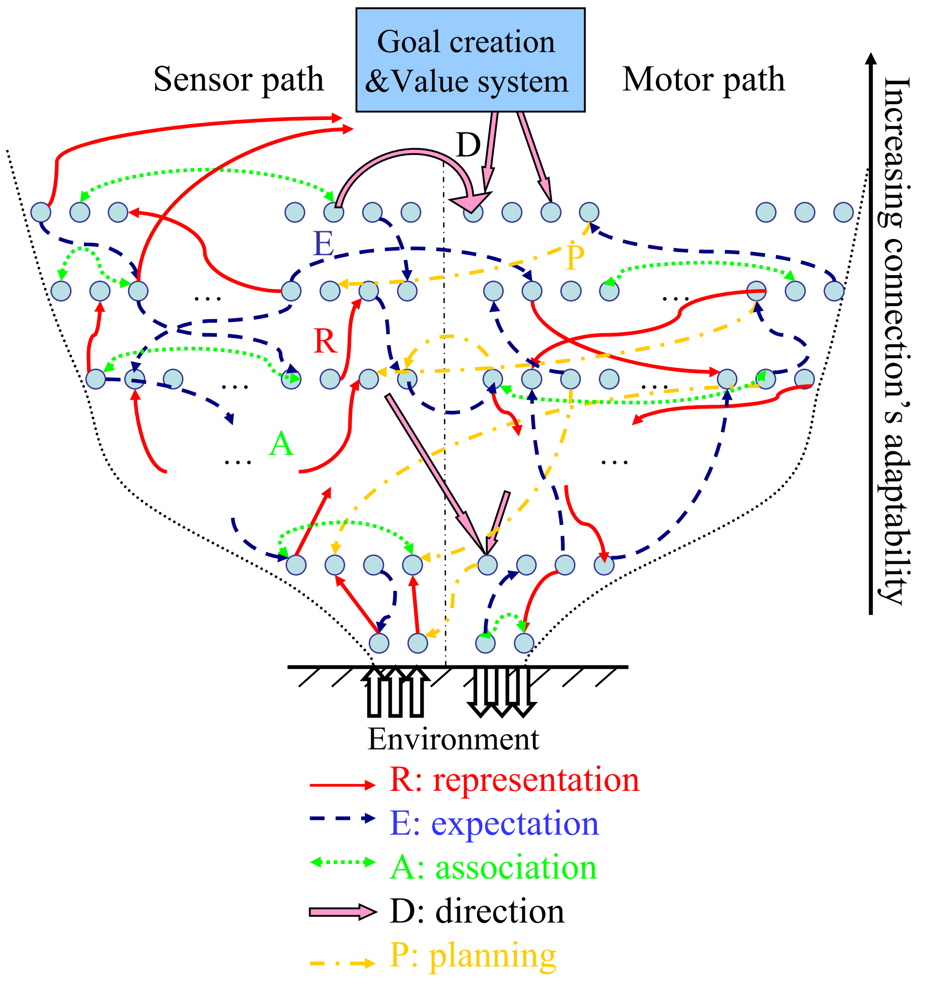


Figure 4.3. Sensory-motor coordination in HSOM.

In this figure, representation connections indicate entity recognition along the upward sensory pathway and represent downwards activations along the motor pathways. Feedback connections represent expectations of future inputs in the sensory and motor pathways. These expectations are provided by both motor neurons as well as higher level sensory neurons. The direction arrows indicate stimulation links from the sensory or from the goal creation pathways to motor neurons, while planning arrows connecting motor neurons to the sensory neurons predict the sensory inputs after the action represented by this motor neuron was completed.

4.2.2.3 Goal creation pathway

The primary objective of neurons in the goal creation pathway is goal creation, evaluation of actions in relation to current goals, and stimulation of the machine to perform useful actions. The growth of goal creation pathway triggers the intentional representation building and the growth of sensory and motor pathways. Primitive level of goal hierarchy is created based on simple external hostile signals from the environment. On different levels of the goal hierarchy, stimuli are internally generated so that machine is triggered to find desired actions and implement them. The found desired actions will be the reason to learn representations of related objects and skills and useful associations between them.

Similar to neurons in the sensory and motor pathways, neurons in the goal creation pathway are organized hierarchically in order to represent different levels of goals and the means of their realization. Lower level goals relate to simple, externally driven objectives. Higher-level goal creation neurons are developed based on lower level goal creation neurons and other neurons in the sensory-motor pathways. They correspond to complex objectives that are learned over the machine's operations and are related to the means to accomplish lower level goals. Since the lower level goals may be satisfied in many different ways, they can correlate with multiple higher level goals. In the EI research, goal creation pathway is less considered and less understood than the other two pathways. However, we propose it as an essential element of building EI. In Chapter 6, we will devote more efforts to developing the concept and structures for the goal creation pathway.

4.3 Conclusions

In this chapter, we presented a definition of EI. This definition of EI clearly differentiates knowledge from intelligence, with emphasis on the ability to acquire knowledge. A framework to design working models for EI is proposed based on our definition of embodiment and embodied intelligence. The design concepts aim to build intelligence on a simple and uniform neural structure. Two elements are proposed as essential for EI, including hierarchical self-organizing memory (HSOM) and goal creation system (GCS) to develop goal-oriented learning, and to stimulate a machine to interact with the environment. Three self-organizing hierarchical structures – sensory, motor, and goal creation pathways form the core of EI. They interact on various levels of abstraction and support the development in the hierarchical memories. The implementation of HSOM and GCS will be discussed more in later chapters.

CHAPTER 5: SPARSE CODING IN A HIERARCHICAL SELF-ORGANIZING MEMORY IN UNSUPERVISED LEARNING

5.1 Introduction

In this chapter, we focus chiefly on bottom-up sensory information processing in the sparsely connected HSOM, which is the sparse-coding procedure needed for object representation building in its sensory pathway. The following discussion is chiefly concerned with the implementation of architectures and algorithms that can provide some of the sparse-coding and classification requirements of an HSOM while retain their speed and efficiency when scaled to brain-size networks. This model employs a Hebbian learning rule [Heb 49], and produces reduced neuronal activity at higher hierarchical levels to build sparse codes for object representation. The memory capacity and the fault tolerance of the proposed algorithms will be investigated. A similar sparsely connected structure, which performs not only bottom-up information processing, but also uses the top-down information, may be used for selective attention-aided perception and invariance building- two critical functions in building sensory representations.

For sparse codes, an important characteristic is the activity ratio f, which shows the fraction of active neurons at any given time. Lower activity ratio increases code sparsity, and at its lowest value, single neuron representation is obtained. The activity ratio also affects the robustness of the representations, the number of distinct categories that can be represented (the representational capacity), and the generalization properties.

Various approaches to reducing neuronal activities, controlling the activity ratio, and obtaining sparse coding have been tried. In "soft" sparse coding, neuronal activities are constrained to smooth distributions that are sharper than Gaussian [Ols 96] [Ols 97]. In networks with sparse connectivity, an optimum activity distribution function depends on connectivity structure and input signal properties expressed by their statistically independent components as presented in [Bel 97] [Zha 04]. In "hard" sparse coding, the percentage of active neurons in a network is kept small [Reh 07]. Either approach requires the application of global activation criteria to all the neurons on a particular hierarchical level. Independent component analysis requires iterative computations to achieve optimum component representations and assumes that the inputs are linear mixtures of independent components. This strong assumption may not apply for natural data (nature images) and applying the methods to these data may deliver unsatisfying results [Zha 04].

Unsupervised, competitive learning, and the extreme case of using global winner neuron to build single-neuron representations of WTA classification is implemented in several existing networks, such as the competitive learning network [Aha 90] and Kohonen self-organizing map (SOM) [Koh 84], which can achieve vector quantization or dimensionality reduction. These networks typically have two-layered fully-connected feedforward structures. The winner output neuron in these methods is found by applying global WTA competition to the output, and then the connection links of the global winner neuron, and its predefined neighboring neurons, are adjusted during learning.

WTA competition implemented using traditional global WTA network [Hay 99] or MAXNET [Hay 99] [Zur 92] requires full connectivity and full comparison of activities of all neurons. Several studies have obtained sparse coding by means of WTA networks with lateral inhibitory links among neurons on the same level [Cou 92] [Xie 02]. Typically, such schemes also employ iterative computations to find the global winner. The competition time increases with the likelihood of similar signal strengths in large WTA networks. And the winner can be any neuron whose activity exceeds some threshold [Xie 02], rather than the neuron that constitutes the best representation. Prompt responses of humans to sensory information suggest that it is very unlikely that the winning neurons are selected through iterative global competitions or through global control of the overall neuronal activity.

A global WTA circuit with full connectivity is expensive in hardware implementation since it requires large design area and high power dissipation. It also suffers from matching problems, especially in systems with a large number of inputs [Fis 05]. To implement a large memory for a machine intelligence device, the circuit would have to compare many analog signal values, which would be inefficient and inaccurate. The competition time and resolution of finding the global winner are greatly affected by the circuit design and the analog values to compare [Sta 93].

In a sparsely-connected HSOM, sparse connections between neurons on different hierarchical levels may fail to transmit enough information up the hierarchy for reliable feature extraction and pattern recognition if the hierarchy is not properly organized. For instance, a hierarchy in which neurons' activities are regulated by a predefined activation threshold may either fail to activate neurons on the top level or activate too many of them. This makes sparse coding and representation building in the sensory pathway difficult.

In a local network model of cognition, called an “R-net” [Vog 97] [Vog 05], each primary neuron randomly projects onto a small fraction of the secondary neurons which in turn project onto the primary neurons, and provide indirect links between the primary neurons in a sparsely connected architecture. While R-nets provide large capacity associative memories, they were not used for feature extraction or sparse coding in the original work. By expanding the concept of R-nets, one or several layers of secondary neurons can be used to provide almost complete, indirect connectivity between the primary neurons on successive levels of the hierarchy. Such an approach is taken in this work.

Global WTA competitions tend to yield a single neuron (or few correlated neurons) representations and are known as localist representations. They are associated with a “grandmother cell” idea of questionable value in connectionists’ machine learning and lack biological support. Cortical neurons tests performed by [Des 89] [Tan 96] [Fel 91] demonstrated that multiple neurons are involved in encoding one input stimuli. Often distributed neuron activities are associated with localized and invariant features of the representations of the input stimuli build in memory. Thus, distributed and sparse representations have important advantages over localist representations as discussed in [Hin 86] [Hum 97].

We wanted to deploy, in the HSOM, a sparsely connected architecture that is biologically inspired and a mechanism that is simple, efficient, and easy to implement in hardware. The model and the algorithm presented in this chapter come from a new synthesis of the above ideas. It investigates the use of hierarchical, self-organizing learning in sparsely connected networks to accomplish sparse recoding of densely coded input patterns. In this sparse structure, the neuronal activities are gradually reduced and the activation pathway is formed for certain sensory inputs. To build localist representation, we implement the method, sparse winner-take-all (SWTA), to find global winners on the top level of a hierarchy of sparse structures through efficient localized WTA competitions. Alternatively, an "oligarchy-take-all" (OTA) competition is employed to build distributed representation in which a group of neurons replace the global winner on the top level. The OTA approach increases memory reliability through coding redundancy in the self-organizing learning.

The suggestion that memories are built through modification of the strengths of synapses between active neurons is commonly referred to as Hebbian Learning [Heb 49]. During memory building, the Hebbian learning adopted in this work uses weight vectors confined to the unit multidimensional sphere. The learning is performed by adjusting the connection links of active neurons. Findings on the plasticity of interneuron connections [Mar 04] suggest the plasticity of synapses is affected by experience and the aging process. We can postulate that the weights that are adjusted frequently should become less plastic (i.e., the amount of adjustment they undergo in response to new patterns decreases). It is expected that this reduction in plasticity is particularly noticeable in the connections of the lower-level neurons that represent basic features. Such features appear in a large number of input patterns and are adjusted frequently. Higher level neurons that can be used to store abstract representations composed of the lower level features are activated less frequently than the lower level neurons and are expected to have connections with greater plasticity.

5.2 Structure and connectivity of the sensory pathway

One or several layers of secondary neurons can be used to provide almost complete, indirect connectivity between the primary neurons on successive levels of the hierarchy, as shown in Figure 5.1. Each lower level primary neuron randomly projects onto a small fraction of secondary neurons that, in turn, fan out onto primary neurons of the next level in the network, producing indirect links between any primary neuron of one level and almost all primary neurons of the next level. In Figure 5.1, two primary levels *h* and *h+1* are shown to be connected through the secondary level *s*.

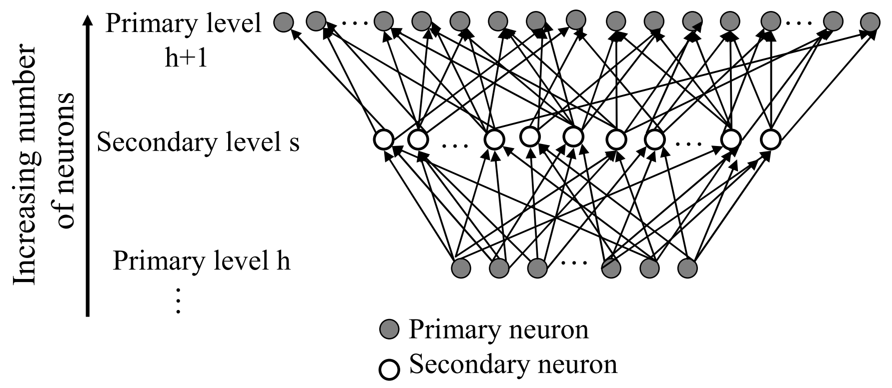


Figure 5.1. Primary and secondary levels in HSOM.

As we proposed in this model, a large memory network may have an increasing number of neurons at higher levels, the number of secondary neurons on the secondary level s, denoted by N_s, may be larger than the number of primary neurons on the lower level h, N_h, and smaller than the number of primary neurons on the higher level $h+1$, N_{h+1}. For example, we can have

$$N_s = \alpha N_h, \quad \text{and} \quad N_{h+1} = \alpha N_s, \tag{5.1}$$

where α denotes a growth factor for the number of neurons on each layer. While we do not stipulate that a higher layer must have a larger number of neurons than the lower layer, we believe that occasionally such a feature of sparse memory may be useful and our model works well in such structures.

5.2.1 Connectivity analysis

Let both the secondary neurons and the primary neurons on the successive layer have an equal numbers of input connections. For example, we can have,

$$l_{in}^{s} = \sqrt{N_h}, \quad \text{and} \quad l_{in}^{h+1} = \sqrt{N_h} \ , \tag{5.2}$$

where l_{in}^{s} and l_{in}^{h+1} denote the numbers of input connections to the secondary and the primary neurons on the secondary level s and the primary level $h+1$, respectively. Since the total number of output connections from a layer must equal the total number of input connections to the next layer, we have

$$N_s l_{in}^{s} = N_h l_{out}^{h}, \text{ and } N_{h+1} l_{in}^{h+1} = N_s l_{out}^{s} . \tag{5.3}$$

The average number of output connections from each neuron to a higher layer is given by

$$l_{out}^{s} = \sqrt{N_{h+1}}, \text{ and } l_{out}^{h} = \sqrt{N_{h+1}} \ . \tag{5.4}$$

The total number of interconnections between each two primary levels can be estimated as,

$$\begin{aligned} l_{all} &= l_{in}^{h+1} N_{h+1} + l_{in}^{s} N_s \\ &= \sqrt{N_h} N_{h+1} + \sqrt{N_h}\sqrt{N_h N_{h+1}} \\ &= \sqrt{N_h N_{h+1}}\left(\sqrt{N_{h+1}} + \sqrt{N_h}\right) \end{aligned} \quad . \tag{5.5}$$

The input weight vectors of neurons (including primary and secondary neurons) place the neurons in a multi-dimensional space. When a particular input pattern is processed, neurons receive inputs from the lower level, and their activations are determined by the inner products of the input vector and their input weight vectors. We compare these inner products to find out the winner. In order for neurons to have a fair competition, and for neurons' weight vectors to represent many categories in the output space, the weight vectors are desired to be unitary (i.e., the summation of the squared input weights is set to 1).

$$\sum_{j=1}^{l_{in}^{s}} \left(w_{ij}^{s}\right)^2 = 1, \text{ and } \sum_{j=1}^{l_{in}^{h+1}} \left(w_{ij}^{h+1}\right)^2 = 1 \tag{5.6}$$

5.2.2 Hierarchical organization

The outputs of several lower level networks can be combined at higher levels. For example, one network with a 2-layer structure can extract basic features on level *h+1*. The activities of *r* such networks can be combined and sent to a higher level, say *h+2,* through secondary level *s+1* for higher level feature extraction or object representation, as in Figure 5.2.

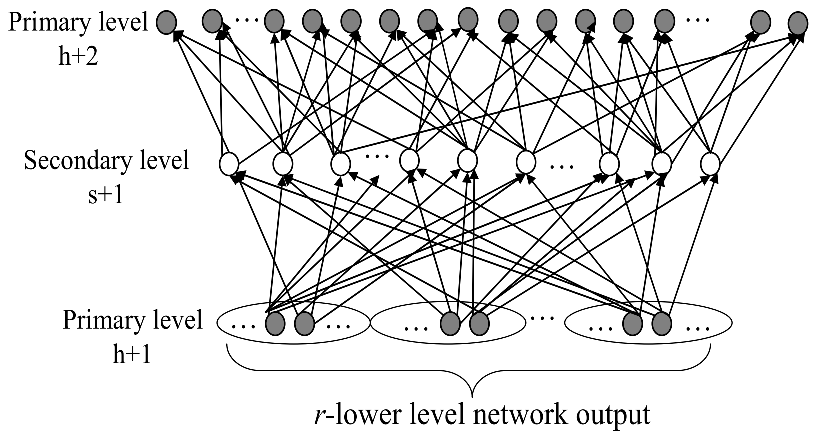


Figure 5.2. *r*-lower level networks combined for higher level feature extraction.

This kind of hierarchy, where a higher layer combines locally selected features extracted by groups of lower layer neurons, was suggested by several authors [Ore 00] [Haw 04] as a desirable form of cognitive architecture. The total number of interconnections between each two primary levels may be very large for large memories. Suppose, for example, we have a network with 10^{11} neurons in the highest level, and 10^{9} neurons in the input level (N_1). Connecting the input to the highest level is comparable to connecting the entire sensorium to the entire cerebrum without regard to specialization of any region. The total number of connections and the number of output connections are given by

$$\begin{aligned} &l_{all} = l_{in}^{h+1} N_{h+1} + l_{in}^{s} N_s = \sqrt{10^9}\,10^{11} + \sqrt{10^9}\sqrt{10^{11}10^9} \approx 3.5\times 10^{15} \\ &l_{in}^{s} = l_{in}^{h+1} = 10^{4.5},\ \text{and}\ l_{out}^{h} = l_{out}^{s} = 3.2\times 10^{5} \end{aligned} \quad . \tag{5.7}$$

The number of connections, here, exceeds the connectivity of pyramidal cells of the human cortex [Rol 89]. However, if such connectivity is required with fewer synapses, we can increase the overall sparsity of the network while maintaining sufficient numbers of connections to transmit the sensory information through the hierarchy by increasing the number of secondary layers. As shown in Figure 5.3, a "stacked" R-net structure, in which two layers of secondary neurons separate primary neuron levels, can further reduce the connection density. With growth factor α, the numbers of neurons in secondary layers s_1 and s_2, and in the next primary layer $h+1$ are,

$$N_{s1} = \alpha N_h,\ N_{s2} = \alpha N_{s1},\ N_{h+1} = \alpha N_{s2}\,. \tag{5.8}$$

For example, we can have the number of input connections and the average number of output connections for all the neurons as,

$$l_{in}^{s1} = l_{in}^{s2} = l_{in}^{h+1} = \sqrt[3]{N_h} \tag{5.9}$$

$$l_{out}^{h} = l_{out}^{s1} = l_{out}^{s2} = \sqrt[3]{N_{h+1}}\,. \tag{5.10}$$

Then the total number of connections between all the neurons is

$$\begin{aligned} l_{all} &= l_{in}^{s1} N_{s1} + l_{in}^{s2} N_{s2} + l_{in}^{h+1} N_{h+1} \\ &= \sqrt[3]{N_h}\left(\sqrt[3]{N_{h+1} {N_h}^2} + \sqrt[3]{N_{h+1}^2 N_h} + N_{h+1}\right) \end{aligned}. \tag{5.11}$$

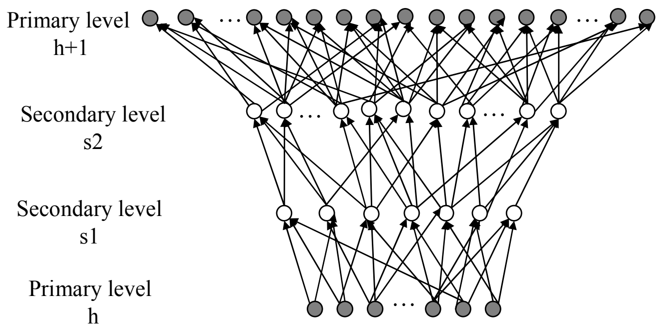


*Figure 5.3.*Three-layer per level structure.

If this three-layer per level organization is again applied to a network with $N_{h+1} = 10^{11}$ and $N_h=10^9$, we get,

$$l_{out}^{h} = 10^{3.67},\ l_{all}^{h} \approx 10^{14}. \tag{5.12}$$

Here, the connectivity is much less than that of a typical pyramidal cell. When the input size, N_h, and the output size, N_{h+1}, are specified, we can build an HSOM of almost any complexity while maintaining almost any connectivity.

The network structure and its connectivity are functions of the number of secondary levels, n_s, as

$$\begin{aligned} &\alpha = \sqrt[(n_s+1)]{\frac{N_{h+1}}{N_h}} \\ &l_{in}^{level} = \sqrt[(n_s+1)]{N_h} \quad (level = s1, s2, \dots, sn_s, h+1) \\ &l_{out}^{level} = \sqrt[(n_s+1)]{N_{h+1}} \quad (level = h, s1, s2, \dots, sn_s) \\ &l_{all} = l_{in} N_h \sum_{i=1}^{n_s+1} \alpha^i \end{aligned} . \tag{5.13}$$

The number of connections to each neuron and the overall connectivity are reduced by adding more secondary levels, as shown in logarithmic scale in Figure 5.4, for the sample network with $N_{h+1} = 10^{11}$ and $N_h=10^9$. We note that the number of connections required in fully connected networks is much higher than those in sparsely connected networks.

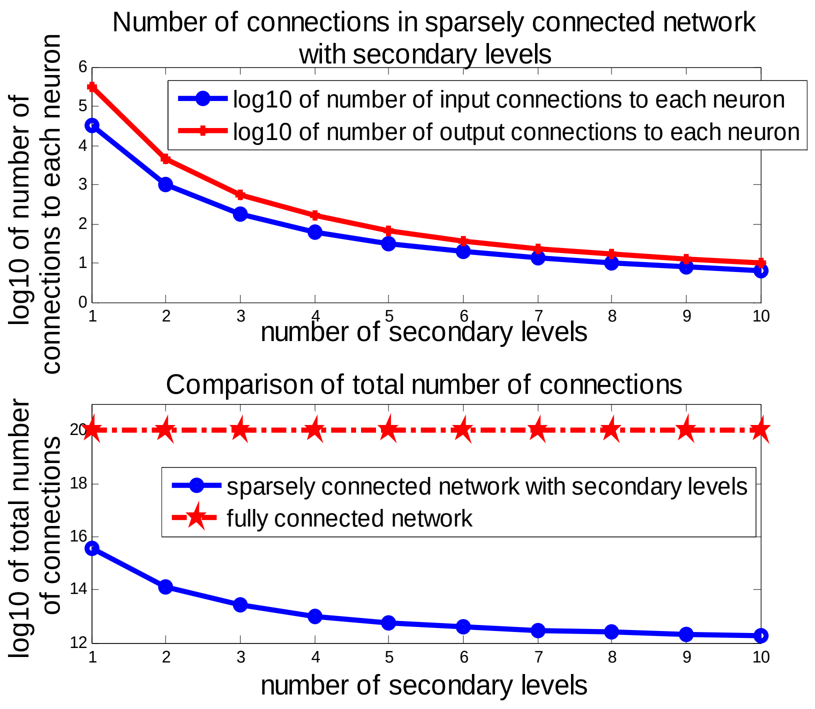


Figure 5.4. Connectivity of the networks with different numbers of secondary levels.

The networks with secondary layers separating primary layers provide sparseness in both structure and data representation while complete information is transmitted up the hierarchy. Such a network ensures high capacity memory, which will be discussed in Section 5.3.4. In general, the sparse structure with WTA reduces maximum memory capacity per neuron, while increasing storage capacity per synapse. A proper trade-off between memory capacity and overall hardware requirements must be considered during the system implementation.

5.3 Hierarchical self-organizing memory with sparse winner-take-all

In unsupervised learning, the intended response of the network to each pattern is not given. When the input data propagate from lower to higher levels, the neurons on the higher layer combine output signals of the neurons on its lower levels and produce different levels of activation. Among the neurons on the top level of a hierarchy, there will be a neuron having the greatest output signal strength, and will be recognized as the global winner that provides the best representation of the input pattern. An efficient, hardware-oriented algorithm that uses local competitions is proposed to find this global winning neuron and the input pattern's activation pathway. The algorithm is implemented for a multilayer sparse network, so we call it a "sparse winner-take-all" (SWTA) algorithm. The algorithm employs three basic steps:

(1). Sending the inputs up through the hierarchy

(2). Finding the global winner and its "winner network" by means of local competitions

(3). Recalculating the neuronal activities and training the network through weight adjustments on connections within the "winner network"

In these three steps, a signal is propagated through the entire network three times. First, a forward propagation is used to obtain signal strengths of all the neurons. Second, a back propagation is used to determine the "winner network" through local competitions. Third, the forward propagation through the local winners identifies the global winner and trains the connection weights. The following sections describe the three basic steps of SWTA algorithm.

5.3.1 Data transmission

In a network built using the structure with secondary and primary levels discussed in Section 5.2, the connections and connection weights are initialized with uniformly random values. To ensure a fair competition, the summation of the squared input weights to each neuron is set to 1. If the initial random weights have $\sum w_j^2 \neq 1$, a scaling factor τ is applied so that we have

$$\tau = \sqrt{\sum w_j^2} \Rightarrow \sum \left(\frac{w_j}{\tau} \right)^2 = 1 \ . \tag{5.14}$$

The input weight vector is on the unit multidimensional sphere, giving each neuron an equal probability of firing, and increasing the memory capacity of the HSOM.

In the first step of the SWTA algorithm, the input information is propagated to the highest level of the network using feed forward computations. In this feed forward computation, each post-synaptic neuron sums its weighted inputs. Each neuron has a transfer function so that the neuron fires and sends a signal to its own post-synaptic neurons when the summed signal strength exceeds an activation threshold, which is typically the medium of the signal range. The neuron's output is expressed in (5.15), where S_j^{layer} and $S_i^{layer+1}$ denote the signal strengths of neuron *j* of level (*layer)* and neuron *i* of level (*layer+1)*, respectively. N_i^{layer} denotes the set of pre-synaptic neurons on level (*layer)* feeding neuron *i* on level (*layer+1)* and w_{ij} denotes the input weights.

$$S_i^{layer+1} = \begin{cases} \sum\limits_{j \in N_i^{layer}} w_{ij} S_j^{layer} & \sum\limits_{j \in N_i^{layer}} w_{ij} S_j^{layer} > threshold \\ 0 & \sum\limits_{j \in N_i^{layer}} w_{ij} S_j^{layer} < threshold \end{cases} \tag{5.15}$$

This feed forward computation establishes initial signal strengths for each neuron of the network. In the second step, local competitions at each layer are used to establish the winner network.

5.3.2. Finding the winner network

In a sparsely connected network, a global winner on the top level can be found without global competition. Each neuron on a certain level connects to a group of post-synaptic neurons on the next higher layer. These post-synaptic neurons are considered as pre-synaptic neuron's "post-synaptic local region". Based on the neurons' output strengths computed in the 1st step of SWTA algorithm, local competitions can be performed within corresponding post-synaptic local regions.

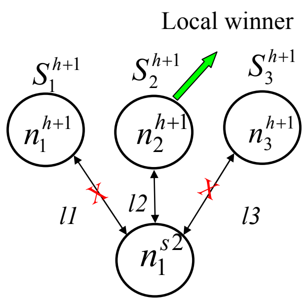
In hardware implementation, this local competition in a post-synaptic local region can be easily implemented using an analog, current-mode WTA circuit [Sta 93] in which the activity of neuron j is represented as the input current I_j^{in}. Figure 5.5 (a) shows an example for a neuron n_1^{s2} on secondary level *s2*, which projects onto 3 post-synaptic neurons on level (*h+1*). For example, if the input node n_2^{h+1} has the highest signal strength S_2^{h+1}, it will become the local winner in this post-synaptic region and will pull out all the current determined by the bias voltage, and the branches connected with losing nodes, *l1* and *l3*, will be logically disconnected. Node n_1^{s2} receives the full signal from n_2^{h+1}, and the signal strength of n_2^{h+1}, S_2^{h+1}, propagates down to n_1^{s2}.

In a multi-level hierarchy illustrated in Figure 5.5(b), on a hierarchical level *(layer)* with N^{layer} neurons, post-synaptic local winners are selected according to each neuron's post-synaptic region on the next level of the hierarchy, as in (5.16a). During the local competition using WTA circuit, each node n_i^{layer} $(i = 1, 2, \ldots N^{layer})$ on level *(layer)* receives the signal strength from its post-synaptic local winner, as in (5.16b). This neuron maintains an active connection only to this local winner and the connections to other neurons in post-synaptic region are logically disconnected, as in (5.16c), where $l_{ji}^{layer+1}$ denotes the logic status of a connection between neuron *i* on level (*layer*) and neuron *j* on level *(layer+1)*, $N_i^{layer+1}$ is a set of post-synaptic neurons on level *(layer+1)* driven by a neuron *i* on level (*layer*), and N_j^{layer} is a set of pre-synaptic neurons of neuron *j* on level (*layer*).

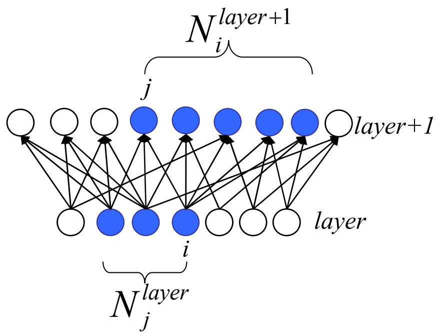
$$S_{winner\ i}^{layer+1} = \max_{j \in N_i^{layerl+1}} \left\{ \sum_{k \in N_j^{layer}} w_{jk} S_k^{layer} \right\} \quad (i = 1, 2, ..N_{layer}) \tag{5.16a}$$

$$S_i^{layer} = S_{winner\ i}^{layer+1} \tag{5.16b}$$

$$l_{ji}^{layer+1} = \begin{cases} 1 & j : local\ winner\ among\ N_i^{layerl+1} \\ 0 & j : not\ \ local\ winner\ among\ N_i^{layerl+1} \end{cases} \tag{5.16c}$$



(a) Winner-take-all circuit in local competition



(b) Interconnection structure to determine a local winner

Figure 5.5. Local winner-take-all circuit.

Localized competitions begin at the highest level. The global winner neuron wins local competitions in each local post-synaptic region in which it is present and its signal strength is passed down to its pre-synaptic neurons. This process of local competitions is repeated at each level until the input layer is reached. By receiving the global winner's signal strength, the pre-synaptic neurons will win the corresponding local competitions as well. Accordingly, all the lower-level neurons which connect to the global winner directly or indirectly will win local competitions and maintain their connections to their pre-synaptic neurons. Specifically, all branches connected to the global winner are kept active while those branches not connected to the global winner are logically disconnected. All these active branches in the hierarchy form the "winner network", as shown in Figure 5.6. Notice that the global winner was not established explicitly at the beginning of the competition. Rather, the global winner begins as a local winner, and progressively wins all competitions.

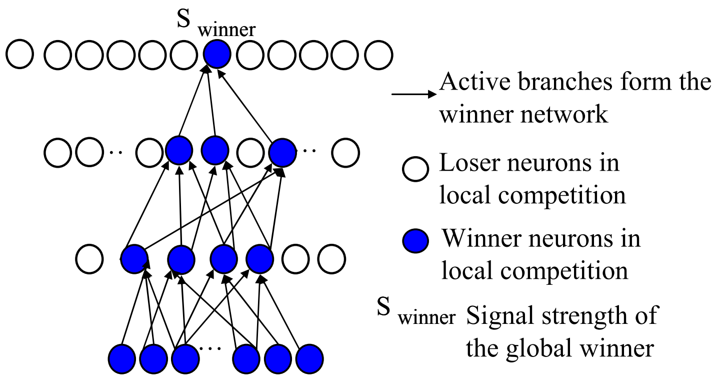

S winner
Active branches form the winner network
Loser neurons in local competition
Winner neurons in local competition
S winner Signal strength of the global winner

Figure 5.6. Winner network.

Definition: A winner network is a sub-network of the original network that contains all branches connected to neurons that received the winner signal in the back propagation step of SWTA algorithm.

If used in a fully connected network without SWTA algorithm, the WTA circuit will be used to find global winners by comparing all the neuron activities in the same circuit. But as stated earlier and as was experimentally established by earlier works, such implementation will greatly increase the computational time for finding global winners for networks with a large number of output neurons.

5.3.3. Learning in the winner network

The logical statuses of connections are determined through local competitions in the 2nd step of SWTA. In the 3rd step of SWTA, the input pattern is forwardly propagated through the network again and neuronal activities are recalculated considering the connections' statuses, as in (5.17).

$$S_i^{layer+1} = \begin{cases} \sum_{j \in N_i^{layer}} w_{ij} l_{ij}^{layer} S_j^{layer} & \sum_{j \in N_i^{layer}} w_{ij} l_{ij}^{layer} S_j^{layer} > threshold \\ 0 & \sum_{j \in N_i^{layer}} w_{ij} l_{ij}^{layer} S_j^{layer} < threshold \end{cases} \tag{5.17}$$

The logically disconnected branches, which have l_{ij}^{layer} as zeros, do not contribute to post-synaptic neuronal activities, and only the active branches in the winner network will send the information up to higher levels. Since all the branches in the winner network are active, the signal strength of the global winner will not be reduced. However, because of the logical disconnection of branches of other output neurons, the signal strength of these neurons decreases and typically falls below threshold. In this way, after the 3rd step of SWTA, the global winner will be identified as the only active neuron on the top level.

Hebbian learning is carried out simultaneously during this step. Weights are only adjusted for connections within the winner networks, since only connections to these neurons contribute to the recognition of this input pattern. The weight vectors are updated so that the activation level of the global winner is reinforced in accordance with (5.18), where $\lambda_{j,i}^{layer+1}$, the plasticity of the connection between neuron *i* of level *(layer)* and neuron *j* of level *(layer+1)*, specifies the learning rate for its weight adjustment.

$$\Delta w_{j,i}^{layer+1} = \lambda_{j,i}^{layer+1} S_i^{layer} S_j^{layer+1} l_{ij}^{layer} \tag{5.18}$$

After updating, the weights are scaled using (5.14) so that they satisfy (5.6). As discussed in Section 5.1, the connection plasticity decreases and the learning rate decays with the number of weight adjustments. For example, we can have

$$\lambda_{j,i}^{layer+1} = \rho^{r}, \tag{5.19}$$

where r denotes the number of times this connection has been updated and ρ (ρ<1) is the original plasticity of the connection.

The procedure of SWTA is performed for processing and training on each input pattern. The logic statuses of connections are determined in order to find the global winner representing the current input pattern. When new input pattern comes into the network, the network activations are reset and determined again for the new input.

In summary, the network builds sparse representations by finding global winner neurons by means of local competitions in sparsely connected networks in which neurons of different primary levels are connected through secondary neurons. The SWTA finds global winners in three steps: propagating data forward, finding the winner network by back-propagating the signal strength from top level neurons, and forward propagation of input signals through active connections in the winner network. The algorithm provides an effective and efficient solution to the problem of finding global winners in large networks, especially those on the scale needed to build human-level intelligence.

In later sections, the learning ability of the HSOM with SWTA will be tested on a hierarchical structure with two primary levels. The representational capacity of single-neuron codes, as the number of distinct categories they can represent, is limited by the number of neurons on the top level of hierarchy. The sparse connectivity will result in reduction of capacity since it reduces the information flowing upwards the hierarchy. The representational capacity of networks with SWTA is also investigated.

5.3.4 Properties of the winner networks

The properties of winner networks depend on the topology of their interconnections and can be studied statistically. When connectivity of the network is sufficiently low, more than one winner network, connected to active neurons at the highest level, can be found. By setting the connectivity above some threshold, dependent on the network size, we can have a single winner with all of the active branches in a single winner network. In the following example, we analyze the conditions under which a single winner is obtained.

Example

We consider a structure with two secondary layers per level with 64 neurons on the input level, the growth factor α in (5.8) set to 2.25 (giving levels that are 1.5 times larger than the previous level in each dimension of two dimensional input), and 729 output neurons. The number of active output neurons at the top level decreases with increasing numbers of input connections for each neuron (as shown in Figure 5.7 by a solid line). For this size of network, we observe that there is typically a single winner neuron at the top level when the number of input connections to each neuron in the network is greater than 6.

In the local competitions, we can choose to keep, as local winners, all the neurons having activations within a certain range of the greatest local activation. This tolerance range in the local competitions affects the number of winners on the top level. For this structure with 64 input, 729 output neurons and 6 input connections for each neuron, the number of active neurons at the highest level increases rapidly with increasing tolerance, as shown in Figure 5.8 by a solid line. While a small number of active output neurons may be beneficial, because they introduce robustness in object representations, too great a number of active neurons may defeat the purpose of sparse coding process needed for information storage in sparsely connected networks.

This experiment was repeated with a different network size, choosing α in (5.8) as 4, and reaching 4096 output neurons. The results are shown by dashed lines in Figure 5.7 and Figure 5.8. We observe a similar trend relating number of winners and signal strength to the number of input connections.

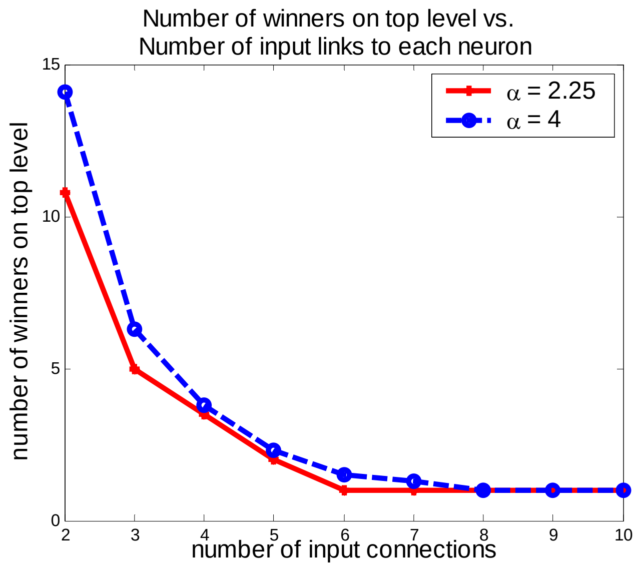


Figure 5.7. Effect of the number of input connections on the number of winners in SWTA.

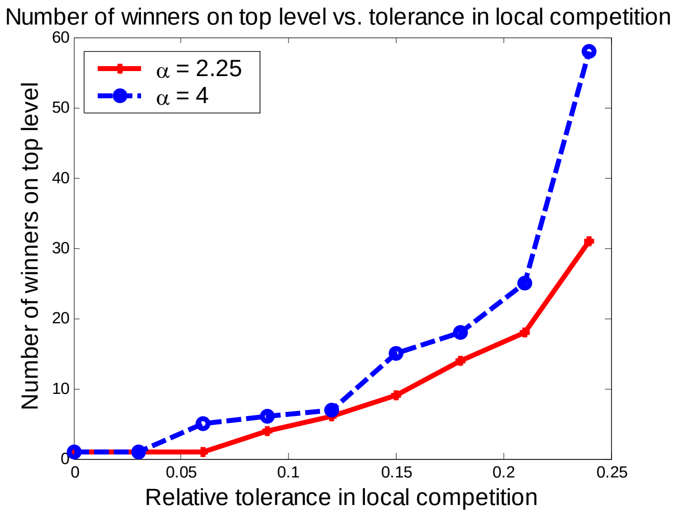


Figure 5.8. Effect of tolerances in local competition on number of winners in SWTA.

In summary, the sparsely connected network in which neurons at different primary levels are connected through secondary neurons builds sparse representations and finds global winner neurons by means of local competitions.

The SWTA finds global winners in three steps: propagating data forward, finding the winner network by back-propagation and local competitions, and forward propagation of input signals through active connections in the winner network, organized as the following Table 5.1.

Table 5.1

Sparse Winner-Take-All Algorithm (SWTA)

1. Data transmission

(feedforward process)

```
FOR Layer = 2: Top Layer
    FOR i=1: number of neurons on (Layer)
        $S_i^{layer+1} = \begin{cases} \sum_{j\in N_i^{layer}} w_{ij} S_j^{layer} & \sum_{j\in N_i^{layer}} w_{ij} S_j^{layer} > threshold \\ 0 & \sum_{j\in N_i^{layer}} w_{ij} S_j^{layer} < threshold \end{cases}$    (5.15)
    ENDFOR
ENDFOR
```

2. Finding the winner

network

(backpropagation process)

```
FOR Layer = Top Layer-1: -1: 1
    FOR i=1: number of neurons on (Layer)
        $S_{winner\ i}^{layer+1} = \max_{j\in N_i^{layerl+1}} \left\{ \sum_{k\in N_j^{layer}} w_{jk} S_k^{layer} \right\} \quad (i = 1,2,..N_{layer})$
        $S_i^{layer} = S_{winner\ i}^{layer+1}$    (5.16)
        $l_{ji}^{layer+1} = \begin{cases} 1 & j: local\ winner\ among\ N_i^{layerl+1} \\ 0 & j: not\ \ local\ winner\ among\ N_i^{layerl+1} \end{cases}$
    ENDFOR
ENDFOR
```

3. Learning in the winner

network

(feedforward process)

```
FOR Layer = 2: Top Layer
    FOR i=1: number of neurons on (Layer)
        $S_i^{layer+1} = \begin{cases} \sum_{j\in N_i^{layer}} w_{ij} l_{ij}^{layer} S_j^{layer} & \sum_{j\in N_i^{layer}} w_{ij} l_{ij}^{layer} S_j^{layer} > threshold \\ 0 & \sum_{j\in N_i^{layer}} w_{ij} l_{ij}^{layer} S_j^{layer} < threshold \end{cases}$    (5.17)
        $\Delta w_{j,i}^{layer+1} = \lambda_{j,i}^{layer+1} S_i^{layer} S_j^{layer+1} l_{ij}^{layer}$    (5.18)
    ENDFOR
ENDFOR
```

5.3.5 Comparison between SWTA and WTA algorithms

The presented SWTA algorithm provides an effective and efficient solution to the problem of finding global winners in large networks, especially those on the scale needed to build human-level intelligence. The number of calculations to process the input information and the number of connections are compared between a sparse network with SWTA and a traditional fully connected network with global WTA in Table 5.2. In this example, assume the network has N_h inputs and the growth factor α . The structural properties of two networks are listed in Table 5.2 as well. It is noted that the computation cost and the design area are both correlated with the overall number of connections in the network.

Figure 5.9 compares the connectivity of a sparse network to that of a fully-connected network as a function of the number of input neurons N_h (logarithmic scale). One can see that the sparsely connected network has a greatly reduced number of connections which results in reduced calculation cost and design area. In comparisons of parallel processing hardware architectures, efficiency is measured by a product of time and area. Thus an architecture that requires smaller area is more efficient than the one with larger area even if the computational time is the same in both. Sparse network with SWTA shows its advantage especially in large networks.

Table 5.2

Comparison between SWTA and global WTA

	Sparse network with SWTA	Fully-connected network with global WTA
Number of layers	4	2
Structure (number of neurons on each level)	$N_h - \alpha N_h - \alpha^2 N_h - \alpha^3 N_h$	$N_h - \alpha^3 N_h$
Number of inputs to each neuron	$\sqrt[3]{N_h}$	N_h
Number of connections	$(\alpha + \alpha^2 + \alpha^3) N_h^{4/3}$	$\alpha^3 N_h^2$
Number of calculations	$(\alpha + \alpha^2 + \alpha^3) N_h^{4/3}$	$\alpha^3 N_h^2$
Design area	$\alpha(\alpha + \alpha^2 + \alpha^3) N_h^{4/3}$	$\alpha\alpha^3 N_h^2$

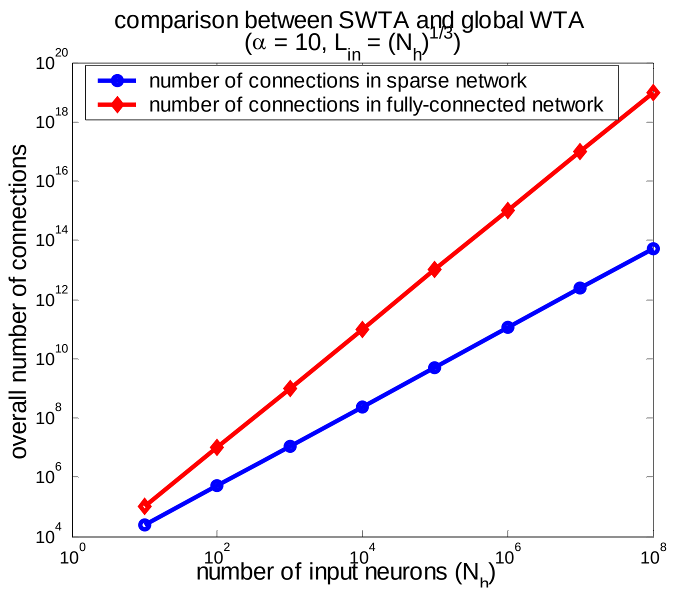


Figure 5.9. Comparison of connectivity between SWTA and global WTA.

Representational capacity of single-neuron codes, as the number of distinct categories stored, is limited by the number of output neurons. The sparse connectivity used in SWTA results in a slight reduction of memory capacity since it reduces the information transmission through the hierarchy. However, the memory per synapse is improved. Further investigation on the representational capacity of networks with SWTA is provided in Section 5.3.7.

5.3.6 Finding a global winner in the network with SWTA

The learning ability of the network with SWTA mechanism was tested on the same 3-layer per level structure with 64 input and 729 output neurons described in the example in Section 5.3.4. Weights were in the range [-1, 1] and the inputs for learning were grey-scale patterns in the range [-1, 1] as well. When a pattern is applied to the network with the SWTA, a single neuron is found as the global winner at the highest level. In this experiment, an 8 by 8 grey-scale random image pattern is applied to the randomly initialized network. The network builds the input representation and learns by adjusting its weights. To evaluate the network's fault tolerance, 5 bits of the original information were assigned random values, and the distorted pattern was presented to the network. A random pattern and its distorted pattern are shown in Figure 5.10.

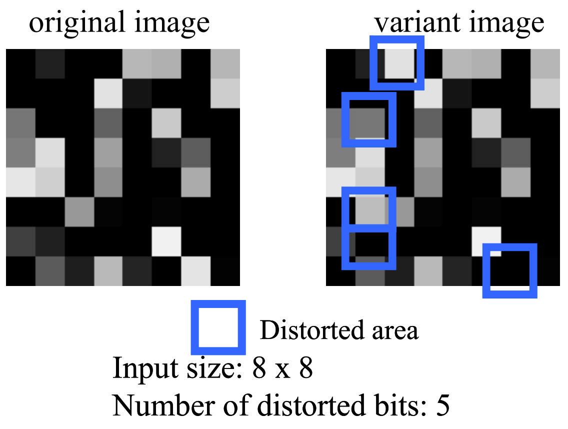


Figure 5.10. Original random pattern and its distortion presented to the SWTA.

The activities of output neurons, after the original pattern is processed are shown in Figure 5.11(a). Activities of the output neurons are then shown in Figure 5.11(b) after finding the winner network and adjusting the weights. After the local competitions and weights adjustment, only the global winner neuron has its signal strength higher than the activation threshold. After learning the original input pattern, the network with SWTA responds to the distorted pattern, as shown in Figure 5.12 before (a) and after (b) the local competitions.

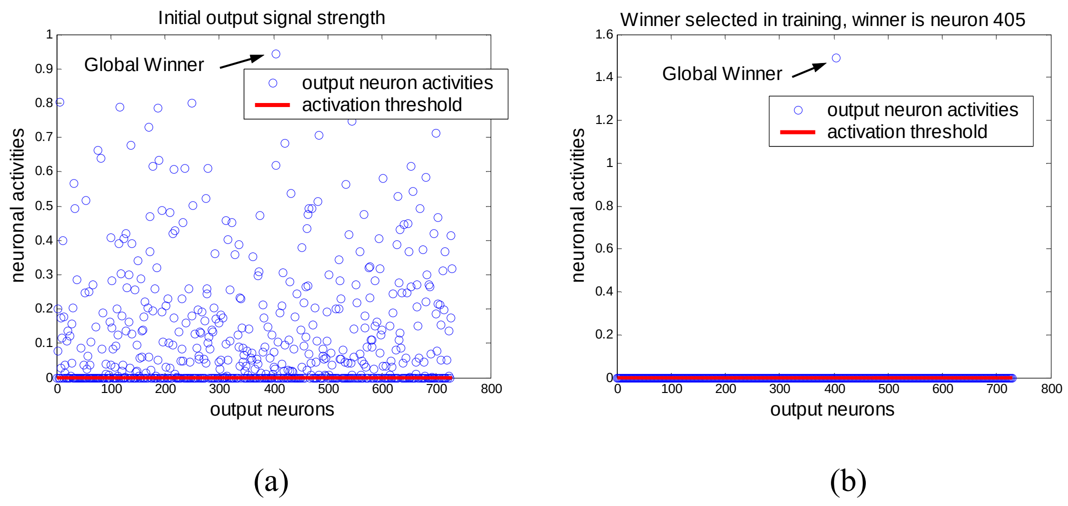


Figure 5.11. Output neuron activities for original pattern.

(a). before (b). after local competitions

Such small levels of pattern distortion can be tolerated by the network which gives the same neuronal representation for both patterns. However, it is anticipated that, as the network learns more patterns, the level of tolerance for distortion of the network will decrease. This expectation was tested by finding the fraction of distorted patterns that were correctly recognized by the network with SWTA as a function of the number of patterns. The performance is obtained based on 10 Monte-Carlo runs (shown in Figure 5. 13). Each stored pattern was tested 10 times with 5 different bits in the pattern randomly changed on each trial. The results agree with the expectation that level of distortion tolerance of the network on classification problems decreases with increasing numbers of training patterns. The performance of a STWA network is evaluated with and without training. During the training, the representation built in the network for a particular pattern is reinforced so that each pattern can be better recognized when a large number of patterns are stored in the network.

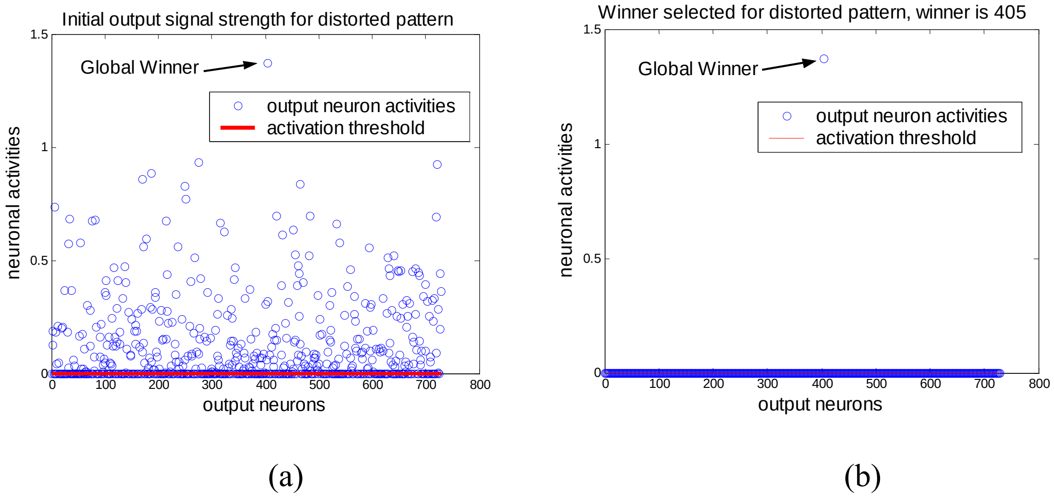


Figure 5.12. Output neuron activities for distorted pattern.

(a). before (b) after local competitions

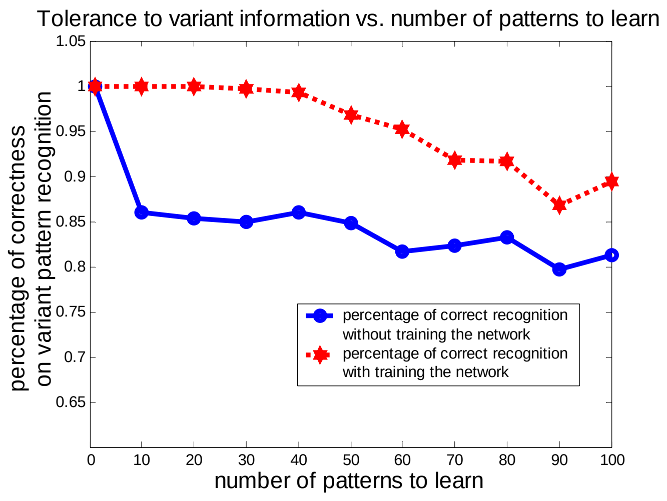


Figure 5.13. Recognition ability of the network with SWTA.

(Percentage of correct recognition of distorted information vs. number of patterns learned)

5.3.7 Representational memory capacity of the network with SWTA

In sparse coding, an important measure of the memory structure is to evaluate how many representations of the input categories can be reliably obtained. Such representations can then be stored in the associative memory, that is capable to associate different parts of the code and recover the missing parts. Associative memory capacity (typically referred to as memory capacity) is then different from the representational capacity. In a fully-connected WTA network, each pattern is represented by a single neuron at the highest level, and the maximum number of distinct categories the network can represent and recognize equals to the number of top-level neurons. It is desired that the representational capacity of SWTA be close to this number as well. In the next experiment, a network with 729 neurons on the top level was exposed to 7000 random patterns. It is found that 452 different neurons at the top level were activated at least once. It means that approximately 57% output neurons can be used to store and represent categories, and the SWTA can store 452 different categories without interference. Figure 5.14(a) shows the neuronal firing frequency distribution, and it is observed that most of these activated neurons are triggered by a small number of patterns. The neuronal firing frequencies of these 452 neurons, sorted by descending order, are shown in Figure 5.14(b). It is noticed that one of the top-level neurons is activated by 197 patterns. The distribution of similarities, in the sense of sum of squared intensity difference [Hil 01], among these 197 patterns is compared with the distribution of similarities among randomly generated patterns in Figure 5.15. In this metric, similar patterns have small intensity differences. As we can see from Figure 5.15, the average intensity difference for patterns which trigger the same output neuron is two standard deviations smaller than that of random patterns. This indicates statistical significance for the similarity of patterns that trigger the same output neuron. Using the results presented in Figure 5.15, one can easily estimate the likelihood that a pattern is from the given class rather than a random pattern.

For a fully connected network with 64 neurons as the input and 729 neurons as the output with no secondary levels, all the 729 neurons at the highest level are activated at least once after applying 7000 random patterns. The neuron firing frequencies and the firing frequency distribution are also shown in Figure 5.14(a) and (b) for comparison. This indicates that the reuse of the common links in storing various input patterns in the sparse structures reduces the representational capacity per neuron of the network with SWTA compared to that of a fully connected memory with global WTA competition on the output layers. However, the representational capacity per synapse of a SWTA network is significantly higher than that of a fully connected WTA network, and is 6.04 times as great in the given example. This representational capacity can be computed using

$$C_{synapse} = \frac{N_{activated \quad neurons}}{l_{all}} \tag{5.20}$$

where $N_{activated \quad neurons}$ is the total number of neurons that can be used to represent categories and l_{all} is the total number of connections.

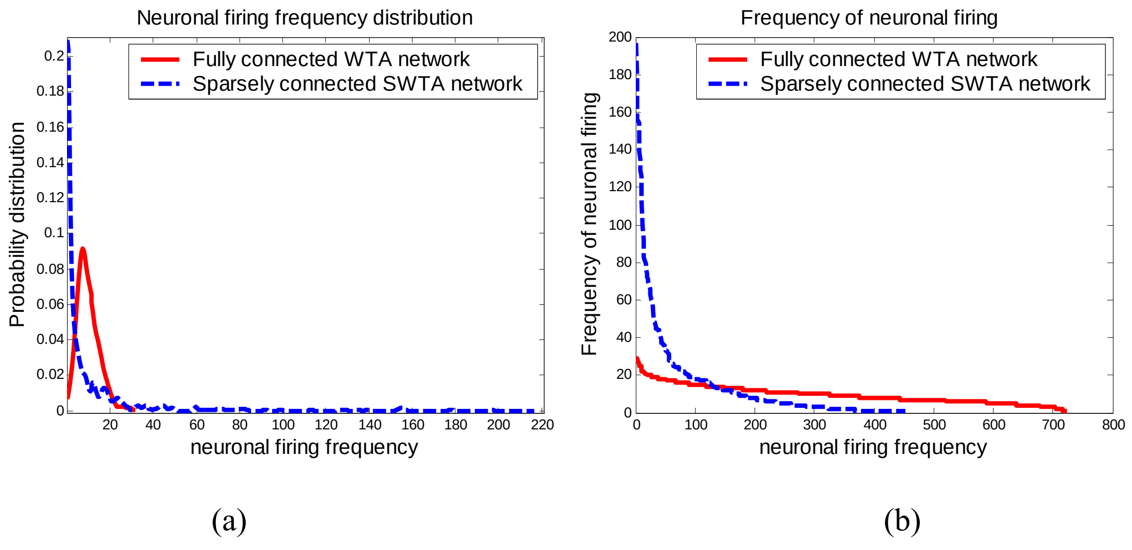


Figure 5.14. Distribution of activities of output neurons of SWTA and WTA.

(N_h=64, α=2.25)

(a). Neuronal activities distribution (b). Frequency of neuron firing

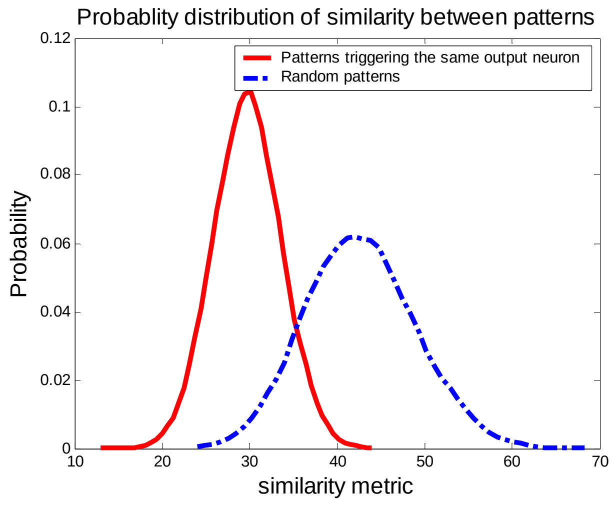


Figure 5.15. Similarities among the patterns that trigger the same output neuron.

The number of input connections to each neuron affects the information transmission from lower levels to higher levels in SWTA. Therefore, it also affects the representational capacity in the sense of the number of neurons on the top level that receive sufficient information to become representations for categories. Assuming that the number of inputs per neuron is estimated by $p \cdot \sqrt[3]{N_h}$, we can find the optimum scaling factor p to maximize the representational capacity of a SWTA network. The representational capacities of a SWTA network as a function of p are shown in Figure 5.16. The optimum representational capacity is obtained when p is approximately 1, which supports the selection on l_{in} in (5.9).

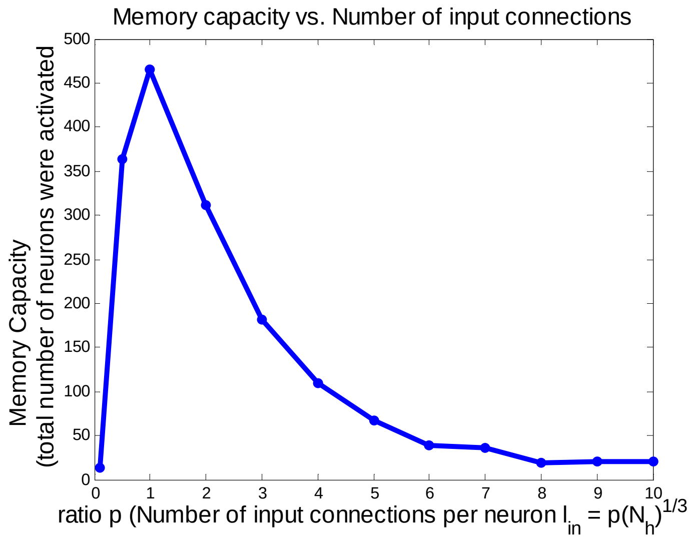
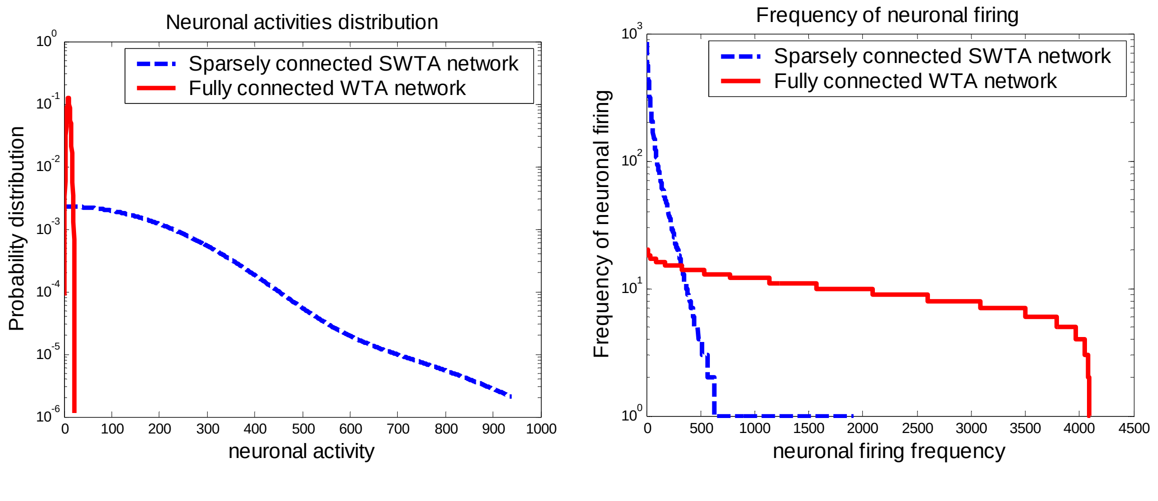


Figure 5.16. Effect of number of input connections per neuron on representational capacity of a SWTA network.

The representational capacity was tested on the network with 64 input neurons and 4096 output neurons described in the example in Section 5.3.4. The 40000 random patterns trigger 1916 different neurons on the top level as global winners at least once. In Figure 5.17, the neuron firing frequency and its distribution are compared with those of the fully connected 64-4096 network, which can activate all 4096 neurons on the top level.



(a) (b)

Figure 5.17. Distribution of activities of output neurons of SWTA and WTA (N_h=64, α=4). (a). Neuronal activities distribution (b). Frequency of neuronal firing

Representational capacity per synapse in this SWTA network, which is calculated by (5.20), is significantly higher (5.7 times as great) than that of a fully connected WTA network.

Typically, sparse connectivity, when we simply reduce the number of connections between two hierarchical levels, may fail to transmit the information sufficiently for higher level neurons to build higher level representations. It produces statistical bias on certain higher-level neurons in the winner-take-all competition. When many of the higher-level neurons always fail in the competition and are not used to store or represent categories, the representational capacity of such sparse network is greatly reduced. However, the secondary levels used in the proposed network provide sufficient interconnections between primary levels so that approximately 50% of the neurons on the higher level can be used for information storage. When the network is scaled up to a size needed for advanced intelligent machines, we can expect that a large number of the higher-level neurons will be used in memory building and the network will possess desirable representational capacity. The loss of representational capacity per neuron is more than compensated by its lower hardware cost expressed by lower total number of connections.

5.4 Hierarchical self-organizing memory with oligarchy-take-all

Encoding the sensory input into only one dominant neuron, as in the network with SWTA, is not very robust. Since the connection weights in the activation pathway are updated after each pattern is learned and the activation pathways may overlap for different patterns, the change in connection weights of one activation pathway will affect another pathway. Accordingly, the signal strength of top level neurons of previously learned patterns may change. Recognition in large WTA networks, whether sparsely or densely connected, is vulnerable to small changes in the activities of the highest level neurons. Moreover, recognition may fail because of noise, faults, and variant representations of the same object. More robust results can be obtained by encoding the sensory input into a group of active neurons, called an "oligarchy" in this work, at the highest level. In the network using the oligarchy-take-all (OTA) algorithm, described below, the winning neurons are found directly in a feed forward process instead of the 3-step procedure used in the SWTA described in Section 5.3.

Distributed representations typified by the oligarchy encoding have several advantages over the localist representations. First, they are more efficient as fewer neurons may code significantly more representations based on the group coding. Second, a new representation can be easily formed by a novel combination of the existing features, while the localist representation requires a new output unit. Third, similar patterns have a number of common units in a distributed representation. Fourth, distributed representation is more accurate since it may code up to 2^n values over n output units while localist represents up to n values. Fifth, sparse representation is more robust, since it has some redundancy. Finally, learning and invariance building is improved as changes in representation are gradual.

5.4.1 Finding the oligarchy

After the neurons on the 1st layer receive the input data, the input activations of post-synaptic neurons are calculated, as in the 1st step of SWTA, by (3.15). Each neuron on the 1st layer finds its set of post-synaptic neurons on the 2nd layer and their signal strengths are compared and the local winner is found. The local competitions in OTA are implemented using the same WTA circuit expressed in (3.16), and branches connected with losing nodes are logically cut off as the signal propagates up the hierarchy. Learning is carried out on the logically connected pathways, and Hebbian learning method in (3.18) is applied. Afterwards, signal strengths of the 2nd level neurons are recalculated and the procedure is continued until the top level of the hierarchy is reached. Only active neurons on each level are able to send information up the hierarchy. Eventually, the activation pathway and the active neurons at the highest level are found through the OTA process. The oligarchy at the highest level, as the most abstract representation of the sensory input, provides redundant, distributed coding of the input pattern. When similar patterns are presented, it is expected that similar groups of neurons will be activated. The OTA algorithm is organized as the following program, shown in Table 5.3.

The properties of oligarchies depend on the topology of their interconnections and the dependence can be studied statistically. By properly choosing the interconnection density for a given network size, we can obtain oligarchies of roughly any desired size. In the following example, we analyze the effect of the number of input connections per neuron on the size of the oligarchy.

Example

Two structures are compared in this test. Each network has two secondary layers per level with 64 neurons on the input level and α in (5.8) as 2.25 or 4. In both networks, the number of active neurons in the oligarchy at the top level, which represents a particular category, decreases with increasing numbers of input connections to each neuron (as shown in Figure 5.18). Therefore, the number of input connections to each neuron can be optimized based on such considerations as the desired coding redundancy and robustness, memory capacity, and implementation cost.

Table 5.3

Oligarchy-Take-All Algorithm

```
FOR Layer = 2: Top Layer
    FOR i=1: number of neurons on (Layer)
      (1). Data transmission (feedforward process)
        $S_i^{layer+1} = \begin{cases} \sum_{j \in N_i^{layer}} w_{ij} S_j^{layer} & \sum_{j \in N_i^{layer}} w_{ij} S_j^{layer} > threshold \\ 0 & \sum_{j \in N_i^{layer}} w_{ij} S_j^{layer} < threshold \end{cases}$   (5.15)
      (2). Finding the winner network (backpropagation process)
        $S_{winner\ i}^{layer+1} = \max_{j \in N_i^{layerl+1}} \left\{ \sum_{k \in N_j^{layer}} w_{jk} S_k^{layer} \right\} \quad (i = 1,2,..N_{layer})$
        $S_i^{layer} = S_{winner\ i}^{layer+1}$   (5.16)
        $l_{ji}^{layer+1} = \begin{cases} 1 & j: local\ winner\ among\ N_i^{layerl+1} \\ 0 & j: not\ \ local\ winner\ among\ N_i^{layerl+1} \end{cases}$
      (3). Learning in the winner network (feedforward process)
        $S_i^{layer+1} = \begin{cases} \sum_{j \in N_i^{layer}} w_{ij} l_{ij}^{layer} S_j^{layer} & \sum_{j \in N_i^{layer}} w_{ij} l_{ij}^{layer} S_j^{layer} > threshold \\ 0 & \sum_{j \in N_i^{layer}} w_{ij} l_{ij}^{layer} S_j^{layer} < threshold \end{cases}$   (5.17)
        $\Delta w_{j,i}^{layer+1} = \lambda_{j,i}^{layer+1} S_i^{layer} S_j^{layer+1} l_{ij}^{layer}$   (5.18)
    ENDFOR
ENDFOR
```

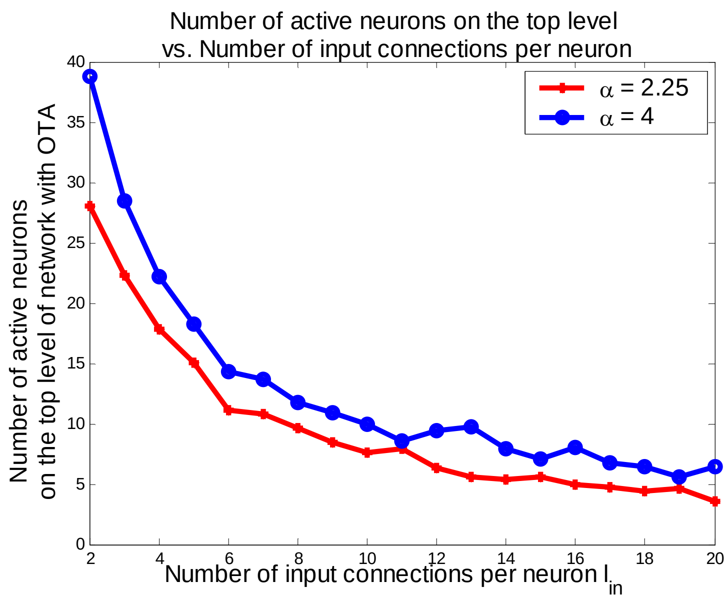


Figure 5.18. Effect of the number of inputs on number of active neurons in OTA.

Oligarchy-take-all idea is similar to k-winners-take-all (kWTA) concept described by [Van83]. However, unlike kWTA, no prior assumption is made about the number of winners on top level. This number is effectively controlled by the network connectivity. Both methods lead to building sparse distributed representations thus are useful for cognitive signal processing. They provide local inhibition that reduces neuron activities in higher layers and yield feedforward pattern competition [Koh 84] [McC 81] [Rum 86b] [Gro 76] [McN 87].

As demonstrated by [Day 95] [Hin 97], balance between a single neuron (localist) representation and distributed representation, requires that a relatively small number of output units are used for representation building. The main objective of our model is to obtain this kind of distributed representation building that is similar to behavior of cortical networks, even if not all the implementation details of our method are biologically plausible.

5.4.2 Grouping active neurons using lateral connections

Even though the representations of patterns are built on groups of neurons so that information is distributed at higher abstraction levels, it has been suggested that information in real brains is not necessarily widely distributed. Therefore, lateral connections are introduced to change the information distribution. Such lateral connections are found at all levels of the visual cortical hierarchy [Sir 95] [Lun 93]. When firing is positively correlated, the lateral connections are expected to amplify the signal strengths of nearby neurons and diminish those of distant neurons. It is noted that, in this work, lateral connections are not used for the purpose of finding winner neurons as in earlier works of [Hay 99] [Xie 02]. They are utilized to change the distribution of neuronal activities so as to cluster the active neurons.

Each neuron is mainly connected to nearby neurons, and has few lateral connections to distant neurons. The connection distribution can follow a Gaussian distribution with a negative offset. Therefore, the neurons have short-range recurrent excitation and long-range recurrent inhibition. The effect of the lateral weights on neuronal activation is made small, by normalizing the lateral weights, so as not to overwhelm the effects of the input signals. For example, we can have $\|W_{YY}\| = \frac{1}{2}\|W_{SY}\|$, where W_{SY} is the mapping from the lower level to y^{th} level neurons, and W_{YY} is the lateral mapping among neurons on the y^{th} level.

Since lateral connections introduce recurrent signals, the neuronal activities are calculated iteratively, the neuronal activities of each time cycle being used to calculate the lateral input of its next time cycle. When the same pattern is presented to the network for learning for *n* cycles, the neuronal activities Y_n are calculated as

$$\begin{aligned}
&Y_0 = S_0 W_{SY} \quad \text{(initial presentation)}\\
&\text{after having the recurrent signals}\\
&Y_1 = Y_0 + Y_0 W_{YY} = S_0 W_{SY}(1+W_{YY}) \quad \text{(after 1st cycle)}\\
&Y_2 = Y_0 + Y_1 W_{YY} = Y_0 + Y_0(1+W_{YY})W_{YY} = S_0 W_{SY}(1+W_{YY}+W_{YY}{}^2) \quad \text{(after 2nd cycle)}\\
&\ldots\\
&Y_n = S_0 W_{SY}(1+W_{YY}+W_{YY}{}^2+\ldots+W_{YY}{}^n) \quad \text{(after nth cycle)}
\end{aligned}, \quad (5.21)$$

where S_0 represents the input from lower level neurons, W_{SY} is the mapping from lower level to y^{th} level neurons, and W_{YY} is the lateral mapping among neurons on the y^{th} level. Let W_H denote the sum in (5.21)

$$W_H = 1 + W_{YY} + W_{YY}{}^2 + \ldots + W_{YY}{}^n . \quad (5.22)$$

W_{YY} can be diagonalized as

$$W_{YY} = V^T D V , \quad (5.23)$$

where V is the eigenvector matrix and D denotes the eigenvalues,

$$D = \begin{pmatrix} \lambda_1 & & 0 \\ & \ldots & \\ 0 & & \lambda_k \end{pmatrix} = diag(\lambda_k)(\lambda_k < 1 \; for \; W_{YY}). \quad (5.24)$$

W_H in (5.22) can thus be expressed as,

$$W_H = \sum_{i=0}^{n} W_{YY}{}^i = \sum_{i=0}^{n} V^T D^i V = V^T \left(\sum_{i=0}^{n} diag(\lambda_k^i) \right) V \quad (5.25)$$

If W_{YY} have eigenvalues λ_k within the unit circle ($\lambda_k < 1$), W_H can converge to a stable value and then Y_n , as the network output after n cycles, can have stable response as,

$$Y_n = S_0 W_{SY} W_H == S_0 W_{SY} V^T \left(diag\left(\frac{1}{1-\lambda_k} \right) \right) V \tag{5.26}$$

The number of lateral connections and the standard deviation of the connection distribution affect the grouping effect of active neurons on every layer including the distribution of the active neurons on the top level. The number of lateral connections for neurons on a certain level and the standard deviation of the lateral connection distribution can be related to the overall number of neurons on that level, as shown in (5.27), where a and b are arbitrary constants.

$$\begin{aligned} l_{lateral}^h &= a \cdot \frac{N_h}{N_{in}} \\ \sigma_{lateral}^h &= b \cdot \frac{N_h}{N_{in}} \end{aligned} \tag{5.27}$$

The clustering effects can be evaluated by the clustering index (CI) [Rip 81]. The CI measures the nearest-neighbor distance between two active neurons within a measurement window, divided by the average distance between a random point in the window and its nearest active neuron [Gro 00]. It is expected that a more clustered distribution will give a higher CI. The CIs of OTA output neurons with and without the lateral connections are compared for various a values in (5.27) while b is fixed. It is noted that in Figure 5.19 that CIs are improved after the grouping effect of the lateral connections. The factor a affected the grouping effect and the optimum clustering can be obtained when using a as 10.

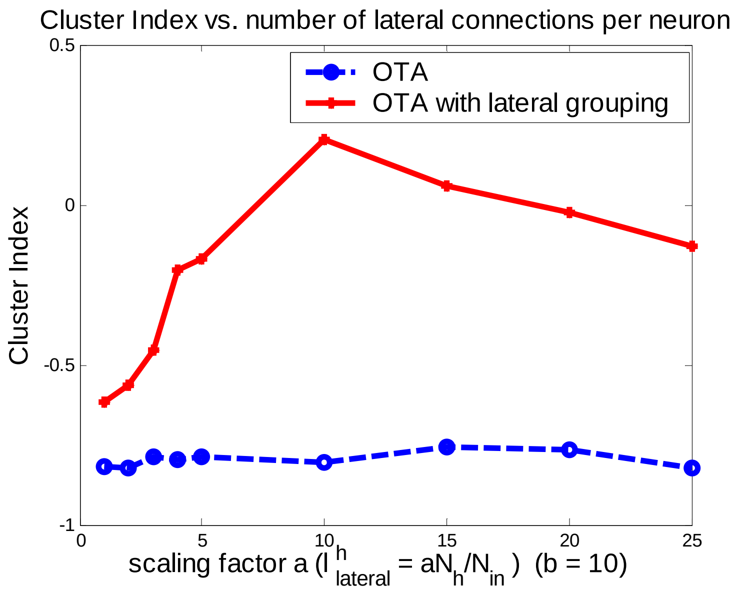


Figure 5.19. Effect of number of lateral connections per neuron on Cluster indices.

5.4.3 Finding a group of winners in the network with OTA

In a network with the OTA algorithm, each pattern is represented by an oligarchy, as a group of neurons, at the highest level, so that information coding is redundant. When a similar pattern is presented to the network, similar groups of neurons are expected to fire. Recognition depends on a determination of the similarities of the oligarchy of a presented pattern to those of previously learned patterns.

In order to show the advantage of this distributed, redundant representation, and demonstrate the learning ability of a network with OTA, a set of handwritten digits from a benchmark database [LeC] was used for training a sparse network with 64 input neurons and 729 output neurons. All patterns have 8 by 8 grey pixel inputs, as shown in Figure 5.20.

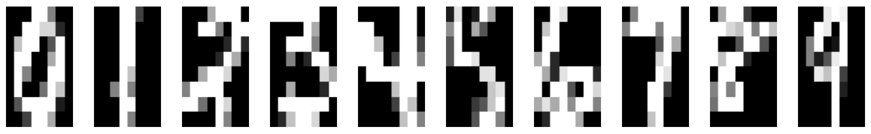
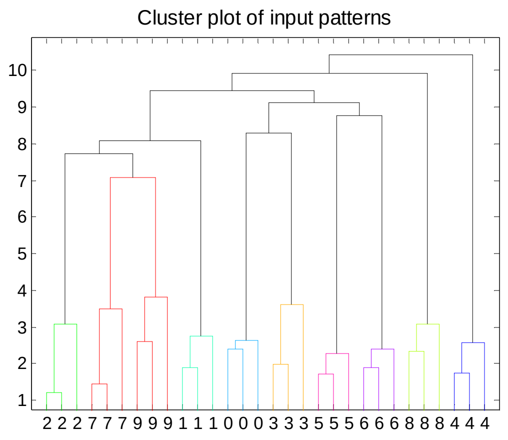


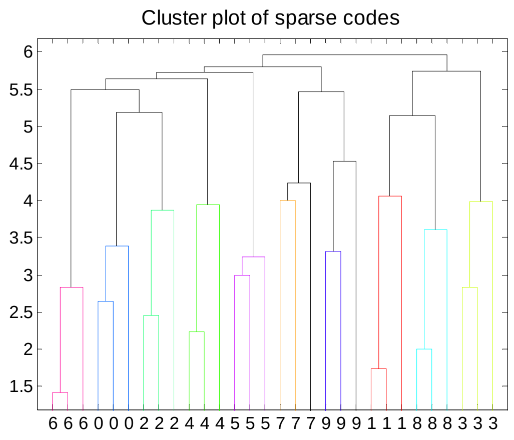
Figure 5.20. Ten handwritten digit patterns.

The number of neurons in the oligarchy activated by each pattern is not pre-determined since the learning process is self-organizing and unsupervised. Based on the structure of this network and the connectivity proposed in this work, each pattern activates 19.1 out of 729 neurons at the highest level (range 14 to 23) on average. These sets of neuronal indices become the representing markers of these 10 different digits. In the recognition process, the class to which an input pattern is assigned is determined by the number of neurons in the resulting oligarchy that are common to each of the marked oligarchies.

It is expected that the sparse network with OTA can recognize a learned pattern with some level of noise tolerance. Similar patterns which are originated from the same digit but with different random noise are desired to produce similar sparse codes on the top level of OTA network. The similarity can be measured by the Euclidean distance between patterns. For example, the cluster plot of ten digits with two groups of noisy patterns is shown in Figure 5.21(a). It is shown that the noisy versions of the same digit are clustered together. Similar clustering can be found on the sparse codes produced by OTA as shown in Figure 5.21(b).



(a)



(b)

Figure 5.21. Cluster plots of input patterns and their sparse codes found by OTA. (a). Cluster plot of input patterns (b). Cluster plot of sparse codes

5.4.4 Representational memory capacity of the network with OTA

As we demonstrated in Section 5.3.7, per synapse representational capacity of network with SWTA is higher than that of a fully connected traditional WTA network. A WTA network with N_{out} output neurons and N_{in} input neurons has a per synapse representational capacity equal to $1/N_{in}$. As the WTA assigns a signal output neuron to each class of training data, WTA representational capacity is both low and intolerant of much output error.

The OTA algorithm permits coding that is error-tolerant. The representational capacity depends on the number of neurons used in the OTA code and the amount of overlap between the codes. The number of neurons in an oligarchy can be controlled by the input connections per neurons, as shown in section 5.4.1. So it must be carefully chosen. In the following example, we select a simple coding scheme to support our claims of high capacity and error tolerance of the OTA.

Let us assume that N_{out} neurons are divided into d disjoint groups of neurons, where each group has $m = \frac{N_{out}}{d}$ neurons. Let us use k such disjoint groups of neurons to code one category of input objects. Thus among d disjoint groups, we can distinguish $C_k^d = \binom{d}{k} = \frac{d!}{k!(d-k)!}$ distinct categories. In addition, each group can tolerate a certain level of different neurons, for instance, $\frac{m}{2} - 1$ neurons, and still be identified as the correct category. Therefore, the entire code may tolerate up to $\left(\frac{m}{2} - 1\right)k$ errors provided that no single group has more than $\frac{m}{2} - 1$ errors.

For example, if we choose N_{out} = 4096, and m =15, we will get approximately 273 disjoint groups of neurons. If we use k =10 disjoint groups to code one category, and each group could tolerate up to 5 errors (50 errors total per code) to be correctly recognized, we will have $C_{10}^{273} = \begin{pmatrix} 273 \\ 10 \end{pmatrix} \cong 5.36 \times 10^{17}$ distinct codes. Memories of this capacity are not yet demonstrated by any neural network model or a learning method. They could explain how human memory can store and quickly retrieve extraordinary amounts of information. This example assumes that the OTA network could activate the required number of independent groups of neurons. As was observed in the SWTA network (approximately 50% of output neurons were activated), this may not be the case. However, even much smaller activation of the output groups in an OTA will yield a large storage capacity with a significant tolerance to errors. In the work presented in [Abb 96], the representational capacity of face coding in monkeys is investigated by recording 14 neurons in the superior temporal sulcus of monkeys in response to 20 images of faces. Based on the experimental data and Monte-Carlo simulation it was presented in [Abb 96] that, for distributed coding, the representational capacity grows exponentially and the information grows linearly with the number of coding neurons. In addition, the number of distinct categories that can be represented with a 50% discrimination accuracy by N neurons is approximately $3(2^{0.4N})$. The OTA algorithm is the approach that can possibly make a sparsely-connected network achieve such level of representational capacity. However, the practical representation capacity of OTA depends on the selection of optimum connectivity, coding scheme, and specific application data set, which is beyond the scope of this work and requires a separate study.

As shown in [Wil 69], sparse codes increase the memory capacity of associative memories. Since a missing part of the code could be recovered, or a noise in the code removed by the associative memory, sparse coding network can be combined with an associative memory to increase the robustness of the sparse coding. For example, the OTA structure can be combined with the original R-net organization, which is a type of associative memory, to improve robustness of the code. R-nets were demonstrated to have large storage capacities for sparsely coded memories. In addition, they have a significant error tolerance and a mechanism that can recover distorted information. As was demonstrated in [Sta 05], when the R-network size reaches 10^9 primary neurons (with an average of 10^4 projections per neuron that is similar to the interconnection density of human brain), the network can store over 10^9 patterns and the optimum storage for these memories is achieved with a pattern size of about 150 neurons. With the recovered information from R-net, the robustness of the OTA coding will be greatly improved.

5.5 Comparing SWTA and OTA

5.5.1 Efficiency of SWTA and OTA

In Section 5.3.5, a sparse network with SWTA is compared with a fully-connected two-layered network with global WTA in terms of the computation and design cost. Here we will compare efficiency of the proposed SWTA and OTA algorithms with a popular self-organizing learning based on WTA. A Kohonen self-organizing map (SOM), which is a common fully-connected competitive network with global WTA, is implemented using MATLAB Neural Networks toolbox. Both types of simulated networks have the same input (e.g. 64 pixels) and output sizes. Their corresponding number of output neurons is changed from 10 to 400. All networks will be trained using the same set of digits for 1 epoch. The simulation times are plotted and compared in Figure 5.22.

As shown in the figure, the simulation time of SOM grows rapidly with the increased network size. The efficiency of SWTA is slightly lower than OTA, since it requires three transmissions of the signals in order to find out the winner neuron. The sparse networks with both SWTA and OTA have greate advantage over SOM especially for large networks.

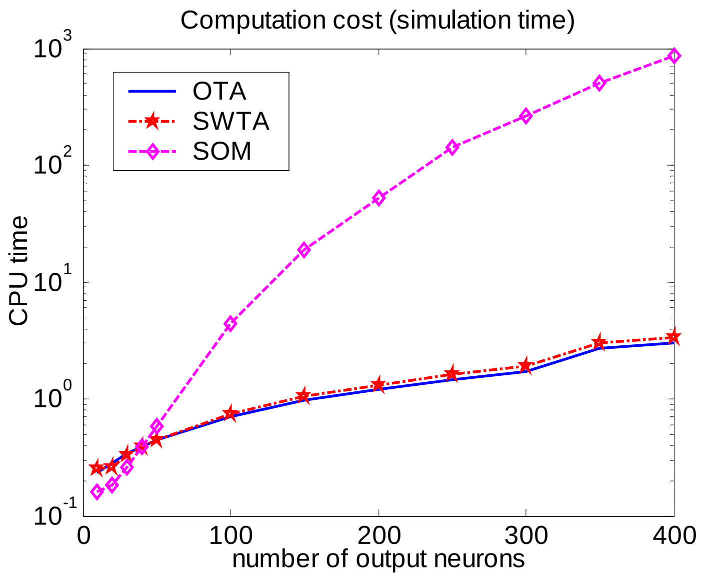


Figure 5.22. Computation cost comparison.

5.5.2 Fault tolerance of SWTA and OTA

It is expected that introducing more noise into the original patterns will degrade recognition performance. However, the tolerance of the network with OTA for such change is expected to be better than that of the SWTA. Figure 5.23 compares the performances of the SWTA and the OTA for different numbers of changed bits in the training patterns based on 10 Monte-Carlo trials. Both sparse networks contain 64 input neurons and 729 output neurons and are trained for 300 epochs. We note that increasing the number of changed bits in the patterns quickly degrades the SWTA's performance on this recognition task. However, the network with OTA has much better fault tolerance.

For reference, a Kohonen self-organizing map (SOM), which has the 2-layered fully connected structure with 64 inputs and 729 outputs, is implemented using MATLAB Neural Networks toolbox. The recognition performances of SOM is evaluated and compared with SWTA and OTA representations in Figure 5.23. The SOM are trained for 300 learning epochs and its performance is averaged from 10 Monte-Carlo runs. It is demonstrated that the OTA's distributed coding scheme have the best fault tolerance over SOM and SWTA. The SOM presents better noise tolerance than SWTA, since its full connectivity can transmit more information up to the top level, but it is still outperformed by OTA.

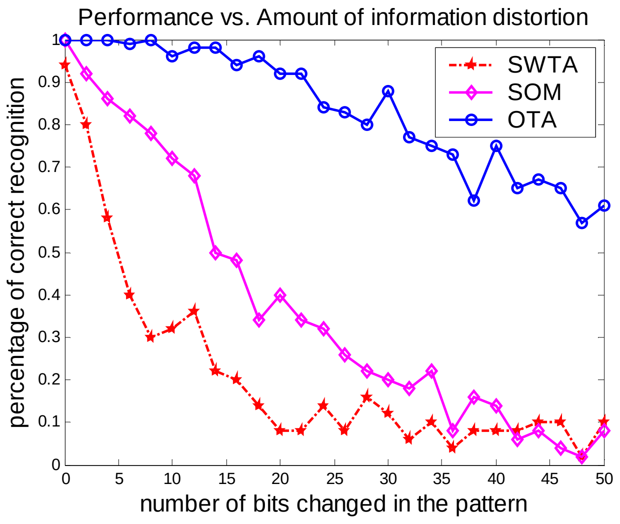


Figure 5.23. Recognition performance comparison.

5.5.3 Tolerance to loss of neurons of SWTA and OTA

Due to the expanding structure of the HSOM and the distributed codes built by OTA, the network with OTA can tolerate a large loss of neurons. Such tolerance of an OTA structure is evaluated and compared with SWTA and SOM networks. The tested OTA network, SWTA network, and SOM have the same structures as used in Section 5.5.2. Ten grey-scaled pixel handwritten patterns organized in 8 by 8 arrays, shown in Figure 5.20, are applied to all these networks for training. In the testing phase, the same group of patterns is applied to various versions of these networks, with the same structures but with a different number of missing neurons in the input pattern. The number of missing neurons varies from 2 to 20 percent of the total number of neurons in each network. The locations of the missing neurons are chosen randomly.

The performance is obtained based on 10 Monte-Carlo runs. As shown in Figure 5.24, with small numbers of missing neurons (e.g., 8 percent of the overall number of neurons in the specified OTA network), the OTA network can still correctly recognize the patterns. With more neurons missing from the network, accuracy gradually decreases. When 20 percent of the neurons are missing, 84% of patterns are still correctly classified in OTA networks. The performance of the SWTA in this test is much lower than that of the OTA. The specified SOM show tolerance to the loss of neurons, but are not as good as the OTA. Therefore, it is observed that the distributed codes generated by the OTA are robust to the loss of neurons.

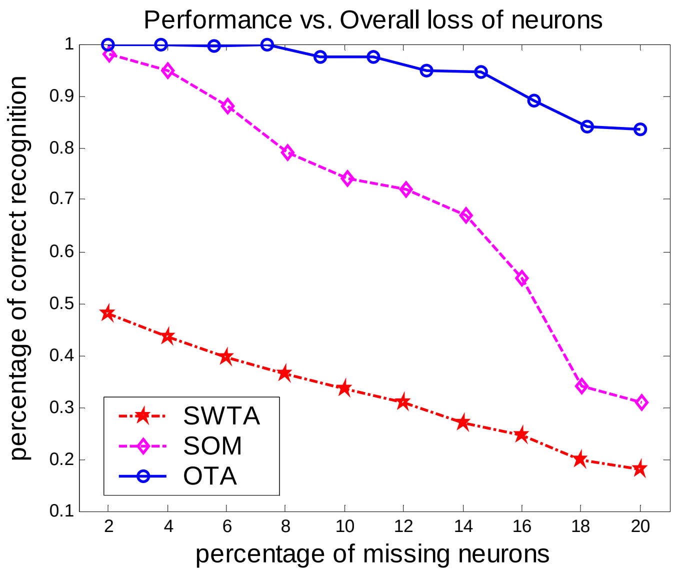


Figure 5.24. Tolerance to loss of neurons of OTA.

5.6 Correlation-based information grouping in OTA

In the proposed organization of the sensory pathways using WTA or OTA algorithms, the number of incoming connections that each neuron connects has is determined and is correlated to the input level size to maximize the network's representational capacity. Positions of these pre-synaptic neurons on the previous layer are randomly selected, which means that each neuron does not have a local receptive field or a self-organized receptive field.

Local receptive fields for neurons are adopted in many earlier works to preserve the input topology, extract features and group information in the learning networks. Such local receptive fields were successfully used for pattern recognition in neural network schemes such as convolutional neural networks LeNet [LeC 89], Linsker model [Lin 86] and pyramidal neural network [Phu 07]. In these earlier works, the networks have local receptive fields with predefined and fixed size through all the layers in the network. While Linsker demonstrated that such an approach produced self-organization and local features (starting with on-center off-surround and off-center on-surround cells on the lower layer), the receptive field is always local over all the layers and the location and size of the local region had to be arbitrarily decided.

In the sensory pathway, it is expected that local correlations dominate on the lower levels of hierarchy. For instance, in the input image, neighboring pixels are more correlated than distant pixels. On higher levels where features are extracted, it is expected that strength of local correlations will diminish and the neurons representing features are active simultaneously responding to the input patterns.

Therefore, we propose to determine the location of the receptive field statistically by checking correlations among various neuron activations using a large set of training data and setting the wiring threshold to have a desired number of the input links to each neuron.

5.6.1. Calculating correlation

The correlation between each two locations should represent similarity of their signal activation strengths. The correlation should be 0 for uncorrelated variables while it should be 1 for the most correlated variables. On the input level, input neurons receive gray-scaled signals in the range of 0 to 255. In order to find the similarity, we can encode the grey scales p_j into angles in the range of 0 to $\pi/2$, as in (5.28). The cosine of the difference between two angles gives the similarity C_{mn} between these two grey scales p_m and p_n, as in (5.29), where N_{in} is the number of the input neurons.

$$p_j = p_j \cdot \frac{\pi/2}{255} \quad (i = 1,2,...N_{in}) \tag{5.28}$$

$$C_{mn} = \cos(p_m - p_n) \tag{5.29}$$

On higher levels, the signals of neurons are in the range of (-1, 1), or specified otherwise as in HSOM. Then the signals are scaled into the range of (0, $\pi/2$) before correlation (5.29) is applied.

The procedure to determine the correlation and the size of the local receptive field is described as follows.

Step 1). Collect a large number (N) of images as the neural network input.

Step 2). Each image is represented by N_x by N_y matrix P_i $(i=1, 2, \ldots N)$, where N_{in} equals to N_xN_y. Start with the first image and its representation matrix P_1.

Step 3). Use (5.29) to determine the correlation matrix $C_{in} \in \Re^{N_{in} \times N_{in}}$ for the current image.

Step 4). Update the total correlation matrix C_T and total count N_t, as in (5.30).

$$\begin{aligned} C_T &= \frac{N_t C_T + C_{in}}{N_t + 1} \\ N_t &= N_t + 1 \end{aligned} \tag{5.30}$$

Step 5). Repeat Step 3)- Step 5) until all images are process.

Step 6). Each row of the resulting C_T contains a vector of single pixel's correlations to all other pixels over all the images.

Step 7). Reshape each row to the image size in order to determine the size of local correlation region.

A similar process can be repeated on the higher levels except that each row of the C_T gives a single neuron's correlation to all the other neurons on a certain level of HSOM after the images are processed by the lower levels.

For example, consider a network structure with 320 input neurons, which can perceive images with the size 20 by 16. The following calculation and discussion will be based on this structure. The 20 by 16 face image from [Fac] is shown in Figure 5.25.

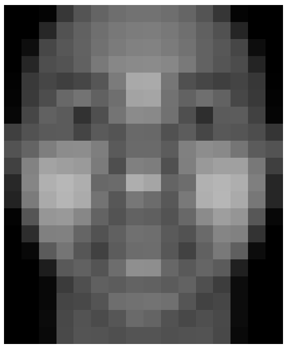


Figure 5.25. Input face image.

By observation, on the input level, the 1st pixel (which is dark) is expected to be closely correlated with the dark area of the picture. Using the procedure above, the correlation of the 1st pixel to the whole picture is shown in Figure 5.26, in which the location of the 1st pixel is marked as star. It is noted that high correlation exists in the expected areas of the input image. One of the pixels on the right cheek area correlates to the whole picture as in Figure 5.27.

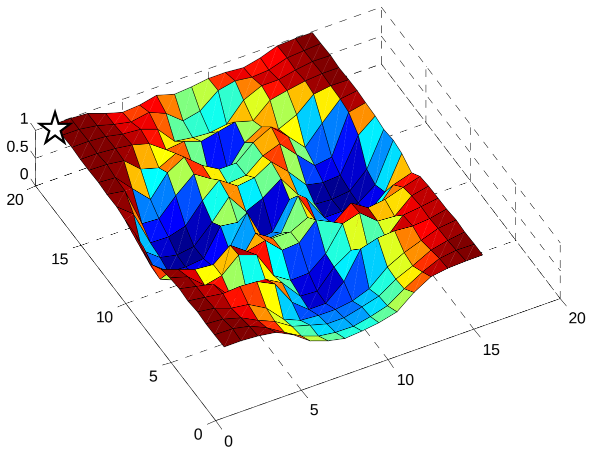


Figure 5.26. The 1st pixel's correlation to the whole image.

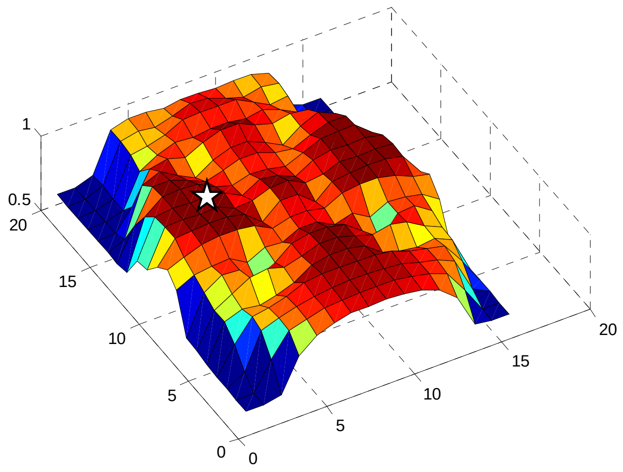


Figure 5.27. The 70th pixel's correlation to the whole image.

If we have a series of images with the same size, as shown in Figure 5.28, the correlation matrices from all pictures should be averaged. The 1st pixel and the 70th pixel's correlation matrices are shown in Figure 5.29. Eventually, after presenting the network with 1000 20x16 input images of all types of patterns, the correlation matrices for the input level are obtained. Several of them are shown in Figure 5.30.

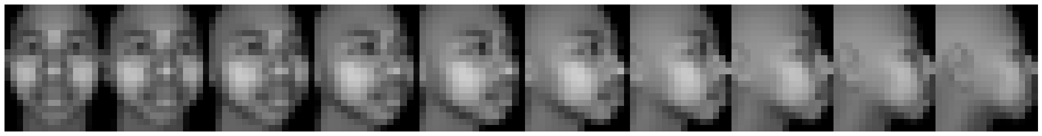
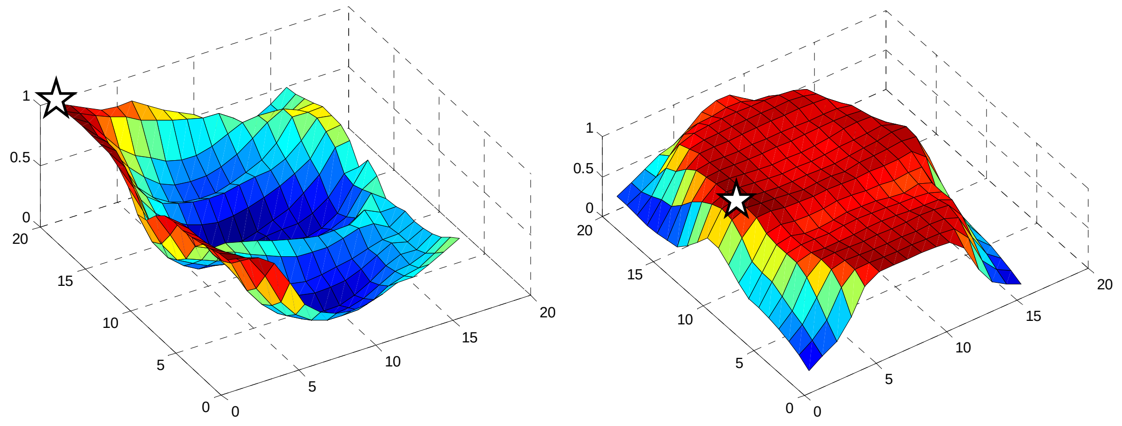


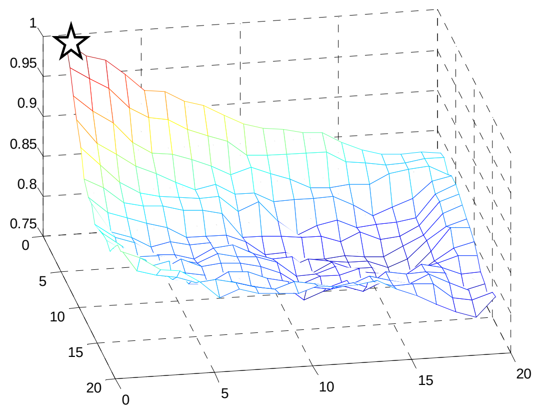
Figure 5.28. A series of face images.



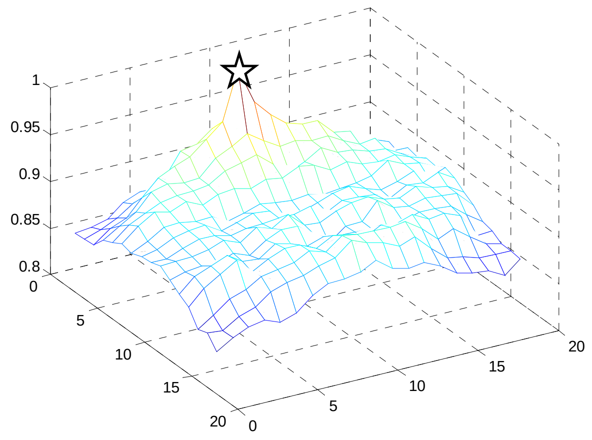
(a) (b)

Figure 5.29. Correlation calculated based on a series of face images.

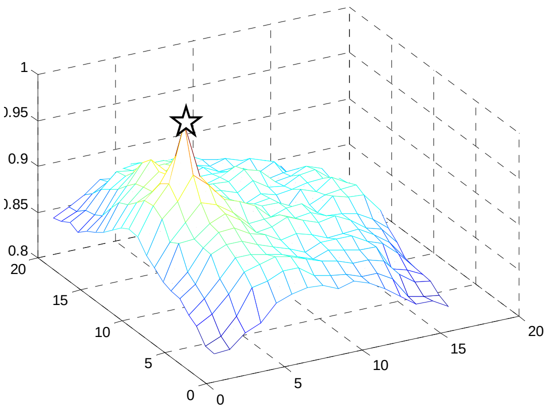
(a). The 1st pixel's correlation to the whole image.
(b). The 70th pixel's correlation to the whole image.



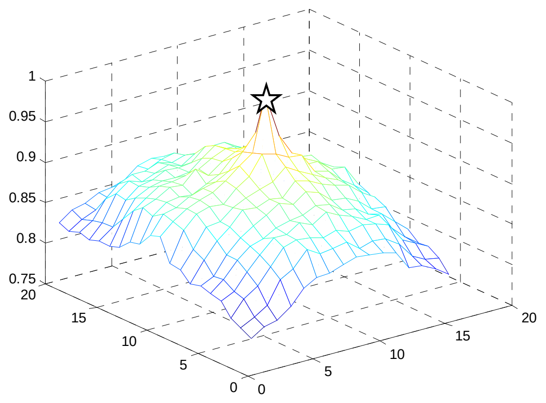
(a)



(b)



(c)



(d)

Figure 5.30. Correlation calculated based on 1000 images.

(a). The 1st pixel's correlation to the whole image
(b). The 161st pixel's correlation to the whole image
(c). The 70th pixel's correlation to the whole image
(b). The 170st pixel's correlation to the whole image

5.6.2. Determining receptive field

The input level of the network receives the whole image. Once again consider the network containing 320 input neurons and overall 4 layers (the input level, 2 secondary layers and the output level), with α in (5.8) equal to 1.21 (giving image representation 1.1 times larger than the previous level in each dimension for two dimensional array). This yields a 320-396-500-594 multi-layered network. Before the correlations among neurons on the 2nd layer are calculated, the receptive field of neurons on the 2nd layer should be determined so that they can be activated by the incoming information.

As shown in Figure 5.31, a neuron on the 2nd layer is projected onto the plane of 1st layer and finds the closest neuron to its projection. This closes neuron is denoted as $N^1_{projection}$. Based on the correlation calculation on the previous layer, the most correlated region of $N^1_{projection}$ can be found. The number of incoming connections has been chosen as in (5.9). Thus, the neuron on the 2nd layer is connected to l^2_{in} neurons on the previous layer within this correlated region. In the specified network structure, each neuron has 7 pre-synaptic neurons, and they are illustrated in Figure 5.31. The same process applies to all the neurons on the 2nd layer in finding their corresponding receptive fields.

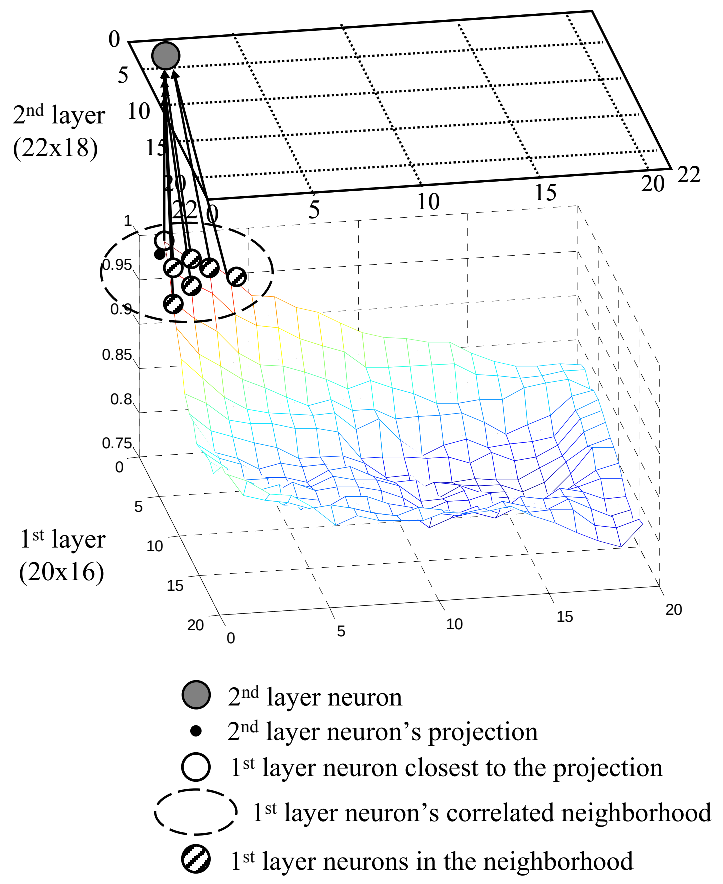

0
5
10
15
20
22
2nd layer (22x18)
0
5
10
15
20
22
1
0.95
0.9
0.85
0.8
0.75
0
5
10
15
20
1st layer (20x16)
0
5
10
15
20
2nd layer neuron
2nd layer neuron's projection
1st layer neuron closest to the projection
1st layer neuron's correlated neighborhood
1st layer neurons in the neighborhood

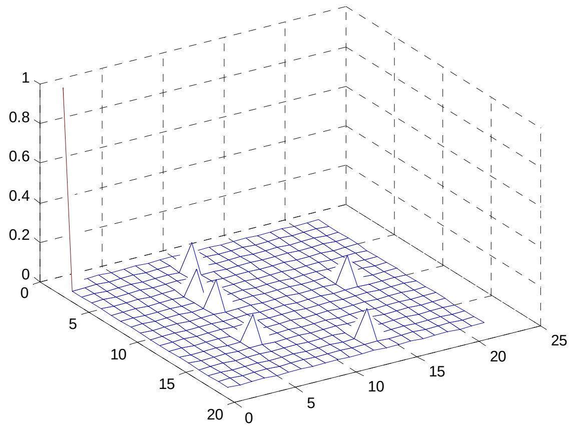
Figure 5.31. The receptive field of a 2nd layer neuron.

The algorithm of OTA is used to calculate the signal responses on the 2nd layer. Then over all the input patterns, the statistical correlations among the activation levels of the neurons are obtained. Due to the local competitions applied when the signals go to the 2nd layer, some of the neurons on the 2nd layer may not be activated at all although many patterns are applied. A silent neuron's output should have no correlation with any other active neurons or silent neurons.

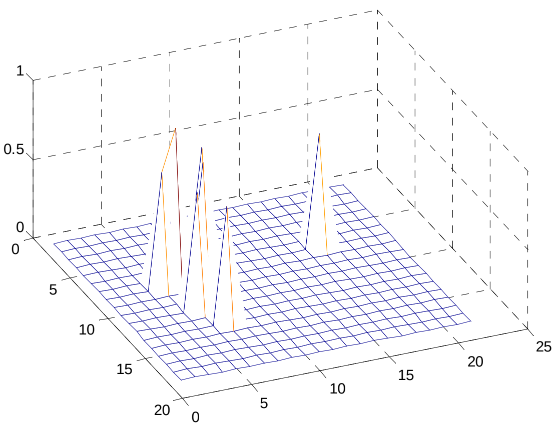
If a neuron is always silent during this process, the correlation of this neuron's location to all the other locations will be zero. Such neurons could be assigned a wiring pattern that averages these of its neighbors.

The correlations among 2nd layer neurons are calculated and l_{in}^{3} most correlated neurons of the 1st and the 160th neurons are shown in Figure 5.32.

It is noted that instead of obtaining the local region as in the first layer, each neuron is more correlated to distant neurons which may represent relevant features. Then a neuron on the 3rd layer is connected to its projection neuron on the 2nd layer and the projection neurons' most correlated neurons, as shown in Figure 5.33.



(a)



(b)

Figure 5.32. Correlations on the 2nd layer.

(a). The 1st neuron's correlation to all the 2nd layer neurons
(b). The 160th neuron's correlation to all the 2nd layer neurons

The correlations among neurons on the higher levels are obtained and neurons on each layer are connected to previous layer in the same way. Notice that using the algorithm in Section 5.6.1, the statistical correlations can be calculated in an on-line fashion. The network can start learning as a randomly-connected structure. After each new pattern is perceived and learned, the correlations among neurons on each layer can be updated and the connections can be updated as well leading to dynamical changes in the network topology.

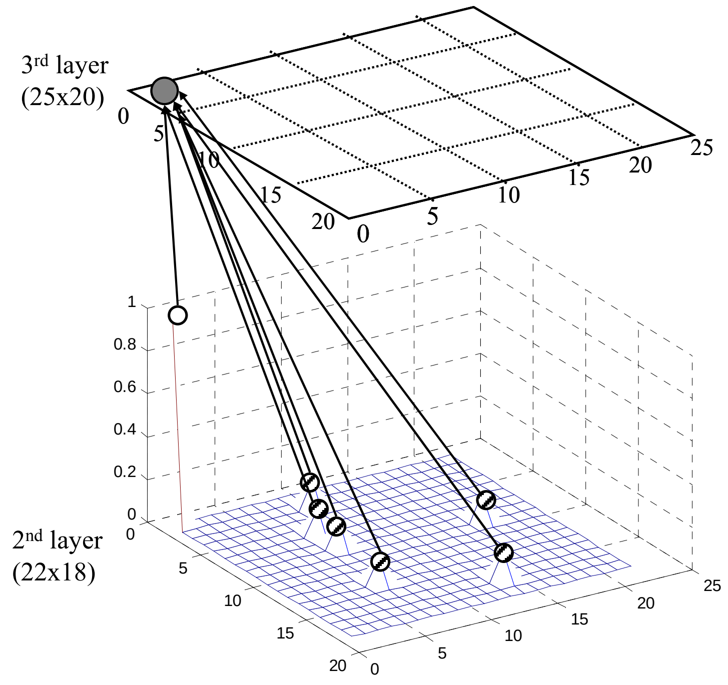


Figure 5.33. The receptive field of a 3rd layer neuron.

5.6.3. Information grouping using OTA

With the correlation-based receptive field, the information is processed and grouped by the network with OTA and the corresponding representations are built. For example, the image shown in Figure 5.34 can activate 36 neurons on the top of a 320-396-500-594 multi-layered sparsely-connected network as its representation. It was observed in the experiment that the scaling of the network with correlated receptive fields does not affect the representations. For example, the intensity-scaled picture shown in Figure 5.34 activates exactly the same group of neurons on the top level, which shows that the OTA network with correlated receptive fields builds representations invariant to linear scaling of stimuli intensity.

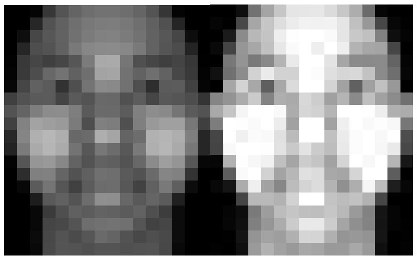


Figure 5.34. An image and its intensity-scaled version.

It is desired and expected that the correlated receptive fields will not affect the representational capacity and the fault tolerance that OTA network has with the random receptive fields. It was observed that the group of 10 patterns shown in Figure 5.20 can activate, on average, 37 neurons on the top level of the network with random receptive fields. The same group of patterns activates on average 34 neurons in the network with correlated receptive fields. Thus we believe that the networks with these two types of receptive fields have similar levels of representational capacity.

In order to show the noise tolerance of the network with correlated receptive fields, the recognition performances are tested for different amount of noise added to the given patterns based on 10 Monte-Carlo trials. The results shown in Figure 5.35 compare the noise tolerance of the networks with random and correlated receptive fields. We note that the correlated receptive fields do not degrade the noise tolerance in the OTA.

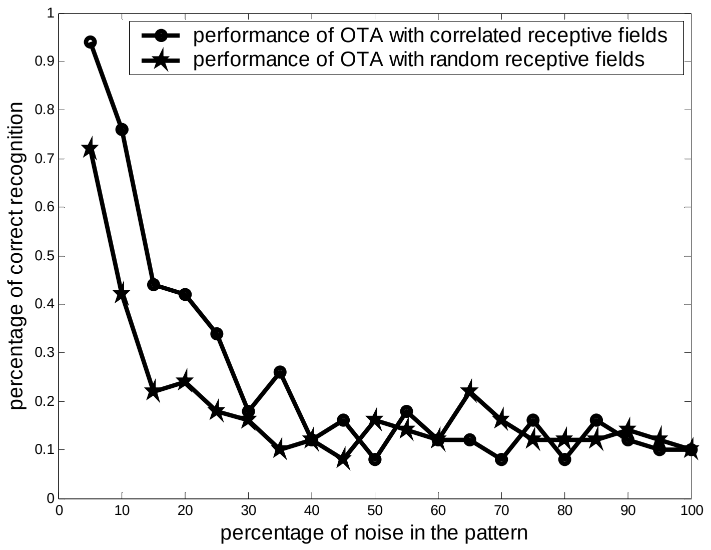


Figure 5.35. Noise tolerance of OTA with correlated receptive fields.

5.7 Attention-aided perception in OTA

It has been investigated that attentions can affect the visual perception. Visual attention, as cognitive control over perception, can be deployed in typically two ways: space-based and object-based [Yan 03], based on the type of the representation on which the selection is carried out. Space-based attention is applied in a particular location of the visual field, while object-based attention is directed to perceptual objects. Both types of attentions are separately supported by various studies, and the study in [Sot 04] found that two types of attentions work in an interactive way.

The behavioral and neurophysiological studies have shown that selective attention is often deployed in an object-based fashion [Roe 98]. When several objects are in the visual scene simultaneously, the attention helps recognizing the attended object while ignoring the rest of the present objects. The mechanism of the top-down attentional based competition was postulated in several earlier works. One candidate for such mechanism was presented in [Ste 00]. It was described as the top-down feedback signal that synchronizes the activity of target neurons that represented the attended object. Then this synchrony increased the efficacy of that neuronal representation at the next perceptual stage.

In this section, a mechanism is proposed to demonstrate how the object-based attention affects the perception in HSOM with the OTA algorithm. As discussed earlier, during the learning, the input patterns are processed in the HSOM with OTA algorithm and are represented as a group of winner neurons on the top level of the network. In the learning stage, each pattern forms a particular group of winner neurons on the top hierarchical level and these particular groups of neurons can be called the pattern's neuronal marker. When a new input comes into the network the winner neuron group is compared with existing markers obtained from the stage of learning and the most similar one is found as the category that it belongs to. When the attention is paid on a particular previously memorized object, in the network, the attention signal is applied on that object's winner neurons to active them so that to synchronize the attended neuronal marker with the currently activated marker. As suggested in [Ste 00], the synchrony will help the attended object to win the competition in the next stage of perception. Hence, the attention-aided perception by object-based attention is a process described as follows.

Step 1). Learning a group of objects

Objects (C_1, C_2, C_3…) are perceived and memorized. Every object is represented by a neuronal marker, denoted as (N_1, N_2, N_3…..).

Step 2). Finding Oligarchy

A pattern containing multiple perceptual objects is presented to the network. A group of winner neurons, N_{win}, are found by OTA algorithm for this input pattern.

Step 3). Applying attention signal

An attention signal is applied on the marker, N_{att}, of the attended object C_{att}. ($C_{att} \in \{C_1, C_2, C_3 ...\}$). This makes the active neurons on the top level include {N_{win}, N_{att}} and have the same level of activation.

Step 4). Finding the winner network (back-propagation process)

```
FOR Layer = Top Layer-1: -1: 1
    FOR i=1: number of neurons on (Layer)
        $S_{winner\ i}^{layer+1} = \max_{j \in N_i^{layerl+1}} \left\{ \sum_{k \in N_j^{layer}} w_{jk} S_k^{layer} \right\} \quad (i = 1,2,..N_{layer})$
        $S_i^{layer} = S_{winner\ i}^{layer+1}$
        $l_{ji}^{layer+1} = \begin{cases} 1 & j: local\ winner\ among\ N_i^{layerl+1} \\ 0 & j: not\ \ local\ winner\ among\ N_i^{layerl+1} \end{cases}$
    ENDFOR
ENDFOR
```

Step 5). Finding final oligarchy N_{final} (feedforward process)

```
FOR Layer = 2: Top Layer
    FOR i=1: number of neurons on (Layer)
        $S_i^{layer+1} = \begin{cases} \sum_{j \in N_i^{layer}} w_{ij} l_{ij}^{layer} S_j^{layer} & \sum_{j \in N_i^{layer}} w_{ij} l_{ij}^{layer} S_j^{layer} > threshold \\ 0 & \sum_{j \in N_i^{layer}} w_{ij} l_{ij}^{layer} S_j^{layer} < threshold \end{cases}$
        $\Delta w_{j,i}^{layer+1} = \lambda_{j,i}^{layer+1} S_i^{layer} S_j^{layer+1} l_{ij}^{layer}$
    ENDFOR
ENDFOR
```

Step 6). The final marker N_{final} is compared with (N1, N2, N3…..) to determine its category.

This experiment is conducted using a grayscale image of spatially superimposed patterns. Since these two patterns occupy the same spatial region, the only method to control the perception (internal representation) is by object-based attention. Firstly, the patterns, including a face image and a house image with 20 by 16 pixels, are learned by the network. The face and the house images are shown in Figure 5.36. These two objects are learned and represented in OTA network using neuronal marker $N_{facee,}$ N_{house}, respectively. A test with these two images superimposed is shown in Figure 5.37.

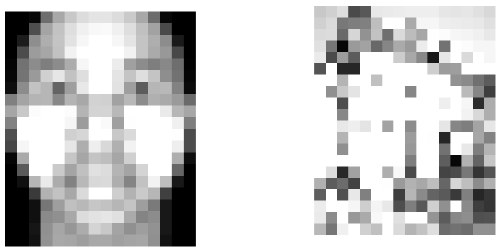


Figure 5.36. The learned face and house images.

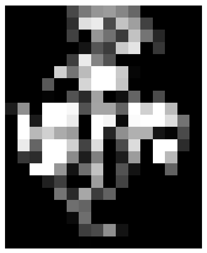


Figure 5.37. The superimposed image.

When the superimposed image is processed using OTA without the affect of object-based attention, it is represented by N_{win} and N_{win} is compared with the face and house markers $N_{facee,}$ N_{house} for recognition. The numbers of common neurons between N_{win} and $N_{facee,}$ N_{house} are shown in Figure 5.38. Therefore, the superimposed image is recognized more as a face object than a house object.

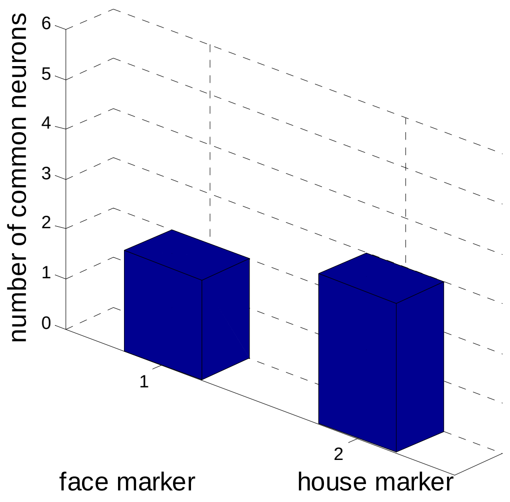


Figure 5.38. The number of common neurons to the markers before the attention on house.

After the attention is applied to the house, the test marker is compared with the markers and the numbers of common neurons with them are shown in Figure 5.39. It was shown that the perception is altered by the applied attention and the superimposed image is identified as a house pattern.

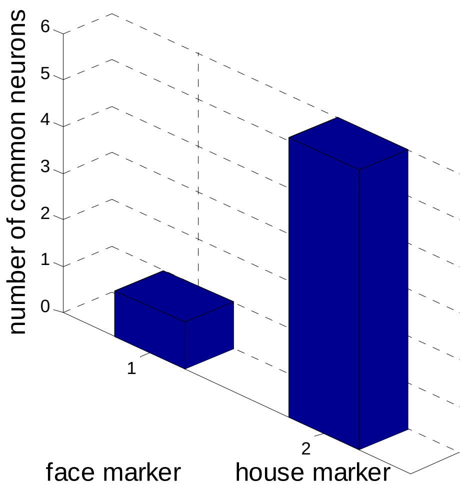


Figure 5.39. The number of common neurons to the markers after the attention on house.

Similarly, if the attention is applied on the face object, the perception after the attention in terms of common neurons to the markers is shown in Figure 5.40. And it shows that the attention makes the superimposed image identified as a face.

A similar procedure can be used for invariant object recognition building through the continuous observation. In human visual perception, within a certain period of time when an object exists in the visual field, different viewpoints of the same object are presented. Through attention on object tracking or boundary detection, the group of images can be determined to belong to the same object. The sense that all these frames of images belong to the same object directs humans to collect the input patterns of these viewpoints to build invariant neural representation in the memory.

This assumption that the observed scene contains the same objects of interest over continuously observed time windows is defined as the sameness principle.

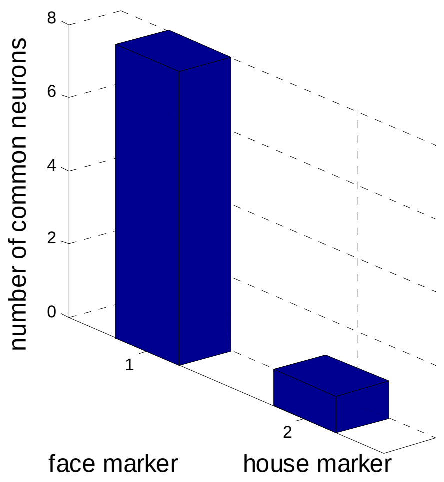


Figure 5.40. The number of common neurons to the markers after the attention on face.

Such process is considered as top-down process, as contrary to the bottom-up process where input images are processed in a feed-forward fashion in sparse coding network. The interaction between bottom-up and top-down processes in invariance building has been studied for building invariance. On one hand, when the variant patterns of the same object are processed and activate a different group of neurons, the top-down process using feedback can affect the neural processing and reinforce the activation of the same group of neurons. On the other hand, such forced feedback to previous layers helps the network development for invariance building. It affects the wiring patterns by changing the correlation among the neurons activities in a given layer and it affects the connection weights by stimulating firing of more desired neurons.

5.8 Conclusions

Sparsely connected R-nets have been used to store information and perform computations that seem analogous to a variety of “higher” cognitive functions [Vog 05]. However, such devices utilize information that has already been sparsely coded, while the inputs from the early stage of sensory information processing are often densely coded. It seems necessary for any sensory device concerned with natural images or sounds to produce sparsely coded information which can be easily handled by R-nets or any other associative memory structures.

We have described a sparsely connected architecture, a hierarchical self-organizing memory, which produces sparse representations of objects. The architecture has low energy consumption in the sense of a much smaller number of connections. The sparse coding and reduction of neuronal activities are generated by means of local competitions, a strategy which is efficient and easily implemented in hardware. The secondary levels, which make interconnections between primary neurons, transmit sufficient information up the hierarchy and provide high memory capacity per synapse while preserving low network connectivity.

In the network with the SWTA algorithm, each pattern activates a single neuron as its representation. The SWTA can be used not only to produce local codes, but also to efficiently find global winners in a sparsely connected structure. In the network with the

OTA algorithm, a pattern triggers a group of distributed neurons and the information is redundantly coded so that recognition is more reliable and robust. Simulations of the proposed models demonstrate that they possess the anticipated mnemonic and coding properties, and an increased level of tolerance to faults and loss of neurons that are essential for the construction of intelligent machines.

The studies of SWTA and OTA in section 5.3.5 are based on the randomly connected networks to eliminate the effects of connection topology on the network coding properties. However, it is also proposed that the neuronal connectivity and receptive fields can be determined based on the statistical correlation among neuron activities. The correlated neurons' information is grouped so that the features are correlated and sent to higher levels. It was also tested that the correlation-based receptive field does not affect the representational capacity and the fault tolerance that the OTA network has with the random receptive field.

Knowing that the top-down process, like attention and the sameness principle can affect the human perception and object recognition, we demonstrated a mechanism of how the object-based attention signal affects the recognition process in HSOM with OTA algorithm. The experiment showed that with attention paid on different object on the superposing images, the image was better recognized as the expected object. The similar mechanism can be implemented for invariance building, in which case the top-down influence comes from the sameness principle instead of attention.

Sparsely connected neural networks are useful devices for learning sparsely coded information. Their large per synapse memory capacities, low energy consumption, efficient algorithms, failure tolerance, and relative ease of implementation in hardware have presumably given sparsely connected networks an adaptive advantage in the evolution of biological brains. It is difficult to imagine how the processing capacities of the roughly 10^{14} synapses of the human brain could be achieved in any densely connected architecture. These same properties seem to make sparsely connected neural networks more likely devices for similar advances in machine intelligence. In addition, the HSOM can implement not only feed-forward processing, but its structure enables the back projection of top-down influence for attention-aided perception and invariance building. In attention-aided perception, the object-based attention is assumed as given top-down signal to be applied on the top level of hierarchy. How this attention is created in the learning or interacting process will be discussed in the next section.

CHAPTER 6: GOAL CREATION AND GOAL-ORIENTED BEHAVIOR FOR EMBODIED INTELLIGENCE

6.1 Introduction

Goal-oriented behavior of intelligent machines was extensively studied in the area of reinforcement learning. Reinforcement learning (RL) [Sut 98] [Bar 03] is an active area of machine learning research that focuses on how a learning agent can learn to approximate an optimal behavioral strategy to maximize the reward while interacting with its environment. One of the characteristics of RL is that it treats the whole problem in a goal-oriented manner. Presented with defined goals but not told what to do explicitly, the agent learns during the interacting, in so-called trial-and-error search. In each step, the learning agent in RL problems senses the state of its environment and chooses actions to influence its environment. RL enables the learning agent to plan, to behave actively, to be situated in an environment and to consider how its actions might fit in a larger picture of continuous interaction with the environment.

The idea of learning from interaction is very fundamental and significant, underlying nearly all theories of learning and intelligence. As stated in [Bar 98], in learning problems, the learning examples of good actions for one situation may not apply to other situations. An agent must be able to obtain new knowledge from experience and interaction with the environment for unseen situations. So, RL is ideal for dealing with uncertain environments or environments that are subject to change.

Basic elements of RL system include a reward function, a policy, a value function and, optionally, a model of the environment [Sut 98]. The reward function is normally given or observable to the agent, while the other three elements are to be learned by the agent. The reward function specifies what the good and bad actions are for the agent. The reward can be either in an immediate or a delayed form. In a biological system, rewards might be identified with pleasure and pain. The policy is a mapping from states of the environment to actions, or a set of states and actions associations. There are two types of values in RL, the value of a state and the value of a state-action pair. The value of a state is the total amount of reward an agent can expect to accumulate over the future, starting from this state *s* and following a certain policy *π*, as shown in (6.1), in which *r* represents the reward received at a certain time, and *γ* is the decay factor along the time. Thus, a value function specifies what is good in the long run.

$$V^{\pi}(s) = E_{\pi}\left\{R_t\left(s_t = s\right)\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\left(s_t = s\right)\right\} \tag{6.1}$$

The value of a state-action pair is defined as the total amount of reward accumulated over the future, starting from this state *s*, taking this action *a*, and following a certain policy *π*, as given by (6.2).

$$Q^{\pi}(s,a) = E_{\pi}\left\{R_t\left(s_t = s, a_t = a\right)\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\left(s_t = s, a_t = a\right)\right\} \tag{6.2}$$

The fourth element of some RL systems is a model of the environment, which are used for planning. Early RL systems were explicitly trial-and-error learners and the model was mostly ignored in the RL designs [Si 01], because the model of environment is not available, subject to change and increase the complexity of the computation. However, with relatively newer developments, the models of planning are incorporated into RL systems. The model of environment is an important part of planning and predicting. In human intelligence, we can consider that, within the complex hierarchical memory that humans have, they can build and hold a model of the environment. Humans build such models of the environment over repetitive training in the lifetime, in supervised, unsupervised or reinforcement fashion. With this model in their memory, humans could make predictions and plan the actions, while making some adjustments based on the information from the environment about the existing situation details.

Many of the earliest RL systems used one of the typical RL methods, actor-critic (AC) method [Wit 77] [Bar 83]. The typical AC architecture [Bar 98] is shown in Figure 6.1. In an AC system, the policy (the actor) and the value function (the critic) are two major parts. The actor is used to select actions based on the states. Having a mapping from current environmental states to the action is a feasible implementation of human's basic actions. Although a typical actor in AC scheme maps the states to a single action decision, the mapping from states to actions can be in the form of states with various action choices with different probabilities.

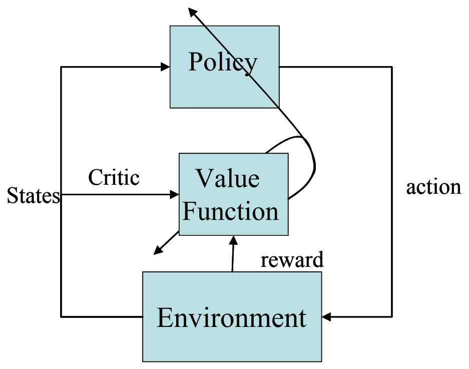


Figure 6.1. Actor-critic architecture

The critic learns about the value function from the reward from environment, so that it is able to evaluate the value function and the actions made by actor. Typically, the critic estimates the state value, described in (6.1). In the action dependent versions of the critic, the critic estimates the state-action value, which is the predicted accumulated future reward by taking certain action at current states, as in (6.2). After each action selection, the critic determines whether this action will improve the value or not. The existence of the critic, as a value system, is a significant component in RL. With the value system, the agent can make predictions about future events and give the values for different predictions so that it can have the optimal choice of action.

In existing designs of intelligent machines, the goal is usually explicitly defined and is given by designers to the learning agent. In RL, the agent does not need to create its own overall goals since it uses a specific, externally provided and externally defined reward. This award implicitly defines its goals and an agent is under complete control of the award giver (a trainer or the environment). However, goal creation is an important and essential consideration for an agent to behave autonomously, conduct intentional and continuous learning in a real-world domain.

In existing designs of intelligent machine, sub-goals are given by designers as well and in most cases cannot be found by the agent itself through experiments. They are strictly related to its externally specified goal. It was demonstrated [Mur 07] that such externally specified subgoals that are explicitly defined and controlled by the system designer yield more effective RL in cases of complex goals.

Due to the top-down approach of learning the values of the state-action pairs, the learning is not a structured process, may require long learning periods as well as equally long periods of time to adopt to new requirements (new goals). RL will scale poorly when applied to problems with high dimensional state space and to train the agents to have complex behavior, which is usually referred to as the curse of dimensionality [Bel 61]. The delayed reward signal most of the time cannot provide enough information for the agent to improve the policy. One promising approach to scale up RL in the existing research is to add hierarchical reinforcement learning (HRL) [McC 96] [McC 94] [Her 00] [Bak 04].

There are several existing directions within the range of HRL research. The first direction in HRL is selective perception [McC 94] [McC 96]. When the model has high-dimensional and large state space, observing and considering the full state space with all state variables will require a great deal of computational resources. Also, during the process that an agent accomplishes a certain part of the complex task, not all the state information is required all the time. To reduce the demand for large amount of resources and computations and to focus on the necessary state information, selective perception may lead designers to more efficient data collection and possibly cheaper design by focusing on the part of state space that is sufficient to solve the current task. In selective perception, due to the switching of attention or perception, short-term memory is required to remember where the attention was focused. Thus, after a certain part of the task is finished, the attention can be switched back. This method requires the full diagram of the agent's policy; that is, the desired complex sequential task will be analyzed and divided into the sub-goals and their corresponding attention switches. Such a diagram, most likely provided by designers, is a finite-state-machine available to the agent and the agent will learn the probability of state transition to form the optimal policy during RL.

Attention-based learning is also desired in order to protect the memory from saturation in the sense that only attended objects are learned; thus the attention should be closely related to the goals. It has been investigated that given a specific task, top-down processes guide the attention to the task-relevant objects in the environment [Cor00] [Hop 00]. In [Nav 02], a goal-oriented attention guidance model was developed. In this model, the saliency map [Koc 85], which encodes the visual salience of visual inputs, and task-relevance map are combined to have attention-guide responding to objects. The model utilized the ontology in a long-term memory (LTM), which represents the relations among objects, to find the relevant objects. Even though such a model is supposedly biologically-plausible, it is very difficult to build since the ontology in LTM is assumed to exist as a given knowledge base. Although the drive of the agent to interact with the environment and the gradual process to build such ontology are significant for building intelligent machines, they are not a part of the learning system presented in [Koc 85]. In addition, the attention-based behavior still relies on the defined goal or a task given by the designers instead of being formulated by agents themselves.

The second direction in HRL is memory-based RL [Her 00]. In large state space, the state information available to the agent may be incomplete, corrupted or the selective attention may produce perceptual aliasing. Adding hierarchical memory about past perceptions is a strategy to deal with such problem. In different levels of hierarchy, past experience is explicitly remembered. Take a robot navigation problem as example. The lowest level strategy includes motions on all the motors on the robot. The 2nd level strategy includes actions of turning left, turning right, and going straight and so on. The highest level strategy includes going to the left part of the room, maneuvering around an obstacle, meeting the cross-section, going to the dead end, and so on. The memory on each level of the hierarchy will remember what actions were taken in the successive steps. And the value system is built for each level, which is updated by the explicit action history in the memory. This hierarchy-based RL is one of the promising ways to go, since the model is divided by the abstraction levels, instead of the states of the finite-state machine. However, there is still one step away from the goal orientated intelligent agent we would desire, because we have to provide the hierarchical structure of the certain tasks. We desire to have an agent which is able to discover the sub-goals and formulate the hierarchical structure of the task automatically given a final goal.

There are several design features we would like to have for the hierarchical RL system for embodied intelligence. Firstly, the final goal of the learning problem can be divided into sub-goals. The sub-goals are accomplished on a certain level of hierarchy. Thus, policies of different levels are constructed in order to achieve the sub-goals. The high-level policies can automatically discover the sub-goals and try to solve the overall task. The low-level policies can be re-used easily, either within the same task or in other tasks [Bak 04]. Accordingly, values systems are built on different levels concerning different level of policies.

Secondly, currently the research in hierarchical RL solves the problem given a hierarchical structure provided by the designers. The agent will learn the policies within its hardwired structure [Bar 03] [Her 00]. It is highly desirable for the agent to learn the hierarchy itself. As stated in HSOM, the representations of patterns are built in a hierarchical way and the abstraction levels of the patterns on different hierarchical levels are not predefined by the designers. It is desired that the procedure that divides the overall goal into different levels of subgoals is an automatic process.

In general, in order to achieve the hierarchical RL with sub-goal automatic discovery and sub-policy specialization, the information stored in the HSOM and a similar hierarchical structure should be utilized. It is obvious that learning complex tasks can be facilitated if simpler tasks are learned first, and that the knowledge acquired in this process should be reused to advance both understanding and skill levels. The machine needs to organize its learning to acquire useful knowledge based on its goals and to create goals for its behavior using a built-in mechanism. In order for the machine to intentionally organize its learning, goal creation is required.

The goal creation issues have been mainly studied in the framework of beliefs, desires and intentions (BDI) [Rao 91] [Rao 92]. In a BDI architecture, beliefs represent existing knowledge the agent has about the environment. Desires are defined as a set of states that agents try to achieve. Intentions are defined as the approaches to achieve the desires. In general, the agents in a BDI system have multiple desires and intentions active simultaneously. The desires and intentions could come in the form of reaction to the environment or proaction from a long-term plan. It is proposed in [Tha 02] that the way to achieve a balance among different desires is to have a library of goals which capture knowledge of limited and well-defined aspects of the world [Luc 98]. The goals in the library are designed to have a uniform type of representation so that goal conflict can be detected and consistency can be maintained. In general, the goal creation issues discussed in the BDI framework deals with the problem of how certain goals are triggered or chosen from the set of available goals instead of being created, for example, in the procedural reasoning system (PRS) [Geo 89] [Geo 87], the motive processing system (MPS) [Bea 93] [Bea 94], and in the motivational goal creation [Nor 95]. Although, the goals in these works are defined in a specific way adopting meaning in various abstraction levels, the hierarchy of goals is not generated by the agent as a part of the learning and interaction process.

In human intelligence, the perception and the actions are intentional and selective processes. The perceptions are built and actions are carried out attempting to meet certain goals or needs. Based on primitive goals and needs, people firstly create simple goals and learn simple actions. Based on the learned perception and skills, they build complex perception and actions to meet complex goals. It can be postulated that such bottom-up process enables a human to find relevant subtasks for a complex task so that the task is divided into procedures that a human can finish step-by-step. The process also generates the human needs and the needs or expectations can affect the human attention on input sensory information. In human reinforcement learning, the rewards are rather subjective than definite, numerical and given by the environment. Different individuals interpret the environmental inputs differently. The mechanism to create the goals and motivations for a human should be relatively simple and embedded inside its brain structures as a part of the mechanism that is capable to create human level intelligence.

Besides the use of goal creation for useful hierarchical RL, it can be used in the studies of motivation for the intelligent machine to behave and interact autonomously in an unknown environment.

In embodied intelligence, a fundamental question is what should be the motivation that a machine has, so that it can develop into an intelligent and knowledgeable agent. An attempt to answer this question introduced "flow" theory, which states that humans get internal reward for activities that are slightly above their level of development [Csi 96]. Another development an intrinsic motivation system for autonomous robots is presented in [Oud 07]. In [Oud 07], a robot explores the environment and learning is activated when its predictions do not match the observed environmental response. The motivation in such exploratory learning systems comes from the desire to minimize the prediction error. Although exploratory learning helps to gain knowledge of the environment, it is void of a specific purpose and does not provide a complete and efficient mechanism of the motivations to develop intelligence.

We propose that it is the hostility of the environment, as expressed in the definition of EI in Section 4.1.2, that is the most effective motivational factor for learning. The pain we receive from the hostile environment motivates us to act, learn and develop in order to reduce this pain. The two conditions are needed together - hostility of the environment and intelligence that learns how to "survive" by reducing the pain signal.

Pain, as the general term for all types of discomfort and unpleasant feelings, is the common experience to all people. On the most primitive level, people feel discomfort when they are hungry so that they learn to eat and to search for food. They feel pain when they touch burning charcoal so that they learn to stay away. Although, on more abstract levels, individuals experience different degrees of stress and anxiety (so that people have very different motives and higher-level goals), these primitive pains essentially help them to survive in the environment and start to obtain skills that will be useful for the sustained survival.

Neurobiology study facilitated by the neuro-imaging techniques, such as positron emission tomography (PET) and functional magnetic resonance imaging (fMRI) etc, supports the suggestion that there are multiple regions of brain involved in the pain system which form the neomatrix, usually called "pain matrix" [Mel 90]. Experiments using fMRI have identified that such a matrix includes a number of cortical structures, the anterior insula, cingulate and dorsolateral prefrontal cortices [Pey 00], and subcortical structures including the amygdala [Der 97] and the thalamus and hypothalamus [Hsi 01].

Two parallel systems are recognized in the pain matrix - the lateral pain system, which processes the physical sensational pains, and the medial pain system, which processes the emotional aspects of pain, including fear, stress, dread and anxiety [Töl 99]. The intensity of the pain is processed throughout the matrix. The physically noxious stimuli activate certain regions in the lateral pain system, and the anticipation of the pains can induce stress and anxiety, which activates the medial pain system. Recently, it has been shown experimentally that the anticipation of a painful stimulus can activate both systems as well [Por 02].

It has been widely accepted since decades ago that pain has sensory-discriminative, affective, motivational, and evaluative components [Mel 68]. The work put forward by [Mes 90] [Mor 93] on a large-scale neurocognitive network model suggests that the cingulate cortex is the main contributor to a motivational map that interacts with a perceptual map provided by the posterior parietal cortex. By affecting the motivation, attention and sensory perception, it is proposed in this learning paradigm that the pain matrix is essentially a part of the goal creation process of EI machine learning machine.

In the proposed learning paradigm, we would expect the machine to use uniform neuronal structure to self-organize the proposed goal creation system (GCS). GCS, as the third pathway in HSOM, stimulates the creation of goals on various abstract levels starting from the given primitive goals. It is responsible for evaluating actions in relation to its goals, stimulating the learning of useful associations and representations for sensory inputs and motor outputs. The goal creation pathway finds the ontology among sensory objects, makes connections among the actions and objects, and creates the needs and affects the agent's attention.

In the following sections the concept and structures for the goal creation system will be further developed. The rest of this chapter is organized as follows. Section 6.2 explains the fundamental characteristics of the proposed goal creation system and states the structure of goal creation system in detail, including the basic unit, connections among neurons, and the growth of the goal creation system hierarchy. Section 6.3 explains how such goal creation system works in the action selection and its role in attention and anticipation based learning. Section 6.4 discusses how such goal creation system fits in the overall goal-driven learning system. Section 6.5 illustrates the GCS using experiment. The proposed model with its future work is concluded in Section 6.6.

6.2 Goal creation system

The built-in goal creation and value system provides a mechanism that triggers learning of intentional representations and associations between the sensory and the motor pathways. When the EI machine realizes that a specific action resulted in a desirable effect related to a current goal, it stores the representation of the perceived objected involved in such action and learns associations between the representations in the sensory pathway and the active action neurons in the motor pathway. If the produced results are not relevant to the current goal, no intentional learning is taking place. Since this happens most of the time during the exploration stage, such a deliberate learning process protects the machine's memory from overloading with unimportant information.

Similar to neurons in the sensory and motor pathways, neurons in the goal creation pathway are organized hierarchically. The neurons are essentially a hierarchy of pain centers. They receive the pain signals and trigger the creation of goals, which represent the needs of the machine and the means to solve the pains. Lower level goals are externally stimulated through primitive sensory inputs. Neurons' activation on these inputs may represent a large number of situations that the EI encounters while interacting with the environment. Higher level goals are developed through associations between activities on the lower level goal creation neurons and other neurons in the sensory-motor pathways. Goals represented on the lower levels correspond to simple, externally driven objectives, while those on the higher levels correspond to complex objectives that are learned over the machine's activities and are related to its understanding of the best ways to accomplish the lower level goals.

6.2.1 Fundamental characteristics of the goal creation system

In the proposed goal creation system (GCS) for intelligent machines, the evolvement and growth of EI value and action systems are stimulated by a simple built-in mechanism based on dedicated sensory inputs, called **"primitive pains"**. Since the pain signal comes from the environment (including the embodiment of EI machine), it is inevitable and gradually increasing unless the machine figures out how to reduce and avoid it. Pain reduction is desirable while pain increase is not. Thus, the agent has a desire to reduce the pain or equivalently to pursue pleasure/comfort. So it is forced by the "primitive pain" to explore the environment seeking solutions to pursue its goal - reduction of the pain. In this process, the machine will accumulate the knowledge about the environment and its own embodiment, and will develop its skills.

The EI machine may have several primitive pains, and each one of them has its own changing intensity, and requires its own solution. A pain threshold *th* can be introduced and a pain higher than the threshold requires the machine to look for solutions. At any given time, the machine suffers from the combination of different pains with different intensities, as shown in Figure 6.2. Different pains vary in time in different ways and the agent needs to take care of the strongest pain signal and sets reduction of this particular pain as its current goal.

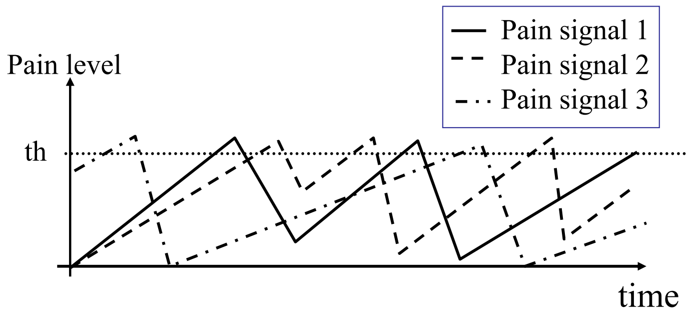


Figure 6.2. Changes in temporal intensity of the primitive pain signals.

We can make references to human learning, where a similar mechanism is used to induce activity-based exploration and learning. The "primitive pain" inputs for a human include pain, hunger, urge, anxiety, fear and other types of physical discomfort. The pain usually happens when something is missing. For instance, we feel hungry when we lack of the sufficient sugar level in our blood. We feel anxious when we lack of enough food or money. Such postulate of deficiency is satisfying our goals as a trigger for action and learning will make the proposed goal creation mechanism biologically plausible even at the level of human intelligence. For example, in a new-born baby, a hierarchical goal creation system and value system has not been developed yet. If the baby is exposed to a primitive pain and suffers, the baby will not be satisfied nor its pain reduced until some action can result in the primitive pain reduction. When the pain is reduced, the baby learns to represent objects and actions that helped to lower that pain.

We also need to find and eat food to sustain our activities. A gradually increasing discomfort coming from the low “sugar level” tells us that we must eat. The pain is getting stronger and forces us more and more to search for solutions. Similar urges pressure us to go to the bathroom, put on clothes when we feel cold, or not touch a burning coal. The pain warns us against incoming threats, but also forces us to take an action. We also feel relief if we take an action that reduces this pain. Thus pleasure and comfort can be perceived as opposite to pain and discomfort.

The intensities of the perceived pains serve as a regulator to set priorities to our actions and thus be responsible for goal creation. For example, the urgent need to go to the bathroom may easily overtake our desire to eat, or even more so to sit through an interesting lecture. In general, the strongest pains will determine the most pressing goals. Thus the pain-based GCS will also yield a natural goal management scheme.

A primitive pain is a signal received from the primitive pain sensors. It stimulates the primitive pain detection center. In solving the pain on the primitive level, the machine is stimulated to explore for actions or to exploit the action that relieves the primitive pain. The exploration at first is based on the random stimulation and activation links or links that were initially (genetically) set to help to reduce the primitive pains. Such genetically set links facilitate learning of higher level skills and correspond to built-in skills. Genetic setting of lower level skills (by pre-wiring sensory-motor responses) may be a preferred solution to designing machines when they need to develop complex skills in a practical case.

Genetically set associations between the primitive pain centers and actions also exist in humans. A baby cries when it is wet or hungry, it also has well developed skills to eat. A burning pain from touching a hot plate triggers an automatic pull back reflex. These sensations and actions become gradually associated with circumstances under which they occurred, leading an intelligent agent to learn basic skills or to improve upon them.

A primitive pain leads the machine to find the solution and then the solution is set as the **primitive goals**. Afterwards, the primitive pain will also be a trigger for developing higher level pain/pleasure centers and a mechanism for the creation of higher level goals. This is based on a fundamental mechanism for need to act and a simple measure for satisfying such a need. We would like to argue that this simple need to act may lead to complex goal creation and its implementation. The mechanism of goal creation in humans, and how the human brain controls his behaviors are not fully theoretically established yet in the field of behavioral science or psychology. It is possible that the mechanism we propose here is not the way people create their goals biologically. However, it is feasible, simple, and it satisfies our need to establish goal creation and to formulate the emergence of a goal hierarchy for machine learning. In addition, this goal creation system stimulates the machine to interact with its environment.

6.2.2 Basic unit of GCS

The proposed goal creation mechanism is based on evolving the idea of a uniform, basic goal creation unit. The GCS unit contains three groups of neurons that interact with each other, including the pain center neurons, reinforcement neurons and corresponding connected neurons in the sensory and motor pathways of the HSOM. The basic goal creation unit structure is shown in Figure 6.3. Although as demonstrated in [Sta 07a], the representations for certain sensory objects or motor actions are built using a group of neurons in HSOM's sensory and motor pathway, they are illustrated as one single neuron in this work for simple illustration.

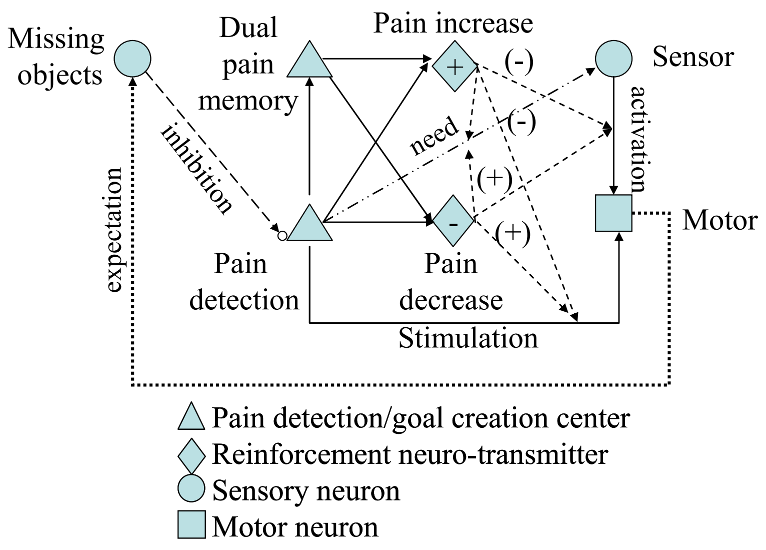


Figure 6.3. Basic goal creation unit.

In the first group, the **pain detection center** (detecting the pain level, denoted as I_P) is stimulated by the pain signal from the sensory input and represents the negative stimulation, such as pain, discomfort, or displeasure. Since the pain exists due to the absence of certain objects, denoted as "missing object" in Figure 6.3, the perceived object can inhibit the pain signal through the "inhibition" link. Thus the pain detection center is activated by the silence of this sensory neuron. A **dual pain memory center** stores the delayed pain level, I_{Pd}. Thus the currently detected pain signal and the pain signal in the last time step (in the previously completed event) are stored and need to be compared in the second group which contains **reinforcement neurons**.

Reinforcement neurons register a decrease or increase in the pain level by comparing signals from the pain detection center and the dual pain memory center. They do not physically connect to any neuron but send positive or negative reinforcement signals to build the associations. The "pain decrease" reinforcement neuron gives a positive reinforcement while the "pain increase" neuron gives a negative reinforcement. The reinforcement signal is calculated in (6.3).

$$r = I_P - I_{Pd} \tag{6.3}$$

The third group contains the corresponding active neurons in the sensory and motor pathways of the HSOM that these pain center neurons connect to.

In a GCS unit, initially, the pain detection center directly stimulates multiple motor neurons. A gradually increasing pain level forces the machine to explore through the motor actions by stimulate the motor neurons with random connection weights W_{MP}, since this is the only chance that the machine will learn a proper action when it has yet to learn anything about its embodiment, its environment and the way to interact with it. The machine explores starting from the action with the strongest activation (strongest weights connecting to given pain stimuli). To carry out such action, certain objects which will be involved in this action must be available. Initially, a motor neuron may be associated with multiple sensory neurons by activation weights W_{MS}. The available (active) sensory neurons send activations to the motor neurons so that a certain sensory-motor combination is implemented. The direct links from the pain center to the motor neurons force exploration or the implementation of certain motor actions all the time.

After the action is taken, once the pain reduction or increase is detected by the second group of neurons, a learning signal r is produced to reinforce or weaken the value of an action and the value of the sensory-action pair by strengthening or weakening the stimulation links from the pain detection center to the motor neurons and the activation links from the sensor to the motor neurons. Pain increase will make the links more inhibitory, while pain decrease will make links more excitatory, as shown in (6.4).

$$\begin{aligned} W_{MP} &= W_{MP} + r \cdot \beta^n \\ W_{MS} &= W_{MS} + r \cdot \beta^n \end{aligned} \tag{6.4}$$

where β denotes a learning rate within range (0, 1] and n denotes how many times the link has been adjusted.

Meanwhile, since the active sensory neuron representing the object which was involved in the action helps reduce the pain, a "need" link, with weight W_{SP} will be created to connect the active pain detection center to the active sensory neuron using Hebbian Learning. On the other hand, the object, which was missing and producing the pain signal earlier, becomes available and the neuron representing the object becomes active after the motor action, an “expectation” link with weight W_{SM} will be created to connect the motor neuron and the missing object.

The “need” link and the “expectation” link will be updated by the reinforcement learning signals as well. The bigger the change of the pain level is, the stronger are the reinforcement signal and the weight adjustment on the involved links. The described interaction of various groups of neurons in the goal creation mechanism and the “stimulation”, “activation”, “expectation” and “need” links are illustrated in Figure 6.3.

To solve the primitive pain from “low sugar level”, after several random trials, the action “eat”, connected with perception of “food”, will be rewarded. As a result, the strength of the stimulation link from primitive pain detection center to “eat” and the activation link from “food” to “eat” will be increased. The “need” link is connected from the pain center to “food” and reinforced when such successful action is exploited and rewarded for several times. And the action “eat” will expect the appearance of sufficient “sugar level”. Whenever the “low sugar level” pain center sends out pain signals, the “eat” will be excited prompting the machine for this action.

This simple mechanism is easy to expand and generalize. In order to generate abstract and complex goals, we will incorporate basic goal creation units into a hierarchy of the goal creation pathway as discussed next.

6.2.3 Building a goal hierarchy

When solving the primitive pain, “food” is needed for and its absence now will lead to anxiety or stress for the machine. A second level pain center is created to represente such stress and is called an **abstract pain center**, shown in Figure 6.4. It connects with the primitive pain centers through an “echo” link since it echoes the pain signal from the primitive pain centers. Each time the primitive pain center is excited, it sends activation to the abstract pain center. Because these centers are not stimulated from the physical pain sensors, they only symbolize the real pain or represent the discomfort for not having the objects that can prevent the primitive pain.

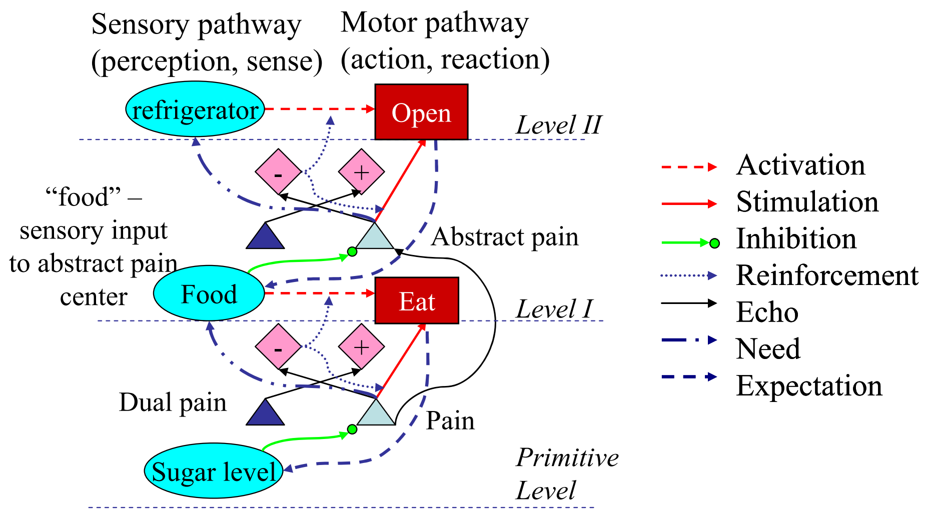


Figure 6.4. Creating the abstract pain signal.

When “food” is available and the agent “eats”, primitive pain is relieved and the echo link will relieve the abstract pain as well. The pain signals disappear and the agent goes back to its normal painless state. As a result an inhibitory link is developed between sensory signal “food” and the abstract pain center, which means the existence of the “food” can inhibit the abstract pain.

When “food” is not available, the agent cannot reduce the physical primitive pain. Then, it tries to find the solution to reduce an “abstract pain”. Although reduction of the abstract pain does not directly reduce the primitive pain on its lower level, it may be a prerequisite for such reduction.

In this experiment, the primitive pain center forces the agent to explore to reduce the abstract pain. Again, exploration is done based on the initial associations between the abstract pain center and motor actions and associations between sensory representations and motor actions. The reinforcement neuro-transmitters connected with this abstract pain center update the interconnection weights. Eventually, the reduction in the abstract pain resulted by the action “open” combined with sensory object “refrigerator” indicates that the pain from absence of “food” will be associated with the state-action pair “refrigerator ”-“open”. It does not matter whether such action (opening refrigerator) was found by pure exploration or by instruction from a teacher. Since once the machine opens the refrigerator, it sees the food and the abstract pain is suppressed, the action will be reinforced. In addition, an expectation link from the motor action “open” to the sensory neuron “food” is built, thus “food” will be expected as the result of the action “open”. It is noticed that this expectation link will be used for planning future actions in which a certain action’s result can be expected. Such process can be illustrated using Figure 6.4.

The goal hierarchy containing three levels can be further expanded vertically. If the agent “opens” the “refrigerator”, but the “food” is not found, the machine needs other options to suppress the abstract pain, and subsequently the primitive pain. The machine may explore by random search or by instruction. Once it “spends” some “money” (in a store), food is available and then the abstract pain (no food) is reduced. Such action is rewarded and will be more strongly stimulated by the abstract pain center. The “food” is eaten, the primitive pain is suppressed, and the pain signals are reduced. However, when “money” is not available, an abstract pain center on level II is activated with an inhibitory link from “money” and directly stimulated by the pain center on level I (that represents “no food” in this example). Subsequently, the machine needs to act to solve the pain on level II. After exploration, the agent finds out that the solution to the pain represented by “no money” is to “work” at a “job”.

The agent may also find that “stealing” other person’s “purse” can provide “money”. However, even if such action suppresses the pain on this pain branch, the agent will be punished by inflicting pain on other pain branch so that this association will be weakened and “working” at a “job” will stand out as the best option. Subsequently, pain centers on higher levels will be created and the hierarchy of pain centers and goal creations will be built. The previous example can be illustrated by Figure 6.5.

In the process of finding efficient solutions for a certain goal, there will possibly be time lag between when the action is taken and when the action start to take effect. The time lag may affect the association between a certain goal and the correct action. However, it is believed that after enough interactive cycles, the agent will eventually make strong associations between the effective action and the goal.

In the proposed GCS, at every step, the machine finds an action that satisfies its goals and such action and the involved objects may result in creating further goals.

Therefore, in this mechanism, the machine simultaneously learns to associate the goals with deliberate actions, the expected results of actions, the means to obtain objects, and hierarchical relations among various objects. It reinforces which objects in its visual field are related to its goals, facilitating and stimulating learning of the desired objects. It helps to establish higher level goals and the means of their implementation. It governs execution of actions to satisfy the goals and manages the goals priorities at a given time.

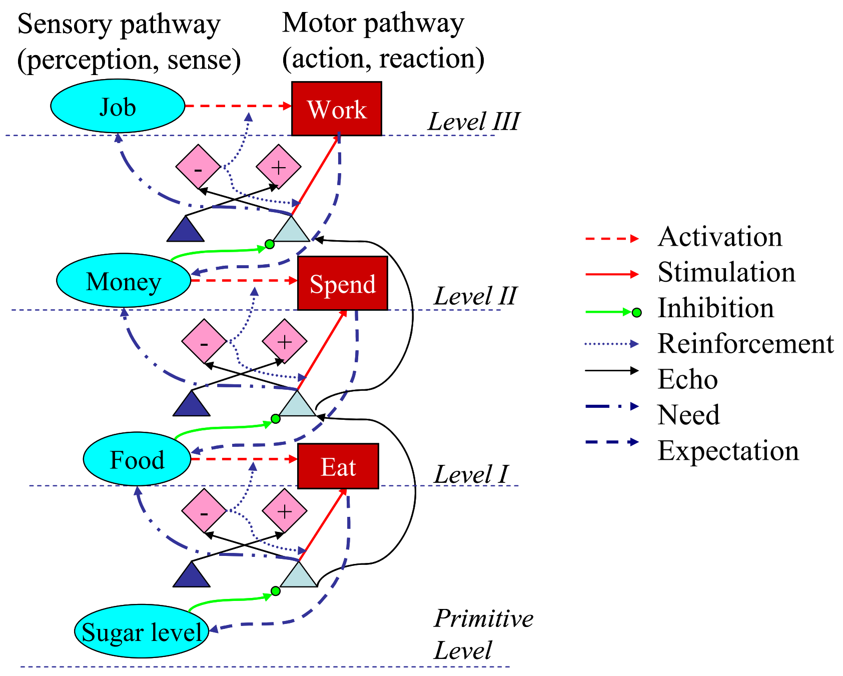


Figure 6.5. Expanding the hierarchy.

6.2.4 Relations to reinforcement learning

In GCS and RL scheme, an agent has different overall objectives. In GCS scheme, a learning agent learns to minimize the pains caused by the hostile environment. On the contrary, a learning agent learns to achieve the maximum reward from the environment in a classical RL structure. When the agent is situated in an environment with multiple inputs, the GCS and RL systems not only have different optimization objectives (min vs max), but also, they will lead to different behavior of the agent. When there are multiple pains at the same time, the GCS agent always tries to reduce the strongest pain. Over the time, he carries out different actions depending on which pain is the strongest every moment. Eventually all the pains will be reduced and be controlled within a certain range. When all the pains become zeros, the agent won't have to do anything until a certain pain increases. However, an RL agent will repeat the same action which produces the reward, even though he fails to obtain the other rewards.

In a RL AC architecture, the agent's critic network learns the values of state-action pairs through the reinforcement signals coming from the environment or a teacher. Then the state/action pairs are evaluated by the critic network. In a contrast to the RL, GCS learns to accomplish only the primitive goals using the external reward, and has reinforcement neurons of each GCS unit to generate internal rewards to build the required associations. Outputs of the reinforcement neurons correspond to the rewards used to instruct the machine regarding desired or undesired state/action pairs in RL.

In a learning system using GCS, instead of a computational-based value system used in typical RL, the value system is essentially embedded in the hierarchical GCS. In the actor-critic (AC) RL paradigm shown in Figure 6.1, the action is chosen by the actor based on the present sensory (state) inputs. The critic evaluates the state-action pair to determine whether the actor needs to improve the selection of actions and how to improve it. Practically, the interactive activities of the agent in AC paradigm are triggered by the sensory (state) inputs, which make it a passive process. Using the GCS, the machines' interaction with its environment becomes an active process since the machine finds the optimum actions according to its internal goals and the pain inputs. To implement the desired actions, certain sensory inputs are necessary. Such state-action pair's value is determined by the strength of the sensory inputs and the strength of the desired action. The need for the sensory inputs becomes a higher level goal. Through building the goal hierarchy, an intelligent machine learns to associate goals with states and the values of different states for accomplishing goals.

Therefore, in this scheme, learning about particular actions may only concern particular goal achievements. Learning that occurs on a higher level of the memory structure to satisfy a higher level goal may have no direct relationship with actions that may satisfy the lower level goals. For instance, an abstract goal of earning money might have been stipulated by a need to buy food or by a need to pay the heating bill or both. In a similar way, accomplishing this higher level goal does not necessary remove hunger or heat the living space. In particular, the only externally observable "primitive pain" may remain unaffected. This isolates externally administered reward related to a "primitive pain" from the reward that satisfies a higher level goal. While we can measure (and thus optimize) the overall reward received externally to satisfy the primitive goals, it is impossible to measure (and therefore optimize) total amount of reward received for higher level goals, as these goals remain mostly unknown to the external observer. This fact differentiates the proposed GCS from the RL scheme, in which the total amount of reward is assumed measurable, so that a system can be optimized to maximize the reward.

This argument somehow negates the existence of a mathematically optimum intelligent agent like the one discussed in [Leg 06]. Since the total amount of reward in the proposed GCS cannot be measured from outside, mathematical proof of optimality presented in [Leg 06] does not apply.

6.3 Anticipation and action selection in hierarchical goal creation network

Visual attention has been investigated in a variety of studies and was shown to affect the human perception. It was known that superior parietal lobule (SPL) is an area of the brain involved in attentional control [Yan 03]. The prefrontal cortex produces a top-down signal which reflects the current goal and sends it to SPL. SPL issues a transient signal for attention control. The signal is sustained in lateral intraparietal (LIP) [Bis 03] and modulates the sensory representations in extrastriate cortex. However, there was no clear and feasible mechanism or working model about how this signal was created and how it is constantly involved in intelligent behavior.

In this section, we will discuss how an agent may choose the most appropriate action to implement its goal. This will lead to such higher level functions of the cognitive process like planning, anticipation, and action selection.

Figure 6.6 shows a single activated path of a goal hierarchy, which is obtained as a result of the mechanism presented in Section 6.2.

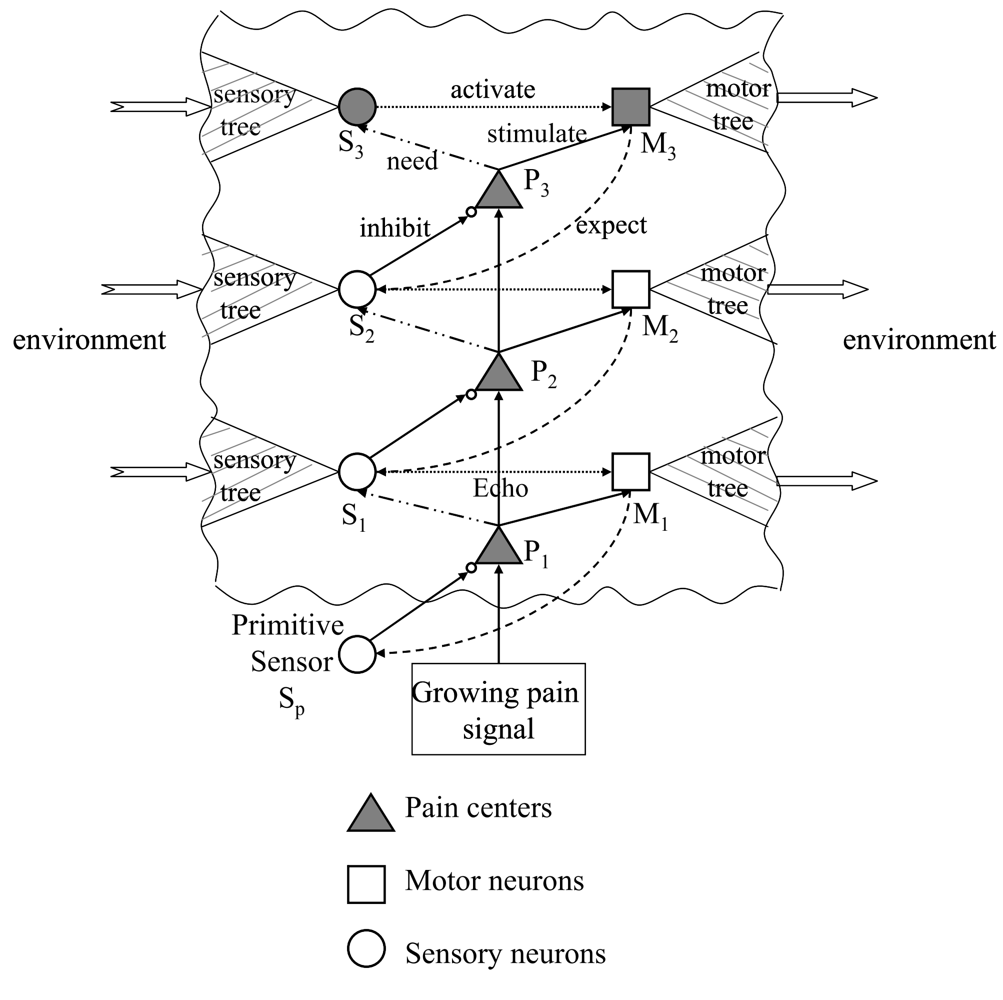


Figure 6.6. Activated path in the pain tree.

We can see from the figure that how GCS interact with sensory and motor pathways on various levels (The goal creation unit is shown using a simplified triangle symbol).

Primitive pain, denoted as P_1 in Figure 6.6, is activated by the absence of primitive sensory S_p. We will illustrate the mechanism responsible for anticipation of the sensory input and selection of action involved in the alleviation of the primitive pain P_1 in this pain tree.

6.3.1 Anticipation and attention in GCS hierarchy

Suppose that, as the machine explores the means to provide S_p to reduce the pain P_1, M_1 is found as the best action which needs activation from the sensory object S_1. Then the stimulation connection from P_1 to M_1 and the activation link from S_1 to M_1 are reinforced. And P_1 generates the need link directed from P_1 to S_1 and M_1 builds the expectation link to S_p. The “need” link sends an additional top-down stimuli coming from the pain center to the needed object, as shown in Figure 6.7.

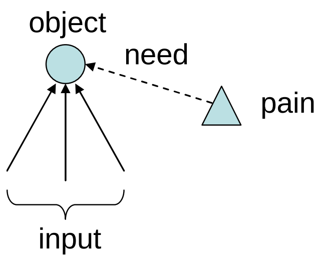


Figure 6.7. Attention signal generated by the goal center.

When S_1 is not available, the lack of S_1 creates the abstract pain P_2 which triggers the machine to look for further actions to solve active P_2. Accordingly, M_2 is found to be the best action to reduce P_2, and P_2 needs S_2. M_2 expects that S_1 becomes available in the sensory pathway after the machine implements action M_2. A similar organization structure is built for P_3. Echo links are used to connect P_2 from P_1, and P_3 from P_2. Notice that in this process, the stimulation and activation links are found through exploration and are trained by reinforcement learning signal issued by neuro-transmitters in the goal center. Both the need and expectation links occur naturally in Hebbian learning as a result of learned observations and actions taken in response to specific goals.

Assume that on the top level an expected sensory input S_3 was found available. This activated the top level motor function M_3 that resulted in the act on the environment. Since the action M_3 has been previously learned to accomplish the goal of reducing the pain P_3, it is expected that the next input from the environment will activate S_2 resulting in inhibition of P_3.

While stimulation link from P_1 to M_1 and activation link from S_1 to M_1 are essential for effective implementation of the GCS scheme, the need link from P_1 to S_1 is not critical in finding M_1 and reducing P_1. The "need" link is useful for building meaningful object representations for accomplishing goals. Using again the "food eating" example, when "food" is not known to the agent and found to be useful to reduce the primitive pain, a neural representation of "food" will be built with the help of "need" link input. Similarly, to reduce higher levels of pains for lack of "food", the representation of "refrigerator" will be built since it is useful for the agent to obtain food. This goal-oriented learning also explains our postulate earlier that the machine only learns something useful once it finds it to be meaningful for the machine's survival in the environment.

The need link also helps to focus the attention on finding particular desired object needed by the action. Therefore, it may have a role in determining the current attention in machine's goal-oriented behaviors, when an object representation was learned to be useful for a machine's goal. As discussed in Section 5.7 such attention signal may help to recognize a desired object perceived by the sensory pathway.

Thus higher level representations related to needed elements are triggered easier since attention focus helps to observe the needed elements and keeps neurons that represent them activated to build this higher-level representation.

With the attention as an additional top-down stimuli coming from the pain center via the need link, the needed object will be recognized easier through integration of the perception with the need. For instance, when you are on a treasure hunt or look for a lost object, your attention may be focused on detecting such object among many others, providing additional excitatory signals to trigger object representation neurons once an object is in the visual field.

6.3.2 Alternative action selection

In the activated pain path shown in Figure 6.6, the expectation links from motors to lower level sensors will play an important role in motor action selection on the same level of abstract pain. As described earlier, an action is considered successful when the pain is reduced in the next time iteration. And when an action is taken, while the pain is increased, the action is determined to be unsuccessful and the next alternative action will be taken. However, any action takes time to finish and to actually reduce the pain and the pain signal can keep increasing during the process of action being undertaken. In this section, we will discuss how the action is determined to be unsuccessful and when the next alternative action needs to be taken.

In the activated pain path, if P_1, P_2 and P_3 represent "lack of enough sugar level", "lack of food" and "lack of money", respectively, M_3 represents the action to solve P_3, we expect that the activation of M_3 will provide the money. Usually, a pain can be alleviated in several different ways. Suppose that we can get money from the bank, from the friend, from the mother in law, or from a stranger. Assume also that each next case is less favorable that the previous one. The initial action M_a is chosen as the action with the strongest stimulation from the pain center P and activation from the sensory object S, as in,

$$M_a = \max\{P \cdot W_{MP} + S \cdot W_{MS}\} \tag{6.5}$$

where W_{MP} denotes the stimulation weights from the pain center to the actions and W_{MS} denotes the activation weights from the sensors to the actions.

We may go to the bank since it is currently our best option and find out that it is closed. If the bank is still perceived through the sensory pathway, the existing mechanism may insist on going to the bank to get the money since getting money from the bank was the strongest action related to the lack of money. Thus, an additional mechanism based on the expectation signal is needed to abandon this action and to try another one. For instance, if P_3 (lack of money) is also linked to getting money from your friend, we need an inhibitory signal blocking the unsuccessful action, as shown in Figure 6.8.

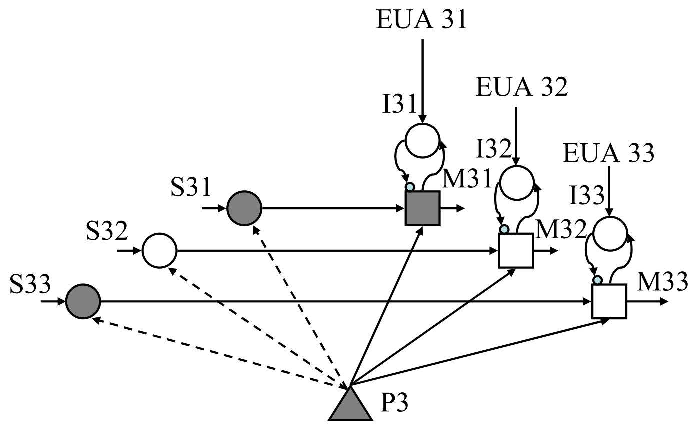


Figure 6.8. Inhibition of an unsuccessful action.

Suppose that M_{31} received the strongest activation from P_3 but the expected result was not provided at the end of the action. At the end of the unsuccessful action, the concurrent activations of M_{31} and its "end of unsuccessful action" (EUA) signal EUA_{31} activate the inhibitory neuron I_{31}, whose activation is determined by (6.6), where m denotes all the possible actions and i denotes the level of goals in this single path of goal hierarchy the machine is acting for.

$$I_{ij} = M_{\mathrm{ij}} \cdot EUA_{ij} \left(j = 1,2,...m \right) \tag{6.6}$$

This neuron I_{31} inhibits the activation of M_{31} allowing other action to take place, as in (6.7).

$$M_{\mathrm{ij}} = P_i \cdot W^{sti}_{M_j P_i} - I_{ij} \left(j = 1,2,...m \right) \tag{6.7}$$

For instance, if S_{33} is observed (mother in law), the action M_{33} will be active and with M_{31} lowered by inhibition (as in (6.7)), M_{33} will win the competition in (6.5) and this action will be selected. Inhibitory neurons (I_{31}, I_{32}…), which give inhibitory inputs to the motor neurons, gradually lose their signal strength as time goes by, as in (6.8), where t denotes the time.

$$I_{ij} = I_{ij} \cdot \alpha^{t} \left(j = 1,2,...m;\ \alpha < 1 \right) \tag{6.8}$$

This feature enables the machine to retry a formerly unsuccessful action when its activation becomes the highest after some time, since the situation may change (for instance bank reopens). In the following Section 6.3.3, we will discuss how the EUA signal is generated using neural circuit.

6.3.3 End of unsuccessful action signal

End of unsuccessful action (EUA) signal has two roles. Its main role is to inhibit an unsuccessful action and force the machine to explore. Its supporting role is to lower the interconnection weight between the pain detection center and related motor action neurons.

All expectation signals are feedback signals fired from active motor neurons to expected sensory neurons. To discuss how the EUA is generated, let us consider the columnar structure of a sensory pathway as discussed in [Gro 98]. An unexpected input (UI) neuron is proposed in this work to represent the mismatch between the real input and the expected input from the lower level sensory neurons. Layer 5 neuron of a column which represents the expected sensory object can receive feedback and be connected with the expectation signal from the motor neurons. It was also discussed in [Gro 98] that in a column, layer 6 neuron is activated only when it receives input from a lower level neuron. Therefore, the UI neurons are connected with layer 5 and layer 6 neurons as shown in Figure 6.9. The value of UI is calculated as in (6.9).

$$UI = n_{layer6} \cdot \overline{n}_{layer5} \tag{6.9}$$

Each motor neuron can lead to multiple expected results. For instance, if motor action M_{31} was activated (go to the bank to get money) we expect that the result of this action is to either get cash, traveler's check, debit card etc, all of which will be a successful completion of this task. So the expectation signal from a motor neuron is sent to all the connected columns on the lower level of the sensory pathway, as shown in Figure 6.9.

In Figure 6.10, M_3 sends the expectation signals to columns 2 - 4 in the sensory columns. Each column in the sensory pathway is simplified to 3 neurons for clarity in the figure. Then the layer 5 neurons of column 2, 3 and 4 are activated. The UI signals are determined considering all combinations of neuronal activations of layer 5 and layer 6 neurons. If any one of the expected sensory inputs is activated, the corresponding UI neuron in this column will not fire, since it receives inhibition from its layer 5 neuron. However, if for instance, unexpected sensory input 1 is activated, it will activate UI neuron in column 1. This may illustrate the case in which we went to bank, but instead we were told that our account was overdrawn and we will get no money.

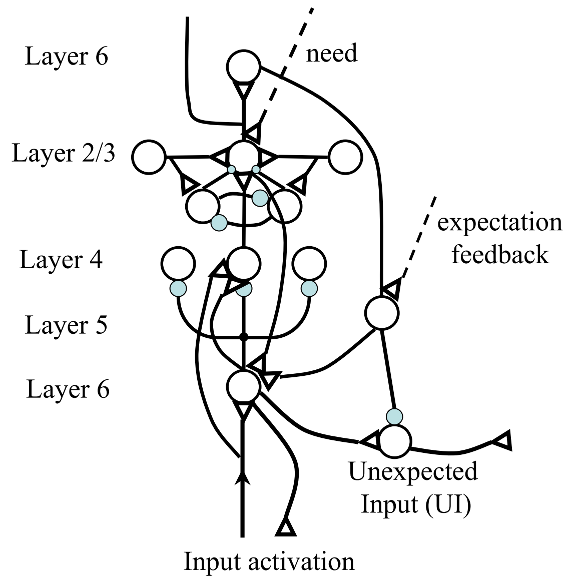


Figure 6.9. Generation of the unexpected input (UI) signal.

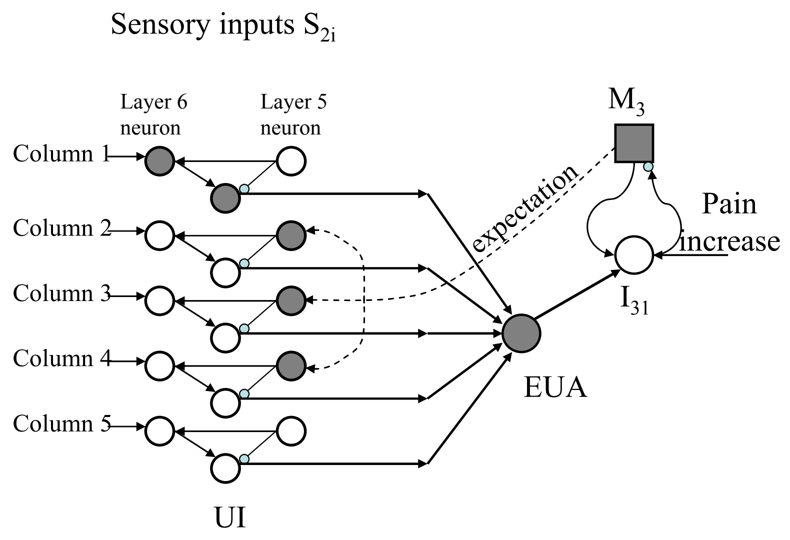


Figure 6.10. Expectation and actual inputs.

End of unsuccessful action (EUA) signal is generated if any of the columns has an active UI, as in (6.10), where *s* denotes all the connected columns in the sensory pathway.

$$EUA = \sum_{j=1}^{s} UI_j \tag{6.10}$$

The activation of EUA neuron results in subsequent action inhibition I_{31}. Finally, an action inhibition neuron may be activated directly by the pain increase signal when the pain level goes above an arbitrary threshold, without a clear sensory signal of unsuccessful action. This may happen, for instance, if the action takes too long time to complete and growing pain triggers its inhibition to abandon this action without its completion.

Notice that a similar feedback mechanism may be used to screen the sensory information for novelty. The machine, by using the feedback signal, similar to the expectation signal, predicts what input representations it expects in the sensory pathway. Any surprises may trigger its attention and may result in learning a new representation or appending the set of acceptable features of the observed object. This attention based screening for novelty may also be a basis for the representational invariance building.

6.3.4 Goal creation and pain network

In previous Section 6.3.3, the goal hierarchy is created based on one primitive pain input. In GCS, there are multiple primitive pain inputs, and normally, the goal hierarchy is complex with many goals and pain centers created both vertically and horizontally. One primitive pain in the goal hierarchy creates and activates one pain tree, which is part of the pain hierarchy. In the example pain tree shown in Figure 6.11, the goal creation unit is shown using a simplified triangle symbol. The nodes on one pain tree can also belong to other pain trees. For instance a need for money could be created as a result of lack of food (related to hunger) as well as satisfy the need to buy clothes (related to feeling cold). Thus, the overlapping of pain trees forms the pain network.

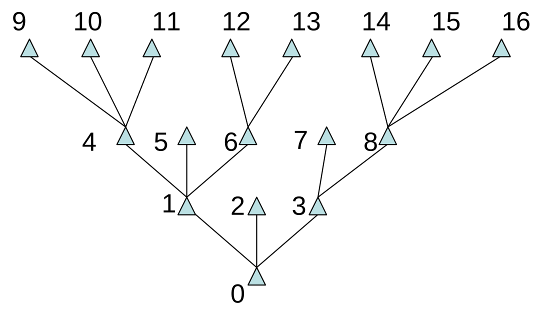


Figure 6.11. An example pain tree.

The goal creation scheme works in a way described as follows. An activated pain node *P*, due to the absence of a certain object *S*, along with the sensory input S_o activates one of the associated motor neurons with the largest signal value, which is found by (6.5). If it is found that this action M_a brings the expected objects and lowers that particular pain *P*, the stimulation and activation links are reinforced, as in (6.4). An expectation feedback link will be built connecting the M_a and the absent object *S*. In addition, the need link is built to connect the pain node and the sensory object S_o. Then, a new pain node which can be inhibited by S_o is created and made to echo *P*. If this pain node was created before, the echo link should be reinforced. A pain tree made of a group of pain nodes is gradually built. These links can be referred to Figure 6.6.

6.4 Goals, subgoals and goal hierarchy

In the goal creation pathway, higher level goals are created to manifest a way to provide the objects necessary for accomplishing lower level goals. They are based on perceptions and actions associated with successful implementation of the lower level goals. The abstract goals are defined as abstract since they are less closely related to the primitive level of goals. Abstract goals do not have direct relationship to lower level goals that once lead to their creation. To realize the lower level goals, it is not necessary to solve its higher-level goals first. It becomes necessary only when certain objects are missing and the lower level actions cannot be performed. On the other hand, to realize a higher level goal, it is not necessary to achieve the lower level goals first, as typically realization of such lower level goals do not provide conditions for attaining higher level goals. Similarly, realization of a higher level goal does not have to satisfy the lower level goal (even the one that triggered its emergence in the goal hierarchy).

These higher level goals are not necessarily the same as “subgoals”. In a goal-driven behavior, there is a need to simplify complex goals by dividing them into subgoals and to implement subgoals in a specific sequence. Many machine learning techniques take this approach. For instance, in [Mur 07], it was used to simplify realization of a complex goal using reinforcement learning. It was demonstrated that if a goal was divided into subgoals, the reinforcement learning was much more effective and was accomplished with a smaller number of exploratory searches. In this work, we use the term “subgoals” to represent the tasks that are elements of a temporal sequence to implement a goal. Subgoals are related to a specific goal on the same level and they are the prerequisite for the final goal. Typically, all the subgoals must be completed to realize the final goal. For example, an agent who needs to go to a specific location in a room needs to go through two doors, A and B, using two corresponding keys, key A and key B. Thus, finding key A to go through door A, and, subsequently, finding key B to go through door B are two necessary subgoals to realize before the ultimate goal is achieved.

It is proposed that subgoals can be found out using the similar goal creation concept, basic goal creation unit and evolving processes in GCS. Given the keys available in the initial exploration, the agent will find out the optimum action is to go through door A using key A when it approaches door A instead of going in other directions. Since key A was involved in the successful action, it creates another level of goal and pain center. The absence of key A generates pain and drives the agent to search for key A. Therefore, search for key A becomes a subgoal. The similar mechanism applies to creating another subgoal as finding the key B.

Structurally, it is easy to confuse higher level goals with complex goals, as both types of goals may lead to a goal hierarchy. We would like to differentiate them by calling higher level goals in GCS as **abstract goals,** and calling the goals that are composed by subgoals as **complex goals.** These two types of goals play different roles in developing embodied intelligence. Application-driven approaches in many intelligent systems focus on complex goals. Subgoals may be further divided into simpler tasks which will lead to a top-down hierarchy of goals and related actions.

Having said that, we must indicate that higher-level goals may become subgoals if they are directly driven by the need to accomplish a lowerlevel goal. For example, finding a store was a subgoal of finding food, but at the same time buying food from a store (and therefore finding the store was a higher-level more abstract goal than observing food). One reason is that finding the store may satisfy other lower-level goals as well. Thus it may be a subgoal of several lower-level goals and, at the same time, a higher level (more abstract goal) than any of them.

6.5 Goal-driven learning system

As discussed earlier, the learning involving GCS is an active process, in which the machine searches for useful actions to solve undesirable pain input perceived by the HSOM sensory pathway. The action taken leads to the change of the environmental states, which is also perceived by the HSOM sensory pathway. The overall diagram of the EI machine interacting with the environment is shown in Figure 6.12. The availability of the needed objects (states) enables the implementation of the action, using a “gate control” fashion. With the needed objects, the action can be carried out and cause the transition of environment states. Otherwise, the GCS will take other action.

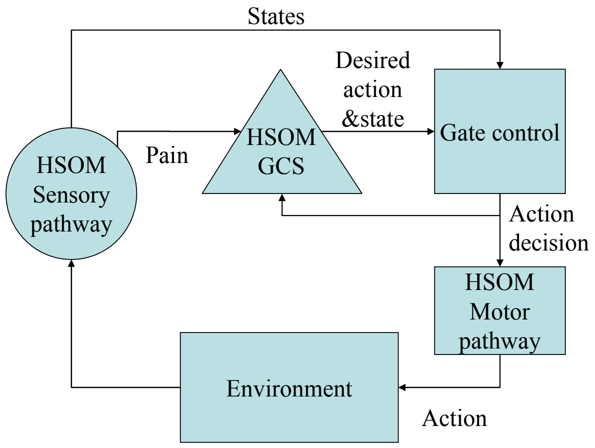


Figure 6.12. EI machine interacts with environment using its three pathways in HSOM.

In previous examples, a single path of the goal hierarchy and related pain tree were illustrated. In the EI design, GCS in HSOM involves multiple pain trees. Pain trees overlap and create a pain (and related goal) network. Thus competition for action exists not only within a single pain tree, but also between different pains within the whole pain network. In general, the stronger the pain, the more attention it gets from EI to lower it. In the learning system, the pain network interacts with the sensory and motor pathways on various levels of abstraction and multiple abstract goals can be pursued concurrently. Such concurrent attainment of various abstract goals may be organized in the GCS despite that, at any given time, the EI focuses its attention on completing one (usually lower level) task. The motor neurons accept stimulations from different pain trees and the strongest pain signals influence the choice of action. The schematic structure of a goal-driven learning system and the interaction among three pathways are shown in Figure 6.13.

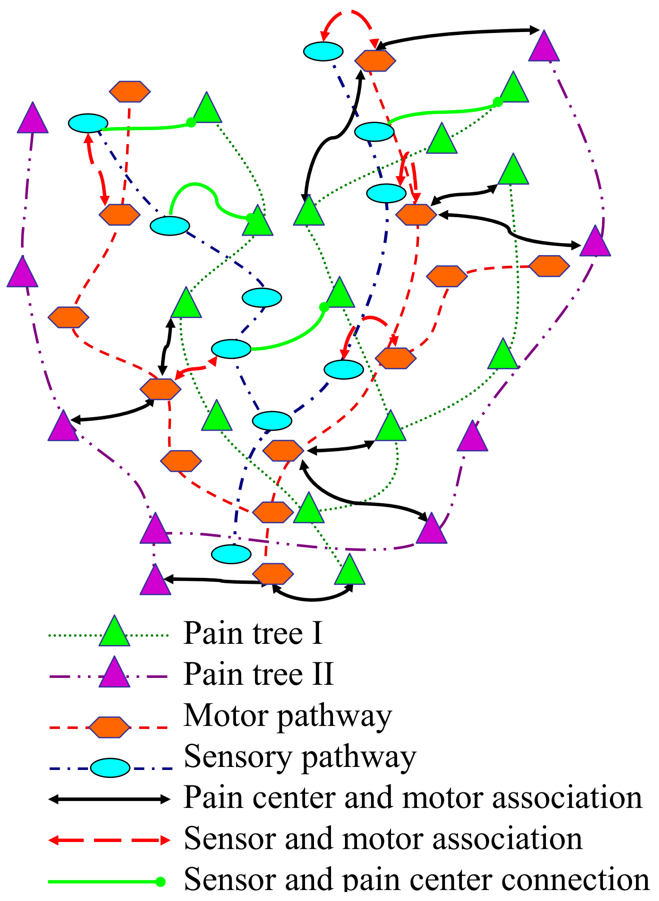

Pain tree I
Pain tree II
Motor pathway
Sensory pathway
Pain center and motor association
Sensor and motor association
Sensor and pain center connection

Figure 6.13. Goal-driven learning scheme.

It is also quite likely that implementation of a higher level goal requires more steps (and more time) and during this implementation another (usually lower level pain) may start to dominate, causing the machine to switch from one task to another one. This requires that the suspended goal maintains memory of its implementation stage, such that the machine can reassume its action after completion of the lower level goal that caused the interruption. Thus using sequential spatio-temporal memory is critical for proper goal management.

During the process of interaction, the reinforcement signals play a significant role in finding new goals, having expectations, and building sensory-motor associations. The reinforcement signals come from the environment, and include autonomous primitive pain signals as well as the teacher's input. The machine's experience built through interactions with the environment affects the associations between different sensory inputs and different goals. The three pathways evolve simultaneously. First, low level representations in the sensory pathway and simple motor actions are developed to manage the primitive pain signals. The machine focuses its attention on objects and actions it can use to lower the pain; thus learning of sensory inputs and motor actions is selective. Not all the objects in the sensory inputs are of interest to the machine, and only those of intent are stored in its memory.

Once representations that are associated with lower-level goals are formed and the machine is capable of formulating its higher-level goals, it may extend its representations to include higher level concepts on the sensory pathway, and learn a sequence of actions to implement them. Thus memories of more abstract entities and useful actions will be formed on the higher levels. Gradually, the complexity of the machine's operations - its intentional memory of objects, actions and goals increases; and its three hierarchical pathways are built concurrently. As a result, complex, long lasting goals may be created and managed by such a system, resulting in a complex, intelligent behavior.

6.6. Simulation and discussion

In this section, let’s assume that the GCS has to learn “how not to go hungry” in an environment in which there are limited food resources and advanced skills are required to get them. We assume that both sensory inputs and motor outputs are symbolic, which means that the machine already learned all the available sensory objects and motor actions, and a source (sensory) or/and a motor action are represented by the symbols. The current state of the environment is determined by the availability of the resources in a binary form. We will also compare effectiveness and performance of learning based on the GCS and RL.

The machine is motivated by one primitive pain signal, which is triggered by low sugar level in the blood. This primitive pain automatically increases if the machine does not eat for some time. All the symbols representing sensory objects and motor actions are shown in Table 6.1. Overall, there are 6 sensory objects and 5 possible motor actions. Initially, all sensors are randomly connected with all the motors and there are 25 possible sensory-motor pairs, as listed in Table 6.2. It is expected that after learning, the association between sensors and motors are learned and only the meaningful pairs will remain. All other sensory-motor pairs are either undesirable or have no effect on the machine’s perceived success. The desired sensory-motor pairs and their effect on the environment are also listed in the following Table 6.1.

Table 6.1

All Sensory Object and Motor Actions

SENSORY	SCALING FACTOR	MOTOR	EFFECT ON THE ENVIRONMENT	
			INCREASES	DECREASES
Sugar level				
Food	10	Eat	Sugar level	Food supplies
Grocery	10	Buy	Food supplies	Money at hand
Bank	2	Withdraw	Money at hand	Spending limits
Job	$1e^8$	Work	Spending limits	Job opportunities
School	$1e^8$	Study	Job opportunities	-

Initially there are plenty of resources around, indicated by a high probability of availability of all the sensory inputs. This allows the machine to learn by exploration how to satisfy its primitive goal (reduce hunger). However, as the machine uses them, the original resources are gradually depleted and need to be replaced. Thus the machine needs to learn how to use other actions to provide the resources. It will use the GCS to find out goal hierarchy and learn how to accomplish different level of goals.

Table 6.2

All Possible Action and Object Combinations

1 Eat food	2 Eat money	3 Eat bank	4 Eat job	5 Eat school
6 Spend food	7 Spend money	8 Spend bank	9 Spend job	10 Spend school
11 Withdraw food	12 Withdraw money	13 withdraw bank	14 Withdraw job	15 Withdraw school
16 Work food	17 Work money	18 Work bank	19 Work job	20 Work school
21 Study food	22 Study money	23 Study bank	24 Study job	25 Study school

A sensory input probability of various items of category c_i (availability of resource) is described by the function:

$$f_{ci}(k_{ci}) = \frac{1}{1 + \frac{k_c}{\tau_c}} \qquad (6.11)$$

where: τ_c is a scaling factor (resource decline rate), k_c stands for the number of the times a resource was used (initially set to zero). Each time a specific action is taken by the machine (for instance the machine withdraws money from the bank account) the corresponding k_c is increased by 1 decreasing the likelihood of this particular resource. This resource can be renewed by invoking a higher-level goal after which k_c at a particular level is reset to 0. For instance if grocery was bought, the counter for food was rest to zero and at the same time the counter for money increased by 1. Also, it is noted that the smaller the scaling factor τ_c is, the sooner the resource gets exhausted. "Sugar level" is considered always low, and triggers increasing primitive pain unless it is renewed by the proper action. Therefore, its likelihood is always 0. In this experiment the scaling factor used for these resources is listed in Table 6.1 as well.

6.6.1 Goal creation system learning scheme in this experiment

Initially, the primitive pain P_1 has random stimulations to all the motor actions and the sensory neurons have random activation connections to all the motor neurons as well. Also, we initialized 6 pain centers due to lack of the 6 corresponding resources. All the pains are randomly connected to all the other pains, assuming other pains can be the higher level pains.

It is intended to show that after the learning, the machine learns the optimum action to solve the primitive pains, finds the connection from each goal to its abstract goals, finds out the ontology among objects related by the pain centers, builds the "need" and "expectations" among motor actions and sensory objects, and builds the meaningful association among sensory and motor neurons.

The initial information and the information to be learned or created through goal creation process are listed in Table 6.3 as follows. The learning process is described in the earlier sections, with a brief flow chart shown in Figure 6.14.

Table 6.3

Initialization and Learned Information in the Goal Creation System Scheme

	Randomly initialized information	Learned information
1	Stimulation weights: W_{MP}	Optimized W_{MP} (from pains to motors)
2	Activation weights: W_{MS}	Optimized W_{MS} (from sensors to motors)
3	Pain connection weights: W_{pps}	Optimized W_{pps}, and hierarchical structure of pain centers
4		"Need" connections W_{SP} (from pains to sensors)
5		"Expectation" connections W_{SM} (from motors to sensors)

In the experiment, in each time iteration, the machine acts only on the most active pain signals among all the active ones. When it is acting according to the j^{th} pain node, P_j, the action is M_a chosen based on the stimulation weights W_{MP} from P_j to all the motors, as in (6.5). The most correlated sensory object from the existing sensory-motor association is chosen. The result of the action will lead to positive or negative reinforcement learning.

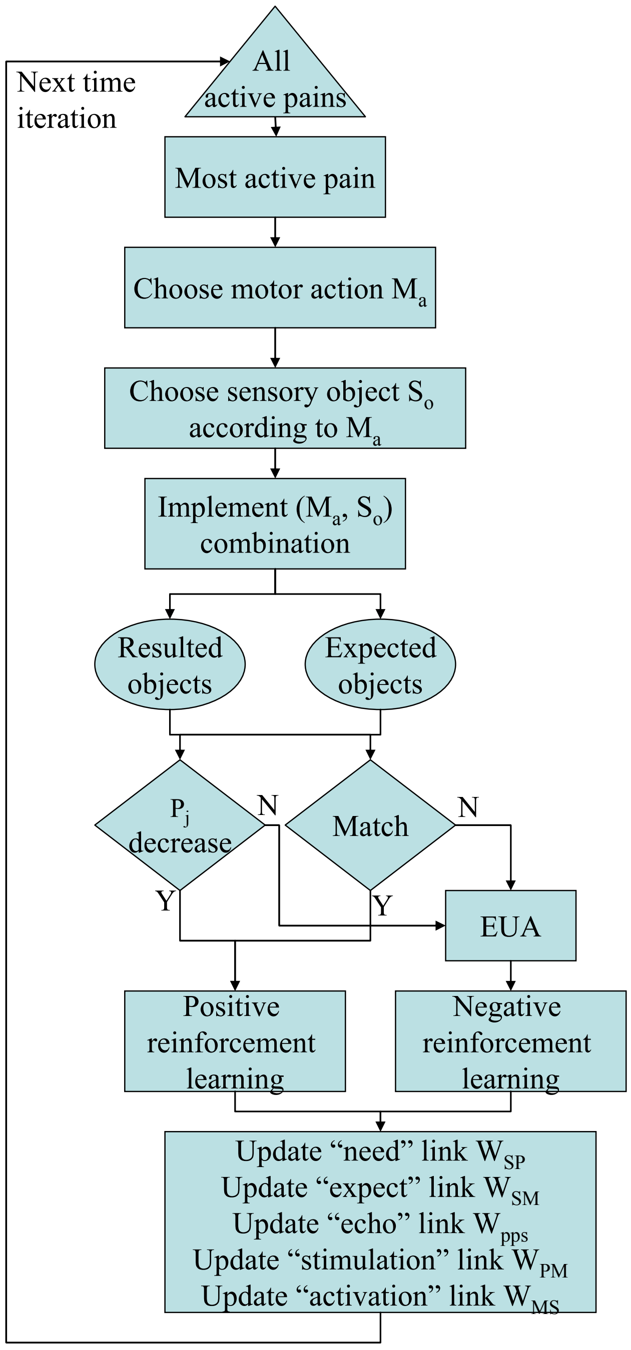

All active pains
Next time iteration
Most active pain
Choose motor action M_a
Choose sensory object S_o according to M_a
Implement (M_a, S_o) combination
Resulted objects
Expected objects
P_j decrease
N
Y
Match
N
Y
EUA
Positive reinforcement learning
Negative reinforcement learning
Update "need" link W_{SP}
Update "expect" link W_{SM}
Update "echo" link W_{pps}
Update "stimulation" link W_{PM}
Update "activation" link W_{MS}

Figure 6.14. Goal creation experiment flow chart.

6.6.2 Reinforcement learning scheme in this experiment

A typical AC system for RL is adopted in this experiment for performance comparison. The algorithm of RL used is actor-critic architecture shown in Figure 6.15. In the RL scheme, the machine is only triggered by the primitive pain as its reinforcement signal (reward). It has only one primitive goal, which is to reduce the primitive pain. The machine does not create any higher-level pains or goals. In this scheme, the optimal mapping from states to actions is to be learned during interaction with the external environment based on the primary primitive reinforcement.

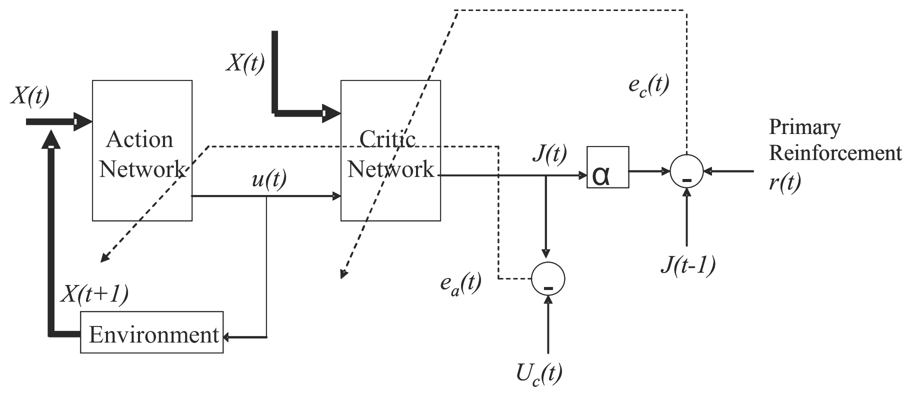


Figure 6.15. Actor-critic architecture for RL.

The actor-critic architecture contains two components: the action network and the critic network, and both networks are implemented by a multi-layer perceptron (MLP) in this experiment. The action network determines the action *u(t)* based on current states *X(t)*. The critic network evaluates the state-action value *J* according to *{X(t), u(t)}*. The

J(X(t), u(t)), also denoted as *J(t)* value is defined as

$$J(X(t),u(t)) = R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{6.12}$$

where $\gamma, 0 \le \gamma \le 1$, is the discount rate.

The critic network is trained using the temporal difference method [Sut 98]. The critic network directs the action network to produce better action so that *J(t)* is maximized; thus the action network is trained. A more detailed implementation of this algorithm can be found in [Si 01].

Since the agent has no prior knowledge, the reasonable associations among different objects and actions are not yet built; therefore all possible object- action combinations are considered. The algorithm is described below.

Reinforcement learning using AC method:

Step 1). The agent receives states information *X(t)* and the reward signal *r(t)* from the environment.

The *X(t)* shows the availability of the resources as a binary vector, with "1" indicating being available. The reward signal *r(t)* is related to the current states *X(t)*. In this experiment, the *r(t)* is the pain input signal, which is due to the "lack of sufficient sugar level".

Step 2). Action network (AN) determines the action *u(t)* from 25 possible action-object combinations based on current states *X(t)*.

The *u(t)* is in the form of a binary vector as well, with "1" indicating the selected action and "0" for the rest of the actions.

Step 3). Critic network (CN) determines the value of this state-action pair *J(t)*.

Step 4). Using the reward signal *r(t)*, CN is trained by TD method. The error function of CN is show in (6.13).

$$\begin{aligned} e_c(t) &= \gamma J(t) - [J(t-1) - r(t)] \\ E_c(t) &= \frac{1}{2} e_c{}^2(t) \end{aligned} \tag{6.13}$$

Step 5). The weights in CN are updated according to gradient descent,

$$\begin{aligned} w_c(t+1) &= w_c(t) + \Delta w_c(t) \\ \Delta w_c^{(2)}(t) &= l_c(t)\left[-\frac{\partial E_c(t)}{\partial w_c^{(2)}(t)}\right] \\ \Delta w_c^{(1)}(t) &= l_c(t)\left[-\frac{\partial E_c(t)}{\partial w_c^{(1)}(t)}\right] \end{aligned} \tag{6.14}$$

where $l_c(t)$ is the learning rate of CN.

Step 6). The CN reevaluate the value *J(t)*.

Step 7). The AN is trained in order to produce action *u(t)* which can have a desired value *J(t)*. The error function is formulated as,

$$\begin{aligned} e_a(t) &= J(t) - U_c(t) \\ E_a(t) &= \frac{1}{2} e_a{}^2(t) \end{aligned} \tag{6.15}$$

where $U_c(t)$ is the desired value.

The updating algorithm is similar to that in CN. By a gradient descent rule,

$$\begin{aligned} \Delta w_a(t) &= l_a(t)\left[-\frac{\partial E_a(t)}{\partial w_a(t)}\right] \\ \frac{\partial E_a(t)}{\partial w_a(t)} &= \frac{\partial E_a(t)}{\partial J(t)} \frac{\partial J(t)}{\partial u(t)} \frac{\partial u(t)}{\partial w_a(t)} \end{aligned} \tag{6.16}$$

where $l_a(t)$ is the learning rate of AN.

Step 8). Apply the determined action *u(t)* to the environment, and the environment will give new states *X(t+1)*. Repeat Step 1) to Step 8).

The performance is shown in the following section.

6.6.3 Performance comparison

6.6.3.1. RL scheme performance

Using the RL scheme, in a simulated trial, the agent runs for 600 time steps. The pain input (hunger) during this trial is shown in Figure 6.16. Initially it takes the RL agent 27 iterations to learn to control the pain input by performing "eat food" action. Then the pain input goes to 0, as shown in Figure 6.16 (a). But as the food resource is depleted the agent needs to learn another action to renew food resource. In iteration 129, the agent accidently takes the action "buy the food" and restores the food supply likelihood, as illustrated in Figure 6.16 (d). The primitive pain is under control as the action "eating" was repeated. However, the food probability is gradually reduced and the agent does not seem to learn how to restore the food. After iteration 322, it never returns to "eat food" action even though the food is available and the primitive pain increases. As it is evident from the simulation result, it takes a long time for the RL mechanism to adjust to the changing environment and learn efficient actions.

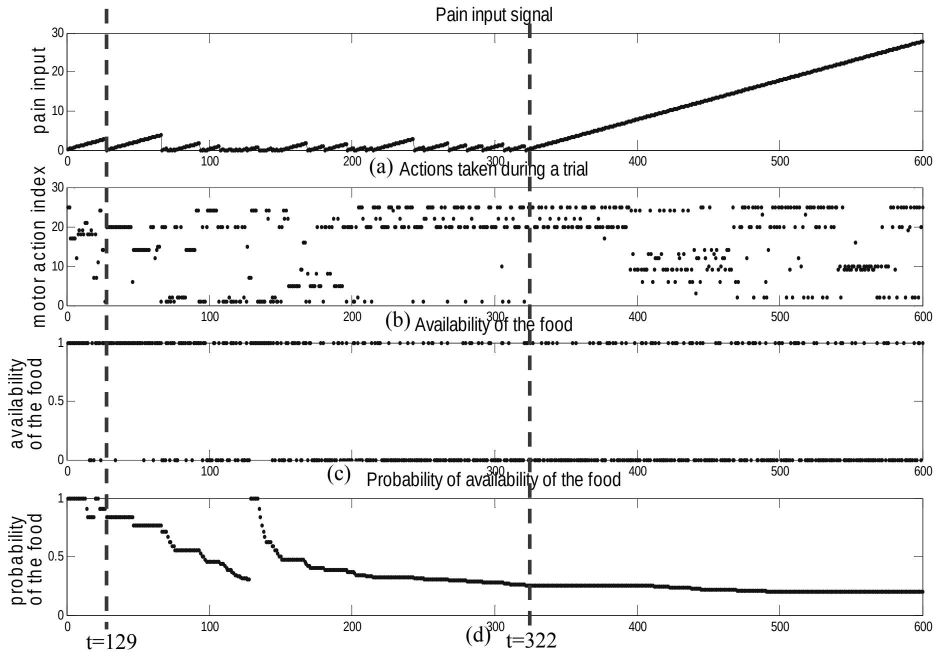


Figure 6.16. Results from RL experiment.

The overall performance by RL scheme on this experiment can be evaluated from multiple trials, as shown in Figure. 6.17. In general, the pain input was not effectively suppressed. It is observed in several runs that the changes in environment happened too quickly for the RL system to keep up developing new skills.

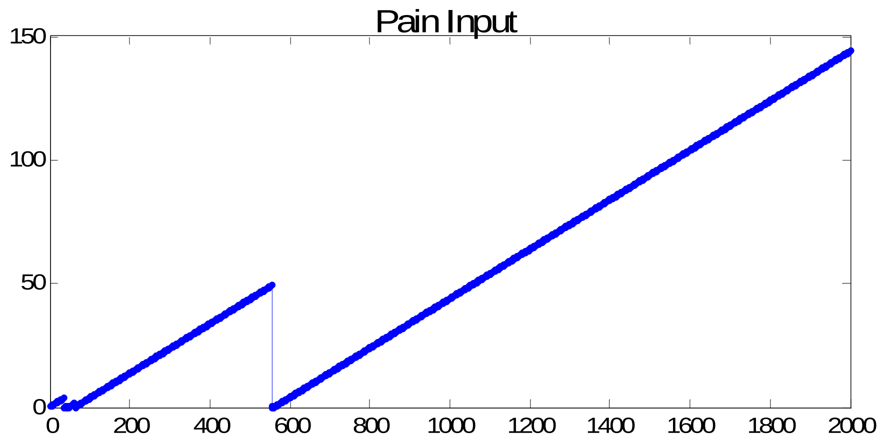


Figure 6.17. The average pain signals in 10 trials for RL scheme.

6.6.3.2 GCS scheme performance

On the other hand, using GCS, as soon as the machine learns how to satisfy a lower level pain, it identifies the environment condition (the sensory-motor pair) that helps to remove the pain (for instance by eating the food) and creates an abstract pain that is activated when these conditions are not met (for instance when no food is present). The agent uses its GCS approach to learn what to do, and how to adjust to changing environment conditions.

A typical result of CGS simulation is shown in Figure 6.18. This figure shows dynamic changes for the pain signals (including the primitive pain) over 600 iterations. At first, the only pain that machine receives is the primitive pain. Once the machine learns that eating food reduces the primitive pain, the lack of food becomes an abstract pain. As there is less and less food in the environment, the primitive pain increases again (since the machine cannot get the food) and the machine must learn how to get the food (buy grocery). Once it learns this, a new pain source is created and so on. Notice that the primitive pain is maintained under control eventually in spite of changing environment conditions. In this presented trial, the machine can learn to create, develop and solve all abstract pains in this experiment within 300 iterations. In this experiment, school opportunity is designed as always available. Therefore, it is noted in Figure 6.18 that the abstract pain for “lack of school opportunity”, although was created when solving lower-level pains, it was never activated and stayed zero.

Figure 6.19 show a scatter plot that illustrates selection of a specific action during a trial of the GCS experiment. The corresponding sensory-action pairs are denoted for the useful actions in the figure. It is seen that the machine makes association between sensor and actions, learns the useful actions while exploring environment and selecting the most useful actions very repetitively. There are still several sensory-action pairs left which are not meaningful and are selected by the machine sometimes. It is expected that with longer period of interaction, the machine will only maintain the meaningful pairs and select them only to have proper interaction with the environment.

The average of all abstract pain signals obtained on the basis of 10 such experiments is shown on Figure 6.20. As we can observe, the machine learns to control all abstract pains and maintains the primitive pain signal on a low level in demanding environment conditions.

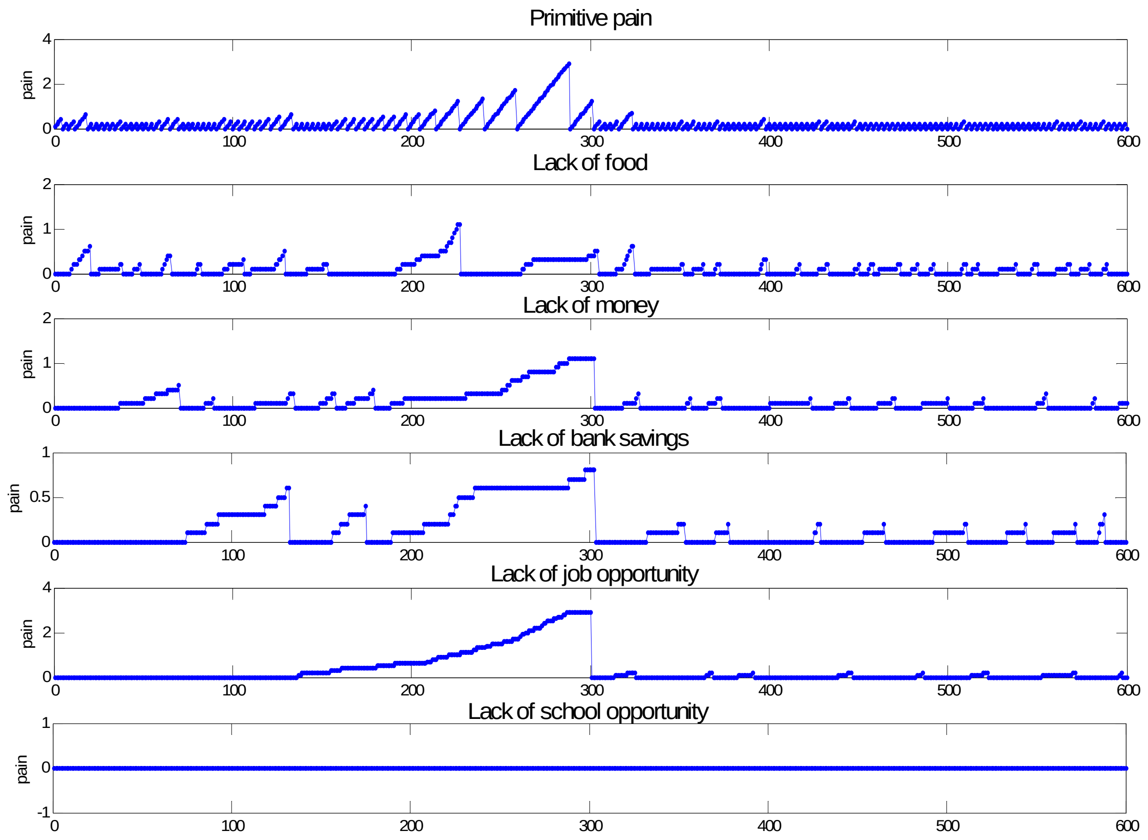


Figure 6.18. Pain signals in GCS experiment.

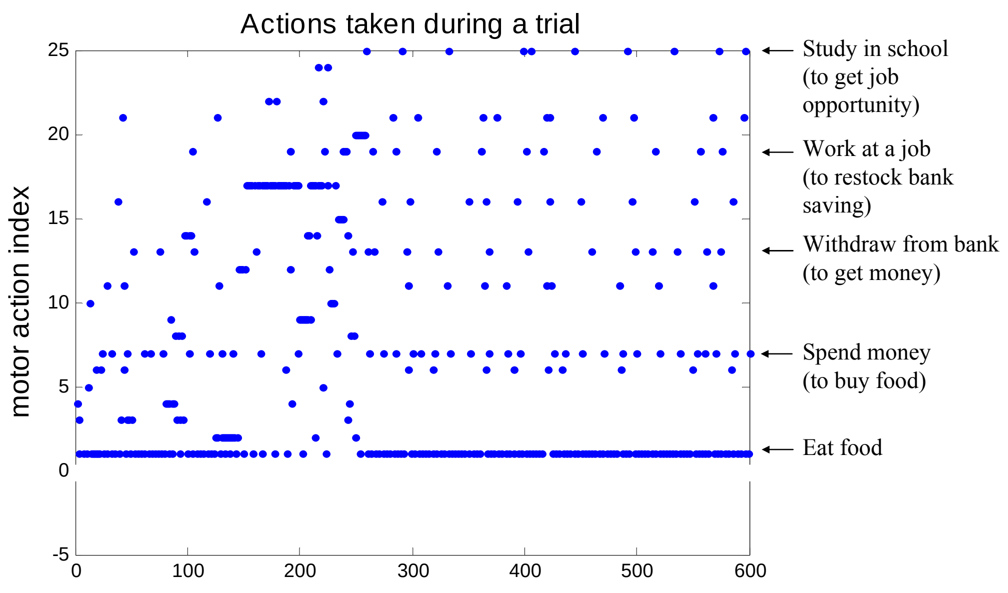


Figure 6.19. Action selections in GCS experiment.

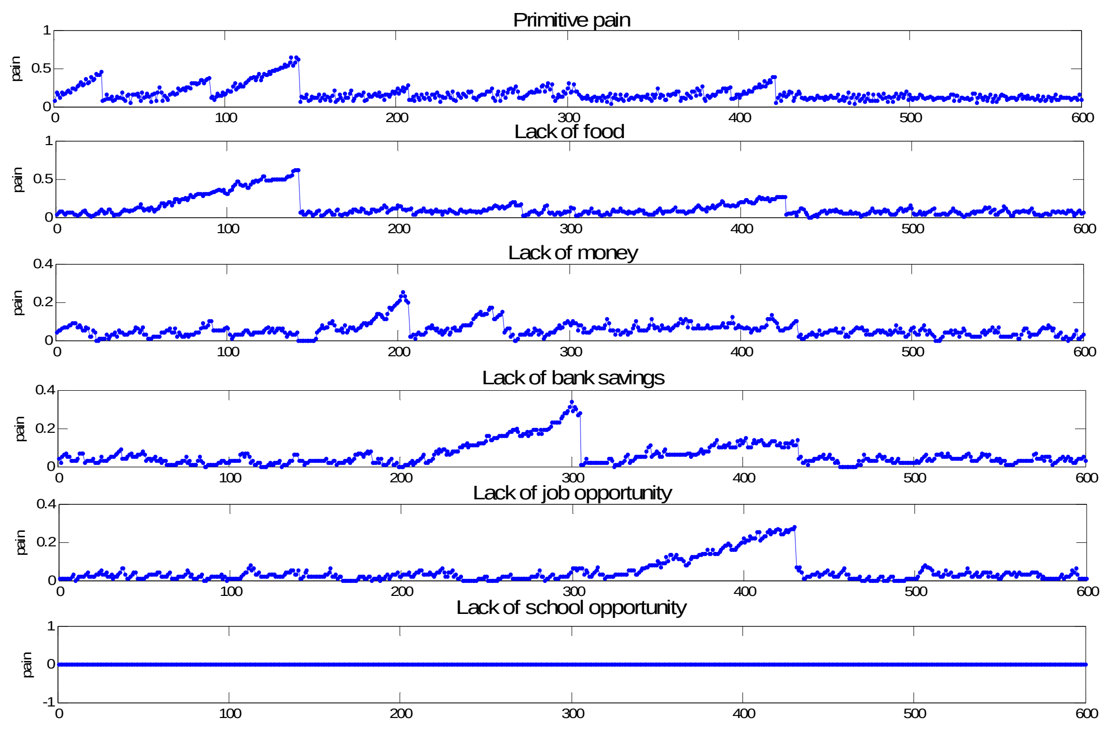


Figure 6.20. The average pain signals in 10 GCS simulations.

6.7. Conclusions

This chapter presented a goal creation system that motivates embodied intelligence to learn how to efficiently interact with the environment. It develops higher-level abstract goals and increases the internal complexity of the stored representations and skills.

The proposed GCS is a mechanism that motivates the creation of more complex knowledge from simpler ones by the pain signals from which the machine suffers. The pain centers in the GCS hierarchy connect the sensory and motor pathways, generate the machine's needs and expectations and find the ontology among objects as well as optimum actions to achieve goals.

Using the GCS learning scheme, the complex knowledge, skills and goals evolve from simpler ones through learning. Thus, in this model, knowledge building is goal-driven activity. Such bottom up goal creation, representation building and action learning adopted in this work is more natural than a top-down approach. It does not define the upper limit of the machine complexity, nor the amount of learning or developed skills, thus it is conducive to developing a general intelligence. The only limitation for developed intelligence in such scheme comes from the limited resources used to build the intelligence memory structures.

In contrast to classical RL, GCS generates an internal reward associated with the abstract goal that the machine was able to accomplish. This makes the reinforcement process not observable, and to some degree makes the machine less controllable than one whose operation is based on classical RL. The machine's actions are more difficult to understand and explain by an external observer, thus the machine behaves more like an independent intelligent being rather than a well-trained robot.

To the contrary, top-down learning approach is based on top-level complex goals implicitly imposes the upper limit of knowledge. Such approaches lead to effective design of an application and goal specific machine. It has severe drawbacks from the point of view of developing general intelligence. Firstly, it requires initializing machine with a substantial knowledge about its environment and complex representation of perceptions in relation to the complex goal. This knowledge and its organizational structure expressed by complex interactions between various representations, must be explicitly entered by a human designer (a computer programmer). Development and maintenance of such knowledge structure is very costly and time consuming. The second drawback is lack of a natural mechanism to create knowledge and to learn more complex representations from simpler ones. While there may be a little need for such mechanism if the environment is well described through the initial built-in knowledge, lack of such mechanisms may be detrimental to further development of a machine's knowledge and skills in changing tasks and environments. It was demonstrated that this type of system learns better and faster than traditional reinforcement learning systems.

In summary, the presented goal creation system motivates a machine to act and develop its cognitive skills in response to externally applied pain signals. It also helps the machine to perceive its environment, learn with a purpose, and respond to changes in environment.

CHAPTER 7: CONCLUSIONS AND FUTURE WORK

Biologically inspired and hardware-oriented, research work and structural design concepts presented in this dissertation create a promising direction in the field of EI by providing:

(1). Hierarchical learning memory with efficient and global optimized training algorithms and quantitative measure to detect overfitting for supervised learning.

(2). Hierarchical self-organizing memory in unsupervised learning based on sparse connectivity.

(3). Goal creation systems to stimulate the continuous, intentional, and active learning and to enable reinforcement learning.

(4). A uniform and promising framework to design working models for EI is proposed based on our definition of intelligence and its embodiment, including the detailed design principles and presenting the HSOM and GCS as two essential elements of EI.

These essential elements for building EI are proposed and studied considering their neuronal structure, learning algorithms, memory capacity, fault tolerance and ability to create goals. The design concepts aim to build intelligence on a simple and uniform neural structure.

Such intelligent machine will have many more successful applications, which can greatly affect and amazingly change our life. In addition, this area, where modern technology including VLSI and FPGA can be applied, will create exponentially growing demand for electronic hardware, will invigorate the electronics industry, and may create an intensive economical growth in the years to come. Hopefully in the near future, the existence of non-biological intelligence will supplement the biological one and empower it many times over its current potential.

Embodied Intelligence can be implemented in hardware or software. In the following comparison, hardware implementation will use multiple processors working concurrently, while software will run on a single central processing unit (CPU). The processing speed of today's hardware is much higher than that needed to simulate a single neuron in a real time. Thus in hardware implementation a hybrid approach can be used, where a group of neurons is simulated at each concurrent processor, and a number of neurons simulated is set to deliver a real time operation. This depends on the speed of operation. For instance, if a concurrent hardware operates at 200 MHz, a single operation may take 5 nsec. So in 5 msec, which is needed for a real time neuronal response, up to 1 million operations can be performed on a single processor. This will set a limit on the number of neurons that each concurrent processor can simulate and put requirements for the system size. For instance, at that speed, a system with human level neuron capacity would require 10^6 processors working concurrently (assuming that each neuron requires only 10 operations for real time processing). With the current distributed computing devices, software implementation can be instantiated into parallel computation, which will greatly improve the system efficiency.

Software implementation is a convenient choice today due largely to inadequate hardware design or programming tools for easy implementation and experimentation with cognitive mechanisms. However, software simulation has inherent limitations for implementing real-time operation of EI with brain level complexity. The major limitation comes from replacing the network of interconnected concurrent processors by a single CPU. Not only does the CPU have to run n times faster to compensate for the combined processing power of *n* neurons, but also it must simulate the complexity of the interconnections as well. With the average number of interconnections growing with the size of the network and the time the machine spends updating the interconnections and simulating signal transformations through these interconnections, interconnection processing may dominate. For instance, to simulate the human brain with 10^{11} neurons and each with an average of 10000 connections per neuron, the CPU must run 10^{15} times faster than biological neurons. With the average response time of a neuron on the level of 5msec, the CPU would have to perform 10^{15} operations in 5 msec. Assuming that a single operation can be performed in one clock cycle, this would require the clock speed of 200,000,000 GHz, (or 10mln times faster than current computers). During the time period, which corresponds to such switching frequency, light travels on the distance of 1.5 nm. Thus even if the switching is performed with the speed of light, the device geometry should be comparable with the particle size of silicon to operate at such speed. Operating on such small scale would require going beyond the single electron switching or spin electronics. Thus a single CPU may never be able to perform the real-time operations of a system with the complexity of the human brain, unless the computational models are simplified. For instance, instead of performing analysis of the entire network at each time step only, selected pathways could be activated and updated in a fashion similar to event driven simulation used in the logical circuits. This requires computational and learning schemes that are based on sparse coding in sparse neural structures. It is very likely that software implementation will be limited to modeling small sized networks necessary to develop mechanisms for EI. It is our opinion that only hybrid hardware approach may reach complexity of human level intelligence in the predictable future.

EI, as an interdisciplinary area, requires not only progress in engineering. A significant growth in other technologies including computer science, nanotechnology, medical scanning, physiology, neuroscience, and psychology, could invigorate progress of this rewarding mission. Building structural and functional organization of EI has a long way to go in order to achieve human level intelligence and to benefit from the developing technologies. Having the promising and uniform framework and design principles prepared, the future work will be done more consistently.

In the future work, a more integrated system will be developed using all the elements and systems proposed and developed in this work. The motor pathway development will be further studied. A closed loop system, including sensory inputs, decision making, action selection and motor outputs, will be built to interact with a certain environment and test over more complex problems and over longer period of time. The performance of the machine on goal creation, complex task accomplishment, and learning capability will be further evaluated. In that way, we can see how the proposed designs in this work can be further developed into an intelligent machine.

I feel fortunate and privileged to get the chance to continue my study and research on machine intelligence, to build integrated reconfigurable intelligent systems using advanced technologies, and to keep pursuing their applications into various areas.

REFERENCES

[Abb 96] L. F. Abbott, E. T. Rolls, M. J. Tovee, “Representational capacity of face coding in monkeys,” *Cerebral Cortex*, vol.6, pp.498-505, 1996.

[Aha 90] S. C. Ahalt, A. K. Arishnamurty, P. Chen, and D. E. Melton, “Competitive learning algorithms for vector quantization,” *Neural Networks*, vol. 3, pp. 277–291, 1990.

[Aka 74] H. Akaike, “A new look at the statistical model identification,” *IEEE Trans. on Automatic Control*, vol.19, no.6, pp.716-723, Dec, 1974.

[Alp 91] E. Alpaydın, “GAL: Networks that grow when they learn and shrink when they forget,” ICSI Berkeley, CA, *Tech. Rep. TR-91-032*, 1991.

[Ama 93] S, Amari, “Neural representation of information by sparse encoding,” *Brain Mechanisms of Perception and Memory from Neuron to Behavior*, Oxford University Press, pp. 630-637, 1993.

[Ama 97] S. Amari, N. Murata, K. Muller, M. Finke, and H. H. Yang, “Asymptotic statistical theory of overtraining and cross-validation,” *IEEE Trans. On Neural Networks*, vol.8, no.5, pp.985-996, Sep, 1997.

[And 93] J. Anderson, *Rules of the Mind*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1993.

[And 05] J. Anderson, “Learning in sparsely connected and sparsely coded system,” *Ersatz Brain Project working note,* 2005.

[Bad 96] R. Baddeley, “An efficient code in V1?” *Nature*, vol.381, pp.560-561, 1996.

[Bak 04] B. Bakker, J. Schmidhuber, “Hierarchical Reinforcement Learning Based on Subgoal Discovery and Subpolicy Specialization,” *Proceedings of the 8th Conf. on Intelligent Autonomous Systems*, Amsterdam, The Netherlands, pp. 438-445, 2004.

[Bar 72] H. B. Barlow, “Single units and sensation: a neuron doctrine for perceptual psychology?” *Perception*, vol.1, pp.371-394, 1972.

[Bar 83] A. G. Barto, R. S. Sutton, and C. W. Anderson, “Neuron like adaptive elements that can solve difficult learning control problems,” *IEEE Trans. Syst., Man, Cybern*. vol.13, pp. 834-847, 1983.

[Bar 98] R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA 1998

[Bar 03] A. Barto, S. Mahadevan, “Recent Advances in Hierarchical Reinforcement Learning,” *Discrete Event Dynamic Systems: Theory and Applications*, Springer Netherlands, vol.13, no. 4, pp. 343-379, October, 2003.

[Bea 93] L.P. Beaudoin and A. Sloman, “A study of motive processing and attention,” In Prospects for Artificial Intelligence: Proceedings of AISB-93, pages 229–38, 1993.

[Bea 94] L.P. Beaudoin. *Motive processing and attention. PhD thesis*, School of Computer Science, University of Birmingham, 1994.

[Bel 61] R. Bellman, Adaptive Control Processes: A Guided Tour, Princeton University Press. 1961.

[Bel 97] A.J. Bell, T.J. Sejnowski, “The independent components of natural images are edge filters,” *Vision Research*. Vol.37, No. 23, pp.3327-3338, 1997.

[Ber 97] M. J. Berry, G. Linoff, *Data Mining Techniques*, NY: John Wiley & Sons, 1997.

[Bis 03] J. W. Bisley, M. Goldberg, “Neuronal activity in the lateral intraparietal area and spatial attention,” *Science,* vol. 299, pp. 81-86, 2003.

[Bog 97] Z. Boger, and H. Guterman, “Knowledge extraction from artificial neural network models,” *IEEE Systems, Man, and Cybernetics Conference*, Orlando, FL, 1997.

[Bro 91] R. A. Brooks, “Intelligence without reason,” *Proc. IJCAI-91*, pp. 569-595, 1991.

[Bro 02] R. A. Brooks, *Flesh and Machines: How Robots Will Change Us*, Pantheon, 2002.

[Cam 01] L. S. Camargo, T. Yoneyama, “Specification of training sets and the number of hidden neurons for multilayer perceptrons,” *Neural Computation*, vol.13, pp.2673-2680, 2001.

[Car 83] S. Card, T. Moran, and A. Newell, *The psychology of human-computer interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates, 1983.

[Cor 00] M. Corbetta, J. M. Kincade, J. M. Ollinger, M. P. McAvoy, and G. L. Shulman, “Voluntary orienting is dissociated form target detection in human posterior parietal cortex,” *Nature Neuroscience*, vol. 3, no. 3, pp. 292-297, March, 2000.

[Cou 92] R. Coultrip, R. Granger, G. Lynch, “A cortical model of winner-take-all competition via lateral inhibition,” *Neural networks*, vol.5, pp.47-54, 1992.

[Csi 96] M. Csikszentmihalyi, *Creativity-Flow and the Psychology of Discovery and Invention,* New York: Harper Perennial, 1996.

[Day 95] P. Dayan, R. S. Zemel, "Competition and multiple cause models," *Neural Computation*, Vol.7, pp. 565—579, 1995.

[Day 01] P. Dayan, L. F. Abbott, *Theoretical neuroscience: computational and mathematical modeling of neural systems*, MIT Press, pp. 378-383, 2001.

[Def 98] S. R. y Cajal, E. G. Jones, J. DeFelipe, *Cajal on the Cerebral Cortex: An Annotated Translation of the Complete Writings (History of Neuroscience, No 1)*, Oxford Univ. Press, New York, 1998.

[Del] Delve Datasets, University of Toronto, [Online] Available: http://www.cs.toronto.edu/~delve/data/datasets.html

[Der 97] S.W. G. Derbyshire, A. K. P, Jones, F. Gyulai, et al. "Pain processing during three levels of noxious stimulation produces differential patterns of central activity," *Pain,* vol. 73, pp.431–445, 1997.

[Des 89] R. Desimone, L. G. Ungerleider, "Neural mechanisms of visual processing in monkeys," In F. Boller, & J. Grafman (Eds.), *Handbook of neurophysiology*, vol. 2, Chap. 14, pp. 267-299, Elsevier, 1989.

[DeW 03] M. DeWeese, M. Wehr, A. Zador, "Binary spiking in auditory cortex", *Journal of Neuroscience*, vol.23, pp.7940-7949, 2003.

[Erd 05] D. Erdogmus, O. Fontenla-Romero, J.C. Principe, A. Alonso-Betanzos, E. Castillo, "Linear-least-squares initialization of multilayer perceptrons through backpropagation of the desired response", *IEEE Trans. On Neural Networks*, vol.16, no.2, pp. 325-337, 2005.

[Fel 91] D. J. Felleman, D. C. Van Essen, "Distributed hierarchical processing in the primate cerebral cortex," *Cerebral Cortex*, vol.1, pp. 1-47, 1991.

[Fie 94] D. J. Field, "What is the goal of sensory coding?" *Neural Computing*, vol. 6, pp. 559-601, 1994.

[Fis 05] A. Fish, V. Milrud and O. Yadid-Pecht, "High-speed and high-precision current winner-take-all circuit," *IEEE Trans. on Circuits and Systems II*, vol. 52, no. 3, March, 2005

[Fol 02] P. Foldiak, "Sparse coding in the primate cortex," *The handbook of brain theory and neural networks*, 2nd edition, MIT Press, 2002.

[Fre 90] M. R. Frean, "The upstart algorithm: A method for constructing and training feedforward neural networks," *IEEE Trans. on Neural Networks*, vol. 2, pp. 198–209, 1990

[For 91] D. B. Forgel, "An information criterion for optimal neural network selection," *IEEE Trans. On Neural Networks*, vol.2, no.5, pp.490-497, Nov, 1991.

[Gal 90] S. I. Gallant, "Perceptron-based learning algorithms," *IEEE Trans. on Neural Networks*, vol. 1, no. 2, pp. 179-191, Jun, 1990.

[Gem 92] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Computation*, vol. 4, pp. 1-58, 1992.

[Geo 87] M. P. Georgeff and A.L. Lansky, "Reactive reasoning and planning," *In Proc. of the 6th National Conf. on Artificial Intelligence*, pp 677–681, 1987.

[Geo 89] M.P. Georgeff and F.F. Ingrand. “Decision-making in an embedded reasoning system,” *Proc. of 11th Int. Joint Conf. on Artificial Intelligence*, vol. 2, pp. 972–978, 1989.

[Geo 05a] D. George, J. Hawkins, “A Hierarchical Bayesian Model of Invariant Pattern Recognition in the Visual Cortex,” *IEEE proceedings of the International Joint Conference on NN*, 2005.

[Geo 05b] D. George, J. Hawkins, “Invariant Pattern Recognition using Bayesian Inference on Hierarchical Sequences,” *Redwood Neuroscience Institute Technical Report*, 2005. Available: http:\\www.cnbc.cmu.edu.

[Gro 76] S. Grossberg, “Adaptive pattern classification and universal recoding I: Parallel development and coding of neural feature detectors,” *Biological Cybernetics*, Vol.23, pp. 121—134, 1976.

[Gro 98] S. Grossberg, *How does the cerebral cortex work? Learning, attention, and grouping by the laminar circuits of visual cortex*, Technical report CAS/CNS-97-023, 1998.

[Gro 00] S. Grossberg, “Linking visual cortical development to visual perception,” *Tech Report CAS/CNS-2000-026*, Boston university, 2000.

[Haw 04] J. Hawkins, S. Blakeslee, *On Intelligence,* Times Books, Henry Holt and Company, New York, NY, 2004.

[Hay 99] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, 1999.

[Heb 49] D. O. Hebb, *The organization of behavior*. New York: Wiley, 1949.

[Her 00] N. Hernandez-Gardiol, S. Mahadevan, “Hierarchical Memory-Based Reinforcement Learning,” *NIPS’02*, pp. 1047-1053, 2000.

[Hil 01] D. L. G. Hill and P. Batchelor, “Registration methodology: concepts and algorithms,” *Medical Image Registration*, J. V. Hajnal, D. L. G. Hill, and D. J. Hawkes, Eds. Boca Raton, FL: CRC, 2001.

[Hin 86] G. E. Hinton, J. L. McClelland, D. E. Rumelhart, “Distributed representations,” In D. E. Rumelhart, J. L. McClelland, & PDP Research Group (Eds.), P*arallel distributed processing. Volume 1: Foundations* (Chap. 3, pp. 77--109). Cambridge, MA: MIT Press, 1986.

[Hin 97] G. E. Hinton, Z. Ghahramani, “Generative models for discovering sparse distributed representations,” *Philosophical Transactions of the Royal Society (London) B*, Vol.352, pp.1177—1190, 1997.

[Hol 92] L. Holmstrom, P. Koistinen, “Using additive noise in back-propagation training,” *IEEE Trans. On Neural Networks*, vol.3, no.1, pp.24-38, Jan, 1992.

[Hop 00] J. B. Hopfinger, M. H. Buonocore, G. R. Mangun, “The neural mechanisms of top-down attentional control,” *Nature Neuroscience*, vol. 3, no. 3, pp. 284-291, March 2000.

[Hor 89] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Networks*, vol.2, pp.359-366, 1989.

[Hsi 01] J.C. Hsieh, C. H. Tu, F. P. Chen, et al. “Activation of the hypothalamus characterizes the acupuncture stimulation at the analgesic point in human: a positron emission tomography study," *Neurosci Lett* vol.307, pp. 105–108, 2001.

[Hua 98] G. Huang, H. A. Babri, "Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions," *IEEE Trans. On Neural Networks*, vol.9, no.1, pp.224-229, Jan, 1998.

[Hua 03] G. Huang, "Learning capability and storage capacity of two-hidden-layer feedforward networks," *IEEE. Trans. Neural Networks,* vol. 14, no. 2, pp. 274-281, Mar, 2003.

[Hum 97] J. E. Hummel, K. J. Holyoak, "Distributed representations of structure: A theory of analogical access and mapping," *Psychological Review*, Vol.104, No. 3, pp. 427-466, 1997.

[Ish 04] K. Ishii, T, Zant, V, Becanovic, P. Ploger, Optimization of Parameters of Echo State Network and its Application to Underwater, Robot, *SIC Annual Conference, Sapporo*, Japan, 2004

[Kan 00] E. R. Kandel, J. H. Schwartz, T. M. Jessell, *Principles of Neural Science*, McGraw-Hill Medical; 4th edition, 2000.

[Kar 00] G.N. Karystinos, D.A. Pados, "On overfitting, generalization, and randomly expanded Training Sets," *IEEE Trans. On Neural Networks*, vol.11, no.5, pp.1050-1057, Sep, 2000.

[Koc 85] C. Koch, S. Ullman, "Shifts in selective visual attention: towards the underlying neural circuitry," *Human Neurobiology*, vol.4, no. 4, pp. 219-227, 1985.

[Koh 84] T. Kohonen, *Self-Organization and Associative Memory*. New York: Springer-Verlag, vol. 8, Springer Ser. Inform. Sci., 1984

[Kwo 97] T. Kwok and D. Yeung, "Constructive algorithms for structure learning in feedforward neural networks for regression problems," *IEEE Trans. on Neural Networks*, vol. 8, no. 3, pp. 630–645, May, 1997.

[Lau 03] S. B. Laughlin, T.J. Sejnowski, "Communication in neuronal networks," *Science*, vol.301, pp.1870-1874, 2003.

[Law 96] S. Lawrence, C. L. Giles, and A. C. Tsoi, "What size neural network gives optimal generalization? Convergence properties of backpropagation," Institute for Advanced Computer Studies, University of Maryland, *Tech. Rep. UMIACS-TR-96-22 and CS-TR-3617*, June, 1996

[Law 97] S. Lawrence, C. L. Giles, and A. C. Tsoi, "Lessons in neural network training: overfitting may be harder than expected," *Proc. of the 14th National Conf. on Artificial Intelligence,* pp. 540-545, 1997.

[LeC] Y. LeCun, C. Cortes, The MNIST database of handwritten digits, http://yann.lecun.com/exdb/mnist/

[LeC 89] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput*., vol. 1, no. 4, pp.541–551, 1989.

[Lee 03] T. Lee, D. Mumford. "Hierarchical Bayesian Inference in the Visual Cortex," *J. Opt. Soc. Am. A Opt Image Sci Vis,* vol. 20, no. 7, pp.1434-1448, July, 2003.

[Leg 06] S. Legg, M. Hutter, "A Formal Measure of Machine Intelligence," *Proc. 15th*

Annual Machine Learning Conference of Belgium and The Netherlands, pp.73-80, 2006.

[Leh 97] E. L, Lehmann, *Testing statistical hypotheses,* New York: Springer, 1997.

[Len 03] P. Lennie, “The cost of cortical computation,” *Current Biology*, vol.13, pp. 493-497, 2003.

[Lew 02] M. S. Lewicki, “Efficient coding of natural sounds,” *National Neuroscience*, vol.5, pp.356-363, 2002.

[Lin 86] R. Linsker, “From basic network principle to neural architecture (series),” *Neurobiology, Proc. Natl. Acad. Sci.* vol. 83, pp.7508-7512, 8390-8394, 8779-8793, 1986.

[Liu 08] Y. Liu, J. A. Starzyk, Z. Zhu, “Optimized Approximation Algorithm in Neural Networks without overfitting”, *IEEE Trans. on Neural Networks*, vol. 19, no. 6, pp. 983-995, June. 2008.

[Lor 66] G. Lorentz, *Approximation of Functions*. New York: Holt, Rinehart and Winston, 1966.

[Luc 98] M. Luck, M. d’Inverno, “Motivated Behavior for Goal Adoption,” *In Multi-Agent Systems: Theories, Languages and Applications –Proc. of the 4th Australian Workshop on Distributed Artificial Intelligence*, C. Zhang and D. Lukose, (eds.), *Lecture Notes in Artificial Intelligence*, vol. 1544, pp. 58-73, Springer-Verlag, 1998.

[Lun 93] J. S. Lund, S. Yoshita, J. B. Levitt, “Comparison of intrinsic connections in different areas of macaque cerebral cortex,” *Cerebral Cortex*, vol.3, no.2, pp.148-162, 1993.

[Mar 04] J. A. Markham, W. T. Greenough, “Experience-driven brain plasticity: beyond the synapse,” *Neuron Glia Biol*, vol.1, no. 4, pp.351-363, 2004.

[McC 81] J. L. McClelland, D. E. Rumelhart, “An interactive activation model of context effects in letter perception: Part 1. An account of basic findings,” *Psychological Review*, vol. 88, no. 5, pp. 375-407, 1981.

[McC 94] R. A. McCallum, “Short-Term Memory in Visual Routines for Off-Road Car Chasing,” *Working Notes of AAAI Spring Symposium Series, “Toward Physical Interaction and Manipulation”*, Stanford University, March 21-23, 1994.

[McC 96] R. A. McCallum, “Learning to Use Selective Attention and Short-Term Memory in Sequential Tasks,” *From Animals to Animats, Fourth International Conference on Simulation of Adaptive Behavior, (SAB'96)*. Cape Cod, Massachusetts. September, 1996.

[McN 87] B. L. McNaughton, R. G. M. Morris, “Hippocampal synaptic enhancement and information storage within a distributed memory system,” *Trends in Neurosciences*, vol. 10, no.10, pp. 408—415, 1987.

[Mel 68] R. Melzack, K. L. Casey, “Sensory, motivational, and central control determinants of pain” In: Kenshalo DR (Ed*). The Skin Senses*. Springfield: C.C. Thomas, pp 423–439, 1968.

[Mel 90] R. Melzack, “Phantom limbs and the concept of a neuromatrix,” *Trends Neurosci*, vol. 13, pp. 88-92, 1990.

[Mes 90] M. M. Mesulam, “Large-scale neurocognitive networks and distributed processing for attention, language, and memory,” *Ann Neurol*, vol. 28, pp. 597–613, 1990.

[Mil 90] W. T. Miller, R. Sutton, and P. Werbos, *Neural Networks for Control*. MIT Press, Cambridge, Massachusetts, 1990.

[Mor 93] R. J. Morecraft, C. Geula, MM. Mesulam, “Architecture of connectivity within a cingulo-fronto-parietal neurocognitive network for directed attention,” *Arch Neurol*, vol. 50, pp. 279–284, 1993.

[Mou 57] V. B. Mountcastle, “Response Properties of Neurons of Cat’s Somatic Sensory Cortex to Peripheral Stimuli,” *J. Neurophysiol*, vol. 20, pp. 374-407, 1957.

[Mou 03] V. B. Mountcastle, “Introduction to a special issue of cerebral cortext on columns”, *Cerebral Cortex*, 13, pp. 2-4, 2003.

[Mur 94] N. Murate, S. Yoshizawa, and S. Amari, “Network information criterion-determining the number of hidden units for an artificial neural network model,” *IEEE Trans. On Neural Networks*, vol.5, no.6, pp.865-872, Nov, 1994.

[Mur 07] J. Murata, Y. Abe, K. Ota, “Introduction and control of subgoals in reinforcement learning,” *Proc. of the 25th IASTED Int. Multi-Conf. AIA*, Feb 12-14, Innsbruck, Austria, 2007.

[Nas 04] S. Nason and J. E. Laird, “Soar-RL, Integrating Reinforcement Learning with Soar,” *Int.l Conf. Cognitive Modeling,* 2004.

[Nav 02] V. Navalpakkam, L. Itti, “A Goal Oriented Attention Guidance Model,” *Proc. 2nd Int. Workshop of Biologically Motivated Computer Vision*, BMCV 2002, Tübingen, Germany, November 22-24, 2002.

[New 90] A. Newell, *Unified Theories of Cognition*. Cambridge, Massachusetts: Harvard University Press, 1990.

[Nor 95] T. J. Norman, D. Long, “Goal Creation in Motivated Agents,” *Intelligent Agents: Theories, Architectures, and Languages (LNAI vol. 890)*, 1995.

[Oja 82] E. Oja, “Simplified neuron model as a principal component analyzer.” *Journal of Mathematical Biology,* vol.15, no. 3, pp.267-273, Nov, 1982.

[Ols 96] B. A. Olshausen, D.J. Field, “Emergence of simple-cell receptive field properties by learning a sparse code for natural images,” *Nature*, vol.381, no.13, pp.607-609, 1996.

[Ols 97] B. A. Olshausen, D.J. Field, “Sparse coding with an overcomplete basis set: a strategy employed in V1,” *Vision Research*, vol.37, no.23, pp.3311-3325, 1997.

[Ols 02] B. A. Olshausen, K. N. O’Connor, “A new window on sound,” *National Neuroscience*, vol.5, pp.292-294, 2002.

[Ols 04] B. A. Olshausen, D. J. Field, “Sparse coding of sensor inputs,” *Current Opinions in Neurobiology*, vol.14, pp.481-487, 2004.

[Ore 00] R. C. O'Reilly, Y. Munakata, *Computational Explorations in Cognitive Neuroscience,* MIT press, pp. 207, 2000.

[Oud 07] P-Y. Oudeyer, F. Kaplan, V. Hafner, “Intrinsic Motivation Systems for Autonomous Mental Development,” *IEEE Transactions on Evolutionary Computation*, vol.11, no. 2, pp. 265-286, April, 2007.

[Pey 00] R. Peyron, B. Laurent, L. Garcia-Larrea, “Functional imaging of brain responses to pain. A review and meta-analysis,” *Neurophysiol Clin,* vol. 30, pp.263–288, 2000.

[Pfe 99] R. Pfeifer and C. Scheier, *Understanding Intelligence*, MIT Press, Cambridge, MA, 1999.

[Phu 07] S. L. Phung, A. Bouzerdoum, “A pyramidal neural network for visual pattern recognition,” *IEEE Trans. on Neural Networks*, vol. 18, no. 2, pp. 329-343, March, 2007.

[Por 02] C. A. Porro, P. Baraldi, G. Pagnoni, et al. “Does anticipation of pain affect cortical nociceptive systems?” *J Neurosci*, vol. 22. pp. 3206–3214, 2002.

[Pre 98] L. Prechelt. “Automatic early stopping using cross validation: quantifying the criteria,” *Neural Networks*, vol.11, no.4, pp.761-777, 1998.

[Pow 98] J. D. Powell, N. P. Fekete, C.-F. Chang, “Observer-based air-fuel ratio control”, *IEEE Control Syst. Mag.*, vol. 18, no. 5, pp. 72-83. 1998.

[Rao 91] A. S. Rao, M. P. Georgeff, “Modeling rational agents within a BDI-architecture,” In R. Fikes and E. Sandewall, editors, *Proc. of the 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning*, KR’91, pp. 473-484, Cambridge, MA, 1991.

[Rao 92] A. S. Rao, M. P. Georgeff, “An abstract architecture for rational agents,” In C.

Rich, W. Startout, B. Nebel, editors, *Proc. of the 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning*, KR'92, pp. 439-449, Boston, MA, 1992.

[Ree 93] R. Reed, "Pruning algorithms – a survey," *IEEE Trans. on Neural Networks*, vol. 4, pp. 740–747, Sep, 1993.

[Reh 07] M. Rehn, F.T. Sommer, "A network that uses few active neurons to code visual input predicts the diverse shape of cortical receptive fields," *Journal of Computer Neuroscience*, 2007.

[Rip 81] B. D. Ripley, *Spatial statistics*. New York: Wiley 1981.

[Ris 86] J. Rissanen, "Stochastic complexity and modeling," *Ann. Statist*, vol.14, no.3, pp.1080-1100, 1986.

[Roe 98] P. R. Roelfsema, V. A. Lamme, H. Spekreijse, "Object-based attention in the primary visual cortex of the macaque monkey," *Nature*, vol. 395, pp. 376-381, 1998.

[Rol 89] E. T. Rolls, "Functions of neuronal networks in the hippocampus and neocortex in memory," *Neural models of plasticity: experimental and theoretical approaches*, pp. 240–265, San Diego: Academic Press, 1989.

[Ros 58] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psych. Review,* vol.65, pp. 386-408, 1958.

[Rum 86a] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations of backpropagtaion errors," *Nature*, vol.323, pp. 533-536, 1986.

[Rum 86b] D. E. Rumelhart, D. Zipser, “Feature discovery by competitive learning,” In D. E. Rumelhart, J. L. McClelland, & PDP Research Group (Eds.), *Parallel distributed processing. Volume 1: Foundations* (Chap. 5, pp. 151--193). Cambridge, MA: MIT Press. 1986.

[Sal 05] M. Salmen, P. G. Ploger, “Echo State Networks used for Motor Control,” in *Proc. of ICRA’05*, Barcelona, Spain, 2005.

[Sam 97] A. Samsonovich, B. L. McNaughton, “Path integration and cognitive mapping in a continuous attractor neural network model,” *Journal of Neuroscience*, vol. 17, pp. 5900-5920, 1997.

[San 89] T. D. Sanger, “Optimal unsupervised learning in a single-layer linear feedforward neural network,” *Neural Networks,* vol. 2, no. 6, pp. 459-473, 1989.

[Sar 15] M. Sartori, P. Antsaklis, “A simple method to derive bounds on the size and to train multi-layer neural networks,” *IEEE Trans. Neural Networks*, vol. 2, no. 4, pp. 467-471, 1991.

[Set 01] R. Setiono, “Feedforward neural network construction using cross validation,” *Neural Computation*, vol. 13, pp. 2865–2877, 2001.

[Si 01] J. Si, Y. Wang, “On-Line Learning Control by Association and Reinforcement,” *IEEE Trans. on Neural Networks*, vol.12, no.2, 2001.

[Sir 95] J. Sirosh, R. Miikkulainen, Y. Choe, *Lateral interactions in the cortex: structure and function*, electronic book, [Online] Available: http://nn.cs.utexas.edu/web-pubs/htmlbook96/ , 1995.

[Sim 01] E. P. Simoncelli, B.A. Olshausen, “Natural image statistics and neural representation,” *Annual review of neuroscience*, vol.24, pp.1193-1216, 2001.

[Smi 98] R. L. Smith. *Intelligent Motion Control with an Artificial Cerebellum. Ph.D dissertation*, University of Auckland, New Zealand, 1998

[Sot 04] D. Soto, M. J. Blanco, “Spatial attention and object-based attention: a comparison within a single task,” *Vision research*, vol. 44, pp. 69-81, 2004.

[Spo 93] O. Sporns, and G. M. Edelman, “Solving Bernstein's problem: A proposal for the development of coordinated movement by selection.” *Child Development, 64*(4), 1993, pp. 960-981.

[Sta 93] J. A. Starzyk and X. Fang, “A CMOS Current Mode Winner-Take-All Circuit with both Excitatory and Inhibitory Feedback”, *Electronics Letters*, Vol.29, No.10, pp.908-910, 1993.

[Sta 05] J. A. Starzyk, and Y. Li, D. D. Vogel, "Neural Network with Memory and Cognitive Functions", *Artificial Neural Networks: Biological Inspirations. Lecture Notes in Computer Science 3696*. pp. 85-90, 2005.

[Sta 06] J. A. Starzyk, Y. Liu, H. He, “Challenges of Embodied Intelligence,” *Proc. Int. Conf. on Signals and Electronic Systems*, ICSES'06, Lodz, Poland, Sep. 17-20, 2006.

[Sta 07a] J. A. Starzyk, Y. Liu, D. Vogel, “Sparse coding in a hierarchical self-organizing memory with sparse connectivity,” *under revision for second submission to IEEE Trans. Neural Networks*, 2007.

[Sta 07b] J. A. Starzyk, H. He, “Anticipation-Based Temporal Sequences Learning in

Hierarchical Structure," *IEEE Trans. Neural Networks*, vol. 18, pp. 344 - 358, 2007.

[Ste 00] P. N. Steinmetz, A. Roy, P. J. Fitzgerald, S. S. Hsiao, K. O. Johnson, E. Niebur, "Attention modulates synchronized neuronal firing in primate somatosensory cortex," *Nature*, vol. 404, pp. 187-190, 2000.

[Ste 01] C. F. Stevens, "An evolutionary scaling law for the primate visual system and its basis in cortical function," *Nature*, Vol. 411, pp. 193-195, May, 2001.

[Sug 93] M. Sugeno, and T.Yasukawa, " A Fuzzy-Logic based approach to qualitative modeling," *IEEE Trans. On Fuzzy Systems*, vol. 1, no. 1, pp. 7-31, Feb, 1993.

[Sut 98] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction,* MIT Press, Cambridge, MA, 1998.

[Swi 96] K. Swingler, *Applying Neural Networks: A Practical Guide*, London: Academic Press, 1996.

[Tam 97] S. Tamura, M. Tateishi, "Capabilities of a four-layered feedforward neural network: Four layers versus three," *IEEE Trans. Neural Networks*, vol.8, no.2, pp.251-255, 1997.

[Tan 96] K. Tanaka, "Inferotemporal cortex and object vision," *Annual Review of Neuroscience*, Vol.19, pp.109-139, 1996.

[Teo 06] E. J. Teoh, K. C. Tan, and C. Xiang, "Estimating the number of hidden neurons in a feedforward network using the singular value decomposition," *IEEE Trans. on Neural Networks*, vol. 17, no.6, pp.1623-1629, Nov, 2006.

[Tha 02] J. Thangarajah, L. Padgham, J. Harland, "Representation and reasoning for goals in BDI agents," *Proc. of the 25th Australasian conference on Computer science*, vol. 4, pp.259 – 265, 2002.

[Töl 99] T. R. Tölle, T. Kaufmann, T. Siessmeier, et al. "Region-specific encoding of sensory and affective components of pain in the human brain: a positron emission tomography correlation analysis," *Ann Neurol*, vol. 45, pp. 40–47, 1999.

[Ung 82] L. G. Ungerleider, M. Mishkin, "Two cortical visual systems," *The analysis of visual behavior*, Cambridge, MA: MIT Press. 1982.

[Van 83] D. C. Van Essen, J. H. R. Maunsell, "Hierarchical organization and functional streams in the visual cortex," *Trends in Neurosciences*, Vol.6, pp. 370-375, 1983.

[Vin 00] W.E.Vinje, J. L. Gallant, "Sparse coding and decorrelation in primary visual cortex during natural vision," *Science*, vol.287, no.5456, pp. 1273-1276, 2000.

[Vin 02] W.E. Vinje, J. L Gallant, "Natural stimulation of the nonclassical receptive field increases information transmission efficiency in V1," *J. of Neuroscience*, vol.22, pp.2904-2915, 2002.

[Vog 97] D. D. Vogel, W. Boos, "Sparsely connected, Hebbian networks with strikingly large storage capacities," *Neural Networks*, vol. 10, no. 4, pp. 671-682, 1997.

[Vog 05] D. D. Vogel, "A neural network model of memory and higher cognitive functions," *Int J Psychophysiology*, vol. 55, no.1, pp.3-21, 2005.

[Wan 99] C. Wang, and J. C. Principe, "Training neural networks with additive noise in the desired signal," *IEEE Trans. On Neural Networks*, vol.10, no.6, Nov, 1999.

[Wan 93] S. Wang, J. Judd, and S. S. Venkatesh, “When to stop: on optimal stopping and effective machine size in learning”, *presented at the Conf. on Neural Information Processing Systems*, Denver, Colorado, 1993.

[Wil 69] D.J., Willsahw, O. P., Buneman, H. C., Longuet-Higgins, “Nonholographic associative memory”, *Nature*, Vol. 222, pp. 960-962, 1969.

[Wit 77] I. H. Witten, “An adaptive optimal controller for discrete-time Markov Environments,” *Information and control*, vol.34, pp. 286-295, 1977.

[Xia 05] C. Xiang, S. Ding, and T. Lee, “Geometrical interpretation and architecture selection of MLP,” *IEEE Trans. On Neural Networks*, vol.16, no.1, pp.84-96, Jan. 2005.

[Xie 02] X. Xie, R. Hahnolser, H. S. Seung, “Selectively grouping neurons in recurrent networks of lateral inhibition,” *Neural computation*, vol.14, pp.2627-2646, 2002.

[Yan 03] S. Yantis, J. T. Serences, “Cortical mechanisms of space-based and object-based attentional control,” *Current opinion in neurobiology*, vol. 13, pp. 187-193, 2003.

[Zha 04] N. Zhang, J. Weng, “Sparse representation from a winner-take-all neural network”, *Proc. Of International Joint Conference on Neural Networks*, Budapest, Hungary, 2004.

[Zur 92] J. M. Zurada, *Introduction to artificial neural systems*, West Publishing Company, 1992.