CHANNEL EQUALIZATION USING SELF-ORGANIZING MAPS

ELIJAH OLUKOHE OTIENO

MASTER OF SCIENCE IN ELECTRICAL AND ELECTRONICS ENGINEERING

PAN AFRICAN UNIVERSITY

INSTITUTE FOR BASIC SCIENCES TECHNOLOGY AND INNOVATION

OCTOBER 2014

Channel Equalization using Self-Organizing Maps

Elijah Olukohe Otieno

EE300-0004/12

A thesis submitted to Pan African University, Institute for Basic Sciences Technology and Innovation in partial fulfillment of the requirement for the degree of M. Sc. in Electrical and Electronics Engineering

October 2014

DECLARATION

This thesis is my original work and has not been submitted to any other university for examination.

Signature……………………………………… Date……………………………………………

Elijah Olukohe Otieno

This thesis report has been submitted for examination with our approval as University supervisors.

Signature……………………………………… Date……………………………………………

Dr. Peter Kamita Kihato

JKUAT, Kenya

Signature……………………………………… Date……………………………………………

Prof. Dominic Onyango Konditi

Multimedia University, Kenya

ACKNOWLEDGEMENT

I am deeply grateful to my supervisors, Dr. Peter K. Kihato of Jomo Kenyatta University of Agriculture and Technology and Prof. Dominic O. Konditi of Multimedia University of Kenya. I would like to thank my first supervisor Dr. Peter K. Kihato for offering me his insights into the topic of self-organizing maps and for his advice and never ending encouragement. I would also like to express my appreciation to Prof. Dominic Konditi of Multimedia University for his support and advice on the conduct of scientific research. Throughout my research, I have gained an appreciation of self-organizing maps and the scientific process, as well as the role of scientific research in society.

I would like to thank members of staff at both Pan African University and Jomo Kenyatta University of Agriculture and Technology for their suggestions and assistance throughout the course. I would like to thank my fellow students for offering their help whenever it was needed.

Finally, I would like to thank my family and friends for their understanding and patience while I undertook my course.

ABSTRACT

In this work, the signals at the output of a wireless channel in the presence of inter-symbol interference were visualized using Self-Organizing Maps with the aim of discovering their properties which can then be used to mitigate the effects of inter-symbol interference on signals sent over a wireless channel. The research tried to find out if there are certain properties in an interfering wireless channel output which can be exploited to mitigate the effects of inter-symbol interference and then it was demonstrated how the found properties can be used in the mitigation. To achieve these objectives, simulations were carried out. Gray coded 16-QAM symbols were transmitted over a channel which introduces severe inter-symbol interference. The in-phase and quadrature components of the channel output were then used to train a self-organizing map. The fully trained map was used to make observations about the general structure of channel output and how it relates to the transmitted constellation. It was found that the channel output resembles a rotated input constellation. Furthermore, some symbols were found in clusters belonging to other constellation points other than their own. An attempt was then made to classify the channel output using the trained map and an analysis was then done on the misclassified symbols to determine what constellation points that symbols from each of the 16 constellation points is likely to be misclassified to. It was found that a symbol is likely to be misclassified to those symbols whose gray codes differ from its own by one bit. Thus the accuracy of gray codes for the used channel was verified.

An examination of the self-organizing map component planes for the in-phase and quadrature components revealed how the classification of the symbols belonging to each of the 16 constellation points is influenced by the value of their in-phase and quadrature components. It was found that the classification of symbols belonging to certain constellation points is strongly affected by the values of their in-phase or quadrature components. The classification of symbols belonging to certain constellation points is found to be weakly dependent on the values of their in-phase or quadrature components while the classification of symbols belonging to certain constellation points is moderately affected by the values of their in-phase or quadrature components. This result can be used for example to minimize the number of symbol classification errors that result when the energy of transmitted signals is reduced. It is demonstrated how this can be done by essentially reducing the magnitude of in-phase or quadrature components for the constellation points whose classification is weakly and moderately dependent on the value of their in-phase or quadrature components. The results show that the symbol classification error obtained is less than that obtained when the energy of all constellation points is reduced indiscriminately.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ISI	Inter-symbol interference
QAM	Quadrature Amplitude Modulation
ANN	Artificial neural networks
BPN	Back-propagation network
MSE	Mean squared error
SOM	Self-organizing maps
BMU	Best matching unit
PAM	Pulse amplitude modulation
EM	Electromagnetic
CIR	Channel Impulse Response
MMSE	Minimum mean squared error
LMS	Least mean squares
CDMA	Code division multiple access

CHAPTER 1

INTRODUCTION

1.1 Background

The wireless channel is a hostile medium of communications because its characteristics change over time and frequency thereby impacting the signal sent over it. The received signal strength varies over time and frequency, a phenomenon called fading. Multipath fading occurs due to the constructive and destructive interference of the multiple signal paths between the transmitter and receiver. Multipath fading results in time-dispersion of received signal. This time dispersion can be characterized by a measure called the delay spread. Wireless channel can be classified as experiencing either flat-fading or frequency-selective fading depending on delay spread. If delay spread is much smaller than the symbol time then flat fading occurs. If delay spread is much larger than symbol time then frequency-selective fading occurs. Frequency-selective fading causes severe inter-symbol interference. Inter-symbol interference is a hurdle in achieving the target of high data rate and reliable communication systems whose magnitude renders channel noise insignificant by comparison [1]. ISI can cause an irreducible error floor when the modulation symbol time is on the same order as the channel delay spread. Before discussing the techniques commonly used to mitigate ISI, we will define another measure of channel characteristics which determine if it is appropriate to use a particular technique to mitigate ISI. Coherence time is a statistical measure of the time duration over which the channel impulse response is essentially time invariant. If the symbol period of the baseband signal is greater than the coherence time of the channel then the channel will change during the transmission of the signal and the channel is said to experience fast fading. On the other hand, if the symbol period is less than the coherence time then the channel is said to experience slow fading. Signal processing provides a powerful mechanism to counteract ISI. Equalization in a broad sense can be defined as any technique used to counteract ISI. The term equalization is however mostly associated with a particular technique used to alleviate the ISI problem caused by delay spread by estimating the channel impulse response and then cancelling its effect on the received signal. In this project we use the term equalization in its broad sense unless stated otherwise.

1.2 Quadrature Amplitude Modulation

Quadrature Amplitude Modulation (QAM) is a passband digital transmission method that impresses two separate k-bit symbols onto the quadrature carriers $cos(2\pi f_c t)$ and $sin(2\pi f_c t)$ respectively, where f_c is the carrier frequency. The general M-ary QAM signal constellation is represented by the finite symbol set $\{s_m = a_m + jb_m\}_{m=1}^{M}$. Alternatively, a QAM symbol can be represented in polar notation as $A_m e^{j\theta_m}$, where $A_m = \sqrt{a_m^2 + b_m^2}$ and $\theta_m = tan^{-1}(b_m/a_m)$. The modulated passband signal $s(t)$ is defined as

$$s(t) = \Re\{A_m(t)e^{j(2\pi f_c t + \theta_m(t))}\} \tag{1.1}$$

where $\Re\{.\}$ represents the real part of a complex number. The modulated signal $s(t)$ is referred to as the narrow-band bandpass signal since $f_c \gg B$ where B is the bandwidth. The narrow-band bandpass signal can be translated to an equivalent baseband signal by expanding (1.1) as follows

$$s(t) = A_m(t)\cos(\theta_m(t))\cos(2\pi f_c t) - A_m(t)\sin(\theta_m(t))\cos(2\pi f_c t) \tag{1.2}$$

$$= u_I \cos(2\pi f_c t) - u_Q \sin(2\pi f_c t)$$

where $u_I = A_m(t)cos\,(\theta_m(t))$ and $u_Q = A_m(t)sin\,(\theta_m(t))$ are the in-phase and quadrature components of $s(t)$, respectively. The complex bandpass envelope is given by

$$u(t) = u_I(t) + ju_Q(t) = A_m(t)e^{j\theta_m(t)} \tag{1.3}$$

which when substituted into (1.1) allows $s(t)$to be re-written as

$$s(t) = \Re\{u(t)e^{j(2\pi f_c t)}\} \tag{1.4}$$

This implies that the knowledge of $u(t)$ and f_c uniquely describes the modulated $s(t)$, where $u(t)$ contains all the useful information.

1.3 Self-Organizing Maps

The self-organizing map is a neural network model and algorithm that implements a characteristic nonlinear projection from the high dimensional space of sensory or other input signals onto a low dimensional array of neurons. The SOM is able to map a structured, high-dimensional signal manifold onto a much lower dimensional network in an orderly manner. The mapping usually preserves the topological relationships of the signal domains. Due to this order, the image of the signal space tends to manifest clusters of input information and their relationships on the map.

Accordingly, the most important applications of the SOM are in the visualization of high dimensional systems and processes and discovery of categories and abstractions from raw data [2]. The later operation is called the exploratory data analysis or “data mining.”

1.4 Problem Statement

The use of self-organizing maps to visualize and explore the characteristics of a wireless channel output has not been done yet such a process could potentially yield hitherto unknown characteristics and relationships in the wireless channel output data. Such information could potentially be used to compensate for channel imperfections.

The findings in this report indicate that indeed interesting relationships exist in the output of an interfering channel output and that such relationships could be used to mitigate inter-symbol interference.

1.5 Objectives

1.5.1 Main Objective

To perform channel equalization using self-organizing maps

1.5.2 Specific Objectives

1) To visualize the output of an interfering channel in a digital communication system using self- organizing maps. For this purpose the digital modulation of choice is 16-QAM.
2) To establish the various characteristics and relationships in the output of an interfering channel with an aim of using such characteristics and relationships to mitigate inter-symbol interference.
3) To demonstrate how the findings in 2 above can be used to mitigate inter-symbol interference

CHAPTER 2

LITERATURE REVIEW

2.1 Artificial Neural Networks

2.1.1 Introduction

The immense capabilities of the human brain in processing information and making instantaneous decisions, even under very complex circumstances and under uncertain environments, have inspired researchers in studying and possibly mimicking the computational abilities of this wonder. What a human can achieve in a very short time span, for instance in terms of pattern recognition and obstacle avoidance within an unknown environment, would take very expensive computer resources and much longer to get comparable results. This is mainly due to the way humans process information. Indeed researchers have shown for many years that brains make computations in a radically different manner to that done by digital computers. Unlike computers which are programmed to solve problems using sequential algorithms, the brain makes use of a massive network of parallel and distributed computational elements called neurons. The large number of connections linking these elements provides humans with the very powerful capability of learning. Motivated by this very efficient computational biological model, scientists have for the last few decades attempted to build computational systems, which can process information in a similar way [3]. Such systems are called artificial neural networks (ANN) or connectionist models. They are composed of a large number of highly interconnected processing elements analogous in functionality to biological neurons and are tied together with weighted connections corresponding to brain synapses.

2.1.2 Features of Artificial Neural Networks

As mentioned previously, an artificial neural network (ANN) is typically composed of a set of parallel and distributed processing units, called nodes or neurons. These are usually ordered into layers, appropriately interconnected by means of unidirectional (or bidirectional in some cases) weighted signal channels, called connections or synaptic weights, as shown in Figure.2.1 below.

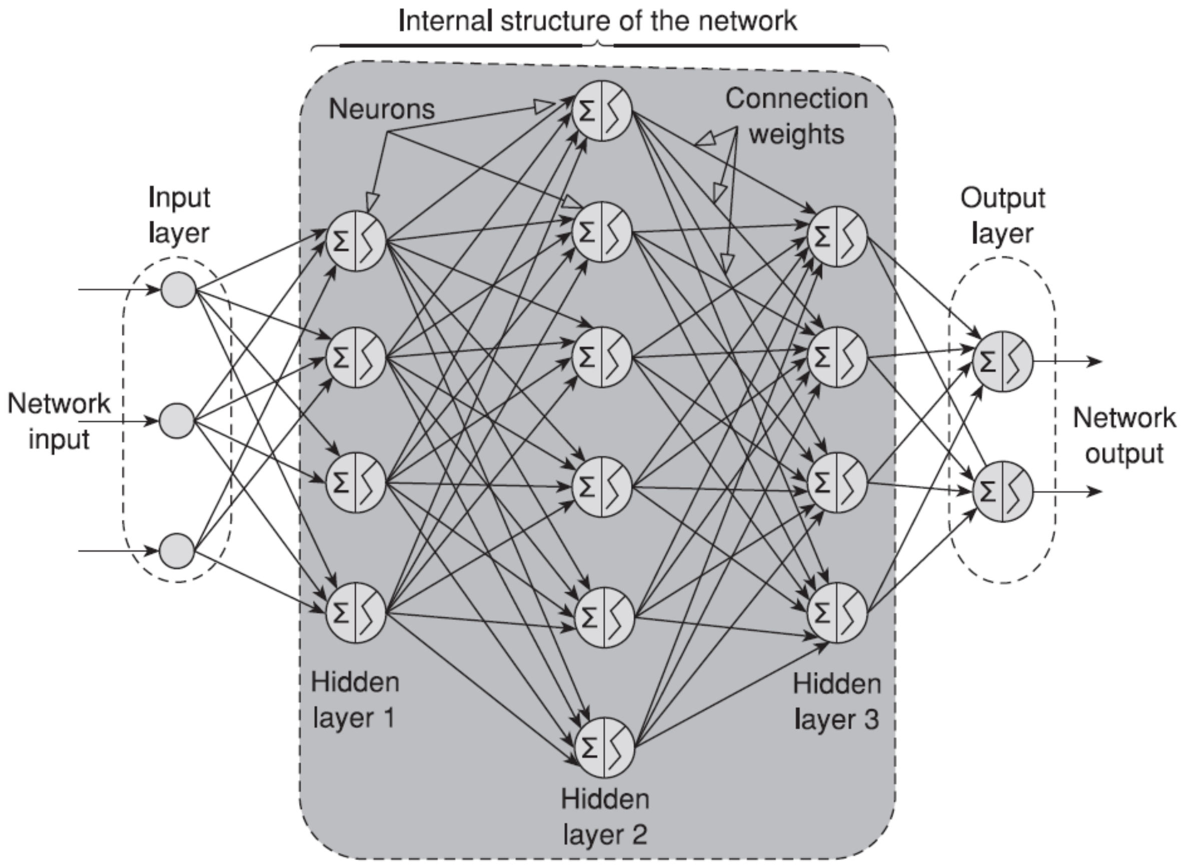

Figure 2-1: Typical representation of a feedfoward (unidirectional) artificial neural network with three inputs and two outputs

The internal architecture of ANN provides powerful computational capabilities, allowing for the simultaneous exploration of different competing hypotheses. Massive parallelism and computationally intensive learning through examples make them suitable for application in nonlinear functional mapping, speech and pattern recognition, categorization, data compression, and many other applications characterized by complex dynamics and possibly uncertain behavior. Neural networks gather their knowledge through detection of patterns and relationships found in the data provided to them. Three important features generally characterize an artificial neural network: the network topology, the network transfer functions, and the network learning algorithm.

2.1.3 Neural network topologies

These correspond to the ordering and organization of the nodes from the input layer to the output layer of the network. In fact, the way the nodes and the interconnections are arranged within the layers of a given ANN determines its topology. The choice for using a given topology is mainly dictated by the type of problem being considered. Some neural networks designers classify ANN according to how the nodes are organized and hence how data is processed through the network. The two well known ANN topologies are the feedfoward and the recurrent architectures.

2.1.3.1 The feedfoward topology

A network with a feedfoward (FF) architecture has its nodes hierarchically arranged in layers starting with the input layer and ending with the output layer. In between, a number of internal layers, also called hidden layers, provide most of the network computational power. The nodes in each layer are connected to the next layer through unidirectional paths starting from one layer (source) and ending at the subsequent layer (sink). This means that the outputs of a given layer feed the nodes of the following layer in a forward path as shown below. Because of their structure, such networks are termed as feedfoward networks. They are also occasionally called open-loop networks given the absence of feedfoward flow of information in their structure.

The feedfoward topology has been very popular due to its association with a quite powerful and relatively robust learning algorithm known as the backpropagation learning algorithm (BPL). The multilayer perceptron network and the radial basis function network are among the well-known networks using the feedfoward topology.

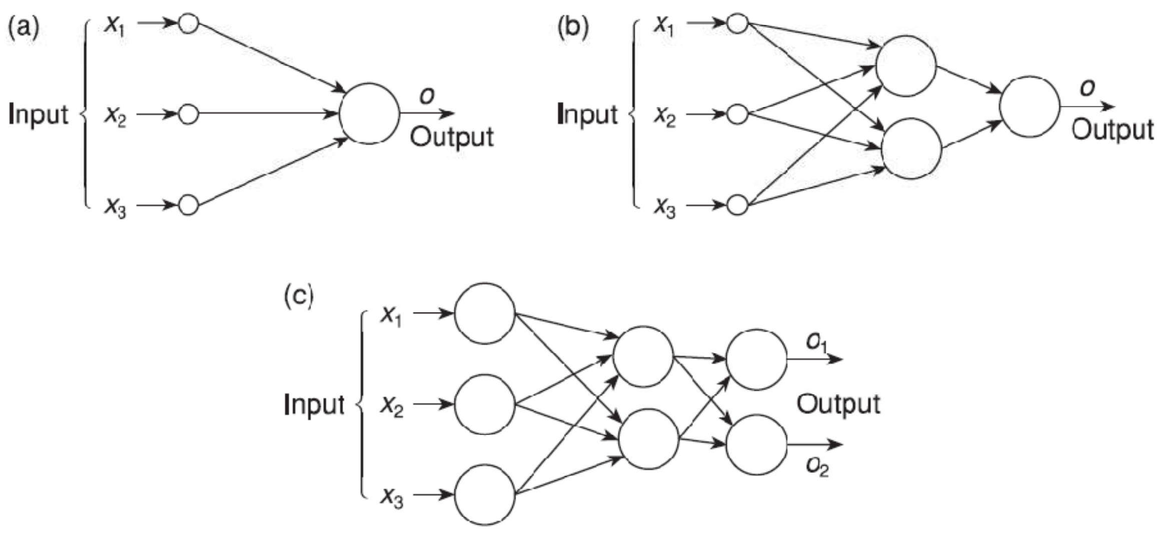

Figure 2-2 A typical structure of neural networks with feedfoward topology, (a) multi-input single output, no hidden layer, (b) multi-input, single output, one hidden layer, (c) multi-input, multi-output, one hidden layer

2.1.3.2 The recurrent topology

Unlike feedfoward networks, recurrent networks (RN) allow for feedback connections among their nodes as illustrated in Figure 2.3 below. They are structured in such a way as to permit storage of information in their output nodes through dynamic states, hence providing the network with some sort of "memory", while FF networks map input into output and are static in the sense that the output of a given pattern of inputs is independent of the previous state of the network, recurrent networks map states into states and as such are very useful for modeling and identifying dynamic systems. This means that the feedback available for a given input node allows it to pass to another state as soon as an output has been delivered at the other end of the network. Several well-known neural networks have been designed based on the recurrent topology. Such networks include the Hopfield network and the time delayed neural networks (TDNN).

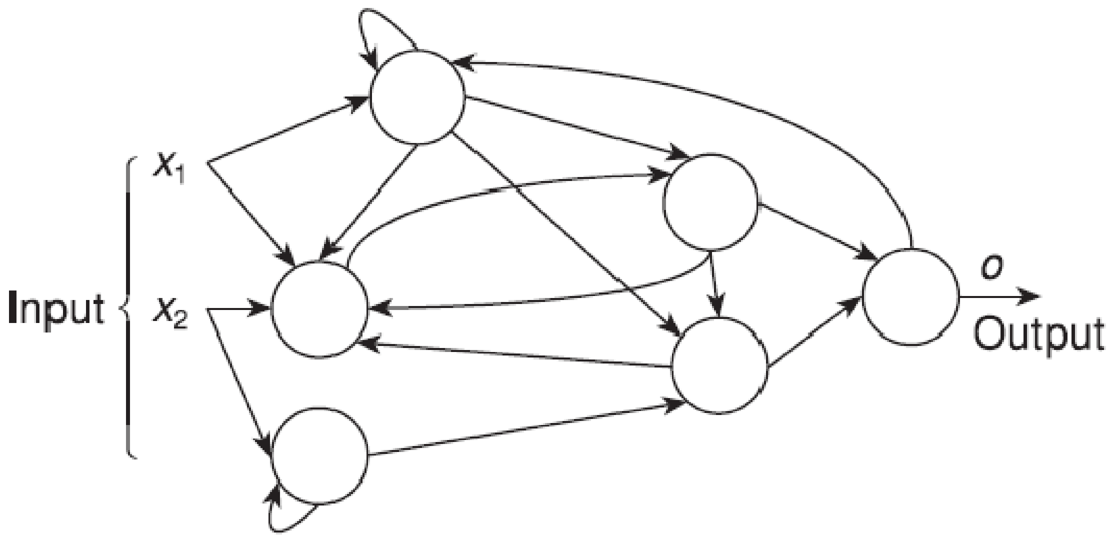

Figure 2-3: A typical neural network with recurrent topology

2.1.4 Neural network activation functions

The basic elements of the computational engine for a neural network are the neurons. These are sorts of simple processors which take the weighted sum of their inputs from other nodes and apply to them a nonlinear mapping (not necessarily linear) called an activation function before delivering the output to the next neuron, see figure below. The output o_k of a typical neuron k having l inputs is given as:

$$\boldsymbol{o_k = f\left(\sum_{l} w_{ik}x_i - \theta_k\right)} \qquad (2.1)$$

where f is the node's activation function, $x_1, x_2, \dots, x_l$ are the node's inputs, $w_{1k}, w_{2k}, \dots, w_{lk}$ are the connection weights, and θ_k is the node's threshold. The processing activity within a given layer is done simultaneously, hence providing the neural network with the powerful capability of parallel computing. The bias effect (threshold value) is intended to occasionally inhibit the activity of some nodes. As neural networks may vary in terms of their structure as described previously, they may as well vary in terms of their activation function.

Depending on the problem at hand and on the location of the node within a given layer, the activation functions can take different forms: sigmoid mapping, signum function, step function or linear correspondence. The mathematical representation for some of these mappings is given below:

$$\mathrm{sigmoid}(\mathrm{x}) = \frac{1}{1 + \exp(-\mathrm{x})} \tag{2.2}$$

$$\mathrm{signum}(\mathrm{x}) = \begin{cases} 1 \text{ if } \mathrm{x} > 0 \\ 0 \text{ if } \mathrm{x} = 0 \\ -1 \text{ if } \mathrm{x} < 0 \end{cases} \tag{2.3}$$

$$\mathrm{step}(\mathrm{x}) = \begin{cases} 1 \text{ if } \mathrm{x} > 0 \\ 0 \text{ otherwise} \end{cases} \tag{2.4}$$

Figure 2.4 below illustrates the profile of six activation functions commonly used in implementations of neural networks.

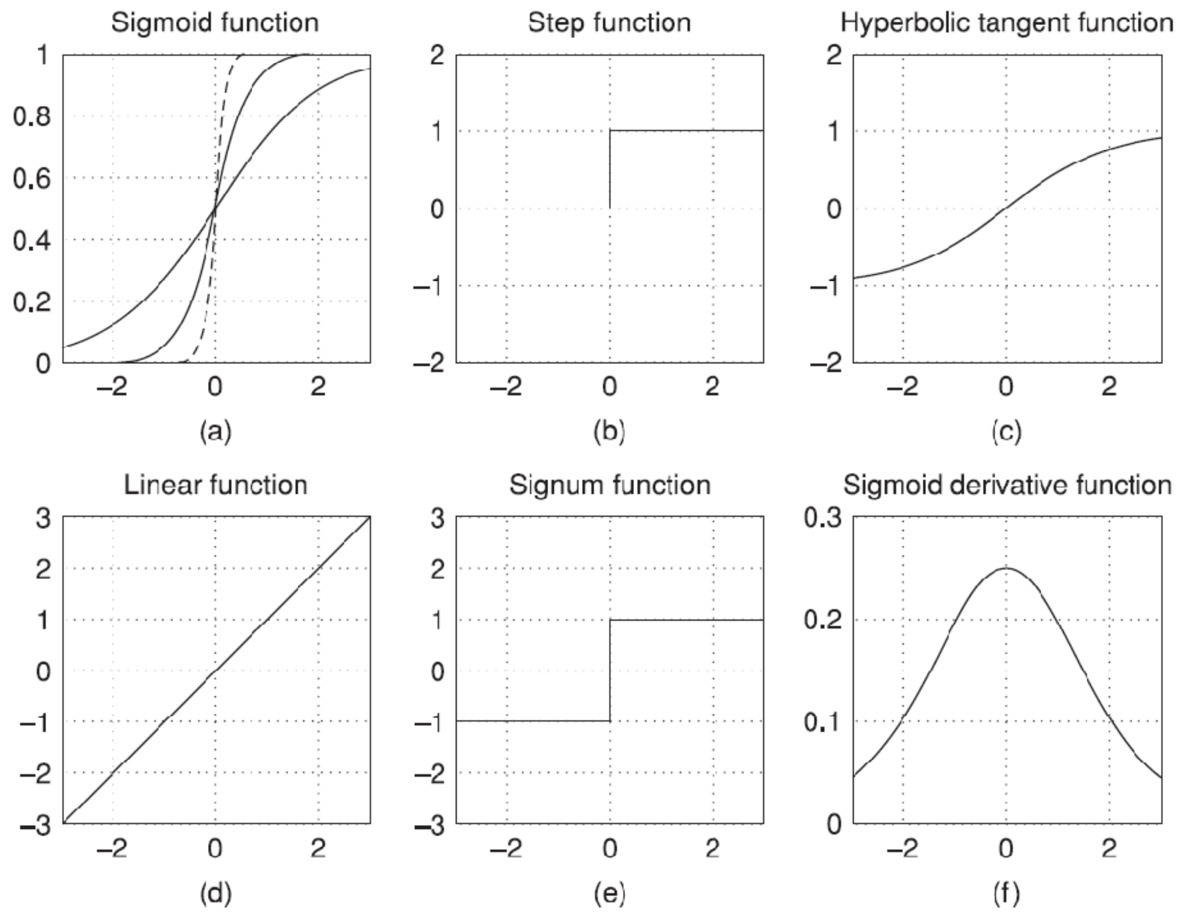

Figure 2-4: Typical profiles of six activation functions

2.1.5 Neural networks learning algorithms

Learning algorithms are used to update the weight parameters at the interconnection level of the neurons during the training process of the network. While some designers have classified neural networks according to topologies or architectures, others have classified them according to the learning algorithm used by the network. The three well known and most often used learning mechanisms are the supervised, unsupervised (or self-organized) and the reinforced.

2.1.5.1 Supervised learning

The main feature of the supervised (or active) learning mechanism (see Figure 2.5 below) is learning by examples. This means that an external teacher provides the network with a set of input stimuli for which the output is a priori known. During the training process, the output results are continuously compared with the desired data. An appropriate learning rule uses the error between the actual output and the target data to adjust the connection weights so as to obtain, after a number of iterations, the closest match between the target output and the actual output. Supervised learning is particularly useful for feedfoward networks. A number of supervised learning algorithms have been suggested in the literature. The backpropagation algorithm, first developed by Werbos in 1974[4] which is also based on the gradient descent optimization technique and the least mean square algorithm [5] are among the most commonly used supervised learning rules.

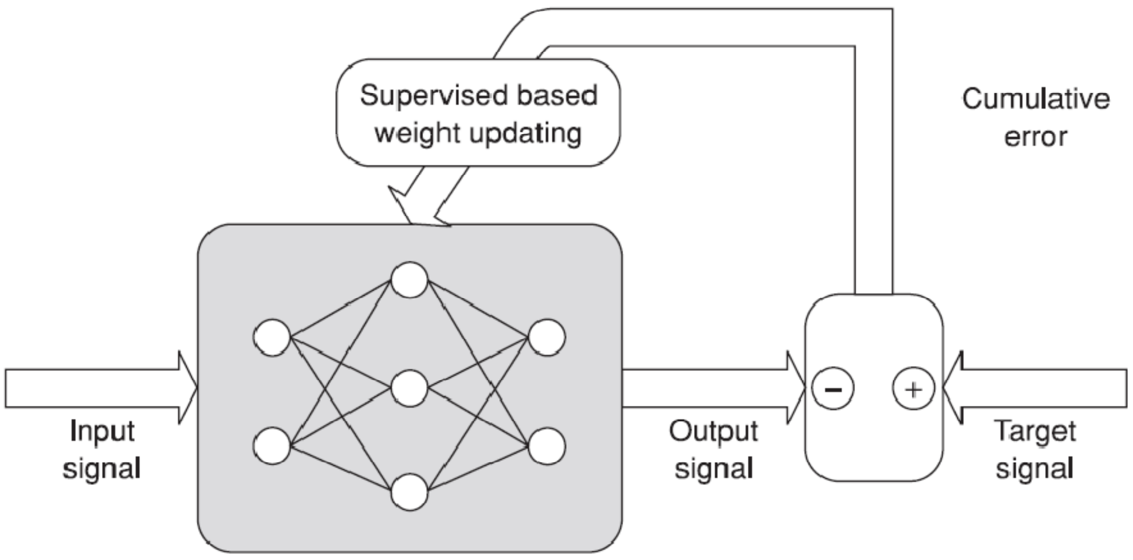

Figure 2-5: Schematic representation of supervised learning

2.1.5.2 Unsupervised learning

Unlike supervised learning, unsupervised or self-organized learning (Figure 2.6 below) does not involve an external teacher and relies instead upon local information and internal control. The training data and input patterns are presented to the system and through predefined guidelines, the system discovers emergent collective properties and organizes the data into clusters or categories. Because of the way the network adjusts its connection weights in response to the presented input data, unsupervised learning algorithms have been known as open-loop adaptation learning schemes. An unsupervised learning scheme operates as follows. A set of training data is presented to the system at the input layer level. The network connection weights are then adjusted through some sort of competition among the nodes of the output layer, where the successful candidate will be the node with the highest value. In the process, the algorithm strengthens the connection between the incoming pattern at the input layer and the node output corresponding to the winning candidate. In addition to the strengthening of the connections between the input layer and the winning output node, the unsupervised learning scheme may be used for adjusting the weights of the connections leading to the neighboring nodes at the output layer. This is controlled by what is referred to as the neighborliness parameter, and it has the major property of making groups of output nodes behave as single entities with particular features.

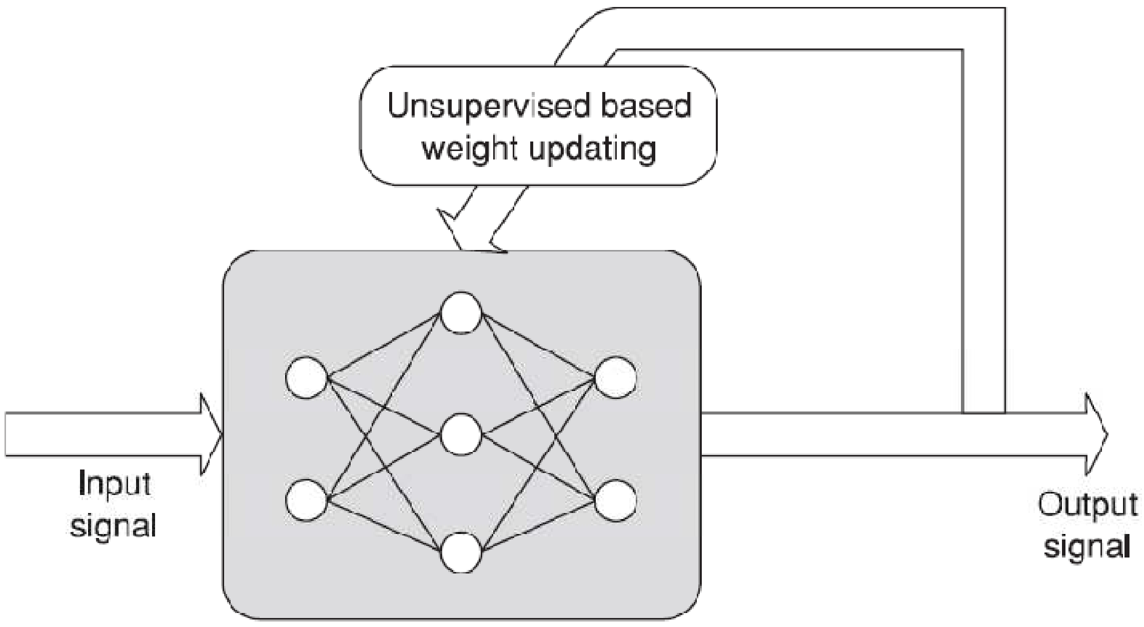

Figure 2-6: Schematic representation of unsupervised learning

2.1.5.3 Reinforcement learning

Reinforcement learning (see Figure 2.7 below) also known as graded learning, has been receiving an increased interest lately because of its many attractive learning features, which mimic in a way the adjusting behavior of humans when interacting with a given physical environment. This is another type of learning mechanism by means of which the network connections are modified according to feedback information provided to the network by its environment. This information simply instructs the system on whether or not a correct response has been obtained. In the case of a correct response, the corresponding connections leading to that output are strengthened, otherwise they are weakened. This type of learning strategy based on the reward/penalty paradigm has several similarities with the biological learning system. Unlike supervised learning, reinforcement learning (LR) does not get information on what the output should be when the network is presented with a given input pattern.

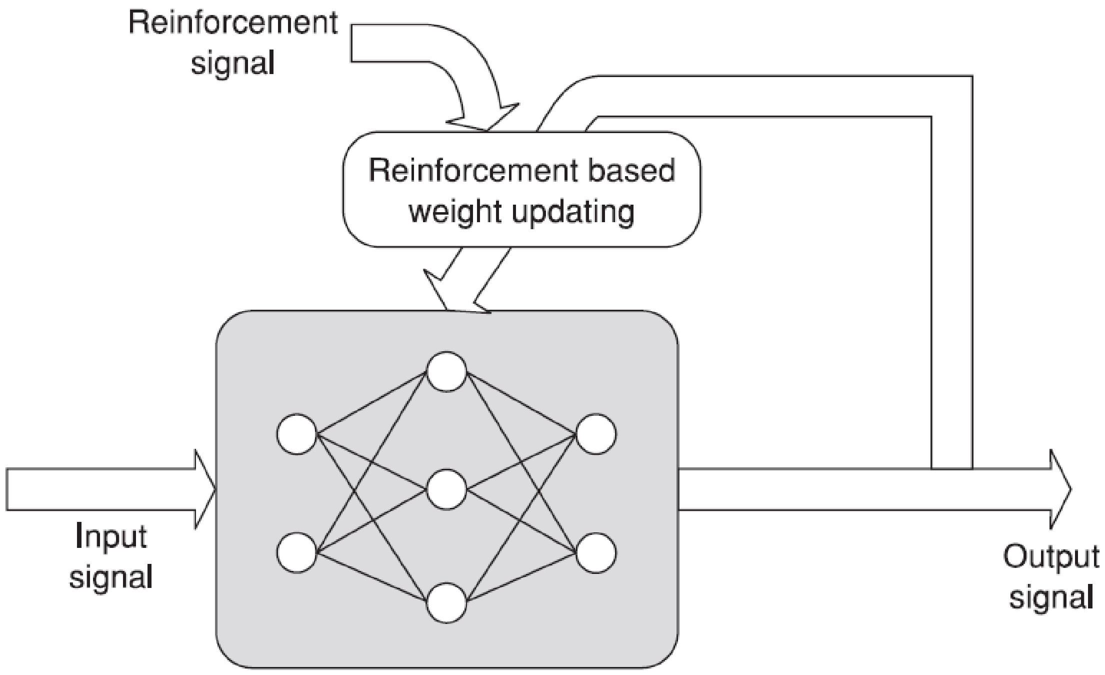

Figure 2-7: Schematic representation of reinforcement learning

Reinforcement learning also differs from unsupervised learning (UL) in that UL doesn't provide the network with information on whether the output is correct or not, but rather operates on the premises of finding pattern regularity among the exemplars provided to the network. Given the nature of this learning scheme, random search strategies have been used to attain the correct output every time the system is presented with an excitation from its environment. This explorational aspect of learning ultimately leads to an improved capability of the system to deliver the expected output every time the system is presented with an input pattern.

2.1.6 Perceptron

Perceptron is an example of an ANN developed as a trainable model of an artificial neuron using a supervised learning procedure in which the system adjusts its weights in response to a comparative signal computed between the actual output and the target output. The perceptron was designed for the purpose of pattern classification of linearly separable sets [6]. By definition, sets are linearly separable if there exists a hyperplanar multidimensional decision boundary that classifies the data input into two classes.

In terms of architecture, the perceptron is composed of a hierarchical three-level structure. The topology of the perceptron includes an input level corresponding to the sensory unit or retina; a second-level unit, also called a feature detector unit, involving nodes connected to the input but with fixed connection weights and thresholds; and a third level unit involving the output layer composed of one single node but with adjustable connection weights. The structure is shown in Figure 2.8 below

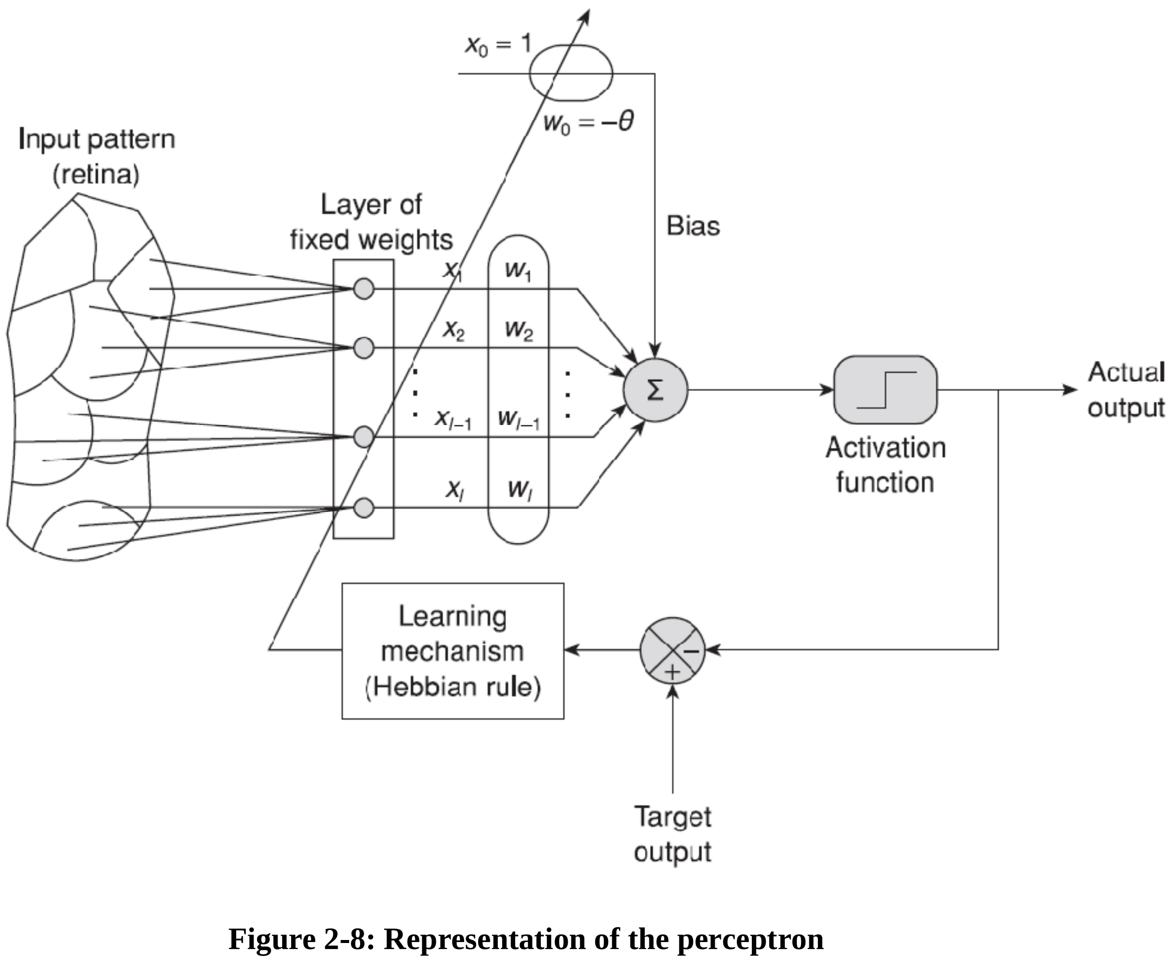

Figure 2-8: Representation of the perceptron

The activation function used by Rossenblatt in the perceptron is the step or the hard limiting activation function, and the learning algorithm used to adjust the weights is the perceptron learning rule. It was shown (perceptron convergence theorem [6]) that as long as the patterns used to train the perceptron are linearly separable, the learning algorithm should converge in a finite number of steps. The steps used to train a perceptron are outlined below:

Initialize weights and thresholds to small random values

Choose an input-output pattern from the training input-output data set $\left(x^{(k)}, t^{(k)}\right)$

Compute the actual output $o = f\left(\sum_{i=1}^{l} w_i x_i - \theta\right)$

Adjust the weights according to the perceptron learning rule:

$$\Delta w_i = \eta \left[t - f\left(\sum_i w_i x_i - \theta\right)\right] x_i$$

In case the weights do not reach state-state values $\Delta w_i = 0$, repeat starting from step 2 and choose another training pattern. This is known as epoch training. This learning procedure is very similar to the Hebbian learning rule [6], with the difference that the connection weights there are not updated if the network responds correctly.

2.1.7 Back-propagation network

2.1.7.1 Background

When an ANN is presented with data, the output will not be the desired output since the network has not undergone training. Since the network weights are initially random, it is likely that the initial output value will be very far from the desired output. To improve the behavior of the network and to know which connection weights need modification and by how much so as to achieve the objective, an algorithm commonly used is the back-propagation algorithm. This is simply a gradient descent method of minimizing the total squared error of the output computed by the network. The principal advantages of back-propagation are simplicity and reasonable speed. Back-propagation is well suited to pattern recognition problems.

The network learns a predefined set of input-output example pairs by using a two-phase propagate-adapt cycle. An input pattern is applied as a stimulus to the first layer of network units. It is propagated through each upper layer until an output is generated. The output is then compared to desired output. The error is transmitted backward from output layer to each node in the intermediate layer that contributes directly to the output. Each unit in the intermediate layer receives only a portion of the error signal. This process repeats itself layer by layer until each node in the network has received an error signal that describes its relative contribution to the total error. One of the significant features of back-propagation is that intermediate layers organize themselves such that different nodes learn to recognize different features of the input space. After training, when presented with arbitrary input patterns that are noisy or incomplete, the units in the hidden layers of the network will respond with an active output if the new input contains a pattern that resembles the feature the individual units learned to recognize during training.

Hidden layers inhibit their outputs if the input pattern does not contain the features they were trained to recognize. Upper layer patterns can be thought of as a pattern with features that can be recognized by units in the subsequent layer. The output pattern generated can be thought of as a feature map that provides an indication of the presence or absence of many different feature combinations at the input.

BPN provides an effective means of allowing a computer system to examine data patterns that may be incomplete or noisy, and to recognize subtle patterns from the partial input. BPN will classify these previously unseen inputs according to the features they share with the training examples.

2.1.7.2 Selection and preparation of training data

A neural network is not useful if it only sees one example of a matching input/output pair. It cannot infer the characteristics of the input data that one is looking for from only one example. Therefore many examples are required. This is analogous to how a child learns the difference between say different types of animals. The child needs to see several examples of each to be able to classify an arbitrary animal. The same is true with neural networks. The best approach to training is to compile a wide range of examples which exhibit all the different characteristics the user is interested in. Adding some noise or other randomness to your example (such as a random scaling factor) prior to training helps to account for noise and natural variability in real data, and tends to produce a more reliable network [7].

If you are using a standard sigmoid node transfer function (not scaled), please note that the desired output must never be set exactly to 0 and 1. This is because the asymptotic values of the hidden layers are set between 0 and 1. This would then mean more data and weights are required to reach the desired output. Again the limits cannot be exceeded. To avoid this limitation, the desired output can be lowered to say 0.9 for the network to reach and even overshoot. The network will converge relatively quickly. It cannot be overemphasized that a neural network is only as good as the training data. Poor training data inevitably leads to an unreliable and unpredictable network.

2.1.7.3 Modification of the neuron connection weights

Consider the example in Figure 2.9 below

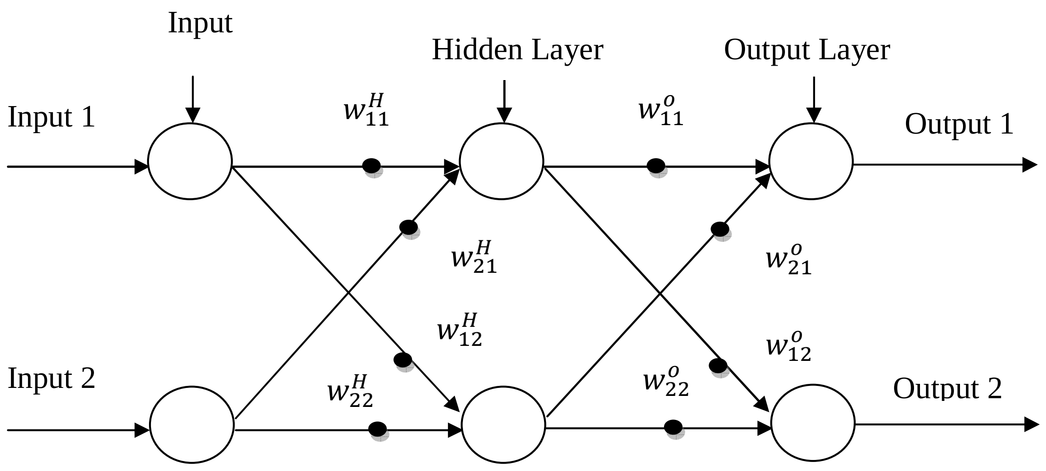

Figure 2-9: 2-Input 2-Output ANN

If I_1,I_2 are the inputs, H_1 H_2 the hidden layer outputs and O_1, O_2 are the output layer outputs respectively then:

Outputs of Hidden Node 1 and 2 are given by [8], [9]

$$\mathrm{H_1} = \mathrm{sgm}\left(\sum_{\mathrm{l}=1}^{2} \mathrm{I_l w_{l1}^H}\right) \tag{2.5}$$

and

$$\mathrm{H_2} = \mathrm{sgm}\left(\sum_{l=1}^{2} \mathrm{I_l w_{l2}^H}\right) \tag{2.6}$$

where

$$\mathrm{sgm(x)} = \frac{1}{1 + \mathrm{e^{-x}}} \tag{2.7}$$

Output-layer outputs are given by:

$$\mathrm{O_1} = \mathrm{sgm}\left(\sum_{m=1}^{2} \mathrm{H_m w_{m1}^O}\right) \tag{2.8}$$

and

$$\mathrm{O_2} = \mathrm{sgm}\left(\sum_{m=1}^{2} \mathrm{H_m w_{m2}^O}\right) \tag{2.9}$$

From equations (2.8) and (2.9), we can calculate the output given a particular set of inputs. This allows us to calculate the mean squared error (MSE) between the actual output and the desired output for the given input in this training example. This gives us the average of what we want and what we got.

The error function can be written as:

$$\mathrm{E} = \sum_{n=1}^{2} (\mathrm{D_n} - \mathrm{O_n})^2 \tag{2.10}$$

where D_n is the n^{th}desired output. or, using (2.5) and (2.8)

$$\mathrm{E} = \sum_{n=1}^{2} \left(\mathrm{D_n} - \mathrm{sgm}\left(\sum_{m=1}^{2} \mathrm{sgm}\left(\sum_{l=1}^{2} \mathrm{I_l w_{lm}^H}\right) \mathrm{w_{mn}^O}\right)\right)^2 \tag{2.11}$$

For a given training the minimum error would be the best target to our training. To find this point, the gradient of the error function with respect to each network weight would be calculated. Adjustments of the weights would then follow the opposite to the gradient.

The gradient is fairly straightforward to calculate, due to the convenient fact that the derivative of the sigmoid function can be expressed in terms of the function itself:

$$\frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) = \frac{e^{-x}}{(1+e^{-x})^2} = \big(1-\mathrm{sgm}(x)\big)\mathrm{sgm}(x) \tag{2.12}$$

The gradient is defined as the vector of partial derivatives of the multivariate function with respect to each of variable. The error function for each network output is calculated as a set of partial derivatives with respect to each associated connection weight. All other variables except one are held constant when we calculate the partial derivative.

$$\frac{\partial O_n}{\partial w^o_{mn}} = \frac{\partial}{\partial w^o_{mn}} \sum_{k=1}^{2} w^o_{kn} H_k = H_m \tag{2.13}$$

The gradient of the error function can be calculated as:

$$\frac{\partial E}{\partial w^o_{mn}} = \frac{\partial}{\partial w^o_{mn}} \sum_{n=1}^{2} (D_n - O_n)^2$$

$$= -2(D_n - O_n)\frac{\partial}{\partial S^o} sgm(S^o)\frac{\partial S^o}{\partial w^o_{mn}}$$

$$= -2(D_n - O_n)\big(\big(1-\mathrm{sgm}(S^o)\big)\mathrm{sgm}(S^o)\big)H_m \tag{2.14}$$

where $S^o = \sum_{k=1}^{2} w^o_{mn}$.

The expression $'-2(D_n - O_n)((1 - sgm(S^o))sgm(S^o))H_m'$ is denoted as δ^o_n.

The new values for the network weights are calculated by multiplying the negative gradient with a step size parameter called the learning rate and adding the resultant vector to the vector of network weights attached to the current layer. This change does not take place, however, until after the middle-layer weights are updated as well, since this would corrupt the weight update procedure for the middle layer [10].

For the middle layer, a new gradient is derived, but this time the output weights are treated as constants rather than the hidden-layer weights.

$$\frac{\partial E}{\partial w_{lm}^{H}} = (\left(1 - \mathrm{sgm}\left(S^{H}\right)\right) \mathrm{sgm}\left(S^{H}\right)) \sum_{n=1}^{2} \delta_{n}^{o} w_{mn}^{o} I_{l} \tag{2.15}$$

The middle weights are updated using the same procedure as for the output layer, and the output layer weights are updated as well. This is a complete training cycle for one piece of training data. The input layer is treated as a buffer for holding the input vector hence has no weights that need modifications.

The sample considered is a (2, 2, 2) network. Larger networks would have longer summations though the learning principle is the same.

2.1.7.4 Repetition

The above procedure causes the output to move a small step towards the desired state of a minimized error. The procedure must be repeated many times until the MSE drops below a specified value. When this happens, the network is performing satisfactorily, and this training session for this particular example has been completed.

Training will be said to be successful when random data is applied to the input terminals repeatedly for many times depending on the application and complexity of the data and other parameters. After training the network, real data new to the network is presented to the input for classification, compression or processing.

A consequence of the BPN algorithm is that there are situations where it can get stuck to ‘local minima’ that traps the algorithm and prevents it from dropping to the actual minimum. If such a situation arises hidden layers can be added or reduced, nodes can be reduced or increased or try another starting point (randomize the network again). Other approaches to the BPN problem are based on alternative determination of MSE (least squares approximation and steepest-descent technique) [11], [12]

2.2 Competitive Learning

The basic idea underlying what is called competitive learning [13] is roughly as follows. Assume a sequence of statistical samples of a vectorial observable $= \boldsymbol{x}(t) \in \mathbb{R}^n$, where t is the time coordinate, and a set of variable reference vectors $\{\boldsymbol{m}_i(t) \colon \boldsymbol{m}_i \in \mathbb{R}^n, i = 1,2,\dots,k\}$. Assume that the $\boldsymbol{m}_i(0)$ have been initialized in some proper way; random selection will often suffice. If $\boldsymbol{x}(t)$ can somehow be simultaneously compared with each $\boldsymbol{m}_i(t)$ at each successive instant of time, taken here to be an integer $t = 1,2,3, \dots$, then the best matching $\boldsymbol{m}_i(t)$is to be updated to match even more closely the current $\boldsymbol{x}(t)$. If the comparison is based on some distance measure $d(\boldsymbol{x}, \boldsymbol{m}_i)$, altering $\boldsymbol{m}_i$ must be such that, if $i = c$ is the index of the best matching reference vector, then $d(\boldsymbol{x}, \boldsymbol{m}_c)$ is decreased, and all the other reference vectors $\boldsymbol{m}_i$ with $i \neq c$ are left intact. In this way the different reference vectors tend to become specifically “tuned” to different domains of the input variable $\boldsymbol{x}$. It will be shown below that if p is the probability density function of the samples $\boldsymbol{x}$, then the $\boldsymbol{m}_i$ tend to be located in the input space $\mathbb{R}^n$ in such a way that they approximate to p in the sense of some minimal residual error.

2.3 Learning Vector Quantization

Is a classical method, that produces an approximation to a continous probability density function $p(x)$ of the vectorial input variable x using a finite number of codebook vectors $m_i, i = 1,2,\dots,k$. Once the “codebook” is chosen, the approximation of x involves finding the reference vector m_c closest to x. One kind of optimal placement of the m_i minimizes E, the expected rth power of the reconstruction error:

$$\mathrm{E} = \int \|\mathrm{x} - \mathrm{m_c}\|^{\mathrm{r}} \mathrm{p(x)dx} \tag{2.16}$$

where dx is the volume differential in the x space, and the index $c = c(x)$ of the best-matching codebook vector (“winner”) is a function of the input vector x:

$$\|\mathrm{x} - \mathrm{m_c}\| = \min_{\mathrm{i}} \{\|\mathrm{x} - \mathrm{m_j}\|\} \tag{2.17}$$

In general no closed-form solution for the optimal placement of the m_i is possible, and iterative approximation schemes must be used.

Using the square-error criterion ($r = 2$), it can also be shown that the following step-wise “delta-rule,” in the discrete time formalism ($t = 0,1,2,\dots$),defines the optimal values asymptotically. Let $m_c = m_c(t)$ be the closest code-book vector to $x = x(t)$ in the Euclidean metric. The steepest-descent gradient-step optimization of Ein the m_c space yields the sequence.

$$m_c(t+1) = m_c(t) + \alpha(t)[x(t) - m_c(t)],$$

$$\mathrm{m_i(t+1) = m_i(t)\ for\ i \neq c} \qquad (2.18)$$

with $\alpha(t)$ being a suitable, monotonically decreasing sequence of scalar-valued gain coefficients, $0 < \alpha(t) < 1$. This then is the simplest analytical description of competitive learning.

2.4 Kohonen Self-Organizing Maps

Kohonen Self-Organizing Maps (or just Self-Organizing Maps, or SOM’s for short) are a type of neural network. They were developed in 1982 by Teuvo Kohonen, a professor emeritus of the Academy of Finland. Self-Organizing Maps are called “Self organizing” because no supervision is required. SOM’s learn on their own through unsupervised competitive learning. They are known as “Maps” because they attempt to map their weights to conform to the given input data. The nodes in a SOM network learn in the sense that they attempt to become like the inputs presented to them. They can also be called “Feature Maps” as in Self-Organizing Feature Maps. Retaining principle “features” of the input data is a fundamental principle of SOM’s, and one of the things that make them so valuable. Specifically, the topological relationships between input data are preserved when mapped to a SOM network. This has a pragmatic value of representing complex data.

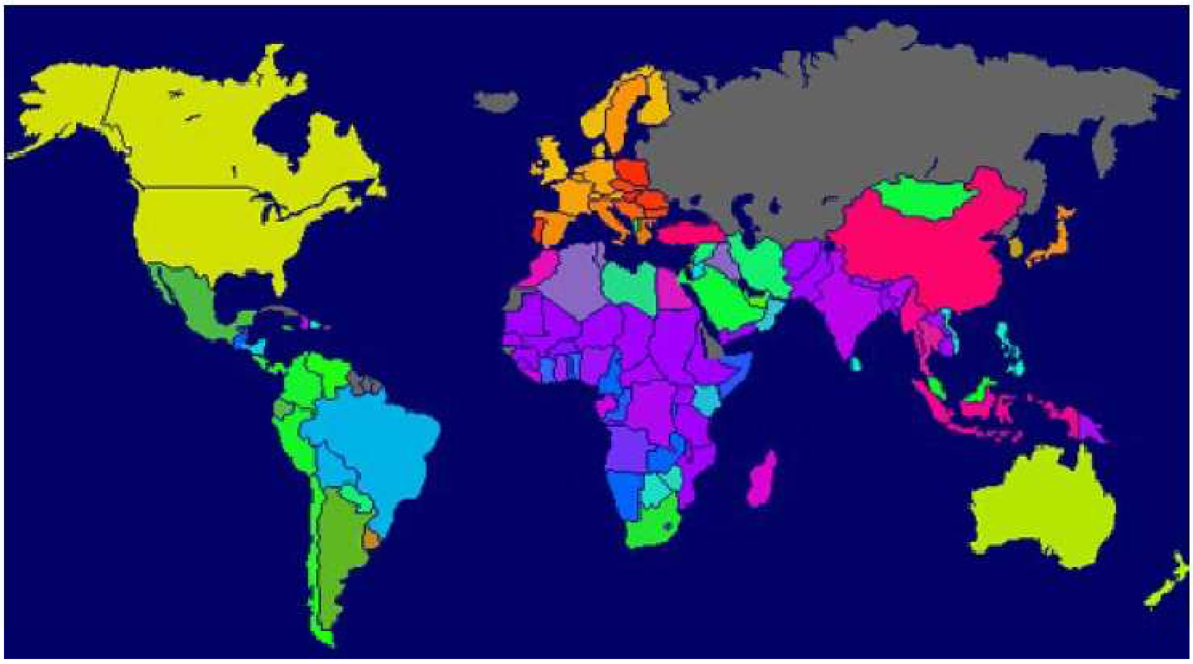

Figure 2-10: A map of the world quality of life

Figure 2.10 is a map of the world quality of life. Yellows and oranges represent wealthy nations while purples and blues are the poorer nations. From this view, it can be difficult to visualize the relationships between countries. However, when represented by SOM as shown in Figure.2.11 below, it becomes easier to see what is going on.

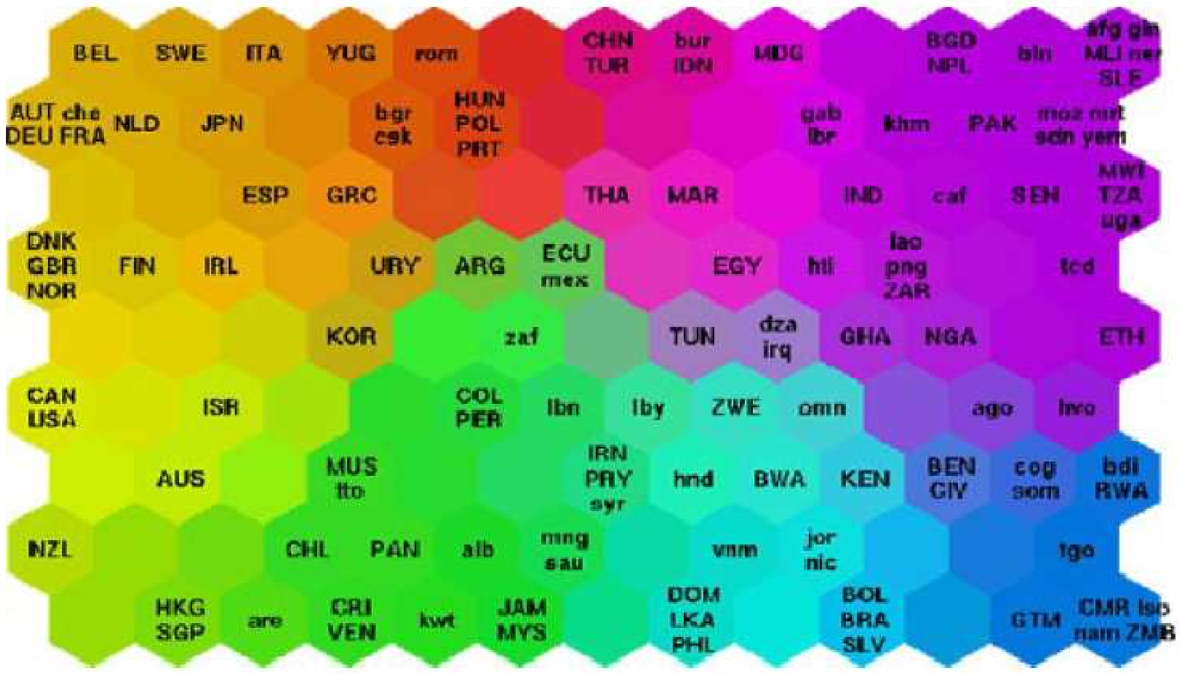

Figure 2-11: SOM representation of world quality of life

In figure 2.11 we can see the wealthiest countries i.e United States, Canada and Western European countries on the left side of the network while the poorest countries can be found on the opposite side of the map (at the point furthest away from the richest countries), represented by the purples and the blues.

Figure.2.11 is a hexagonal grid. Each hexagon represents a node in the neural network. This is typically called a unified or u-matrix and is probably the most popular method of displaying SOMs. Another intrinsic property of SOM’s is known as vector quantization. This is a data compression technique. SOMs provide a way of representing multi-dimensional data in much lower dimensional space-typically one or two dimensions. This aides in their visualization, as humans are more proficient at comprehending data in lower dimensions than higher dimensions as can be seen in the comparison of Figure 2.10 to Figure 2.11.

2.4.1 Structure of a Self-Organizing Map

The structure of a SOM is fairly simple and is best understood through the use of an illustration as shown in Figure 2.12 below

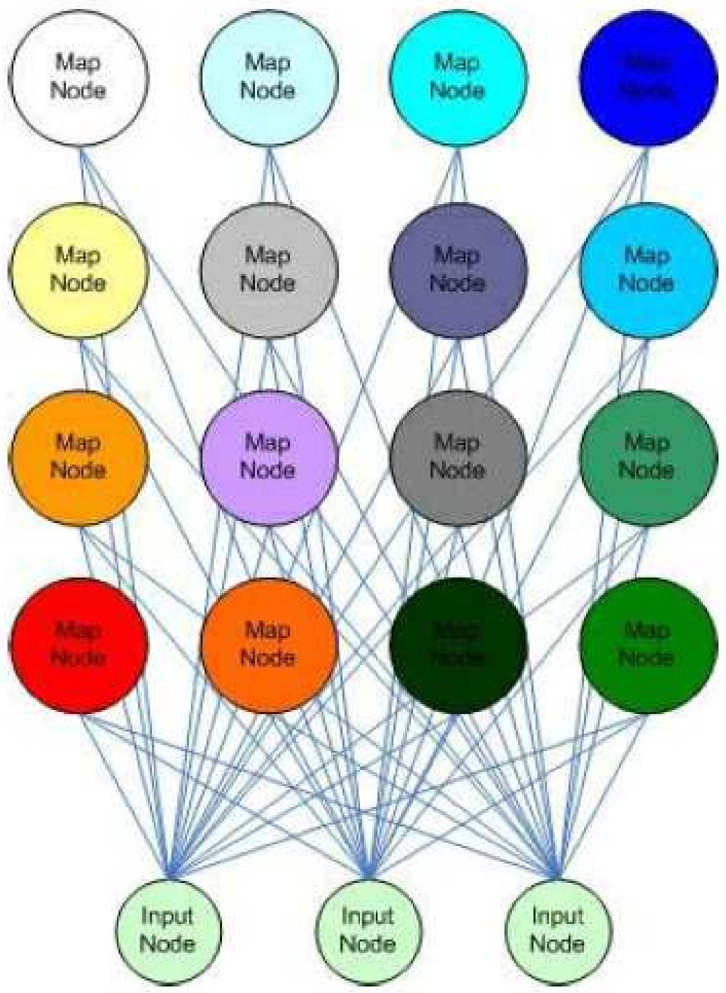

Figure 2-12: Structure of the self-organizing map

Figure.2.12 is a 4×4 SOM network (4 nodes down, 4 nodes across). It is easy to overlook this structure as being trivial but there are a few key things to notice. First, each map node is connected to each input node. For this small 4×4 network, that is $4 \times 4 \times 3 = 48$ connections. Secondly, notice that map nodes are not connected to each other. The nodes are organized in this manner of a 2-D grid as it makes it easy to visualize the results. This representation of the map is by the SOM algorithm. In the drawn configuration, each map node has a unique (i,j) coordinate. This makes it easy to reference a node in the network and to calculate the distances between nodes. Because of the connections only to the input nodes, the map nodes are oblivious as to what values their neighbors have. A map node will only update it's weights (explained next) based on what the input vector tells it.

The following relationships describe what a node essentially is:

1.$network \subset mapNode \subset float\ weights[numWeights]$

2.$inputVectors \subset inputVector \subset float\ weights[numWeights]$

1 says that the network (the 4×4 grid above) contains map nodes. A single map node contains an array of floats, or it's weights. numWeights will become more apparent during application discussion. The only other common item that a map node should contain is it's (i,j) position in the network. 2 says that the collection of input vectors (or input nodes) contains individual input vectors. Each input vector contains an array of float's or its weights. Note that numWeights is the same for both weight vectors. The weight vectors must be the same for map nodes and input vectors or the algorithm will not work.

2.4.2 The SOM algorithm

The Self-Organizing Map algorithm [14] can be broken up into 6 steps

1) Each nodes weights are initialized.
2) A vector is chosen at random from the set of training data and presented to the network
3) Every node in the network is examined to calculate which one's weights are most like the input vector. The winning node is commonly known as the $Best\ Matching\ Unit\ (BMU)$.
4) The radius of the neighborhood of the BMU is calculated. This value starts large and is typically set to be the radius of the network, diminishing each time step. (2.20, 2.21).
5) Any nodes found within the radius of the BMU, calculated in 4), are adjusted to make them more like the input vector (2.22, 2.23). The closer a node is to the BMU, the more its weights are altered (2.24).
6) Repeat 2) for N iterations

The equations utilized by the algorithm are as follows:

Equation (2.19)-Calculate the BMU.

$$\mathrm{distFromInput}^2 = \sum_{\mathrm{i}=0}^{\mathrm{n}} (\mathrm{I_i} - \mathrm{W_i})^2 \tag{2.19}$$

$$I = current\ input\ vector$$

$$W = nodes\ weight\ vector$$

$$n = number\ of\ weights$$

Equation (2.20)-Radius of the neighborhood.

$$\sigma(t) = \sigma_o e^{-t/\lambda} \tag{2.20}$$

$$t = current\ iteration$$

$$\lambda = time\ constant\ (Equation\ 2b)$$

$$\sigma_o = radius\ of\ the\ map$$

Equation (2.21)-Time constant

$$\lambda = \text{numIterations}/\text{mapRadius} \tag{2.21}$$

Equation (2.22)-New weight of a node

$$W(t+1) = W(t) + \Theta(t)L(t)(I(t) - W(t)) \tag{2.22}$$

Equation (2.23)-Learning rate

$$L(t) = L_o e^{-t/\lambda} \tag{2.23}$$

Equation (2.24)-Distance from BMU

$$\Theta(t) = e^{\left(-\frac{\text{distFromBMU}^2}{2\sigma^2(t)}\right)} \tag{2.24}$$

There are some things to note about these formulas. (2.19) is simply the Euclidean distance formula squared. It is squared because we are not concerned with the actual numerical distance from input. We just need some sort of uniform scale in order to compare each node to the input vector. This equation provides that, thus eliminating the need for the computationally expensive square root operation for every node in the network.

Equations (2.20) and (2.21) utilize exponential decay. At $t = 0$ they are at their maximum. As (t =the current iteration number) increases, they approach zero. This is exactly what we want. In (2.20) the radius should start out as the radius of the lattice, and approach zero at which time the radius is simply the BMU node. (2.21) is almost arbitrary. Any constant value can be chosen. This provides a good value, though, as it depends directly on the map size and the number of iterations to perform.

Equation (2.22) is the main learning function. $W(t+1)$ is the new ‘educated’ weight value of a given node. Over time, this equation essentially makes a given node weight more like the currently selected input vector, I. A node that is very different from the current input vector will learn more than a node very similar to the current input vector. The difference between the node weight and the input vector are then scaled by the current learning rate of the SOM, and by $\Theta(t)$.

$\Theta(t)$, Equation (2.24), is used to make nodes closer to the BMU learn more than nodes on the outskirts of the current neighbourhood radius. Nodes outside of the neighbourhood radius are skipped completely. $distFromBMU$ is the actual number of nodes between the current node and the BMU, easily calculated as:

$$\mathrm{distFromBMU}^2 = (\mathrm{bmuI} - \mathrm{nodeI})^2 + (\mathrm{bmuJ} - \mathrm{nodeJ})^2 \tag{2.25}$$

This can be done since the node network is just a 2-D grid of nodes. With this in mind, nodes on the fringe of the neighborhood radius will learn some fraction less than 1.0. As $distFromBMU$ decreases, $\Theta(t)$ approaches 1.0. The BMU itself will have a $distFromBMU$ equal to 0, which gives $\Theta(t)$ its maximum value of 1.0. Again, this Euclidean distance remains squared to avoid the square root operation.

There exists a lot of variation regarding the equations used with the SOM algorithm. There is also a lot of research being done on the optimal parameters. Some things of particularly heavy debate are the number of iterations, the learning rate and the neighborhood radius. Kohonen himself has suggested however that the training should be split into two phases. In phase 1 the learning coefficient is reduced from 0.9 to 0.1 and the neighborhood radius from half the diameter of the lattice to the immediately surrounding nodes. In phase 2 the learning rate is reduced from 0.1 to 0.0 but over double or more the number of iterations in phase 1. In phase 2, the neighborhood radius value should remain fixed at 1 (the BMU only). Phase 1 allows the network to quickly fill out the space, while Phase 2 performs the ‘fine tuning’ of the network to a more accurate representation.

2.4.3 Applications

2.4.3.1 Color Classification

The color classification SOM is used as the primary example in most tutorials published on SOM's. This is for a very good reason. By going through this example, the concept of SOMs can be solidly grasped. The reason color classification is fairly easy to understand is because of the relatively small amount of data utilized, as well as the visual aspect of the data.

Color classification SOMs only use three weights per map and input nodes. These weights represent the (r,g,b) triplet for the color. For example, colors may be presented to the network as (1,0,0) for red,(0,1,0) for green, etc. The goal for the network here, is to learn how to represent all of these input colors on it's 2-D grid while maintaining the intrinsic properties of a SOM such as retaining the topological relationships between input vectors. With this in mind, if light blue and dark blue are presented to the SOM, they should end up next to each other on the network grid.

To illustrate the process, we will step through the algorithm for the color classification algorithm. Step 1 is the initialization of the network. Figure 2.13 shows a newly initialized network where each square is a node in the network.

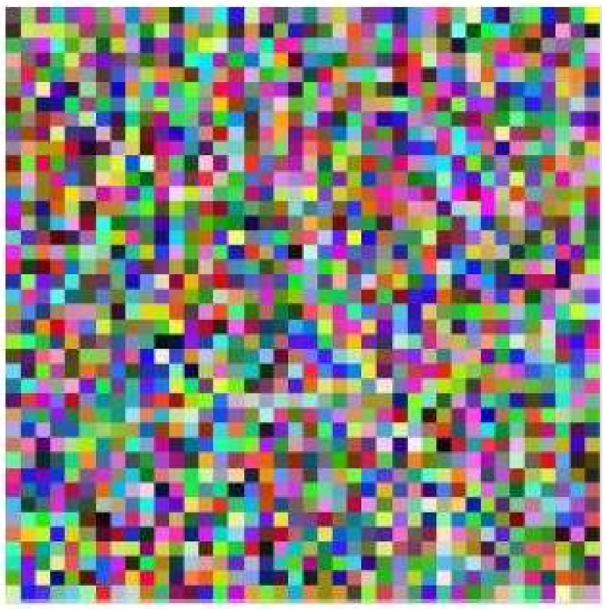

Figure 2-13: Newly initialized color classification SOM

The initialization method used here is to assign a random value between 0.0 and 1.0 for each component (r, g, and b) of each node. Step 2 is to choose a vector at random from the input vectors. Eight input vectors are used in this example, ranging from red to yellow to dark green. Next, step three goes through every node and finds the BMU, as described earlier. Figure 2.14 shows the BMU being selected in the 4×4 network. Step 4 of the algorithm calculates the neighborhood radius. This is also shown in Figure 2.14. All the nodes tinted red are within the radius. Step 5 then applies the learning functions to all these nodes. It is based on their distance from the BMU. The BMU (dark red) learns the most while nodes on the outskirts of the radius (light pink) learn the least. Nodes outside of the radius (white) don't learn at all.

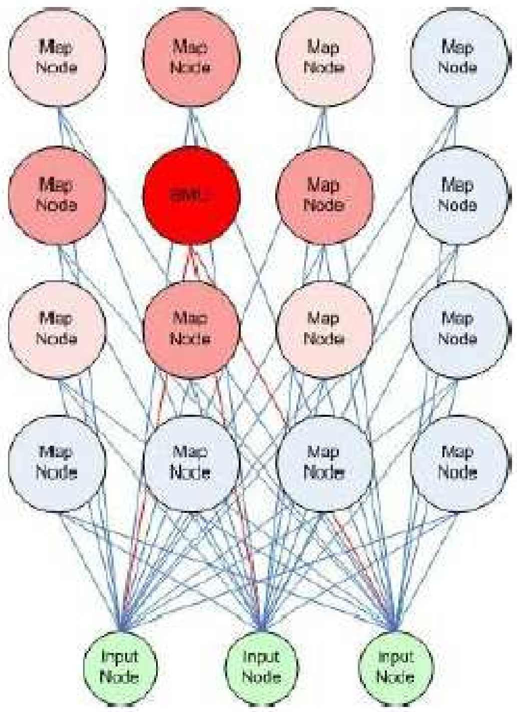

Figure 2-14: Illustration of BMU and neighboring nodes for color classification SOM

We then go back to step 2 and repeat. Figure 2.15 shows a trained SOM, representing all eight input colors. Notice how light green is next to dark green, red is next to orange. An ideal map would probably have light blue next to dark blue. This is where the error map comes in, which is described next.

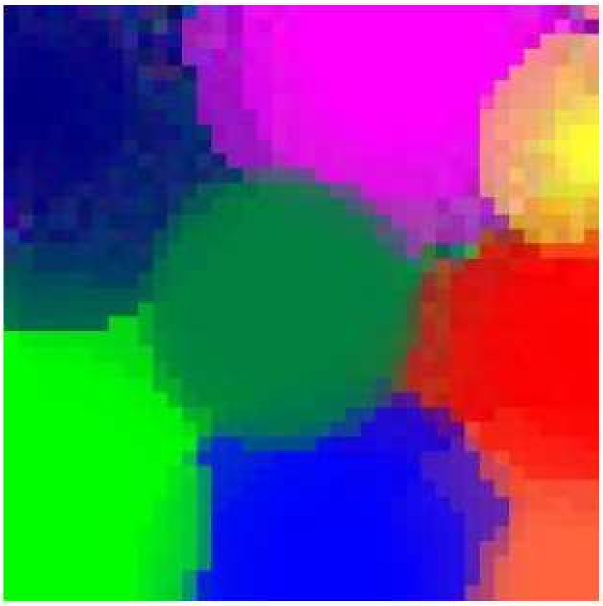

Figure 2-15: Trained color classification SOM

Each time a SOM is trained, it can produce a completely different result given the same input data. This is because the network is initialized with random colours, presenting a unique set-up prior to each training session. Also, this occurs because input vectors to be presented to the network are chosen at random. With this in mind, some SOM's may turn out to be 'better' than others where 'better' is a measure of how well the topological data is preserved. One method for gauging the superiority of one SOM over another is to calculate the error map. This will give us some numeric value. SOMs with lower values can be said to be 'better' mappings. A SOM with a value of zero would be a perfect mapping where the entire network is the exact same color.

To calculate an error map, loop through every map node of the network. Add up the distance (not the physical distance but the weight distance. This is exactly the same as how the BMU is calculated) from the node we're currently evaluating, to each of it's neighbours. Average this distance. Multiply this by 3 (the number of weights used), assuming no square root is used to calculate the distance between adjacent nodes. If the square root operation is used, multiply by $\sqrt{3}$ instead. Assign this value to the node. This gives each map node a nice value between 0.0 and 1.0. These values can then be used as the grayscale values for each square of the Error map window. Pure white represents the maximum possible distance between adjacent nodes, while black shows that adjacent nodes are of the same colour. Shades of gray in between give an even finer explanation, with darker grays being a better map than a map with light grays. Figure 2.16 shows an example. Notice the lines and how they line up with Figure 2.15.

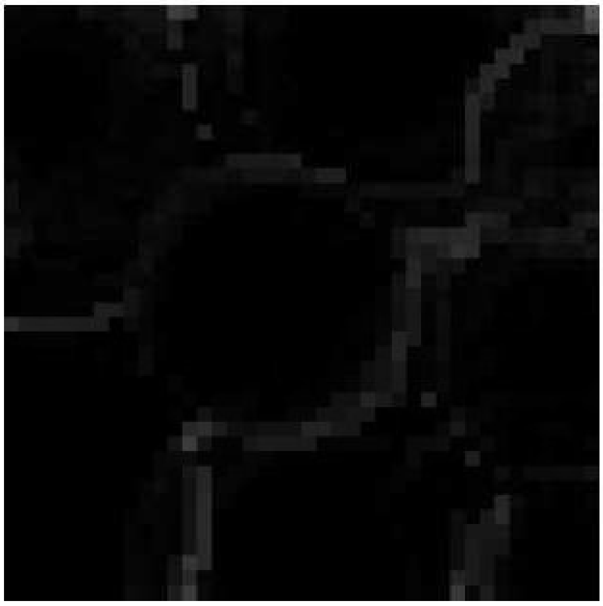

Figure 2-16: Error map for color classification SOM

2.4.4 Batch SOM

An important variant of the basic SOM is the batch algorithm in which the whole training set is gone through at once and only after this the map is updated with the net effect of all samples. This algorithm executes much faster in software than the normal sequential algorithms, and the results are typically just as good or even better [15]. The learning steps are defined as follows:

1) For the initial reference vectors, take for instance the first K training samples. Where $K$is the number of reference vectors.
2) For each map unit i, collect a list of copies of all those training samples x whose nearest reference vector belongs to the topological neighbourhood set N_i of unit i.
3) Take for each new reference vector the mean over the respective list.
4) Repeat from 2 a few times

This algorithm is particularly effective if the initial values of the reference vectors are already roughly ordered, even though they would not yet approximate to the distribution of the samples [16]. It should be noticed that the above algorithm contains no learning rate parameter; therefore it seems to yield stabler asymptotic values for the m_i than the original SOM.

Definition of the size of the neighborhood set N_i can be similar as in the basic SOM algorithms. "Shrinking" of N_i in this algorithm means that the neighbourhood size is decreased while the steps 2 and 3 are repeated. At the last iteration N_i may contain the element i only.

2.4.5 SOM simulations

The self-organizing map is a sheet-like artificial neural network, the cells of which become specifically tuned to various input signal patterns or classes of patterns through an unsupervised learning process. In the basic version only one cell or local group of cells at a time gives the active response to the current input. The locations of the responses tend to become ordered as if some meaningful coordinate system for different input features were being created over the network. The spatial location or coordinates of a cell in the network then corresponds to a particular domain of input signal patterns. SOM can thus be used to visualize metric ordering relations of input samples. A typical application of SOM is in the analysis of complex experimental vectorial data such as process states, where the data elements may even be related to each other in a highly nonlinear fashion. Here we define the process of creating a SOM as contained in [17].

2.4.5.1 Data pre-processing

Sometimes there is missing data to complete each input vector. However, such incomplete training examples still contain useful information. The distribution statistics of the available vector components can still be determined from partial data, for example. Using the self-organizing map algorithm one can easily utilize partial training data [18], [19]. For incomplete input data vectors the SOM_PAK has the possibility to mark the missing values by a pre-defined string which is 'x' by default. The SOM_PAK routines will compute the distance calculations and reference vector modification steps using the available data components.

The input data should also be scaled so that corresponding components have approximately the same dynamic range. This assures that for each component, the difference between two samples contribute approximately an equal amount to the summed distance measure between an input sample and codebook vector.

2.4.5.2 Initialization

Kohonen presents three different types of network initializations: random, initial samples, and linear initialization.

Random initialization

Random initialization means that random values are assigned to codebook vectors. This is particularly useful if nothing or little is known about the input data at the time of initialization.

Initial samples initialization

Initial samples of the input data set can be used for codebook vector initialization. This has the advantage that the points automatically lie in the same part of the input space with the data.

Linear initialization

The codebook vectors are initialized to lie in the same input space that is spanned by two eigenvectors corresponding to the largest eigenvalues of the input data. This has the effect of stretching the SOM to the same orientation as the input data.

2.4.5.3 Map topology

Refers to the locations of units in the possible topological structures. Two topologies are used in SOM_PAK namely the rectangular and hexagonal topologies as illustrated in Figure 2.17 below. The choice of a topology is a matter of taste [17] but the hexagonal topology is more effective for visual display.

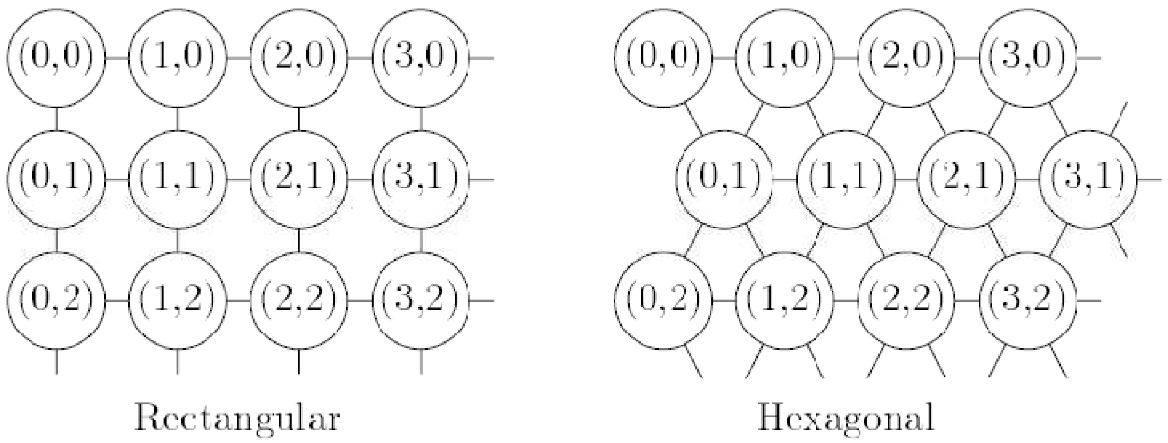

Figure 2-17: SOM topologies

The distance between two units in the map is computed as an Euclidean distance in the (two dimensional) map topology.

In section 2.4.5, it was pointed out that in the SOM algorithm, after the Best Matching Unit has been found, then a neighborhood $h_{ci}(t)$ function is used to determine the nodes in the neighborhood of the BMU which should be updated. Two types of neighborhood functions are available: bubble and Gaussian. Let the BMU be referred to by the index c and its neighborhood by N_c (which is time variable and hence denoted by $N_c(t)$). See Figure 2.18 below for an illustration of neighborhoods of varying sizes about the BMU. In the bubble function, $h_{ci} = \alpha(t)$ if node index $i \in N_c$ and $h_{ci} = 0$ if node index $i \notin N_c$, where $\alpha(t)$ is some monotonically decreasing function of time $(0 < \alpha(t) < 1)$. In gaussian function

$$\mathrm{h_{ci}} = \alpha(\mathrm{t}).\exp\left(-\frac{\|\mathrm{r_c} - \mathrm{r_i}\|^2}{2\sigma^2(\mathrm{t})}\right) \tag{2.26}$$

where $r_c \in \Re^2$ and $r_i \in \Re^2$ are the radius vectors of nodes c and i respectively. $\alpha(t)$ is another scalar-valued "learning rate" and the parameter $\sigma(t)$ defines the width of the kernel. The latter corresponds to the radius of N_c above.

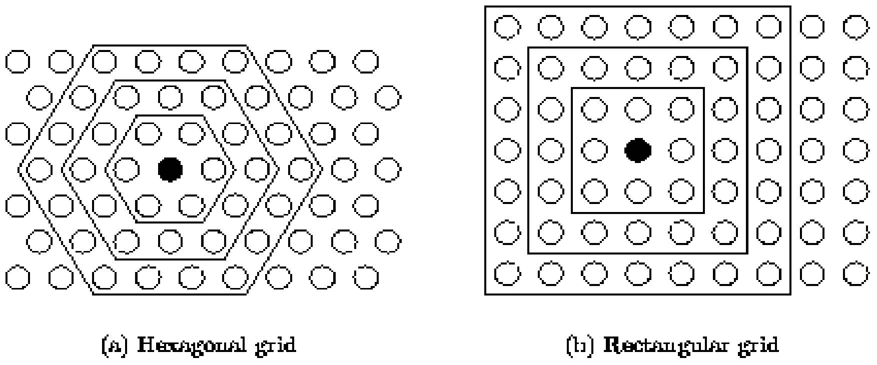

Figure 2-18: Neighborhoods (size 1, 2 and 3) of the unit marked with black dot: (a) hexagonal lattice, (b) rectangular lattice.

2.4.5.4 Quality of learning

Very different learning processes can be defined starting with different initial values $m_i(0)$, and applying different sequences of the training vectors $x(t)$ and different learning parameters. As such, it is clear that some optimal map for the same input data must exist. Also when comparing maps that have the same "stiffness" (same h_{ci}), the best map is expected to yield the smallest average quantization error because it is then fitted best to the same data. The average quantization error or the mean of $||x - m_c||$ defined via inputting the training data once again is then a useful performance index. Therefore an appreciable number (say, several tens) of random initializations of the $m_i(0)$ should be tried, and the map with the minimum quantization error selected.

2.4.5.5 SOM visualization

U-matrix

U-matrix (unified distance matrix) representation of the Self-Organizing Map visualizes the distances between the neurons. The distance between the adjacent neurons is calculated and presented with different colorings. A dark coloring between the neurons corresponds to a large distance and thus a gap between the codebook values in the input space. A light coloring between the neurons signifies that the codebook vectors are close to each other in the input space. Consequently, light areas can be thought as clusters and dark areas as cluster separators. This presentation can be helpful when one tries to find clusters in the input data without having any a priori information about the clusters.

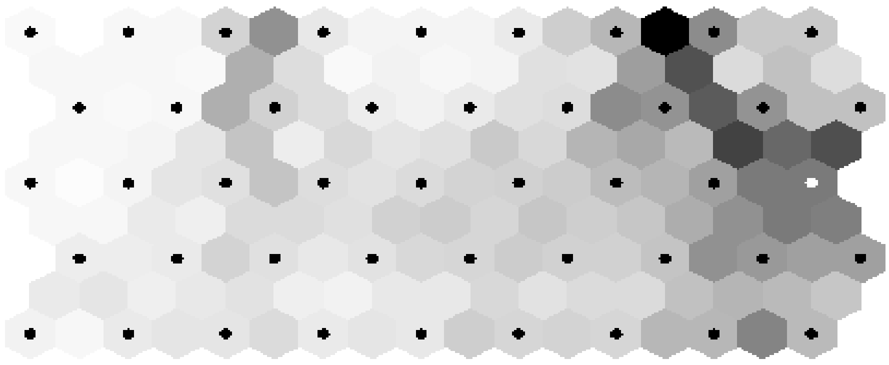

Figure 2-19: U-matrix representation of the Self-Organizing Map

In Figure 2.19 above, the neurons of the network are marked as black dots. The diagram reveals that there is a separate cluster in the upper right corner. The present clusters are separated by a dark gap. Teaching a SOM and representing it with the U-matrix thus offers a fast way to get insight of the data distribution.

U-matrix algorithm

Let n be a neuron on the map, $NN(n)$ be the set of immediate neighbors on the map and $w(n)$ the weight vector associated with neuron n, then

$$\mathrm{U-height(n)} = \sum_{\mathrm{m \in NN(n)}} \mathrm{d(w(n) - w(m))} \tag{2.27}$$

where $d(x, y)$ is the distance used in the SOM algorithm to construct the map. The U-matrix is a display of the U-heights on top of the grid positions of the neurons on the map. An U-matrix is usually displayed as a grey level picture or as a three dimensional landscape.

Component planes

Is a map imaging of the values of a selected component of the som in grayscale, i.e an imaging of corresponding components of the final map node vectors. It can thus be thought of as a sliced version of the self-organizing map. In grayscale representation, dark values represent relatively small values while white values represent relatively large values. We can see if two components relate by comparing component planes. If the outlook is similar, the components strongly correlate.

2.5 Applications of self-organizing maps to telecommunications

Modern telecommunications can benefit from neural networks in many ways. The SOM principle has been used to improve the separability of quantized signal states [20], to equalize channel properties [21], to compress information [22] and to reduce co-channel interference [23]. Other applications are in bandwidth compression [24] and adaptive equalization of PAM and QAM signals [25]. SOM has also been used for clone detection [26], [27] and complexity analysis [28] of telecommunications software. Configuration and monitoring of telecommunication traffic networks is a particularly suitable application area for SOM [29], [30], [31] [32].

In this work, SOM has been used to gain insights into the structure of an interfering wireless channel output and to utilize the information gained in mitigating inter-symbol interference.

2.5.1 Wireless Channel Fading

In this section we will describe wireless channel fading with the aim of understanding the specific types of fading, namely: frequency-selective and slow fading, for which equalization is a suitable mitigating technique.

A characteristic of the wireless channel is the presence of many different paths between the transmitter and the receiver (See Figure 2.20).

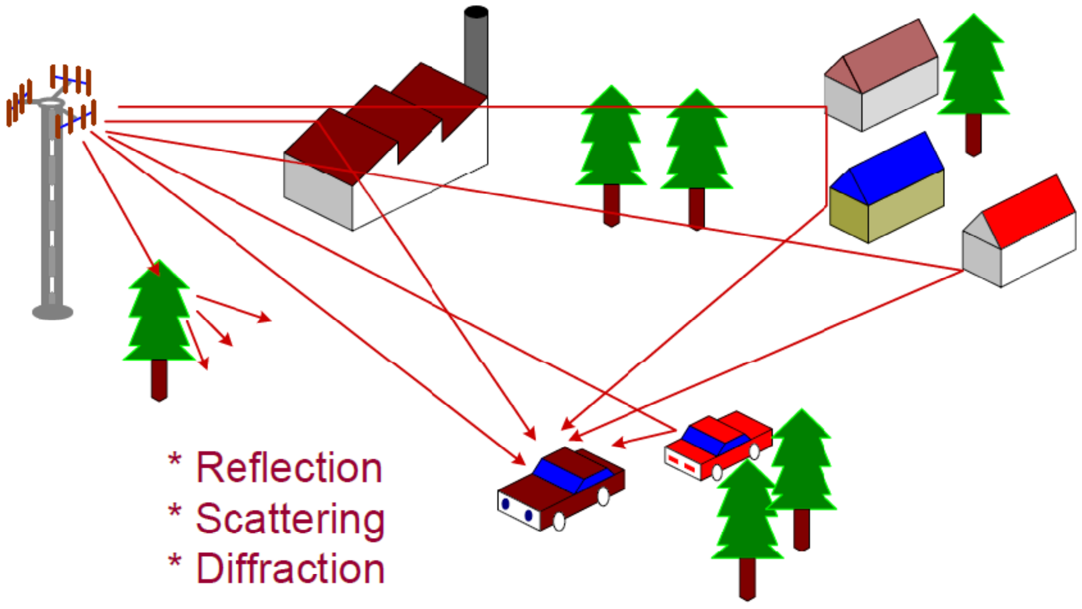

Figure 2-20: Multipath channel in wireless communications

Basic Electromagnetic (EM) wave propagation phenomena such as scattering that occurs along these paths further increases the number of the paths between the communicators.

Common propagation phenomena encountered are:

1) Reflection: EM waves are reflected when impinging on objects in their paths if the physical size of the objects are much greater than the wavelength of the EM waves.
2) Diffraction: Characterized as the sharp changes in the propagation path of EM waves that occur when they hit an obstacle with surface irregularities such as sharp edges.
3) Scattering: Occurs when EM waves visit a cluster of objects smaller in size than the wavelength, such as water vapor and foliage. Scattering causes many copies of the EM wave to propagate in various directions.

The signal power in a wireless channel varies over time and over frequency. The variations are of two types

1) Large-scale fading, due to path loss of signal as a function of distance and shadowing by large objects such as buildings and hills. This fading occurs as the mobile moves through a distance of the order of the cell size, and is typically frequency dependent.

2) Small-scale fading, due to the constructive and destructive interference of the multiple signal paths between the transmitter and receiver. Small scale fading occurs at distances of the order of the carrier wavelength and is frequency dependent.

We will focus on small scale fading rather than large scale fading for the reason that small scale fading is more relevant to the design of reliable wireless transmission, the objective of this research. Large scale fading is more relevant to telecommunication network planning.

2.5.1.1 Small-scale Fading

Due to multipath propagation, more than one version of the transmitted signal arrive at the mobile receiver at slightly different times. The interference induced by these multiple copies, also known as multipath waves, has become the most significant cause of distortion known as fading and Inter-Symbol Interference (ISI). The radio signal experiences rapid changes of its amplitude over a relatively short period of time.

The waves travelling different paths, therefore travelling different distances, sum up at the receiver antenna (or antenna array in some cases) to generate ISI of such a magnitude that the effects of large scale path loss can be completely ignored by comparison.

There are a variety of ways to statistically model the wireless channels in order to represent the random behavior of multipath fading. One simple and popular model represents the fading channel with a linear and time-varying Channel Impulse Response (CIR) denoted by the function $h(t,\tau)$

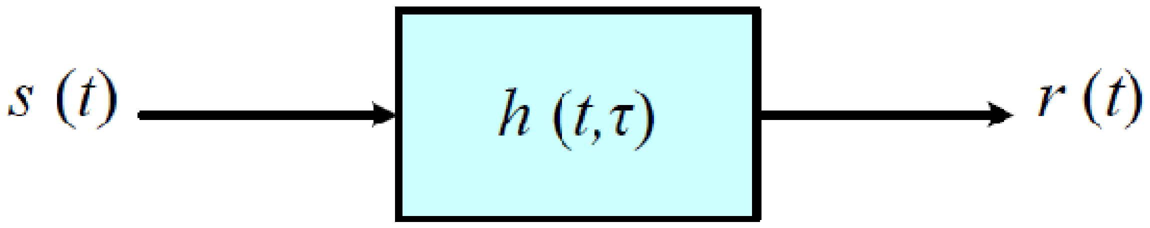

Figure 2-21: Channel modeling by impulse response

Time Dispersion Parameters

A perfect channel from a communications point of view is one that has a constant gain and a linear phase response, or at least possesses these features over a desired frequency range or bandwidth. Such a frequency range should be larger than the frequency spectrum of the transmitted signal to preserve the signal spectral characteristics. Consequently, such an ideal channel can be symbolically shown as $h(t,\tau) = g_o\delta(t)$ with g_o as a constant

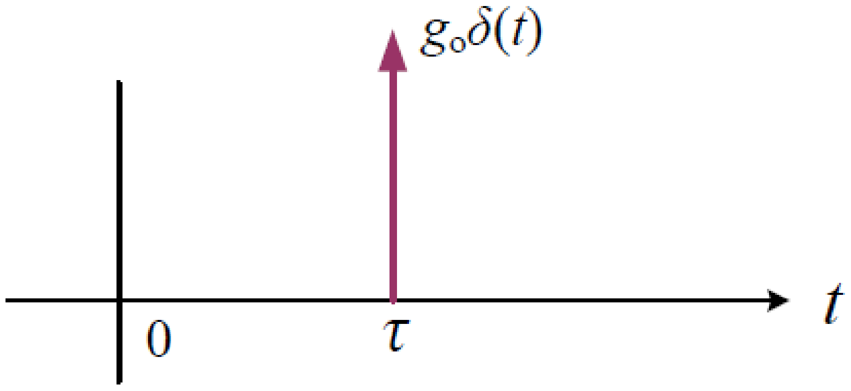

Figure 2-22: An ideal channel impulse response

Such channel impulse response implies only one received signal (delayed by), causing no ISI even when the gain varies with time as the varying CIR of $h(t,\tau) = g(t)\delta(\tau)$ where $g(t)$ is a relatively slowly varying function of time and in general may be complex-valued .If we assume that the multipath channel includes N different paths and let the power and delay of k^{th} path be given by P_k and τ_k respectively, then the weighted average delay (also known as mean excess delay) is defined as :

$$\bar{\tau} = \frac{\sum_{k=1}^{N} g_k^2 \tau_k}{\sum_{k=1}^{N} g_k^2} \tag{2.28}$$

The second statistical moment of the delay may also be computed by:

$$\overline{\tau^2} = \frac{\sum_{k=1}^{N} g_k^2 \tau_k^2}{\sum_{k=1}^{N} g_k^2} \tag{2.29}$$

The channel delay spread, that is the rms value of the delay is given by:

$$\sigma_\tau = \sqrt{\overline{\tau^2} - (\overline{\tau})^2} \tag{2.30}$$

The channels with time-dependent response (CIR) given by $h(t, \tau)$ will have time-dependent frequency response $H(\omega, t)$ (In fact, as will be stressed later, the CIRs change very slowly with respect to time in most practical cases) with

$$H(\omega, t) = \int_{-\infty}^{+\infty} h(\tau, t) e^{-j\omega\tau} d\tau \tag{2.31}$$

To determine the wireless channel characteristics in the frequency domain we first need to determine the correlation coefficient or factor of the channel frequency response, based on a change in frequency of the size $\Delta\omega$ or $2\pi\Delta f$.

$$P(\Delta\omega) = \frac{\mathbb{E}\{H^*(\omega, t)H(\omega + \Delta\omega, t)\}}{\mathbb{E}\{H^*(\omega, t)H(\omega, t)\}} = \frac{\mathbb{E}\{H^*(\omega, t)H(\omega + \Delta\omega, t)\}}{\mathbb{E}\{|H(\omega, t)|^2\}}$$

$$= \frac{\int_{-\infty}^{+\infty} |h(\tau, t)|^2 e^{-j\Delta\omega\tau} d\tau}{\int_{-\infty}^{+\infty} |h(\tau, t)|^2 d\tau} \tag{2.32}$$

The coherence bandwidth is the counterpart of the delay spread in the frequency domain, and it is the range of frequencies over which the channel gain remains about the same, or as is commonly known, over the range of frequencies the gain is flat, with a linear phase. Fortunately, the coherence bandwidth of the channel denoted by B_C can be approximated based on the specified correlation coefficient value. The case when the correlation coefficient is about zero, $(\Delta\omega) \approx 0, \Delta\omega = 2\pi B_C$. The coherence bandwidth for this case is approximated by $B_C \approx 1/\sigma_\tau$, which implies that changing the frequency by B_C results in a completely different (and statistically independent) gain. For the more common value of $P(\Delta\omega) \approx 0.5(or\ 50\%)$, the coherence bandwidth is estimated by $B_C \approx 1/5\sigma_\tau$ which implies that the channel gain at ω and ω $+ B_C$ are similar. Finally, when considering $P(\Delta\omega) \approx 0.9(or\ 90\%)$ the coherence bandwidth can be approximated as $B_C \approx 1/50\sigma_\tau$. In this case, the channel gain at ω is almost exactly the same as the gain at $\omega + B_C$.

Based on the value of coherence bandwidth B_C , and given signal bandwidth B_S , one can evaluate the channel category. When $B_C > B_S$, the channel is considered flat or flat fading.

Denoting the symbol time duration by T_S then the minimum signal bandwidth can be estimated as $B_S = \frac{1}{T_S}$. This signal bandwidth therefore has to be quite a bit less than the channel coherence bandwidth so that the channel can be assumed to be flat fading. A rule of thumb is given by

$$\mathrm{B_S} = \frac{1}{\mathrm{T_S}} \leq \frac{1}{10\sigma_\tau} \text{ or equivalently by } \frac{\sigma_\tau}{\mathrm{T_S}} \leq 0.1 \tag{2.33}$$

In the case of a flat fading channel, no compensation for the channel distortion is required. Consequently, the symbol rate in the channel has an upper bound $R_S \leq \frac{0.1}{\sigma_\tau}$. When the above condition is not met, that is $B_S > B_C$, the ISI exists and the received signal is distorted.

In most common multipath channels the ISI distortion effects are significant and dominate the channel noise. Such channels are said to be highly dispersive, and engineering efforts are focused on eliminating ISI by a process known as equalization in communication discipline, or de-convolution in some other applications such as geophysics.

In conclusion, we divide the channel behavior with respect to the signal bandwidth and according to the channel delay spread as follows:

When$B_S \ll B_C$ and $T_S \gg \sigma_\tau$ the channel is known as flat fading or frequency-not-selective

When$B_S > B_C$ and $T_S < \sigma_\tau$ the channel is known as non-flat fading or frequency selective

Frequency Dispersion Parameters

The mobility of the communicator originates another parameter known as Doppler shift in frequency, or simply some change in frequency due to the mobile velocity. When denoted by f_d , the Doppler shift is computed as $f_d = \frac{v}{\lambda} cos\theta$ in which v is the relative mobile speed, λ is the radio wavelength, and θ is the angle between the wave direction and the mobile direction. The change in frequency is positive when the mobile approaches the transmitter and negative when the mobile is departing.

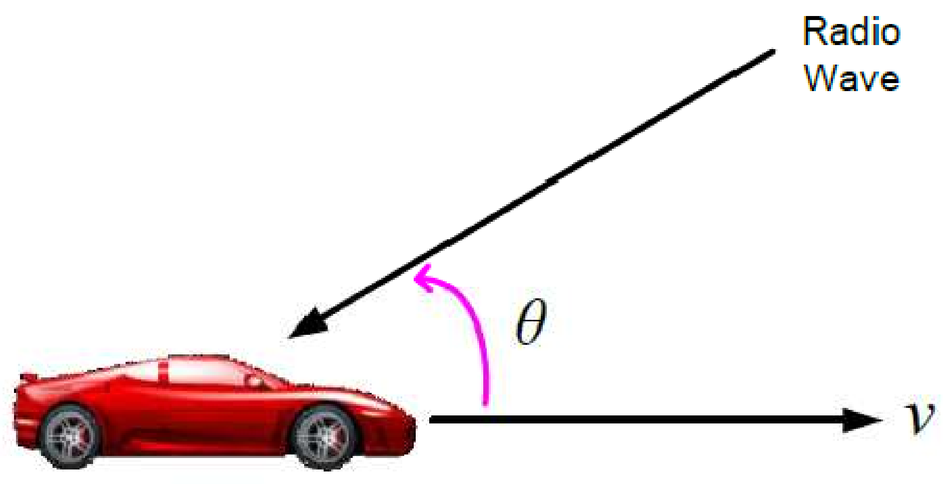

Figure 2-23: Doppler shift geometry

It is obvious that different paths have different Doppler shifts that possess random natures, as the angle θ can be considered random and in most cases uniformly distributed.

There are a number of copies of the transmitted waves at the mobile antenna, each travelling along different paths, which are characterized by various relative speeds and angles. Moreover, in specific scenarios, the surrounding objects might be moving and generating time varying Doppler shifts on multiple components.

The corresponding random change in frequency causes spectral broadening known as Doppler spread. Doppler spread is therefore defined as the range of frequencies over which the Doppler shift is not zero. We denote by B_d the maximum Doppler shift or Doppler spread of a specific wireless channel.

It is possible to categorize the wireless channel with respect to Doppler spread B_d as follows:

1) If the signal bandwidth is much greater than Doppler spread, that is $B_S \gg B_d$, the fading is known as slow fading and, hence, the effects of Doppler spread are negligible. In slow fading, the channel (in particular, CIR) changes at a much slower rate and can be assumed to be static over several symbol time durations.

2) If, on the contrary, the effects of the Doppler spread are significant and cannot be ignored in the case that $B_S < B_D$, the CIR changes rapidly with respect to the symbol time duration. Such channel is called fast fading

The time domain properties of the wireless channel can be further specified by defining another parameter, coherence time, which is the duration of time in which the CIR is invariant. Two samples of the channel are highly correlated if their time separation is less than the coherence time. The given definition itself depends on the time correlation coefficient. In the time domain, the correlation coefficient as a function of time difference Δt is given by:

$$\mathrm{P}(\Delta t) = \frac{\mathbb{E}\{h(t)h^*(t + \Delta t)\}}{\mathbb{E}\{|h(t)|^2\}} \tag{2.34}$$

Generally, the coherence time is inversely proportional to the Doppler spread.

$$\mathrm{T_C} \approx \frac{1}{\mathrm{B_d}} \tag{2.35}$$

If the coherence time for which the time correlation coefficient of (2.28) remains above 0.5 or 50% it is approximated by:

$$\mathrm{T_C} \approx \frac{9}{16\pi \mathrm{B_d}} \tag{2.36}$$

As a rule of thumb, the geometric average of (2.29) and (2.30) is used for digital communication.

$$\mathrm{T_C} = \sqrt{\frac{1}{\mathrm{B_d}} . \frac{9}{16\pi \mathrm{B_d}}} \approx \frac{0.423}{\mathrm{B_d}} \tag{2.37}$$

The channel characteristics can be categorized using the coherence time T_C , and the symbol time duration T_S as follows:

When $T_S < T_C$, the complete signal or symbol is affected similarly by the channel, and the channel is known as slow fading.

When $T_S > T_C$, different parts of a signal are affected differently because the channel changes faster compared to a symbol duration. Consequently, the channel is called fast fading

In summary, wireless channels can be divided into four types. Based on the delay spread, the channel is either flat (not frequency selective) or frequency selective, and based on Doppler spread (or, equivalently, coherence time) the channel is known as slow or fast fading.

As we will see, commonly encountered wireless channels in modern mobile communication systems are determined to be selective and slow fading. That is, the channels are usually highly dispersive. However, the variation of channels is slow with respect to time.

2.5.2 Adaptive Equalization

Adaptive equalizers [33] compensate for signal distortion attributed to inter-symbol interference (ISI) which is caused by multipath within time-dispersive channels. As shown in Figure 2.23 below, there are several methods in which equalizers can achieve adaptation of their tap coefficients. These include the transmission of a training sequence and decision- directed adaptation

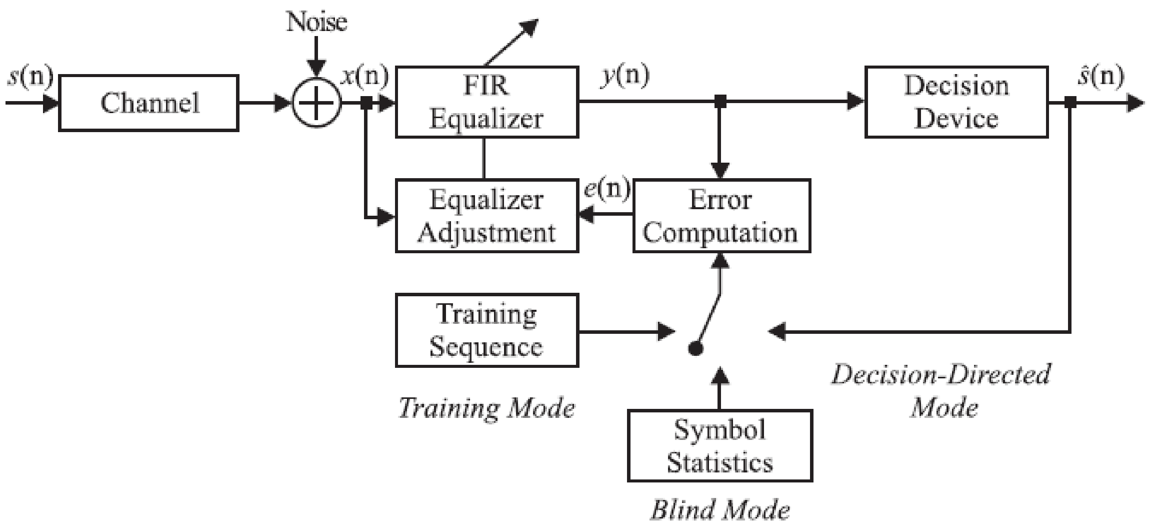

Figure 2-24: Simplified baseband model of a digital communication system

The basic data communications process can be explained by Figure 2.23. A k-bit binary sequence is mapped to a symbol $s(n)$ which is pulse-shape filtered and modulated to a band-limited communication channel. The received symbol $x(n)$ is corrupted by inter-symbol interference ISI and Gaussian white noise. The equalizer removes the distortion caused by the channel by estimating the channel inverse. The equalizer output $y(n)$ is sent to a decision device, which results in the received symbol estimate $\hat{s}(n)$. The error computation, which determines the error signal $e(n)$ used to adjust the equalizer tap coefficients, depends on the equalizer mode of operation and the corresponding equalization algorithm employed. For a non-blind adaptive equalizer such as those based on the least mean squares (LMS) algorithm, the equalizer will initially operate in the training mode. In this mode $s(n)$ is a training sequence known by the receiver and $e(n)$ will be calculated using the difference between $y(n)$ and $s(n)$. After convergence, the equalizer will be switched to the decision-directed (DD) mode where $e(n)$ will be computed based on the difference between $y(n)$ and the estimated symbol $\widehat{s}(n)$. For blind adaptive equalizers, there is no training sequence, initially $e(n)$ is a non-linear function of $y(n)$. After convergence, as for non-blind equalizers, the blind equalizer can be switched to the DD mode.

2.5.2.1 Minimum Mean-Squared-Error Equalization

The mean squared error and variants of its cost function are widely used used for equalizer design [34], [35] due in part to the simplicity of the MSE cost function and its unimodal performance surface. The result of minimum mean squared error (MMSE) equalization is an exact solution for the equalizer tap coefficients, providing the theoretical minimum for MSE based equalization algorithms. The MSE criteria attempts to minimize the expected squared magnitude of the recovery error $e(n) = d(n) - y(n)$, where $d(n)$ is the desired signal. The desired signal is the transmitted signal delayed by δ so that $d(n) = s(n - \delta)$.The MSE cost function is defined as

$$\mathcal{J}^{\mathrm{mse}} = \mathrm{E}\{\mathrm{e}^2(\mathrm{n})\} \tag{2.38}$$

$$= E\{d^2(n) - 2d(n)y(n) + y^2(n)\}$$

$$= E\{d^2(n) - 2d(n)\boldsymbol{w}^T(n)\boldsymbol{x}(n) + \boldsymbol{w}^T(n)\boldsymbol{x}(n)\boldsymbol{x}^T(n)\boldsymbol{w}(n)\}$$

$$= E\{d^2(n)\} - 2E\{d(n)\boldsymbol{w}^T(n)\boldsymbol{x}(n)\} + E\{\boldsymbol{w}^T(n)\boldsymbol{x}(n)\boldsymbol{x}^T(n)\boldsymbol{w}(n)\}.$$

In the equation above $w(n) = [w_0(n), w_1(n), \dots, w_{N-1}(n)]^T$ is an $N \times 1$ vector of equalizer tap coefficients where N is the length of the equalizer.

$\boldsymbol{x}(n) = [x_0(n), x_1(n), \dots, x_{N-1}(n)]$ is the equalizer input. When the filter coefficients are fixed, the cost function in (2.32) is not time-varying and can be re-written as

$$\mathcal{J}^{\text{mse}} = \mathrm{E}\{\mathrm{d}^2(\mathrm{n})\} - 2\mathbf{w}^{\mathrm{T}}\mathrm{E}\{\mathrm{d}(\mathrm{n})\mathrm{x}(\mathrm{n})\} + \mathbf{w}^{\mathrm{T}}\mathrm{E}\{\mathrm{x}(\mathrm{n})\mathrm{x}^{\mathrm{T}}(\mathrm{n})\}\mathbf{w} \qquad (2.39)$$

$$= E\{d^2(n)\} - 2\boldsymbol{w}^T\boldsymbol{p} + \boldsymbol{w}^T\boldsymbol{R}\boldsymbol{w}$$

where $\boldsymbol{p} = E\{d(n)x(n)\}$ is the cross-correlation vector between the desired signal and the input signal and $\boldsymbol{R} = E\{x(n)x^T(n)\}$ is the input correlation matrix.

The gradient of the MSE cost function with respect to the equalizer tap weights is defined as

$$\nabla_{\mathbf{w}}\mathcal{J}^{\text{mse}} = \frac{\partial\mathcal{J}^{\text{mse}}}{\partial\mathbf{w}} = [\frac{\partial\mathcal{J}^{\text{mse}}}{\partial \mathrm{w}_0} \quad \frac{\partial\mathcal{J}^{\text{mse}}}{\partial \mathrm{w}_1} \cdots \frac{\partial\mathcal{J}^{\text{mse}}}{\partial \mathrm{w}_{\mathrm{N}-1}}] \qquad (2.40)$$

$$= -2\boldsymbol{p} + 2\boldsymbol{R}\boldsymbol{w}$$

The optimal equalizer tap weights $\boldsymbol{w_o}$required to obtain the minimum mean-squared error (MMSE) can be determined by equating (2.34) to zero and solving for $\boldsymbol{w_o}$as follows

$$0 = 2\mathbf{R}\mathbf{w}_{\mathrm{o}} - 2\mathbf{p} \rightarrow \mathrm{w}_{\mathrm{o}} = \mathbf{R}^{-1}\mathbf{p} \qquad (2.41)$$

where the input correlation matrix $\boldsymbol{R}$ is assumed to be invertible. The MMSE, which is denoted by ξ_{min} is obtained by substituting (2.35) for $\boldsymbol{w}$ in (2.33) as follows

$$\xi_{\min} = \mathrm{E}\{\mathrm{d}^2(\mathrm{n})\} - 2\mathbf{w}_{\mathrm{o}}^{\mathrm{T}}\mathbf{p} + \mathbf{w}_{\mathrm{o}}^{\mathrm{T}}\mathbf{R}\mathbf{w}_{\mathrm{o}} \qquad (2.42)$$

$$= E\{d^2(n)\} - 2[\boldsymbol{R}^{-1}\boldsymbol{p}]^T\boldsymbol{p} + [\boldsymbol{R}^{-1}\boldsymbol{p}]^T\boldsymbol{R}[\boldsymbol{R}^{-1}\boldsymbol{p}]$$

$$= E\{d^2(n)\} - 2\boldsymbol{p}^T\boldsymbol{R}^{-1}\boldsymbol{p} + \boldsymbol{p}^T\boldsymbol{R}^{-1}\boldsymbol{p}$$

$$= E\{d^2(n)\} - \boldsymbol{p}^T\boldsymbol{R}^{-1}\boldsymbol{p}$$

$$= E\{d^2(n)\} - \boldsymbol{p}^T\boldsymbol{R}^{-1}\boldsymbol{p}$$

$$= E\{d^2(n)\} - \boldsymbol{p}^T\boldsymbol{w_o}$$

Optimal tap coefficients could be obtained by solving (2.35) but it requires the inversion of large matrices in practice, which is computationally costly. There are algorithms which can search for the near optimal or optimal coefficients in an iterative manner without carrying out the matrix inversion in (2.35). An example of such an algorithm is the method of steepest descent which is expressed as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(-\nabla_{\mathbf{w}}\mathcal{J}^{\mathrm{mse}}) \tag{2.43}$$

where μ is a constant step size. In order to converge the step size is chosen to satisfy the following requirement

$$0 < \mu < \frac{1}{\lambda_{\max}} \tag{2.44}$$

where λ_{max} is the maximum eigenvalue of the input correlation matrix $\boldsymbol{R}$. The method of steepest descend requires an estimate of the gradient which is not available in practice and thus hinders its application.

2.5.2.2 Trained Least-Mean-Squares Algorithm

The least-mean-squares algorithm (LMS) is a search method whose cost function simplifies the gradient calculation in (2.34) by replacing the expected values with the instantaneous values. There are two modes in which the LMS operates: training and tracking. The training mode occurs only once during startup for point-to-point communications and the desired signal to adapt the equalizer tap coefficients. After initialization is complete, the LMS algorithm switches to the tracking mode, where the equalizer taps are adjusted based on the difference between the equalizer output and the estimated symbol. The LMS algorithm will now be derived for both the real-valued and complex-valued cases.

Real-Valued LMS

Recall from (2.34) that the gradient of $\mathcal{J}^{mse}$ is defined as

$$\nabla_w \mathcal{J}^{mse} = -2\boldsymbol{p} + 2\boldsymbol{R}\boldsymbol{w}$$

An estimate of the gradient, $\nabla_w \mathcal{J}^{mse}$ can be obtained by replacing $\boldsymbol{R}$ and $\boldsymbol{p}$ with their instantaneous estimates $\widehat{\boldsymbol{R}} = x(n)x^T(n)$ and $\widehat{\boldsymbol{p}} = d(n)x(n)$, respectively. The gradient estimate $\nabla_w \mathcal{J}^{mse}$ is equivalent to the LMS cost function which is defined as

$$\nabla_{\mathrm{w}} \mathcal{J}^{\mathrm{lms}} = -2\widehat{\mathbf{p}} + 2\widehat{\mathbf{R}}\mathrm{w}(\mathrm{n}) \tag{2.45}$$

$$= -2\big(d(n)x(n)\big) + 2\big(x(n)x^T(n)\big)\boldsymbol{w}(n)$$

$$= -2x(n)(d(n) - x^T(n)\boldsymbol{w}(n))$$

Substituting (2.39) into (2.37), the LMS equalizer tap adjustment algorithm for real-values signals is defined as

$$\mathrm{w}(\mathrm{n}+1) = \mathrm{w}(\mathrm{n}) + \mu\big(\mathrm{d}(\mathrm{n}) - \mathrm{y}(\mathrm{n})\big)\mathrm{x}(\mathrm{n}) \tag{2.46}$$

where μ is restricted to the limits defined in (2.38).

Complex-Valued LMS

The LMS cost function for the complex case is defined as

$$\mathcal{J}^{\mathrm{lms}} = |\mathrm{e}(\mathrm{n})|^2 = \mathrm{e}(\mathrm{n})\mathrm{e}^*(\mathrm{n}) \tag{2.47}$$

$$= \big(d(n) - \boldsymbol{w}^T(n)\boldsymbol{x}(n)\big)(d^*(n) - \boldsymbol{w}^H(n)\boldsymbol{x}^*(n))$$

$$= |d(n)|^2 - \boldsymbol{w}^H(n)\boldsymbol{x}^*(n)d(n) - \boldsymbol{w}^T(n)\boldsymbol{x}(n)d^*(n) + \boldsymbol{w}^T(n)\boldsymbol{x}(n)\boldsymbol{x}^H(n)\boldsymbol{w}^*(n)$$

$$= |d(n)|^2 + \boldsymbol{w}_R^T(n)\big(\boldsymbol{x}^*(n)d(n) + \boldsymbol{x}(n)d^*(n)\big) - \boldsymbol{w}_I^T(n)\big(\boldsymbol{x}(n)d^*(n) - \boldsymbol{x}^*(n)d(n)\big)$$
$$+ \boldsymbol{w}_R^T(n)\boldsymbol{x}(n)\boldsymbol{x}^H(n)\boldsymbol{w}_R(n) + \boldsymbol{w}_I^T(n)\boldsymbol{x}(n)\boldsymbol{x}^H(n)\boldsymbol{w}_I(n)$$

where $(.)^H$ the complex conjugation and transposition operator is, while $(.)_R$ and $(.)_I$ are the real and imaginary components of a complex number, respectively. Taking the gradient of (2.41) with respect to the real and imaginary components:

$$\nabla_{\mathbf{w}}\mathcal{J}_{\mathrm{R}}^{\mathrm{lms}} = \frac{\partial\mathcal{J}^{\mathrm{lms}}}{\partial\mathbf{w}_{\mathrm{R}}(\mathrm{n})} = 2\mathbf{x}(\mathrm{n})\mathbf{x}^{\mathrm{H}}(\mathrm{n})\mathbf{w}_{\mathrm{R}}(\mathrm{n}) - (\mathbf{x}^*(\mathrm{n})\mathrm{d}(\mathrm{n}) + \mathbf{x}(\mathrm{n})\mathrm{d}^*(\mathrm{n})) \tag{2.48}$$

$$\nabla_{\mathbf{w}}\mathcal{J}_{\mathrm{I}}^{\mathrm{lms}} = \frac{\partial\mathcal{J}^{\mathrm{lms}}}{\partial\mathbf{w}_{\mathrm{I}}(\mathrm{n})} = 2\mathbf{x}(\mathrm{n})\mathbf{x}^{\mathrm{H}}(\mathrm{n})\mathbf{w}_{\mathrm{I}}(\mathrm{n}) - \mathrm{j}(\mathbf{x}(\mathrm{n})\mathrm{d}^*(\mathrm{n}) - \mathbf{x}^*(\mathrm{n})\mathrm{d}(\mathrm{n})) \tag{2.49}$$

Using (2.42) and (2.43), the complex gradient of (2.41) is defined as

$$\nabla_{\mathbf{W}}\mathcal{J}^{\mathrm{lms}} = \nabla_{\mathbf{w}}\mathcal{J}_{\mathrm{R}}^{\mathrm{lms}} + \mathrm{j}\nabla_{\mathbf{w}}\mathcal{J}_{\mathrm{I}}^{\mathrm{lms}} \tag{2.50}$$

$$= 2\boldsymbol{x}(n)\boldsymbol{x}^H(n)\boldsymbol{w}_R(n) - \big(\boldsymbol{x}^*(n)d(n) + \boldsymbol{x}(n)d^*(n)\big) + j2\boldsymbol{x}(n)\boldsymbol{x}^H(n)\boldsymbol{w}_I(n) + (\boldsymbol{x}(n)d^*(n)$$
$$+ \boldsymbol{x}^*(n)d(n))$$

$$= 2\boldsymbol{x}(n)\boldsymbol{x}^H(n)\big(\boldsymbol{w}_R(n) + j\boldsymbol{w}_I(n)\big) - 2\boldsymbol{x}^*(n)d(n)$$

$$= 2\boldsymbol{x}(n)\boldsymbol{x}^H(n)\boldsymbol{w}(n) - 2d(n)\boldsymbol{x}^*(n)$$

$$= -2\big(d(n) - \boldsymbol{w}^T(n)\boldsymbol{x}(n)\big)\boldsymbol{x}^*(n)$$

The LMS equalizer tap adjustment algorithm is defined as

$$\mathrm{w(n+1) = w(n) + \mu\big(-\nabla_w \mathcal{J}^{lms}\big)} \tag{2.51}$$

$$= w(n) + \mu\big(d(n) - y(n)\big)x^*(n)$$

where once again μ is restricted to the limits given in (2.38). Therefore the complex equalizer tap adjustment algorithm for the method of steepest descent algorithm is defined as

$$\mathbf{w}(\mathrm{n}+1) = \mathbf{w}(\mathrm{n}) + \mu \mathrm{e(n)}\mathbf{x}^*(\mathrm{n}) \tag{2.52}$$

where $e(n)$ is the error signal of the particular algorithm.

2.5.3 Symbol classification at the receiver

2.5.3.1 Principle of digital communication

Consider the communication system shown in Figure 2.24 below. Every T seconds the system sends $K = log_2 M$ bits of information through the channel for a data rate of $R = K/T$ bits per second (bps).There are $M = 2^K$ possible sequences of K bits and we say that each bit sequence of length K comprises a message $m_i = \{b_1, b_2, \dots . b_k\} \in M$ where $M = \{m_1, m_2, \dots . m_K\}$ is the set of all such messages.The messages have probability p_i of being selected for transmission where $\sum_{i=1}^{M} p_i = 1$.

Suppose message m_i is to be transmitted over the channel during the time interval [0,T).Since the channel is analog the message must be embedded into an analog signal for channel transmission. Thus, each message $m_i \in M$ is mapped to a unique analog signal $s_i(t) \in S = \{s_1(t), \dots, s_M(t)\}$ where $s_i(t)$ is defined on the time interval [0, T) and has energy

$$\mathrm{E_{s_i}} = \int_0^{\mathrm{T}} \mathrm{s_i^2(t)dt}, \quad \mathrm{i = 1, \dots, M} \tag{2.53}$$

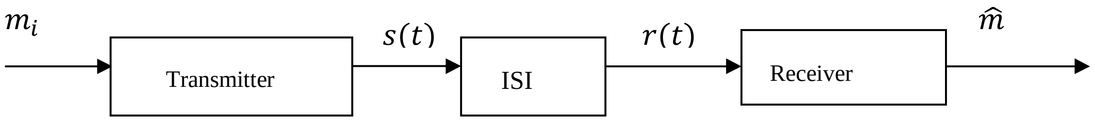

Figure 2-25: Communication system model

Since each message represents a bit sequence, each signal $s_i(t) \in S$ also represents a bit sequence, and detection of the transmitted signal $s_i(t)$ at the receiver is equivalent to detection of the transmitted bit sequence.

In the communication system model shown above, the transmitted signal is sent through a channel which introduces inter-symbol interference. Given $r(t)$ the receiver must determine the best estimate of which $s_i(t)$ was transmitted during each transmission interval $[KT, (K + 1)T)$.This best estimate for $s_i(t)$ is mapped to a best estimate of the message $m_i(t) \in M$ and the receiver then outputs this best estimate $\widehat{m} = \{\widehat{b}_1, \ldots.., \widehat{b}_k\} \in M$ of the transmitted bit sequence.

Intuitively, the receiver minimizes the probability of detection error by decoding the received signal as the signal in the set of possible transmitted signals that is closest to the one received [36]. Determining the distance between the transmitted and received signals requires a metric for the distance between signals. If we could find a way to represent digitally modulated signals as vectors in an appropriately defined vector space, then we could analyze signals in finite-dimensional vector space using classical notions of distance for vector space instead of analyzing signals infinite-dimensional space which is considerably more difficult. In the next section we show how to represent digitally modulated signals as vectors in finite dimensional space. This is called geometric representation of signals.

2.5.3.2 Geometric representation of signals

The basic premise behind a geometric representation of signals is the notion of a basis set. Specifically, using a Gram-Schmidt orthogonalization procedure, it can be shown that any set of M real energy signals $S = (s_1(t), \ldots.., s_M(t))$ defined on [0,T) can be represented as a linear combination of $N \leq M$ real orthonormal basis functions $\{\emptyset_1(t), \ldots.., \emptyset_N(t)\}$. We say that these basis functions span the set S.Thus we can write each $s_i(t) \in S$ in terms of its basis function representation as

$$s_i(t) = \sum_{j=1}^{N} s_{ij}\emptyset_j(t), \quad 0 \le t < T \tag{2.54}$$

where

$$s_{ij} = \int_0^T s_i(t)\emptyset_j(t)dt \tag{2.55}$$

is a real coefficient representing the projection of $s_i(t)$ onto the basis function $\emptyset_j(t)$ and

$$\int_0^T \emptyset_i(t)\emptyset_j(t)dt = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{2.56}$$

For most modulation techniques normally encountered in practice $N = 1\ or\ 2$ .For the case when $N = 2$ such as for QAM modulation, the basis set consists of the sine and cosine functions:

$$\emptyset_1(t) = \sqrt{\frac{2}{T}}\cos(2\pi f_c t) \tag{2.57}$$

and

$$\emptyset_2(t) = -\sqrt{\frac{2}{T}}\sin(2\pi f_c t) \tag{2.58}$$

The $\sqrt{\frac{2}{T}}$ factor is needed for normalization so that $\int_0^T \emptyset_i^2\,(t)dt = 1, i = 1{,}2$.

We denote the coefficients s_{ij} in (2.49) above as a vector $\boldsymbol{s}_i = (s_{i1}, s_{i2}, \ldots\ldots, s_{iN}) \in R^N$ which is called the signal constellation point corresponding to the signal $s_i(t)$. All the constellation points $\{\boldsymbol{s}_1, \ldots\ldots, \boldsymbol{s}_M\}$ corresponding to all the M messages constitute the signal constellation. Given the basis functions $\{\emptyset_1(t), \ldots., \emptyset_N(t)\}$ there is a one-to-one correspondence between the transmitted signal $s_i(t)$ and its constellation point $\boldsymbol{s}_i$. Specifically $s_i(t)$ can be obtained from $\boldsymbol{s}_i$ by (2.48) and $\boldsymbol{s}_i$ can be obtained from $s_i(t)$ by (2.49).Thus it is equivalent to characterize the transmitted signal by $s_i(t)$ or $\boldsymbol{s}_i$.

The representation of $s_i(t)$ in terms of its constellation point $\boldsymbol{s}_i \in R^N$ is called its signal space representation and the vector space containing the constellation is called the signal space. With this signal space representation we can analyze the infinite-dimensional functions $s_i(t)$ as vectors $\boldsymbol{s}_i$ in finite-dimensional vector space R^2.

2.5.3.3 Received vector

From the communication system model shown above, the received signal $r(t) = s_i(t) + n(t), 0 \leq t < T$. Using geometric representation of signal concepts, the received signal can be represented by the vector $\boldsymbol{r} = (r_{\boldsymbol{1}}, \dots\dots, r_N)$ where $r_j = s_{ij} + n_j, j = 1, \dots., N$.

$$s_{ij} = \int_0^T s_i(t)\emptyset_j(t)dt \tag{2.59}$$

and

$$n_j = \int_0^T n(t)\emptyset_j(t)dt \tag{2.60}$$

The vector $\boldsymbol{r}$ is called the received vector and is used at the receiver to determine which signal was transmitted. This can be done by mapping $\boldsymbol{r}$ in the constellation diagram and determining which constellation point $\boldsymbol{s}_i$ is closest to $\boldsymbol{r}$.The closest constellation point is then taken to correspond to the signal that was transmitted and hence the message that was transmitted.

2.5.3.4 Classification (Detection) of symbols

In the previous section, it is mentioned that the received vector r is used at the receiver to determine the signal that was transmitted. This can be done by mapping $\boldsymbol{r}$ in the constellation diagram and determining which constellation point $\boldsymbol{s}_k$ is closest to $\boldsymbol{r}$.The closest constellation point is then taken to correspond to the signal that was transmitted and hence the message that was transmitted. A received vector $\boldsymbol{r}$ corresponds to message $m_i, i = 1,2, \dots, M$ if

$$\sum_{j=1}^{N}(r_j - s_{kj})^2 \quad \textit{is minimum for } k = i \tag{2.61}$$

where r_j's,$i = 1, \dots, N$ are the components of the received vector $\boldsymbol{r}$ and s_{kj}'s ,$j = 1, \dots, N$ are the components of the signal constellation point $\boldsymbol{s}_k, k = 1, \dots, M.M$ is the number of signals in the set of all possible unique signals that can be transmitted. (2.62) can be written as

$$\sum_{j=1}^{N}(r_j - s_{kj})^2 = \underbrace{\sum_{j=1}^{N} r_j^2}_{\text{Independent of k, need not be computed}} - 2\underbrace{\sum_{j=1}^{N} r_j s_{kj}}_{\text{Inner product of } \boldsymbol{r} \text{ with } \boldsymbol{s}_k \text{ need to be computed}} + \underbrace{\sum_{j=1}^{N} s_{kj}^2}_{E_k, energy\ of\ k^{th} signal. For\ equal \text{ energy pulses this need not be computed. For QAM it will be computed}} \tag{2.62}$$

Thus an equivalent form of the optimum decision rule (2.61) is to choose $\hat{m} = m_i$ if

$$\sum_{j=1}^{N} r_j s_{kj} - \frac{1}{2} E_k \ \textit{is maximum for } k = i \tag{2.63}$$

A correlation receiver implements the optimum decision rule of (2.63) by first finding $\boldsymbol{r}$ using the procedure described in section 1.3 and then computing the metric of (2.63) and taking a decision in the receiver. The corresponding receiver structure is illustrated in Figure 2.25 below

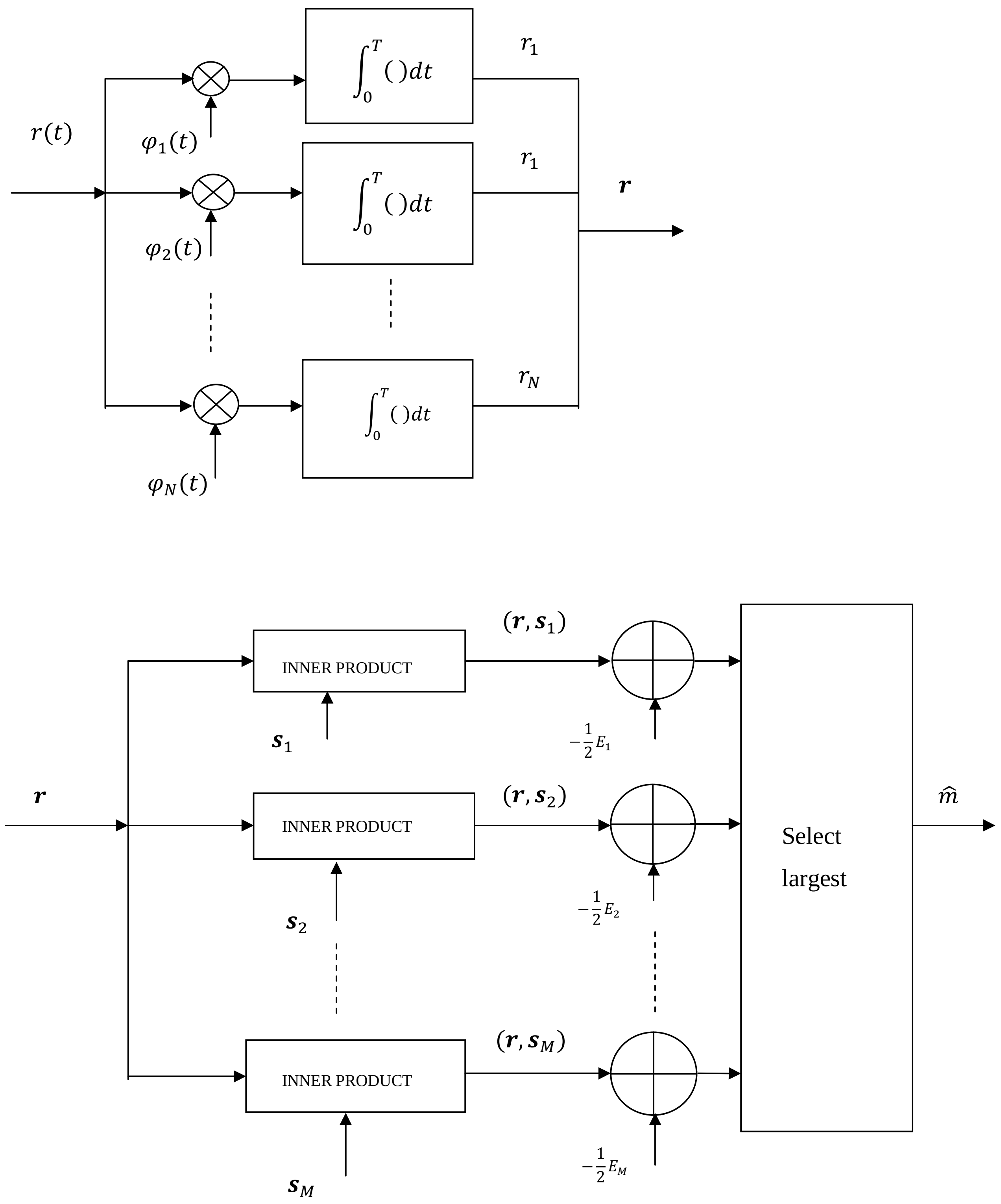

Figure 2-26: Receiver structure

CHAPTER 3

METHODOLOGY

3.1 Simulations

3.1.1 Generation of training data

Un-modulated 16-QAM signals whose gray coded constellation is shown below are generated.

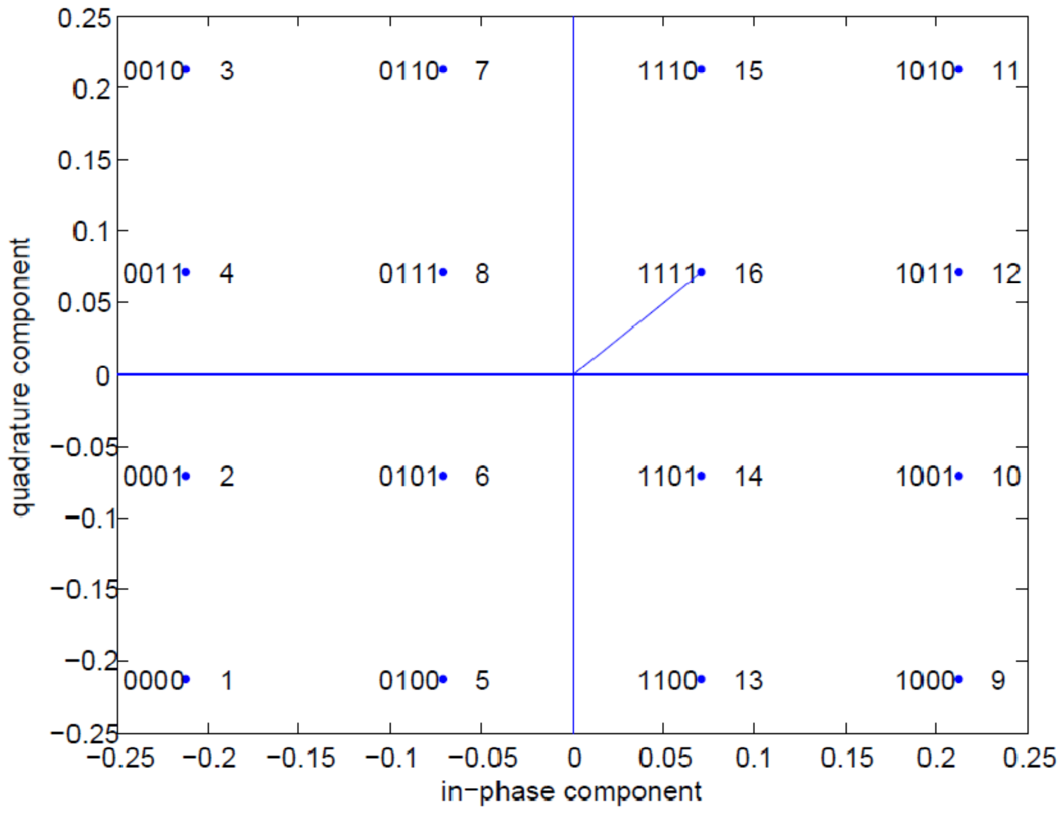

Figure 3-1: 16-QAM constellation

The 16 signal constellation points have been shown with their gray codes to the left and the decimal equivalent to the right. Each constellation point will henceforth be referred to by their decimal equivalent. In clustering terminology we will refer to each of the constellation points as a class, so that there are 16 classes for the drawn constellation and each class is referred to by its decimal equivalent. Note that the energy of a specified constellation point is given by the length of the vector originating from the origin (point (0, 0)) to the specified point. For example, the energy of constellation point 16 is given by the length of the drawn vector. The average constellation energy is given by the sum of the energy for all the constellation points divided by the number of constellation points.

In the second step, a value is randomly generated on the discrete uniform distribution in the range 1-16, and then the constellation point with the decimal equivalent to this value is sent over the symbol interval. For the experiments in this paper, about 1000 such symbols are sent over a wireless channel.

The wireless channel used is a 3G wideband CDMA channel [37] for a vehicle travelling at 120 km\hr. This channel introduces severe inter-symbol interference. Below (Table 3.1) is a table of relative powers in dB of the rays of this channel versus the delay in nanoseconds.

Table 3-1: Specifications of a 3G wideband CDMA channel

Delay(ns)	*Power(dB)*
0	*0.0*
244	*-2.4*
488	*-6.5*
732	*-9.4*
936	*-12.7*
1220	*-13.3*
1708	*-15.4*
1953	*-25.4*

The channel output for the number of transmitted symbols was then used to train a SOM. For the complex channel output corresponding to each sent symbol, a two component vector is formed where the first component is the real part of the channel output and the second component is the imaginary part. A collection of all such vectors for the channel output forms the training samples. Figure 3.1 below is a plot of the in-phase versus the quadrature component for all channel output. As seen from the plot, the dynamic range of the two components (inphase and quadrature component) is equal and thus no normalization of data before training the SOM is necessary.

3.1.2 Training the SOM

For training the SOM, the SOM_PAK program package [18] was used, SOM_PAK contains all programs necessary for the correct application of the self organizing map algorithm in the visualization of complex experimental data.

A map was created of 160 units arranged in a rectangular topology with 16 units in the x-direction and 10 units in the y-direction. Random initialization of the map was used. The training of SOM was carried out in two phases. In the first phase the parameter used were: 30,000 steps, initial learning rate parameter was 0.05 and initial radius of the training area of SOM was set to 8 units. In the second phase the parameters used were: 70,000 steps, initial learning rate parameter is 0.02 and the initial radius of the training area of SOM is 2 units.1000 maps were created using the parameters and the best one was picked as the one which gives the lowest average quantization error. The quantization error is the Euclidian distance between a reference vector of the trained map and the input vector which most resembles it. The training samples are used for the purpose of computing the average quantization error. The final map picked had a quantization error of 0.017460.

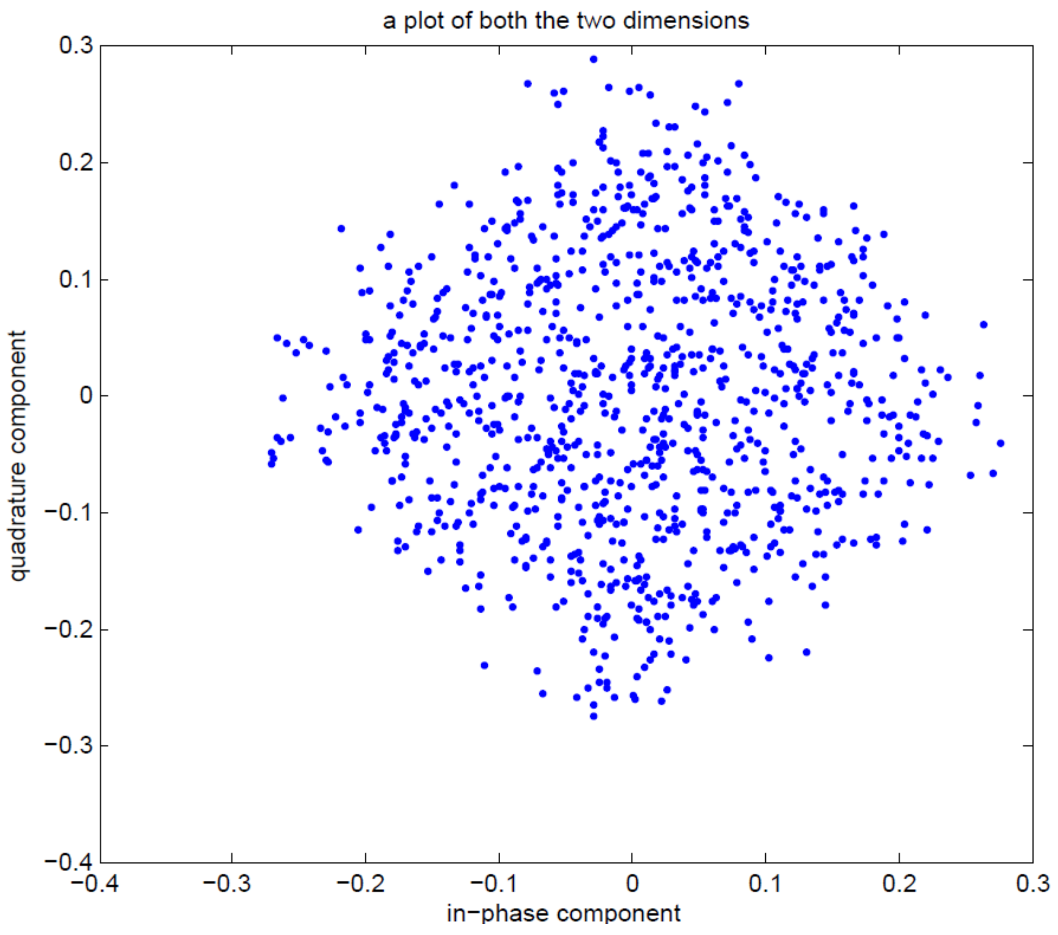

Figure 3-2: A plot of the training data

The trained map was used for the analysis and visualization of the channel output with the aim of discovering any properties which may be used to mitigate inter-symbol interference.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Analysis and visualization of channel output

Self-organizing maps preserve the topological relationships in the input data, as such they can be used to make observations about the structure of the input data which in our case is an interfering channel output. Self-organizing maps also can be used to discover categories of data in the training data and consequently to classify input data. To classify the wireless channel output, training data was presented again to the fully trained map, an input vector is picked from the training data and presented to the trained map, then the winning node is computed as the node whose vector is closest to the input vector. This procedure is repeated for all training data. Then each node of the trained map is given the label of the class to which majority of the symbols for which it wins belong. Figure 4.1 below is a map showing the 2-dimensional locations of the final map vectors and the numerical labels of each node.

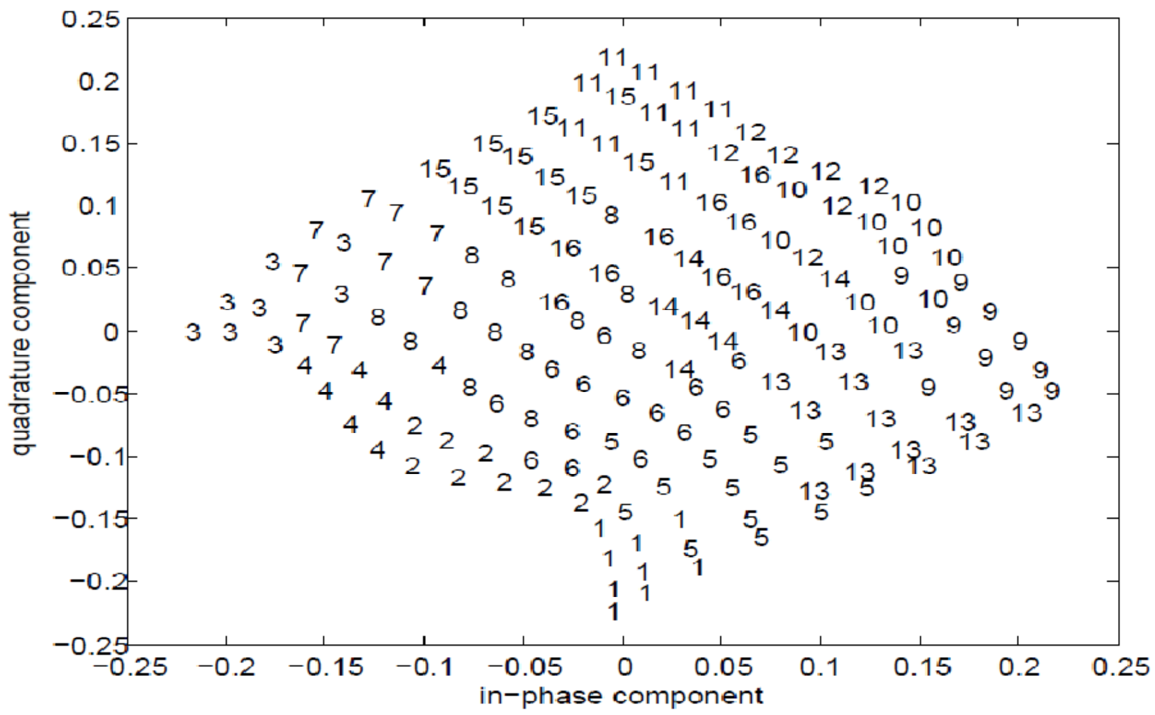

Figure 4-1: Structure of channel output

Comparing Figure 4.1 to the 16-QAM constellation map shown reproduced in Figure 4.2 below for convenience, it is seen that the channel output is a rotated version of channel input. The topological relationships in channel input are roughly maintained.

Next, the trained map is used to classify the channel input, and the resultant classes are compared with the known classes of channel input, then misclassification of symbols is noted. Then an error analysis is done on the misclassified symbols to determine the classes to which a symbol belonging to a given constellation point is likely to be misclassified to.

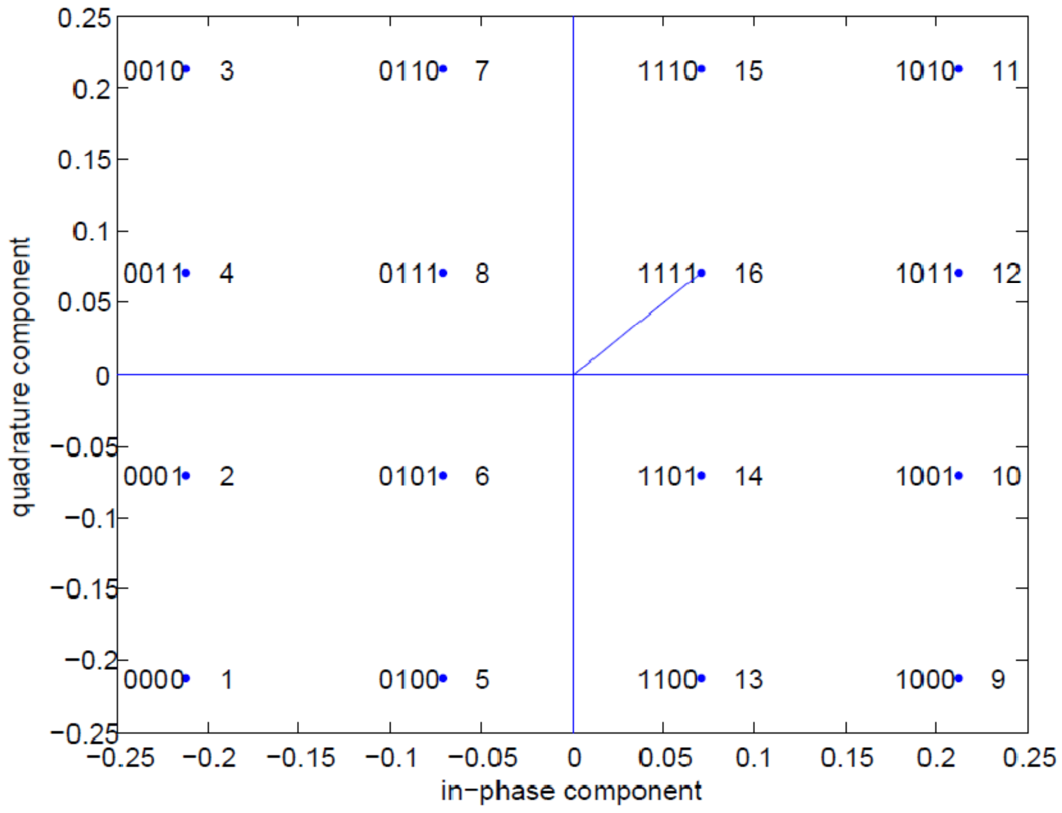

Figure 4-2: 16-QAM constellation

The results are shown in table below:

Table 4-1: Results of symbol misclassification analysis

class	*likely to be misclassified to*
1	*2,5*
2	*1,4,6*
3	*4,7*
4	*3,8,2*
5	*1,6,13*
6	*5,2,14,8*
7	*3,8,15*
8	*4,6,7,16*
9	*10,13*
10	*9,12,14*
11	*12,15*
12	*10,11,16*
13	*5,9,14*
14	*6,10,13,16*
15	*7,11,16*
16	*8,12,14,15*

From Table 4.1 it is seen that symbols are likely to be misclassified to those symbols whose gray codes differ from its own by 1 bit. In other words, the gray coding technique has been verified.

4.2 Component Maps

The component maps for the first and second component of the SOM final weights is illustrated in Figure 4.3 and Figure 4.4 respectively. Also shown is the label of each node corresponding to the majority of the classes whose representation it wins. The lighter regions indicate areas where classification of the present classes is strongly dependent on a component of the reference vectors while the dark regions indicate the areas where classification of the classes are weakly dependent on a component of the reference vectors

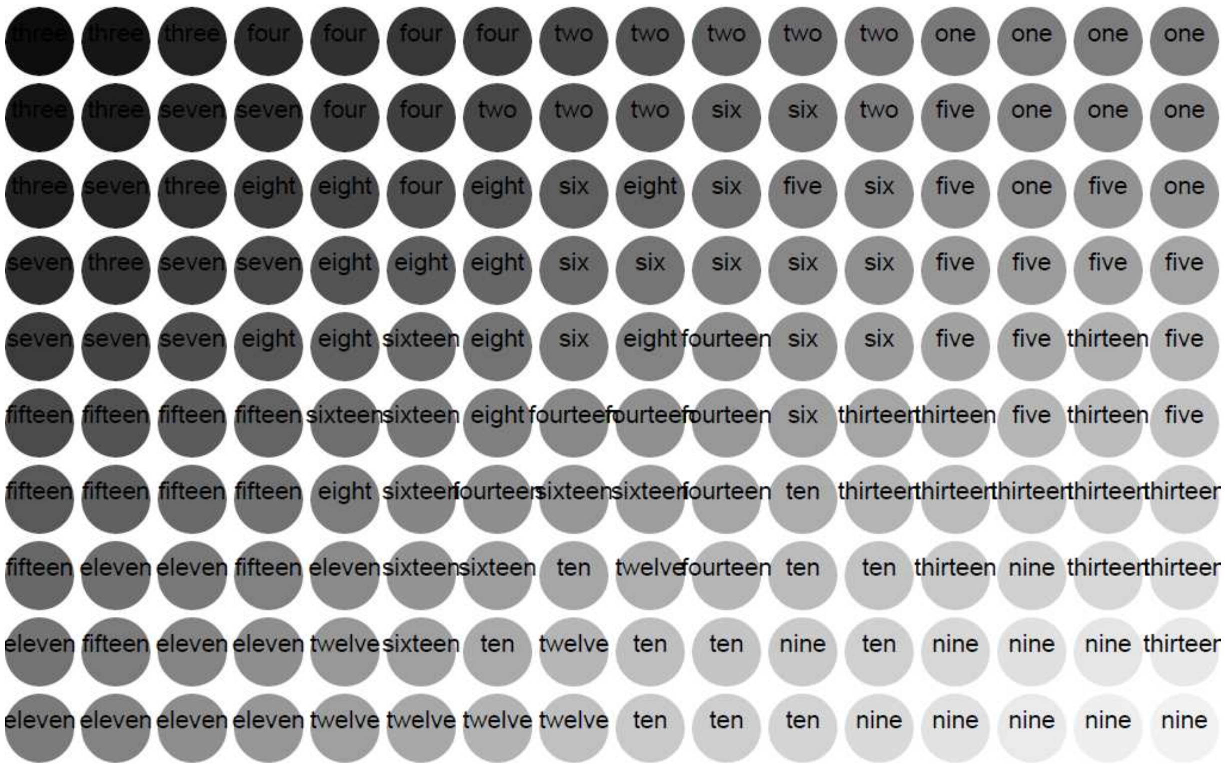

Figure 4-3: Component map for the first component

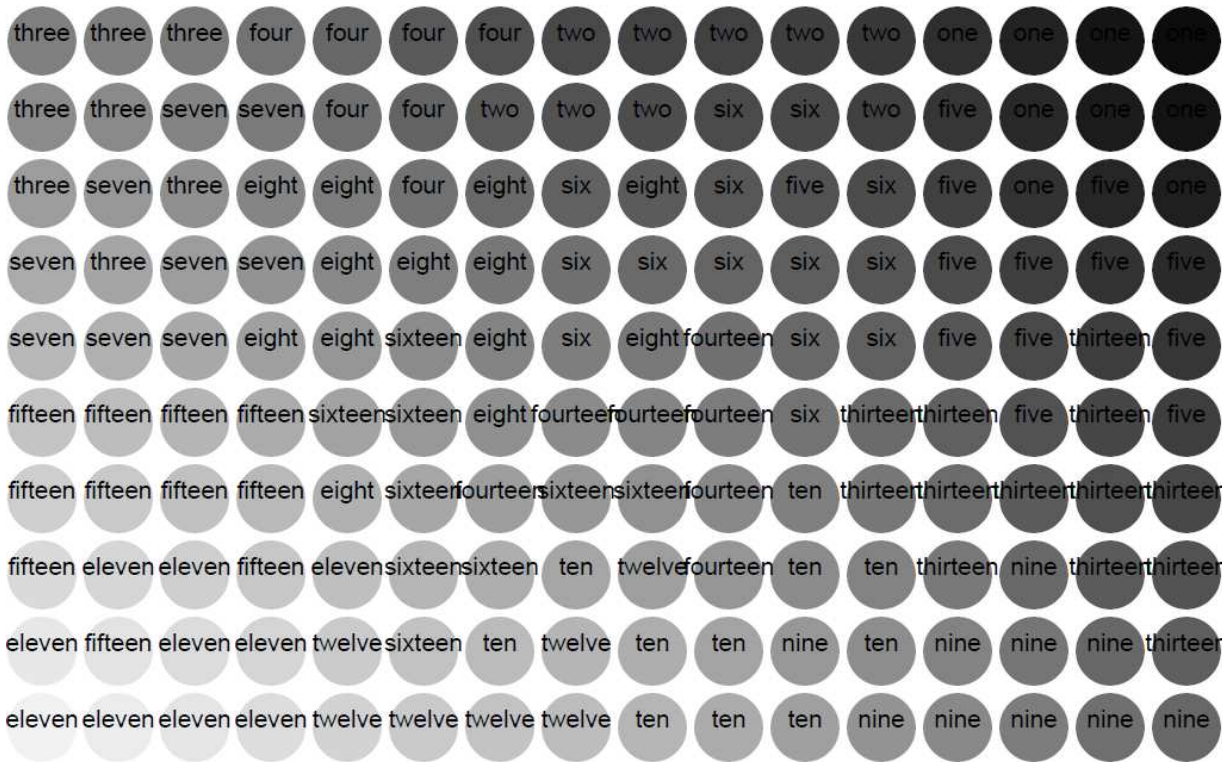

Figure 4-4: Component map for the second component

From Figure 4.3, it is seen that the classification of classes 9, 10 and 13 is strongly dependent on the first component of map weight vectors. Note that the first component of the map weight vectors correspond to the in-phase component of channel output which in turn correspond to the in-phase component of channel input. Similarly, the second components of map weight vectors correspond to the quadrature component of channel input. In the same Figure 4.3 it is also seen that classes 3, 4 and 7 are weakly dependent on the real part of the channel output. Classification of the remaining classes is here taken to be moderately dependent on the first component. From Figure 4.4, we see that classification of classes 1, 2, and 5 are least dependent on the second component of map weight vectors while classes 11, 12 and 15 are most strongly dependent on the second component. The rest of the classes are moderately dependent on the second component of the map weight vectors.

In summary, it is noted that the classification of various classes of channel output is influenced to varying extent by the value of symbol in-phase and quadrature component respectively. Simulations show that if the magnitude of the in-phase component of all the classes of the input 16-QAM constellation is increased, then output symbols corresponding to input classes 9,10, and 13 have the most energy increase, followed by the channel output corresponding to the classes that are moderately responsive to the in-phase component. Output symbols corresponding to classes that are least influenced by the in-phase component showed the least energy increase. The Table 4.2 below shows the average energy increase per symbol for the 3 categories of channel output when the magnitude of the in-phase component of all input symbols is increased by 0.0025.

Table 4-2: Average increase in energy per symbol for the symbols whose misclassification is weakly, moderately and strongly influenced by the value of the in-phase component

Category	*Energy increase per symbol*
Weak	*-0.0130*
Medium	*0.0071*
Strong	*0.0166*

From the above discussion, it is seen that classes which are strongly influenced by a component of the map weight vectors have the most desirable response to a change in that component in the input symbols whereas classes which are least influenced by a component of the map weight vectors have the least desirable response to a change in that component in the input symbols.

In the next section we explore how this result can be used in the design of a constellation which minimizes the symbol classification error while reducing the average constellation energy.

4.3 Design of a 16-QAM constellation

In the previous section a component map for the first and second component of map weight vectors reveals that the classification of different classes of channel output is influenced to varying levels by the value of their in-phase and quadrature components. Starting with 16-QAM symbols and transmitting them over a channel that introduces inter-symbol interference, it was shown for instance that the classification of symbols belonging to constellation points 9, 10, and 13 is strongly influenced by the value of their in-phase component while those belonging to constellation points 3, 4, and 7 are weakly influenced by the value of their in-phase component. The rest of the constellation points are considered to be moderately affected by the value of their in-phase component. Further, we see that classification of classes 1, 2, and 5 are least dependent on the value of their quadrature component while the classification of classes 11, 12 and 15 are most strongly dependent on the value of their quadrature component. The rest of the classes are moderately dependent on the value of their quadrature component. A possible application of this result is in the design of a digital modulation constellation given the knowledge of the type of channel likely to be encountered in a digital communication system. For a digital modulation constellation, a reduction of the average constellation energy results in an increase in the symbol error rate at the receiver. A simple way to explain this is that when the average constellation energy is reduced, then it becomes more difficult to distinguish between the different amplitude levels and phases of the transmitted symbols. The question we ask and answer is: starting from a given average constellation energy, what is the technique by which the average constellation energy can be reduced while at the same time minimizing the resultant increase in symbol classification errors at the receiver?

Beginning with a standard rectangular 16-QAM constellation, we show how to construct a constellation with a smaller average energy while minimizing the resulting increase in receiver symbol error rate. If it is known for instance that the classification of symbols belonging to certain constellation points is weakly influenced by the value of their in-phase component, then an attempt to reduce energy of transmitted signals is best done by reducing the value of the in-phase component of symbols belonging to those constellation points. The resulting symbol error rate is then compared to that obtained by a wholesome reduction of the energy of all constellation points such that the average constellation energy is the same as that of selective reduction of the in-phase components of specific constellation points described above.

For the simulations, we start with a gray coded 16-QAM shown in Figure 4.2. Besides each constellation point is its 4-digit binary gray code to the left and its decimal equivalent to the right. A constellation point shall be referred to by its decimal equivalent. The average energy of the constellation is 0.2118.In the second step, a value is randomly generated on the discrete uniform distribution in the range 1:16, then the constellation point with the decimal equivalent to this value is sent over the symbol interval. 800 such symbols are sent. The symbols are sent over a 3G wideband CDMA channel [2] for a vehicle travelling at 120 km\hr. This channel introduces severe inter-symbol interference. In the receiver an LMS equalizer is used to remove the inter-symbol interference. The error incurred after equalization is about 100 symbols. To achieve the aims outlined in the previous section, the following three experiments were performed.

Experiment 1

The in-phase components of constellation points 3, 4 and 7 were reduced by 0.0025, these are the constellation points whose classification is least dependent on the in-phase component. The quadrature component of constellation points 1, 2 and 5 were also reduced by the same amount. These are the constellation points whose classification is least affected by the value of their quadrature components. The in-phase and quadrature components of constellation points moderately dependent on in-phase and quadrature component were also reduced so as to maximize the total constellation energy reduction while not drastically increasing the symbol classification errors. The new average constellation energy is then calculated. Then 800 symbols are sent over the channel and the number of misclassified symbols at the receiver is noted. In order to generalize the results, we repeat the procedure for 10,000 randomly generated vectors, each consisting of 800 symbols and the symbol misclassification is counted for all the 10,000 instances. The results are plotted in Figure.4.5.

Experiment 2

For this experiment, all the constellation points are multiplied by a suitable scalar so that the resulting constellation has an average energy equal to that of the constellation used for experiment 1.In other words, here we have a wholesale reduction of the energy of all constellation points so that the average energy of resulting constellation is equal to the case of experiment 1 where only specific components of some constellation points had their magnitude reduced depending on how they are influenced by the value of their in-phase and quadrature components.800 symbols are then sent over the channel and the number of misclassified symbols is counted. As in experiment 1, we repeat the procedure for 10,000 vectors, where each vector consists of 800 symbols. The result is plotted in Figure.4.5.

Experiment 3

For this experiment, symbols from the original constellation shown in Figure.1 are transmitted and the resulting symbol misclassification is noted. The result is plotted in Figure.4.5.

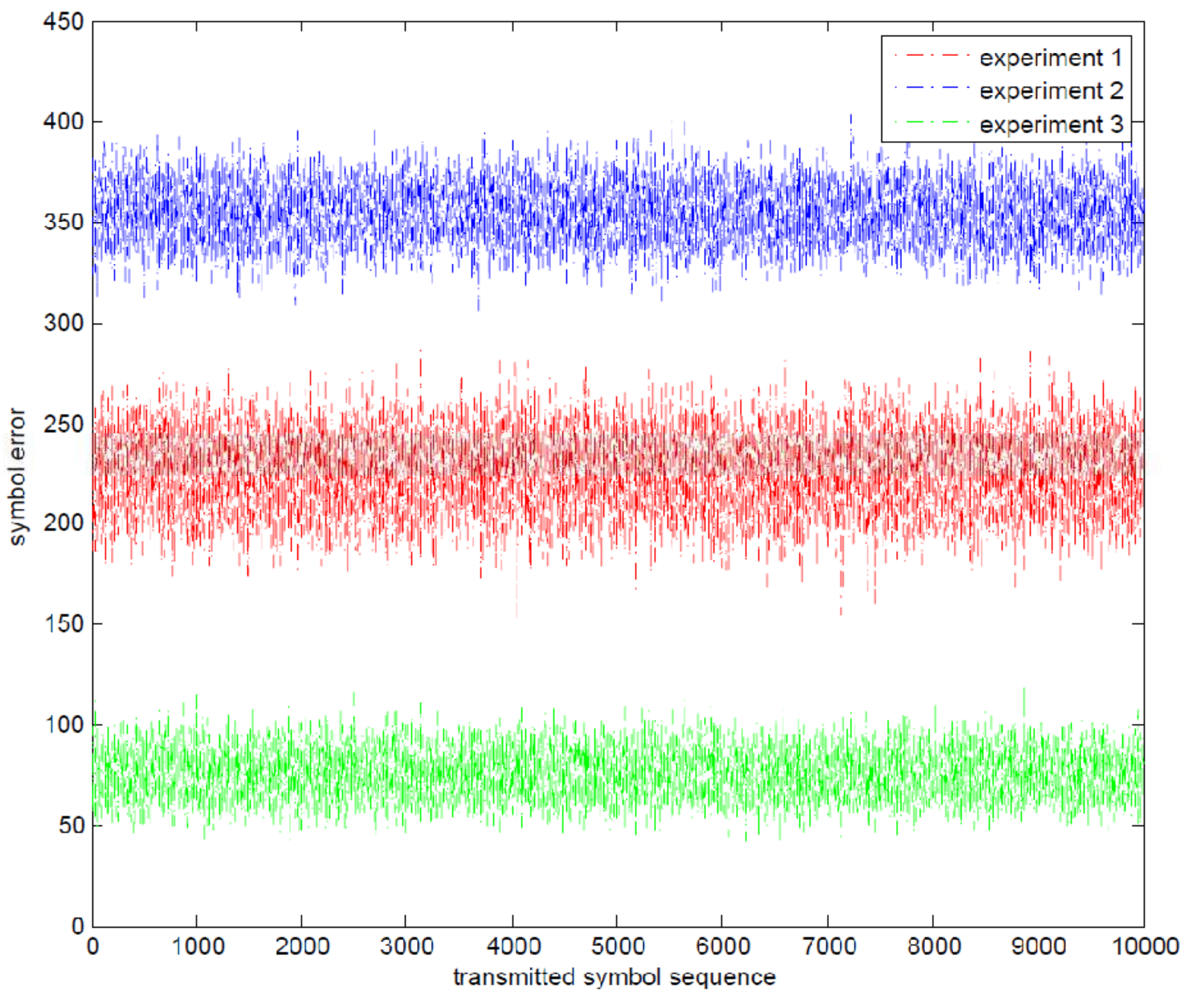

Figure 4-5: Symbol classification errors for experiments 1, 2 and 3

As shown in Fig.4.5, more symbol errors are incurred when the constellation energy is reduced homogeneously as in experiment 2 than when only specified constellation points have an energy reduction as in experiment 1. This is because there is a smaller increase in number of errors when constellation points whose classification is weakly dependent on a component have that component reduced than when all constellation points have an energy reduction including the ones whose classification is strongly dependent on the values of their in-phase and quadrature components. As expected, the least error is incurred when using symbols from the original constellation point without any energy reduction.

In section 1.1, it was shown through computation that symbols belonging to constellation points whose classification is weakly dependent on a component have the least desirable response when that component is altered in the input symbols and symbols belonging to constellation points whose classification is strongly dependent on a component have the most desirable response when that component is altered in the input symbols. For example, symbols whose classification is weakly dependent on the in-phase component have the least energy increase when the magnitude of the in-phase component of corresponding constellation points is increased. The same is true for when the magnitude of the in-phase component is decreased. An equivalent way to demonstrate this experimentally is to show that when the magnitude of the in-phase components for symbols whose classification is least dependent on the in-phase component is reduced, then we have the least increase in symbol classification errors. That is the essence of the next experiment

Experiment 4

In this experiment, the aim is to compare the increase in error per unit reduction of constellation energy when a reduction of the in-phase and quadrature components is done in the manner of experiment 1 and when in addition, the in-phase and quadrature components of the constellation points most strongly influenced by the values of their in-phase and quadrature components is reduced. The result is plotted in Figure 4.6.

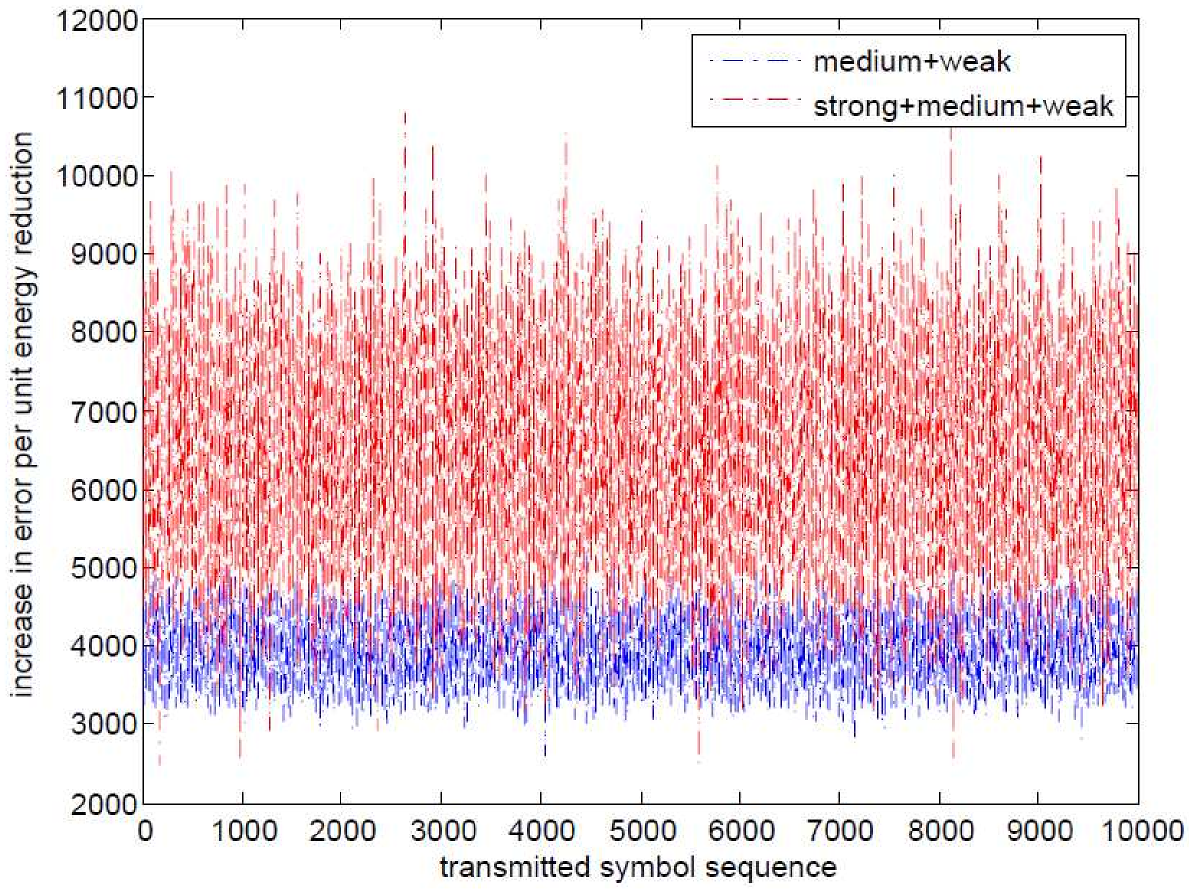

Figure 4-6: Comparison of increase in error per unit energy reduction for experiment 4

In Fig.4.6, it is evident that when constellation points whose classification is strongly influenced by the in-phase or quadrature component have the magnitude of that component reduced, then there is a drastic increase in number of symbol errors than when only components that are weakly or moderately influenced by in-phase or quadrature components have their magnitudes decreased.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

Visualization and analysis of an interfering wireless channel output using self-organizing maps was done. Using 16-QAM symbols as the input to the wireless channel, then visualization of the output using a trained SOM revealed that the output is a rotated version of the input constellation. The SOM was also used to classify the channel output and an error analysis was then done on the misclassified symbols to determine which constellation points that an output symbol belonging to a given constellation point is likely to be misclassified to. It was found that a symbol is likely to be misclassified to those symbols belonging to constellation points whose gray codes differ from that of the symbol under consideration by one bit. In other words, the validity of the gray coding for the wireless channel used is verified. A component map analysis of the SOM map reveals how the classification of various classes of channel outputs is influenced by the values of their in-phase and quadrature components. It is found that the classification of symbols belonging to constellation points 9, 10, and 13 is strongly influenced by the value of their in-phase component while those belonging to constellation points 3, 4, and 7 are weakly influenced by the value of their in-phase component. The rest of the constellation points are considered to be moderately affected by the value of their in-phase component. Further, we see that classification of classes 1, 2, and 5 are least dependent on the value of their quadrature component while the classification of classes 11, 12 and 15 are most strongly dependent on the value of their quadrature component. The rest of the classes are moderately dependent on the value of their quadrature component. Using this result, a technique was demonstrated on how to reduce the average constellation energy for an interfering channel while minimizing the resulting increase in symbol classification errors. Thus a method is found to mitigate inter-symbol interference.

5.2 Contribution of this work

The study has extended the application of SOM into the visualization and analysis of wireless communication channels.The findings documented in this report and summarized in the preceding section shows new insights into the structure of a wireless channel output as well as establishing how the classification of symbols is affected by the values of two parameters namely their in-phase and quadrature components. Also presented is a technique demonstrating how this knowledge can be used to design an energy efficient digital modulation constellation.

5.3 Recommendations for further research work

Visualization and analysis was done for an output of a particular wireless channel. It would be interesting to see how the results turn out for a practical situation in which the channel varies over time. Also as has been demonstrated, self-organizing maps can be used to gain valuable insight about the wireless channel output and so the tool is recommended as a way of gaining additional insights on the output of a wireless channel, which may lead to discovery of properties which could be utilized to mitigate against channel imperfections.

REFERENCES

[1] Theodore S. Rappaport, Wireless communication systems, principles and practice, 2nd Edition

[2] T. Kohonen, E. Oja, O. Simula, A. Visa and J. Kangas, “Engineering applications of the self-organizing map,” in Proc. IEEE, Vol. 84, No. 10, Oct. 1996, pp. 1358.

[3] F.O. Karray and C. de Silva, Soft Computing and Intelligent Systems Design, Harlow, England: Pearson, 2004.

[4] P. Werbos, “Beyond regression: new tools for prediction and analysis in the behavioral sciences,” Ph.D. Dissertation, Harvard University, 1974.

[5] B. Widrow and M. Jr. Hoff, “Adaptive switching circuits,” IRE WESCON Convention Record, 1960, pp. 96-104.

[6] S. Haykin, Neural Networks, A Comprehensive Foundation, Englewood Cliffs, NJ: MacMillan Publishing, 1994.

[7] G. F. Page, J. B. Gomm, D. Williams, Application of Neural Networks to Modeling and Control, Chapman and Hall, 1993

[8] Dayhoff, E. Judith, Neural Network Architectures: An Introduction, Australia: Van Nostrand Reinhold, 1990.

[9] A. Ultsch, D. Korus, “Integration of neural networks with knowledge-based systems,” in Proc. IEEE Int. Conf. on NN, vol.4, 1995, pp.1828-1833.

[10] Xiao-Feng Gu, Lin Liu, Jian-Ping Li, Yuan-Yuan Huang, Jie Lin, “Data classification based on artificial neural networks,” in Int. IEEE Conf. on Computing and Intelligence Analysis, 2008, pp.223-226.

[11] Chang-Shian Chen, Chao-Chung Yang, Chin-Hui Liu, “The interval estimation of parameters for back-propagation network to flood discharge forecasting,” in Int IEEE Conf. on NN, 2006, pp.3729-3735.

[12] Back propagation network:

http://www.itee.uq.edu.au/~cogs2010/cmc/chapters/BackProp/index2.html#Background

[13] T. Kohonen, “The Self-Organizing Map,” in Proc. IEEE, Vol. 78, Sept. 1990, pp. 1466–1469.

[14] S. M. Guthikonda. (2005, September), Kohonen self-organizing maps [Online] Available:http://www.shy.am/wp-content/uploads/2009/01/kohonen-self-organizing-maps-shyam-guthikonda.pdf

[15] SOM implementation in SOM Toolbox: www.cis.hut.fi/projects/somtoolbox/documentation/somalg.shtml

[16] T. Kohonen, “Things you haven’t heard about the self-organizing map,” in Proc. ICNN’93, Int. Conf. on NN, 1993, pp. 1147-1156.

[17] T. Kohonen, J. Hynninen, J. Kangas and J. Laaksonen, (1995, 7th April), SOM_PAK the self-organizing map program package [Online]. Available: http://www.isegi.unl.pt/ensino/docentes/fbacao/som_pak.pdf.

[18] T. Samad and S. A. Harp, “Self-organization with partial data,” Network: Computation in Neural Systems, 1992

[19] S. Kaski, T. Kohonen, Structures of Welfare and Poverty in the World Discovered by the Self Organizing Map, Report A24, Helsinki University of Technology, Laboratory of Computer and Information Technology, 1995.

[20] T. Kohonen, E. Oja, O. Simula, A. Visa and J. Kangas, Finnish Pat. 83,577, U.K. Pat. 2,233,865.

[21] T. Kohonen et al., “Combining linear equalization and self organizing adaptation in dynamic discrete signal detection,” in Proc. IJCNN-90-San Diego Int. joint Conf. on NN, 1990, vol. 1, pp. 223-228.

[22] G. Burel and J. Y. Catros, “Image compression using topological maps and MLP,” in Proc ICNN’93 Int. Conf. on NN, Piscataway, NJ, 1993, vol. 2, pp. 727-731.

[23] K. Raivio, J. Henriksson and O. Simula, “Neural detection of QAM modulation in the presence of interference,” in Proc. ICNN’95 Int. Conf. on NN, 1995.

[24] A. Habibi, “Neural networks in bandwidth compression,” in Proc. SPIE-The Int. Soc. for Optical Engineering, SPIE: Bellingham, WA, 1991, vol. 1567, pp. 334-340.

[25] M. Peng, C. L. Nikias, and J. G. Proakis, “Adaptive equalization for PAM and QAM signals with neural networks,” Conf. Record of the 25th Asilomar Conf. on Signals, Syst. and Computers, vol. 1, New York: IEEE Comput. Soc., 1991, pp. 496-500.

[26] S. Carter, R. J. Frank, and D. S. W. Tansley, “Clone detection in telecommunications software systems: A neural net approach,” in Proc. Int. Workshop on Applicat. of NN to Telecommun. Hillsdale, NJ: L Erlbaum, 1993, pp. 273-287.

[27] P. Barson et al., “Dynamic competitive learning applied to the clone detection problem,” in Proc. Int. Workshop on Applications of Neural Networks to Telecommun., 2nd ed. Hillsdale, NJ: L Erlbaum, 1995, pp. 234-241.

[28] S. Field, N. Davey, and R. Frank, “A complexity analysis of telecommunications software using neural networks,” in Proc. 2nd Int. Workshop on Applicat. of Neural Networks to Telecommun., Hillsdale, NJ: L Erlbaum, 1995, pp. 226-233.

[29] N. Ansari and Y. Chen, “A neural network model to configure maps for a satellite communication network,” in Proc. GLOBECOM’90, IEEE Global Telecommun. Conf. and Exhibit..Communications: Connecting the Future, Piscataway, NJ, 1990, vol. 2, pp. 1042-1046.

[30] N. Ansari and D. Liu, “The performance evaluation of a new neural network based traffic management scheme for a satellite communication network,” in Proc. GLOBECOM’91, IEEE Global Telecommun. Conf. Countdown to the New

Millenium: Personal Communications Services (PCS), Piscataway, NJ, 1991, vol. 1, pp. 110-114.

[31] H. Tang and O. Simula, "The optimal utilization of multi-service SCP," in Proc. IFIP-TCG Working Conf. on Intell. Networks, Copenhagen, Denmark, 1995, pp. 157-167.

[32] T. Kohonen, E. Oja, O. Simula, A. Visa and J. Kangas, "Neural adaptation for optical traffic shaping in telephone systems," in Proc. ICNN'95, Int. Conf. on NN, Perth, Australia, 1995.

[33] K. Banovic, "Blind adaptive equalization for QAM signals: new algorithms and FPGA implementation," M.S. thesis, Dept. Elect. and Comput. Eng., Univ. of Windsor, Windsor, Canada, 2006.

[34] B. Widrow and S. D. Sterns. Adaptive Signal Processing. Prentice Hall, New York, 1985.

[35] P. S. R. Diniz. Adaptive Filtering. Kluwar Academic Publishers, Norwell, Massachusetts, 2002.

[36] A. Goldsmith. Wireless Communications. Cambridge University Press, 2005.

[37] W. H. Tranter et al., "Modeling and simulation of waveform channels," in Principles of Communications Systems Simulations, New Jersey: Prentice Hall, 2004, ch. 14, sec. 8, pp. 571.

PUBLICATIONS FROM THIS WORK

[1] E. Olukohe, P. K. Kihato and D. O. Konditi, "Insights gained by visualization of a wireless channel output using Self-organizing maps," Journal of Sustainable Research in Engineering, Unpublished.