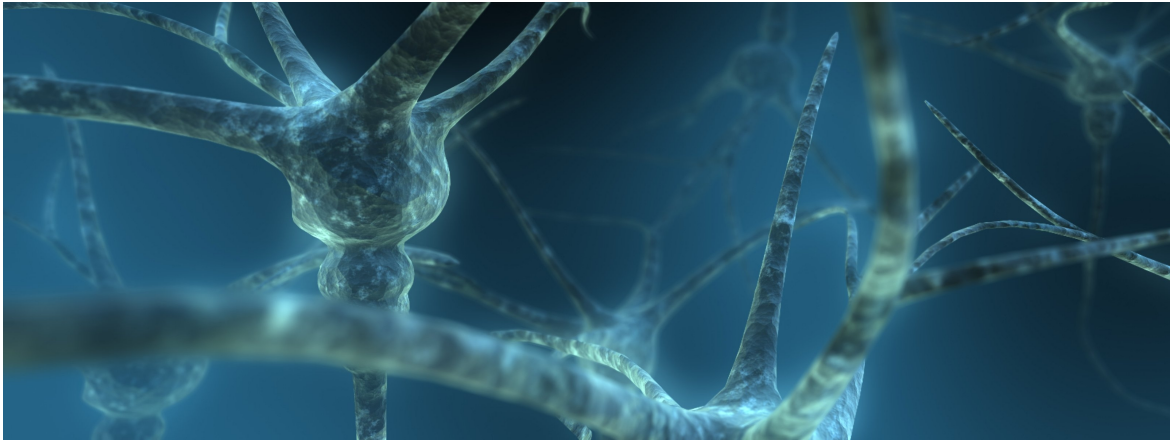


People's Democratic Republic of Algeria  
Ministry of Higher Education and Scientific Research  
Mentouri University of Constantine, Engineering Faculty, Computer  
Science Department



---

# INCREMENTAL LEARNING USING PROBABILISTIC NEURAL NETWORKS AND ITS APPLICATION TO FACE RECOGNITION

---

Presented by **ELHAM AL.KHOLANI**

Option : ARTIFICIAL INTELLIGENCE AND SOFTWARE ENGINEERING.

A thesis submitted in partial fulfillment of the requirements for the  
Degree of Master in Computer Science.

Supervisor: PR.MOHAMED BATOUCHE

Chairman: DR.SAIDOUNI DJAMEL EDDINE\*

Reporter: DR.A.CHAOUI\*

Member: DR.KHOLLADI MOHAMED KHIREDDINE\*

Member: DR.CHIKHI SALIM\*

\* Associate Professor of Computer Sciences, Mentouri University.

# Dedication

I would like to thank my family for the support they provided me through my entire life and in particular, I must acknowledge my precious parents, my uncle and aunt , my husband and daughter And my sister Amira and cousin Ali for supporting and encouraging me to pursue this degree. Without their support and love, I would not have finished this thesis.

# Acknowledgements

FIRST and foremost I would like to thank the lord, Allah, for enabling me to complete this project. This thesis would not have been possible without the support of many people, I would like to express my gratitude to my supervisor, Dr. Mohamed BATOUCHE, whose expertise, understanding, guidance and patience, added considerably to my graduate experience.

I would also like to acknowledge the advice and guidance of D.Mohamed. A very special thanks goes all who helped and supported me ,especially Dr. S. CHIKHI , Dr. M. BOUFAIDA and all the members of the Faculty of Computer Science for all of their computer and technical assistance throughout my graduate program.

I would also like to thank Dr. D. SAIDOUNI who gave me the honour of presiding the jury and special thanks also to the members of the jury Dr.A.CHAOUI , Dr. S. CHIKHI and Dr.M.KHOLLADI.

Appreciation also goes out to Haoua CHERIBI , Rokia BENDANA and all my colleagues for all of their computer and technical assistance throughout my graduate program. I would also like to thank Hana Al.saidi and all my friends who never lost faith in me and presented their endless support and encouragement.

# Contents

<b>Introduction</b>	<b>4</b>
<b>1 Learning Theory</b>	<b>1</b>
1.1 Introduction	1
1.2 Human and machine learning Definitions	1
1.3 The motivation of research in Machine Learning	2
1.4 Importance of Machine Learning	3
1.5 Wellsprings of machine learning	3
1.6 Overview Of Learning Theory	4
1.6.1 Types of Learning	4
1.6.1.1 Speed-up learning	5
1.6.1.2 Learning by taking advice	5
1.6.1.3 Learning from examples	5
1.6.1.4 Clustering	5
1.6.1.5 Learning by analogy	5
1.6.1.6 Discovery	5
1.6.2 Major Paradigms of Machine Learning	5
1.6.2.1 Supervised learning	5
1.6.2.2 Unsupervised learning	6
1.6.2.3 Reinforcement learning	6
1.6.2.4 Batch vs. online learning	6
1.6.3 Three main learning problems	6
1.6.3.1 Classification	7
1.6.3.2 Regression	8
1.6.3.3 Novelty Detection	9
1.6.4 A Framework for learning systems	10
1.6.5 Designing a Learning System	10
1.6.6 Some Approaches to Machine Learning	11
1.6.6.1 Decision Tree Learning	11
1.6.6.2 Artificial Neural Networks (more details will be discussed in chapter 3)	12
1.6.6.3 Bayesian Learning	12
1.6.6.4 Genetic Algorithms	13
1.6.6.5 Relational Learning	13
1.6.6.6 Modern" learning algorithms	13
1.7 Open Problems in Learning	13
1.8 Application Successes	14
1.9 Conclusion	15

<b>2</b>	<b>Incremental learning</b>	<b>16</b>
2.1	Introduction . . . . .	16
2.2	On-line vs. Off-line Learning . . . . .	17
2.3	Incremental learning vs. non-incremental learning(Batch learning) . . .	17
2.4	Incremental Learning . . . . .	18
2.4.1	Incremental Learning Tasks . . . . .	18
2.4.2	Examples of incremental learning tasks . . . . .	19
2.4.2.1	User modelling/profiling . . . . .	19
2.4.2.2	Robotics . . . . .	19
2.4.2.3	Intelligent agents . . . . .	19
2.4.2.4	Software project estimation . . . . .	19
2.4.3	The main characteristics of an incremental learning task . . . .	20
2.5	Incremental Learning Algorithms . . . . .	20
2.5.1	Purpose and definition . . . . .	20
2.6	Some incremental learning algorithms . . . . .	22
2.6.1	CANDIDATE-ELIMINATION . . . . .	22
2.6.2	COBWEB . . . . .	22
2.6.3	ID5 . . . . .	23
2.6.4	ILA . . . . .	23
2.7	Design Issues . . . . .	23
2.7.1	Ordering Effects . . . . .	23
2.7.2	Learning Curve . . . . .	24
2.7.3	Open-world Assumption . . . . .	24
2.7.4	Incremental learning from examples . . . . .	24
2.7.5	Incremental conceptual clustering . . . . .	28
2.8	Conclusion . . . . .	28
<b>3</b>	<b>Neural Networks</b>	<b>30</b>
3.1	Introduction . . . . .	30
3.2	Biologically, How Human Brain Learns? . . . . .	30
3.3	From Human Neuron to Artificial Neuron . . . . .	31
3.4	A simple Artificial neuron . . . . .	31
3.5	Why we use Neural Networks . . . . .	32
3.5.1	Motivations . . . . .	32
3.5.2	Artificial Neural networks versus conventional computers . . . .	33
3.5.3	The Ability of ANNs . . . . .	33
3.6	Representing 'knowledge' in a Neural Network : . . . . .	34
3.7	ANN Components . . . . .	35
3.7.1	Input Layer . . . . .	35
3.7.2	Hidden Layer (s) . . . . .	36
3.7.3	Output layer . . . . .	36
3.7.4	Connections /Arcs . . . . .	36
3.7.5	Weights . . . . .	36
3.7.6	Summation Function . . . . .	36
3.7.7	Activation (transfer) functions . . . . .	36
3.7.8	Bias, offset, threshold . . . . .	37
3.8	Learning Process . . . . .	37
3.8.1	Connectionist Learning Algorithms . . . . .	37

3.8.2	Learning Rules . . . . .	39
3.8.2.1	Hebb's Rule . . . . .	39
3.8.2.2	Hopfield Law . . . . .	40
3.8.2.3	The Delta Rule . . . . .	40
3.8.2.4	The Gradient Descent Rule . . . . .	40
3.8.2.5	Kohonen's Learning Law . . . . .	41
3.9	Types of connectionist Models . . . . .	41
3.10	ANN Architectures . . . . .	42
3.10.1	Feed-forward networks . . . . .	42
3.10.1.1	Single-layer perceptron . . . . .	42
3.10.1.2	Multi-layer perceptrons . . . . .	43
3.10.1.3	ADALINE . . . . .	45
3.10.1.4	Radial basis function (RBF) . . . . .	45
3.10.2	Recurrent (feedback) networks . . . . .	46
3.10.2.1	Simple recurrent network . . . . .	47
3.10.2.2	Hopfield network . . . . .	47
3.10.2.3	Kohonen self-organizing network . . . . .	48
3.10.3	Stochastic neural networks . . . . .	48
3.10.4	Probabilistic neural networks(PNN) . . . . .	48
3.10.5	Backpropagation Algorithm . . . . .	48
3.10.6	The Gradient Descent Algorithm . . . . .	49
3.11	Overtraining , how can we avoid it ! . . . . .	49
3.12	Strengths and Weaknesses of Neural Network Models . . . . .	50
3.13	Applications . . . . .	51
3.13.1	Clustering . . . . .	51
3.13.2	Classification/Pattern recognition . . . . .	51
3.13.3	Function approximation . . . . .	51
3.13.4	Prediction/Dynamical Systems . . . . .	52
3.14	Conclusion . . . . .	52
<b>4</b>	<b>Incremental Learning Based On Neural Networks</b>	<b>53</b>
4.1	Introduction . . . . .	53
4.2	Assessing The quality of a Neural Network solution . . . . .	53
4.3	Batch vs. Incremental Learning . . . . .	54
4.4	learning algorithm criteria . . . . .	54
4.5	Incremental Neural learning framework . . . . .	55
4.6	Catastrophic Interference . . . . .	55
4.7	Incremental Learning and Neural Network: . . . . .	56
4.8	Start big and remove . . . . .	58
4.9	Start small and add . . . . .	59
4.10	Overview of Incremental Neural Network Algorithms : . . . . .	59
4.10.1	Evolving Neural Networks: . . . . .	61
4.10.2	Resource Allocating Network with Long-Term Memory: . . . . .	61
4.10.3	Evolving connectionist systems . . . . .	61
4.10.4	Sleep and Awake methods . . . . .	62
4.10.5	The ART (adaptive resonance theory) net . . . . .	62
4.10.6	The PNN (probilistic neural network) . . . . .	62
4.10.7	The cascade correlation net . . . . .	62

4.10.8	Incremental backpropagation network . . . . .	62
4.10.9	Learn++ . . . . .	63
4.11	Conclusion . . . . .	63

## **5 An Incremental Learning Model Using probabilistic Neural Network For Face Recogniton 64**

5.1	Facial recognition system . . . . .	64
5.1.1	Introduction . . . . .	64
5.1.2	Notable users and deployments . . . . .	64
5.1.3	Early development . . . . .	65
5.1.4	General Framework . . . . .	66
5.1.5	Variations in Facial Images . . . . .	66
5.1.6	Comparative study . . . . .	67
5.1.7	Incremental Learning vs. Batch Learning in the context of Face recognition: . . . . .	68
5.2	Probabilistic models in Machine Learning : . . . . .	68
5.3	statistical parameters Measurments . . . . .	69
5.3.1	Measures of Location (central tendency) . . . . .	70
5.3.1.1	Mean . . . . .	70
5.3.1.2	Median . . . . .	70
5.3.1.3	Mode . . . . .	70
5.3.1.4	Quartile (Fractiles) . . . . .	71
5.3.2	Measures of Variation . . . . .	72
5.3.2.1	Range . . . . .	73
5.3.2.2	Variance . . . . .	73
5.3.2.3	Standard Deviation . . . . .	73
5.3.2.4	Interquartile Range . . . . .	73
5.3.3	Moments of a Distribution . . . . .	74
5.3.3.1	Skewness . . . . .	74
5.3.3.2	Kurtosis . . . . .	75
5.4	Incrementality notion and statistical parameters . . . . .	75
5.5	Architecture of PNN Network . . . . .	76
5.6	Advanced PNN . . . . .	78
5.7	Incremental APNN . . . . .	79
5.7.1	Representation of Incremental APNN . . . . .	80
5.7.1.1	Input data representation . . . . .	80
5.7.2	Representation of input layer . . . . .	82
5.7.3	Representation of hidden layer . . . . .	83
5.7.3.1	The notion of inctrumentality of statistical parameters in a neourn in hidden layer . . . . .	86
5.7.3.2	Summation and Activation Function (Transfert Function) . . . . .	87
5.7.4	Pattern layer (Summation layer) . . . . .	88
5.7.5	Output layer (Decision Layer ) . . . . .	89
5.7.6	Learning phase (How does it work ! ) . . . . .	89
5.7.7	Demonstration Example For our Network . . . . .	89
5.7.8	Recognition phase . . . . .	90
5.7.9	Learning phase . . . . .	90

5.7.9.1	Level One: Add new image to existening subclass . . .	90
5.7.9.2	Level two: Add new image as a new subclass to existening class . . . . .	91
5.7.9.3	Level Three: Add new image as a new class and a new subclass as well . . . . .	92
5.8	Incremental Learning scheme . . . . .	93
5.9	Feature extraction . . . . .	95
5.9.1	Block extraction . . . . .	95
5.10	Testes . . . . .	97
5.10.1	The database of faces ORL . . . . .	97
5.11	Result . . . . .	101
5.11.1	Rate . . . . .	101
5.11.2	Number of blocks . . . . .	101
5.11.3	Size of NN . . . . .	102
5.11.4	Time of learning (ms) of one image . . . . .	102
5.11.5	Time of recognition (ms) of one image . . . . .	103
5.11.6	Rate with the Size of block . . . . .	103
5.11.7	Time of learning/Time of recognition . . . . .	104
5.11.8	Number of training image . . . . .	104
5.11.9	Rate with Sp . . . . .	105
5.11.10	Number of blocks . . . . .	105
5.12	Conclusion . . . . .	105
	<b>Conclusion</b>	<b>106</b>
	<b>Bibliography</b>	<b>110</b>
	<b>Glossary</b>	<b>111</b>



# List of Tables

3.1 NN vs. conventional computers. . . . .	33
5.1 Approachs comparison . . . . .	101

# List of Figures

1.1	Classification . . . . .	7
1.2	Regression . . . . .	9
1.3	Novelty Detection . . . . .	9
2.1	Incremental learning vs. non-incremental learning(Batch learning) . . .	18
3.1	Biological Neuron illustration . . . . .	31
3.2	Artificial Neuron Model . . . . .	32
3.3	Neural Network Components . . . . .	35
3.4	Sample Activation functions . . . . .	37
3.5	Processing Element.This figure is adapted from NeuralWare’s simulation model used in NeuralWorks Profession II/Plus . . . . .	39
3.6	perceptron . . . . .	43
3.7	(a) Linearly separable (b) Non-Linearly separable . . . . .	43
3.8	Multi-layer perceptrons . . . . .	44
3.9	Different Non-Linearly Separable Problems . . . . .	44
3.10	ADLINE . . . . .	45
3.11	Two unnormalized Gaussian radial basis functions in one input dimension. The basis function centers are located at $c_1=0.75$ and $c_2=3.25$ . . . . .	46
3.12	Hopfield Network . . . . .	47
3.13	PNN Architecture . . . . .	48
3.14	Overfitting/Overtraining in supervised learning; Training error is shown in blue, validation error in red. If the validation error increases while the training error steadily decreases then a situation of overfitting may have occurred . . . . .	50
4.1	INL: an Incremental Neural Learning framework . . . . .	55
4.2	Taxnomy of Incremental Neural Learning framework . . . . .	58
5.1	Mode. . . . .	71
5.2	Mean, Median, Mode . . . . .	71
5.3	Fractiles . . . . .	72
5.4	Quartile . . . . .	72
5.5	Interquartile Range . . . . .	74
5.6	Skewness . . . . .	75
5.7	Kurtosis . . . . .	75
5.8	Diagram of a PNN network . . . . .	77
5.9	Structur of a PNN . . . . .	78
5.10	Pattern layer of PNN . . . . .	79
5.11	Pattern layer of APNN . . . . .	79

5.12	Input data representation . . . . .	80
5.13	$U = \{C_1, C_2\}$ . . . . .	81
5.14	Input layer . . . . .	82
5.15	Input layer . . . . .	83
5.16	Hidden layer . . . . .	84
5.17	$H_1 = \{I_1, I_2, I_3\}$ . . . . .	84
5.18	$H_1 = \{I_1, I_2, I_3\}$ . . . . .	85
5.19	Add Another 8. . . . .	86
5.20	Class of 8. . . . .	87
5.21	Class of 1. . . . .	87
5.22	Neuron in hidden layer fires just to its main class in pattern layer. . . . .	88
5.23	Pattern layer. . . . .	89
5.24	Output layer. . . . .	89
5.25	Demonstration Example For our Network. . . . .	90
5.26	Add new image to existence subclass. . . . .	91
5.27	Add new image to existence subclass. . . . .	91
5.28	Add new image as a new subclass to existence class. . . . .	92
5.29	Add new image as a new class and a new subclass as well. . . . .	92
5.30	Diagram of Incremental Learning scheme . . . . .	94
5.31	Block extraction . . . . .	96
5.32	Example of a change of lighting . . . . .	98
5.33	Example of a change of scale . . . . .	98
5.34	Example of a change of facial expressions . . . . .	98
5.35	Example shift face . . . . .	99
5.36	Example of individuals carrying or without glasses . . . . .	99
5.37	Example change hair and beard presence . . . . .	99
5.38	Example of individuals of different ages, gender and skin color . . . . .	100
5.39	Whenever size of extract blocks increases , Number of extracted blocks decreases . . . . .	101
5.40	Whenever the extracted block size increases, the NN size decreases. . . . .	102
5.41	Whenever the extracted block size increases, Time of learning decreases. . . . .	102
5.42	Whenever the extracted block size increases, Time of recognition decreases. . . . .	103
5.43	Whenever the extracted block size increases too much ex.(40x40), The rate of recognition starts decrease. Due to the extream general viewpoint . Likewise, Whenever the extracted block size decreases too much ex.(2x2) , The rate of recognition starts decreases as well . Due to the extream local viewpoint . . . . .	103
5.44	Whenever the extracted block size increases, The time of learning converges to the time of recognition. . . . .	104
5.45	Whenever Number of training set of the image increases, NN recognition rate increases. and this is similar to the human learning . . . . .	104
5.46	Whenever number of using of SP increases (ex. mean , median,... are calculated together upon the image), Rate recognition increases. . . . .	105
5.47	Whenever number of SP increases ,Size of NN increases . . . . .	105

## Abstract

The objective of this thesis is to show the importance and efficiency of "Incrementality" notion in "Machine Learning" field . we also present "Artificial Neural Networks" as one of the successful and interesting approaches in "Machine Learning" , and how the "Incrementality" assists to improve the quality of learning and network capacity by consuming Spatial And Time components. In this thesis we provide a brief introduction on Machine Learning,Incremental Learning ,Neural Networks and how Incremental Learning was applied in the Neural Networks . A model of Incremental Learning using Probabilistic Neural Network was applied for Face Recognition application and we succeeded in obtaining 96% as a rate of recognition on data base ORL (Olivetti Research Laboratory). Incrementality notion in this model inspired from statistical parameters function, Our architecture based on PNN , but it utilizes Advanced PNN transfer function in hidden layer, It also uses constructive learning (start small and add), The incrementality is not limited only in adding neurons , It also appear strongly in updating weights incrementally.

**Keywords :** *Machine learning, Incremental learning, Neural Network, Probabilistic Neural Network, Face recognition.*

# Introduction

As a broad subfield of Artificial Intelligence (AI), Machine Learning (ML) is concerned with the development of algorithms and techniques that allow computers to learn. ML is trying to reshape our view of Computer Science more generally. By shifting the question from *"how to program computers"* to *"how to allow them to program themselves"*.

Many techniques and methods in ML inspired the notions from human learning and derive from the efforts of psychologists to make more precise their theories of animal and human learning through computational models. It seems likely also that the concepts and techniques being explored by researchers in machine learning may illuminate certain aspects of biological learning. Also Machine learning can be viewed as an attempt to automate parts of the scientific methods like statistics.

Someone may ask "Why should MACHINES have to learn!", There are several reasons why Machine Learning is important. Of course, as we mentioned above that the achievement of learning in machine might help us to better understand learning in general and therefore in humans and animals and to produce computer systems that are able to adapt to changing conditions. It is also a good source of ideas on how computers can be improved practically. Being adaptable is important to allow computers to respond to changes in the environment. This includes in its interaction with humans, making using a computer less frustrating and generally more efficient. currently, Speech Recognition, computer vision, Robot control and Accelerating empirical sciences are a sample of successful applications of ML and available as a commercial systems .

ML algorithms vary in their goals, learning strategies, the knowledge representation languages they employ and the type of training data they use. in the other hand, ML algorithms that do not require training are referred to as unsupervised algorithms e.g. clustering and discovery algorithms. Those that require training with a set of pre-classified examples are referred to as supervised learning algorithms e.g. decision tree learning and version space algorithms.

There are many concepts in ML that make the learning process more efficient, "Incremental Learning" is one of those concepts which tries to mimic Human learning process. Human learning is an incremental process. We learn many tasks over a life-long time and the accumulated knowledge guides our subsequent learning. When a person encounters an instance of concept that is never seen before, he first tries to find a match in his knowledge base (the brain). If he cannot find a good match, he will update his knowledge by learning from this new instance. He never throws away what

he has already learned before. In fact, he makes himself adaptive to the new case by just changing a small part of his knowledge.

In ML we say An algorithm is nonincremental if it reprocesses all earlier training instances in order to learn from each new instance. It is incremental if it can build and refine a concept gradually as new training instances come in, without reexamining all instances seen in the past.

"incremental learning" is often used for on-line, constructive, or sequential learning.

The efficiency of Incremental Learning arose when a large amount of data was available to machine learning community , that tends to improve performance by reducing the use of resources In regards to spatial resources, many problems can consist of a large evidence, which cannot fit in memory , and an incremental handling of this evidence is straightforward and convenient solution(there are, of course, other solutions, such as sampling or caching).Secondly, there is also a temporal resources improvement, since induction is much more computationally expensive than deduction. Incrementality allows the establishment of a hypothesis in the early stages of the learning process. If this hypothesis is stable, the next work will be deductive in order to check that the following evidence is consistent with the current hypothesis. Moreover, there are other reasons for using incremental learning approach : it may be impossible to have all examples initially or even its number cannot be known. In this sense, incrementality is essential when the number of examples is infinite or very large. This is the case knowledge discovery from databases.

There are a lot of Algorithms that have been proposed and applied in order to solve ML problems , and are competing to obtain the best result using different concepts in learning , Artificial Neural network is one of the successful approaches in ML , Neural networks take a different approach to problem solving than that of conventional computers. Conventional computers use an algorithmic approach i.e. the computer follows a set of instructions in order to solve a problem. Neural networks process information in a similar way the human brain does. The Network is composed of a large number of highly interconnected processing elements(neurons) working in parallel to solve a specific problem. Neural networks learn by example. They cannot be programmed to perform a specific task. Other advantages can be considered:

- *Adaptive learning* : An ability to learn how to do tasks based on the data given for training or initial experience.

- *Self-Organization* : An ANN can create its own organization or representation of the information it receives during learning time.

- *Real Time Operation* : ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.

- *Fault Tolerance via Redundant Information Coding* : Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

Classical Neural networks, with their remarkable ability to solve complicated problems and computations, have some disadvantages some in size of structure (number

of neurons) that lead to reserve more space in the memory. On the other hand, the number of iterations in its training phase may lead to slow down the neural network processing and is time consuming. There are also the problems of over and under fitting. If a network does not have enough connections then the network will not be able to learn the desired function, if however the network possesses too many connections then the architecture may appear to learn the desired function but the resulting generalization properties of the neural network will be poor. Such a network has over-fitted the examples and is acting more like a look-up table. There are three factors that affect the quality of a neural network solution :

- *Success achieved*: on test data indicates how well the network generalizes to data unseen during training; which one wants to maximize. This generally is taken as the only performance criterion.

- *Network complexity* : by itself can be very difficult to assess but two important factors are network size and processing complexity of each unit. Network size gives the memory required which is the product of the number of connections and the number of bits required to store each connection weight. Processing complexity depends on how costly it is to implement processing occurring in each unit, e.g., sigmoid vs. threshold nonlinearity, fanin, fanout properties, precision in storage and computation, etc. This has a negative effect on the quality as one prefers smaller and cheaper networks.

- *Learning time* : is the time required to learn the given training data till one gets a reasonable amount of performance. This is to be minimized also.

Many Neural learning algorithms are not incremental. One of the greatest impediments in building large, scalable learning systems based on neural networks is that when a network trained to solve task A is subsequently trained to solve task B, it "forgets" the solution to task A. In other words, Disruption in neural networks, called "forgetting" or "catastrophic interference", often occurs during incremental learning. It is caused by the excessive adaptation of connection weights to new data. One way of overcoming this problem is that only representative training data are kept in memory and some of them are trained with newly given training data.

For these reasons have mentioned above , researchers have tried to involve incrementality notion with the powerful of Neural network for obtaining a robust learning system. In this context The phrase "incremental learning" has been used to refer in the literature to as diverse concepts as incremental network growing and pruning of classifier architectures, in the sense of constructive learning (start small and add) and destructive learning (start big and remove), also incremental learning may refer to selection of most informative training samples , In other cases, some form of controlled modification of classifier weights has been suggested, typically by retraining with misclassified signals.

Incremental learning aims to improve network capacity by consuming Spatial And Time components.

In this thesis, We propose an Incremental Learning depending on Probabilistic Neural Networks (PNN) [5], PNN has gained interest because it offers a way to interpret the network's structure in the form of a probability density function and it is easy to implement. PNN utilizes statistical parameters, For example: mean , vari-

ance,..ect. Statistical parameters function in an incremental manner without the need to prompted them to do so,. it also depends on constructive learning (starts small and increases) that can be highly effective for escaping bad local minima of the objective function. Every PNN networks have four layers: Input layer, Hidden layer, Pattern layer, Decision layer . We have applied the idea for face recognition and we succeeded in obtaining 96% as a rate of recognition on data base ORL (Olivetti Research Laboratory).This approach can be used in supervised or unsupervised paradigm.

Thus, the rest of thesis will be structured as follows: in the first chapter, a brief overview of Learning Theory and Machine Learning is introduced. Chapter 2 will provide a description of Incremental Learning , definitions and motivations , then we present some techniques that have been used to learn in an incremental manner. In the third chapter, we give an overview of Neural Networks, motivations , different types of Neural Networks that have been mentioned and their applications also is introduced . Next chapter gives an overview involving Incrementality notion in Neural Networks, famous NN algorithms and techniques that involve incremental learning concept are also introduced . in the last chapter we propose an incremental NN depending on Probabilistic Neural Netowerk that was applied for face recognition.



# LEARNING THEORY

## 1.1 Introduction

**L**EARNING in its broadest sense is defined as building a model from incomplete information in order to predict as accurately as possible some underlying structure of the unknown reality. In practice, for this definition to take sense to a computer scientist or statistician, the information about the structure (experience) and the model have expressed numerically. Thus from statistical point of view, the problem of learning becomes a problem of function estimation. As such, we refer to an estimator of functions as a *Learning Machine*.

## 1.2 Human and machine learning Definitions

*Learning* term in a dictionary definition includes phrases such as to gain knowledge, or understanding of, or skill in, by study, instruction, or experience. the process of gaining understanding that leads to the modification of attitudes and behaviors through the acquisition of knowledge, skills and values, through study and experience. In human learning, there is more than one type of learning, identified three domains of educational activities:

- **Cognitive:** mental skills (Knowledge), knowledge acquired by systematic study in any field of scholarly application
- **Affective:** growth in feelings or emotional areas (Attitude), the act or process of acquiring knowledge or skill.
- **Psychomotor:** manual or physical skills (Skills).[18]

In contrast, **Learning in machine** :

- Learning: [Simon] Changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently the next time.

Mitchell (1997): A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$ , if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ .

Witten and Frank (2000): things learn when they change their behavior in a way that makes them perform better in the future.

- Improvement of performance in an environment through acquisition of knowledge in that environment.
- Machine Learning is the study of computer algorithms that improve automatically through experience. Mitchell, 1997 .
- Machine learning is programming computers to optimize a performance criterion using example data or past experience. Alpaydin, 2004 .

There are several parallels between human and machine learning, Certainly, many techniques in machine learning derive from the efforts of psychologists to make more precise their theories of animal and human learning through computational models. It seems likely also that the concepts and techniques being explored by researchers in machine learning may illuminate certain aspects of biological learning.[18]

Some ML attempt to eliminate the need for human intuition in the analysis of the data, while others adopt a collaborative approach between human and machine. Human intuition cannot be entirely eliminated since the designer of the system must specify how the data is to be represented and what mechanisms will be used to search for a characterization of the data. Machine learning can be viewed as an attempt to automate parts of the scientific method. Some machine learning researchers create methods within the framework of Bayesian statistics.

In conclusion: Machine Learning system we might say, very broadly, that a machine learns whenever it changes its structure, program, or data based on its inputs or in response to external information) in such a manner that its expected future performance improves.

ML usually refers to the changes in systems that perform tasks associated with artificial intelligence (AI). Such tasks involve recognition, diagnosis, planning, robot control, prediction , etc. the *changes* might be either enhancements to already performing systems or .... synthesis of new systems.

### 1.3 The motivation of research in Machine Learning

There are two main motives for research in this area: to better understand learning in general and therefore in humans and animals and to produce computer systems that are able to adapt to changing conditions. Understanding learning in humans and animals is not only a goal in its own right, it is also a good source of ideas on how computers can be improved practically. Being adaptable is important to allow computers to respond to changes in the environment. This includes in its interaction with humans, making using a computer less frustrating and generally more efficient.[19]

## 1.4 Importance of Machine Learning

There are several reasons why ML is important. as we mentioned some reasons referred to help us understand how animals and humans learn, but there are important engineering reasons as well. Some of these:[18,19]

- Some tasks cannot be defined well except by example; that is, we might be able to specify input/ output pairs but not a concise relationship between inputs and desired outputs. We would like machines to be able to adjust their internal structure to produce correct outputs for a large number of sample inputs and thus suitably constrain their input/output function to approximate the relationship implicit in the examples.
- It is possible that hidden among large piles of data are important relationships and correlation. Machine learning methods can often be used to extract these relationships(data mining).
- Human designers often produce machines that do not work as well as desired in the environments in which they are used. In fact, certain characteristics of the working environments might not be used for on-the-job improvement of existing machine designs.
- The amount of knowledge available about certain tasks might be too large for explicit encoding by humans. Machines that learn this knowledge gradually might be able to capture more of it than humans would want to write down.
- Environments change over time. Machines that can adapt to a changing environment would reduce the need for constant redesign.
- New knowledge about tasks is constantly being discovered by humans. Vocabulary changes. There is a constant stream of new events in the world. Continuing redesign of AL systems to conform to new knowledge is impractical, but machine learning methods might be able to track much of it.

## 1.5 Wellsprings of machine learning

Work in machine learning is now converging from several sources. These different traditions each bring different methods and different vocabulary which are now being assimilated into a more unified discipline. Here is a brief listing of some of the separate disciplines that have contributed to machine learning:[19]

- **Statistics:** A long-standing problem in statistics is how best to use samples drawn from unknown probability distributions to help decide from which distribution some new sample is drawn. A related problem is how to estimate the value of an unknown function at a new point given the values of this function at a set of sample points. Statistical methods for dealing with these problems can be considered instances of machine learning because the decision and estimation rules depend on a corpus of samples drawn from the problem environment.

- **Brain Models:** None-linear elements with weighted inputs have been suggested as simple models of biological neurons. Networks of these elements have been studied by several researchers including [McCulloch Pitts, 1943, Hebb, 1949, Rosenblatt, 1958] and, more recently by [Gluck Rumelhart, 1989, Sejnowski, Koch Churland, 1988]. Brain modelers are interested in how closely these networks approximate the learning phenomena of living brains. We shall see that several important machine learning techniques are based on networks of nonlinear elements often called neural networks. Work inspired by this school is sometimes called connectionism, brain style computation, or sub-symbolic processing.
- **Adaptive control theory:** Control theorists study the problem of controlling a process having unknown parameters which must be estimated during operation, and control process must track these changes. Some aspects of controlling a robot based on sensory inputs represent instances of this sort of problem.
- **Psychological Models:** Psychologists have studied the performance of humans in various learning tasks. An early example is the EPAM network for storing and retrieving one member of a pair of words when given another [Feigenbaum, 1961]. Related work led to a number of early decision tree [Hunt, Marin, Stone, 1966] and semantic network [Anderson Bower, 1973] methods. More recent work of this sort has been influenced by activities in artificial intelligence .

## 1.6 Overview Of Learning Theory

### 1.6.1 Types of Learning

*Deductive learning* works on existing facts and knowledge and deduces new knowledge from the old. This is best illustrated by giving an example. For example, assume:

$A = B$

$B = C$

Then we can deduce with much confidence that

$C = A$

Arguably deductive learning does not generate *new* knowledge at all, it simply memorises the logical consequences of what is known already.

Whereas *Inductive learning* takes examples and generalises rather than starting with existing knowledge. For example, having seen many cats, all of which have tails, one might conclude that all cats have tails. This is unsound step of reasoning but it would be impossible to function without using induction to some extent. In many areas it is an explicit assumption. There is scope of error in inductive reasoning, but still it is a useful technique that has been used as the basis of several successful systems. One major subclass of inductive learning is concept learning. This takes examples of a concept and tries to build a general description of the concept. Very often, the examples are described using attribute-value pairs. ID3 and Version Spaces are inductive learning algorithms. [18,35]

There are **six main types of learning**:

### 1.6.1.1 Speed-up learning

A type of deductive learning that requires no additional input, but improves the agent's performance over time. There are two kinds, rote learning, and generalization (e.g., EBL). Data caching is an example of how it would be used.

### 1.6.1.2 Learning by taking advice

Deductive learning in which the system can reason about new information added to its knowledge base. McCarthy proposed the "advice taker" which was such a system, and TEIRESIAS [Davis, 1976] was the first such system.

### 1.6.1.3 Learning from examples

Inductive learning in which concepts are learned from sets of labeled instances.

### 1.6.1.4 Clustering

Unsupervised, inductive learning in which "natural classes" are found for data instances, as well as ways of classifying them. Examples include COBWEB, AUTO-CLASS.

### 1.6.1.5 Learning by analogy

Inductive learning in which a system transfers knowledge from one database into a that of a different domain.

### 1.6.1.6 Discovery

Both inductive and deductive learning in which an agent learns without help from a teacher. It is deductive if it proves theorems and discovers concepts about those theorems; it is inductive when it raises conjectures

## 1.6.2 Major Paradigms of Machine Learning

### 1.6.2.1 Supervised learning

Supervised learning is a machine learning technique for creating a function from training data. The training data consist of pairs of input objects (typically vectors), and desired outputs. The output of the function can be a continuous value (called regression), or can predict a class label of the input object (called classification). The task of the supervised learner is to predict the value of the function for any valid input object after having seen a number of training examples (i.e. pairs of input and target output). To achieve this, the learner has to generalize from the presented data to unseen situations in a reasonable way .Wikipedia (2006).

In other words, We have some empirical data (images, medical observations, market indicators, socioeconomical background of a person, texts), say  $x_i \in X$ , usually together with labels  $y_i \in Y$  (digits, diseases, share price, credit rating, relevance) and we want to find a function that connects  $x$  with  $y$ .

### 1.6.2.2 Unsupervised learning

Supervised learning is a method of machine learning where a model is fit to observations. It is distinguished from supervised learning by the fact that there is no a priori output. In unsupervised learning, a data set of input objects is gathered. Unsupervised learning then typically treats input objects as a set of random variables. A joint density model is then built for the data set. Wikipedia (2006) in other words, We have some data  $x_i$  and we want to find regularities or interesting objects in general from the data.[18] Examples:

- Find natural groupings of  $X_s$  ( $X$ =human languages, stocks, gene sequences, animal species,..) Prelude to discovery of underlying properties Summarize the news for the past month
- Cluster first, then report centroids.
- Sequence extrapolation: E.g. Predict cancer incidence next decade; predict rise in antibiotic-resistant bacteria

Methods

- Clustering (n-link, k-means, GAC,...)
- Taxonomy creation (hierarchical clustering)
- Novelty detection (meaningful outliers)
- Trend detection (extrapolation from multivariate partial derivatives)

### 1.6.2.3 Reinforcement learning

Refers to a class of problems in machine learning which postulate an agent exploring an environment in which the agent perceives its current state and takes actions. The environment, in return, provides a reward (which can be positive or negative). Reinforcement learning algorithms attempt to find a policy for maximizing cumulative reward for the agent over the course of the problem. Wikipedia (2006)

### 1.6.2.4 Batch vs. online learning

All training examples at once or one at a time (with estimate and update after each example).

## 1.6.3 Three main learning problems

This formulation of the learning problem is rather general. It encompasses many specific problems, the main ones are:

pattern recognition ,regression estimation ,density estimation.[1, ]

### 1.6.3.1 Classification

*Goal:* We want to find a function  $f : x \rightarrow \mathfrak{R}$  which will tell us whether a new observation belongs to specific class and possibly the confidence with which this is so.

*Questions:* How to rate  $f$ , how to find  $f$ , how to interpret  $f$ , show Figure[1.1].

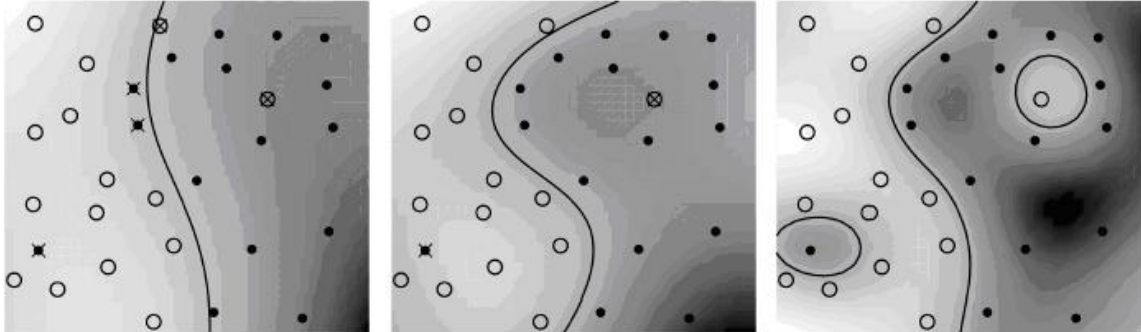


Figure 1.1: Classification

- **Learning is not (only) Memorizing**

- *Goal*, Infer from a training set to the general connection between  $x$  and  $y$ .
- *Simple and Dumb Learning* Memorize all the data we are given. This clearly gets the hypotheses on the training set right (popular strategy in the 80s), but what to do for new  $x$ ?
- *Slightly Better Strategy* Memorize all the data and for new observations predict the same as the closest neighbor  $x_i$  to  $x$  does (Nearest Neighbour algorithm).
- *Problem* What about noisy and unreliable data? Different notions of proximity? We want to generalize our training data to new and unknown concepts.

- **Errors in Classification** We say that a classification error occurs, if  $f(x) \neq y$ , i.e. if  $f(x)$  predicts an outcome different from  $y$ .

- **Data Generation** Assume that the  $x$  are drawn from some distribution and that  $y$  is either deterministic (for every  $x$  there is only one true  $y$ ) or random ( $y = 1$  occurs in a fraction  $p$  of all cases and  $y = -1$  in a fraction of  $1 - p$  cases). This distribution could simply be due to someone writing 0 and 1 on a piece of paper. Or the different processes that smudge and degrade paper when an envelope ends up in the postbox.

*Goal* We want to rate our classifier  $f$  with respect to this data generating process.

- **Applications**

- *Optical Character Recognition* The goal is to classify (handwritten) characters (note that here  $f$  has to map into  $\{a, \dots, z\}$  rather than into  $\{-1, 1\}$  automatically (form readers, scanners, post).

- *Spam Filtering* Determine whether an e-mail is spam or not (or whether it is urgent), based on keywords, word frequencies, special characters (\$, !, uppercase, whitespace), . . .
- *Medical Diagnosis* Given some observations such as immune status, blood pressure, etc., determine whether a patient will develop a certain disease. Here it matters that we can estimate the probability of such an outcome.
- *Face Detection* Given a patch of an image, determine whether this corresponds to a face or not.

### 1.6.3.2 Regression

*Goal* We want to find a function which will tell us the value of  $f$  at  $x$ . show (Figure[1.2]).

- Applications
  - *Getting Rich* Predict the stock value of IBM/CISCO/BHP/TELSTRA . . . given today's market indicators (plus further background data).
  - *Wafer Fab* Predict (and optimize) the yield for a microprocessor, given the process parameters (temperature, chemicals, duration, . . .).
  - *Network Routers* Predict the network traffic through some hubs/routers/switches. We could reconfigure the infrastructure in time . . .
  - *Drug Release* Predict the requirement for a certain drug (e.g. insulin) automatically.



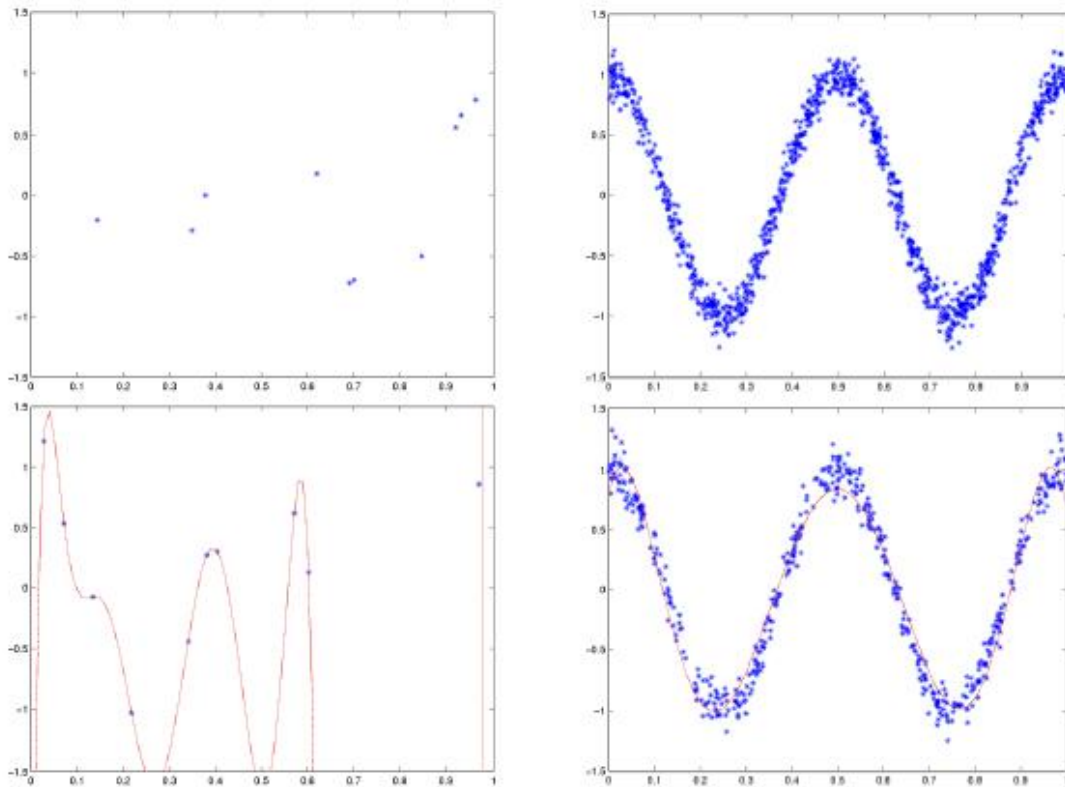


Figure 1.2: Regression

### 1.6.3.3 Novelty Detection

*Goal* Build estimator that finds unusual observations and outliers and Build estimator that can assess how typical an observation is. (Figure [1.3]).

*Idea* Data is generated according to some density  $p(x)$ . Find regions of low density. Such areas can be approximated as the level set of an auxiliary function. No need to estimate  $p(x)$  directly use proxy of  $p(x)$ . Specifically: find  $f(x)$  such that  $x$  is novel if  $f(x) \leq c$  where  $c$  is some constant.

*Data* Observations  $(x_i; y_i)$  generated from some  $P(x)$ , e.g. (network usage patterns) (handwritten digits) (alarm sensors) (factory status).

*Task:* Find unusual events, clean database, distinguish typical.

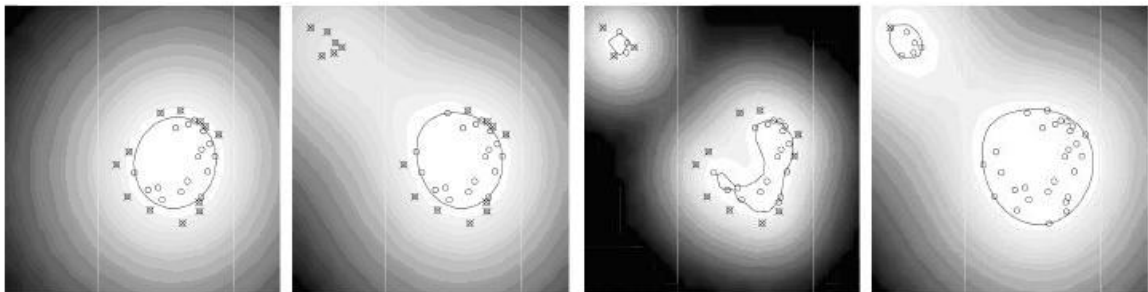


Figure 1.3: Novelty Detection

- Applications
  - *Network Intrusion Detection* Detect whether someone is trying to hack the network, downloading tons of MP3s, or doing anything else unusual on the network.
  - *Jet Engine Failure Detection* You can't (normally) destroy a couple of jet engines just to see how they fail.
  - *Database Cleaning* We want to find out whether someone stored bogus information in a database (typos, etc.), mislabeled digits, ugly digits, bad photographs in an electronic album.
  - *Fraud Detection* Credit Card Companies, Telephone Bills, Medical Records  
Self calibrating alarm devices Car alarms (adjusts itself to where the car is parked), home alarm (location of furniture, temperature, open windows, etc.)

#### 1.6.4 A Framework for learning systems

Learning system Framework can be considered as following [19,35] :

- Environment
  - task: clustering, classification, prediction (one-step vs multi-step).
  - performance measure: accuracy of prediction vs quality and adequacy of classification.
  - amount of supervision: supervised vs unsupervised learning
  - domain characteristics: complexity of target concept, presence of irrelevant features, presence of noise
- Representational choices
  - instances: sets of attributes, sets of relational literals
  - concepts: conjunctive, threshold, competitive, disjunctive concepts
- Learning component
  - learning biases: search biases vs representational biases
  - search strategies through space of possible concepts
  - processing mode: incremental vs non-incremental learning

#### 1.6.5 Designing a Learning System

In designing a learning system, there are four major issues to consider:

- *components* which parts of the performance element are to be improved
- *representation* of those components
- *feedback* available to the system

- *prior* information available to the system

All learning can be thought of as learning the representation of a function. In designing a learning system, we have to deal with (at least):

- Training experience:
  - Direct or indirect evidence (supervised or unsupervised).
  - Controlled or uncontrolled sequence of training examples.
  - Representativity of training data in relation to test data.
- Target function and Learned function:
  - The problem of improving performance can often be reduced to the problem of learning some particular target function.
  - In many cases we can only hope to acquire some approximation to the ideal target function.
- Learning algorithm:
  - In order to learn the (approximated) target function we require:
  - A set of training examples (input arguments)
  - A rule for estimating the value corresponding to each training example (if this is not directly available)
  - An algorithm for choosing the function that best fits the training data.

### 1.6.6 Some Approaches to Machine Learning

- Decision trees
- Artificial neural networks
- Bayesian learning
- Instance-based learning (cf. MBL)
- Genetic algorithms
- Relational learning (cf. ILP).

#### 1.6.6.1 Decision Tree Learning

A simple structure for inductive learning. Given an instance of the problem, a decision tree returns a yes or no decision about it; thus decision trees are Boolean classifiers. Each branching node in the tree represents a test on some aspect of the instance. Decision trees classify instances by sorting them down the tree from the root to some

leaf node, where:

- Each internal node specifies a test of some attribute.
- Each branch corresponds to a value for the tested attribute.
- Each leaf node provides a classification for the instance.

Decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances.

- Each path from root to leaf specifies a conjunction of tests.
- The tree itself represents the disjunction of all paths.

]]

### 1.6.6.2 Artificial Neural Networks (more details will be discussed in chapter 3)

Learning methods based on artificial neural networks (ANN) are suitable for problems with the following characteristics:

- Instances are represented by many attribute-value pairs.
- The target function may be discrete-valued, real-valued, or a vector of real- or discrete-valued attributes.
- The training examples may contain errors.
- Long training times are acceptable.
- Fast evaluation of the learned target function may be required.
- The ability of humans to understand the learned target function is not important[]

### 1.6.6.3 Bayesian Learning

Two reasons for studying Bayesian learning methods

- Efficient learning algorithms for certain kinds of problems
- Analysis framework for other kinds of learning algorithms

Features of Bayesian learning methods:

- Assign probabilities to hypotheses (not accept or reject)
- Combine prior knowledge with observed data

- Permit hypotheses that make probabilistic predictions
- Permit predictions based on multiple hypotheses, weighted by their probabilities.[]

#### 1.6.6.4 Genetic Algorithms

Genetic algorithms search the space of individuals for good candidates. The goodness of an individual is measured by some fitness function. Search takes place in parallel, with many individuals in each generation. The approach is a hill-climbing one, since in each generation the offspring of the best candidates are preserved. []

#### 1.6.6.5 Relational Learning

Most learning methods are limited to propositional logic, where each variable represents an atomic proposition. Relational learning (aka inductive logic programming) applies to predicate logic and can learn rules involving variables. []

#### 1.6.6.6 Modern" learning algorithms

- Concept learning algorithms Version spaces (combined with SVMs),
- Statistical algorithms Gaussian mixtures, Naïve Bayes, ...
- Neural networks multilayer perceptron, Kohonen maps, ...
- Advanced algorithms Support Vector Machines, kernel approaches
- ...

## 1.7 Open Problems in Learning

The most popular problems in learning are [19] :

- **choice of good generalization languages** : bad choices lead to too large a search space, an inability to find the concept, or exponential resource consumption; good choices lead to good results, but it isn't clear how to design good languages in general.
- **using expectations and prior knowledge** : generalizations can be acceptable wrt training instances, but also acceptable wrt prior expectations (e.g., model-based constraints built into the language); the problem is to find ways to combine prior knowledge with training data to constrain learning.
- **inconsistency problems** : when (1) the generalization language can't describe the target generalization or (2) the training instances contain errors .
- **partially learned generalizations** in general cases, unlikely the training data will determine a unique generalization; problem is how to cope with this .

## 1.8 Application Successes

One measure of progress in Machine Learning is its significant real-world applications, such as those listed below. Although we now take many of these applications for granted, it is worth noting that as late as 1985 there were almost no commercial applications of machine learning.[35]

- **Speech recognition:** Currently available commercial systems for speech recognition all use machine learning in one fashion or another to train the system to recognize speech. The reason is simple: the speech recognition accuracy is greater if one trains the system, than if one attempts to program it by hand. In fact, many commercial speech recognition systems involve two distinct learning phases: one before the software is shipped (training the general system in a speaker-independent fashion), and a second phase after the user purchases the software (to achieve greater accuracy by training in a speaker-dependent fashion).
- **Computer vision:** Many current vision systems, from face recognition systems, to systems that automatically classify microscope images of cells, are developed using machine learning, again because the resulting systems are more accurate than hand-crafted programs. One massive-scale application of computer vision trained using machine learning is its use by the US Post Office to automatically sort letters containing handwritten addresses. Over 85% of handwritten mail in the US is sorted automatically, using handwriting analysis software trained to very high accuracy using machine learning over a very large data set.
- **Bio-surveillance:** A variety of government efforts to detect and track disease outbreaks now use machine learning. For example, the RODS project involves real-time collection of admissions reports to emergency rooms across western Pennsylvania, and the use of machine learning software to learn the profile of typical admissions so that it can detect anomalous patterns of symptoms and their geographical distribution. Current work involves adding in a rich set of additional data, such as retail purchases of over-the-counter medicines to increase the information flow into the system, further increasing the need for automated learning methods given this even more complex data set.
- **Robot control:** Machine learning methods have been successfully used in a number of robot systems. For example, several researchers have demonstrated the use of machine learning to acquire control strategies for stable helicopter flight and helicopter aerobatics. The recent Darpa-sponsored competition involving a robot driving autonomously for over 100 miles in the desert was won by a robot that used machine learning to refine its ability to detect distant objects (training itself from self-collected data consisting of terrain seen initially in the distance, and seen later up close).
- **Accelerating empirical sciences:** Many data-intensive sciences now make use of machine learning methods to aid in the scientific discovery process. Machine learning is being used to learn models of gene expression in the cell from high-throughput data, to discover unusual astronomical objects

## **1.9 Conclusion**

In this chapter we have presented a brief overview of Machine Learning and Learning theory with the different existing definitions and paradigms ,we also presented the main learning functions classification,regression and Novelty detection . we also gave a brief summary of some approches in machine learning. At the end we introduced open problems in machine learning and applications success.

# INCREMENTAL LEARNING

## 2.1 Introduction

**H**UMAN learning is an incremental process. We learn many tasks over a life-long time and the accumulated knowledge guides our subsequent learning. When a person encounters an instance of concept that is never seen before, he first tries to find a match in his knowledge base (the brain). If he cannot find a good match, he will update his knowledge by learning from this new instance. He never throws away what he has already learned before. In fact, he makes himself adaptive to the new case by just changing a small part of his knowledge.[2]

The notion of **Incrementality** in machine learning is a powerful and technique that tends to improve performance by reducing the use of resources. In regard to spatial resources, many problems can consist of a large evidence, which cannot fit in memory, and an incremental handling of this evidence is straightforward and convenient solution (there are, of course, other solutions, such as sampling or caching). Secondly, there is also a temporal resources improvement, since induction is much more computationally expensive than deduction. Incrementality allows the establishment of a hypothesis in the early stages of the learning process. If this hypothesis is stable, the next work will be deductive in order to check that the following evidence is consistent with the current hypothesis. Moreover, there are other reasons for using incremental learning approach: it may be impossible to have all examples initially or even its number cannot be known. In this sense, incrementality is essential when the number of examples is infinite or very large. This is the case knowledge discovery from databases. [2,6]

Incrementality in machine learning involves the number of past examples a learning algorithm should reprocess during each learning step. An algorithm is nonincremental if it reprocesses all earlier training instances in order to learn from each new instance. It is incremental if it can build and refine a concept gradually as new training instances come in, without reexamining all instances seen in the past. Langley [1996] proposes the notion of degree of incrementality to smooth out this dichotomy: a learner is incremental in degree  $k$  if it reprocesses at most  $k$  previous cases to learn from a new instance.



## 2.2 On-line vs. Off-line Learning

Offline learning happens when all training items are given at the same time. In other word, In off-line learning, all the data are stored and can be accessed repeatedly (Batch learning is always off-line), On the contrary, Online learning occurs when training data items are presented one at a time. In on-line learning, each case is discarded after it is processed (On-line training is always incremental). There are intermediate situations where data items are presented in short chunks.[32]

The first studies of online systems were justified by the nature of human learning. A system was supposed to imitate the learning process of humans and people experience events in sequence. More recently, online systems are found to be useful in more and more real-world environments since companies gather new data every day and these data should be used in order to improve the knowledge base.

**N.B:** Batch learning is always off-line. On-line training is always incremental. Incremental learning can be done either on-line or off-line. [33]

## 2.3 Incremental learning vs. non-incremental learning (Batch learning)

A distinction can be made for learning algorithms which can either handle all database instances at a time in a non-incremental manner or one instance at a time in an incremental fashion. More precisely, non-incremental algorithms, given the whole training data set, output a domain model after processing data, possibly multiple times. This sort of algorithms stop learning when they have processed the dataset and assume that they have reached a good domain model which will not be revised. On the other hand, incremental algorithms never assume that they reach a final learning stage. They keep improving their domain model by processing new data items as they are available. Incremental algorithms process each single item of data as it is available without reprocessing previously seen ones.

In this way, during the whole learning process there is a domain model available, although incomplete, that can be used for whatever task it is intended.

Incremental Learning is often used for on-line, constructive, or sequential learning.

Naturally, incremental algorithms are best suited for online environments (note incremental learning should be differentiated from online learning and constructive learning), and non-incremental algorithms suite best off-line ones. In spite of that, one could adapt one non-incremental algorithm to work in an online environment by storing new instances together with the old ones and re-running the learning process on the whole set. One could also use an incremental algorithm in an off-line environment by running the algorithm for each instance of the database. Both sorts of algorithms have their advantages. On the one hand non-incremental learners can collect statistics about all training instances and thus perform a more informed search than an incre-

mental learner could do. On the other, in an off-line environment incremental learners use less memory space and computer time than non-incremental ones while obtaining knowledge structures of similar quality ,Figure[2.1] shows how the process works .[2]

Each mode of learning has its advantages: nonincremental or batch learners usually avail of overall information about the training sample to take more enlightened decisions; on the other hand, there is an increasing demand for incremental learners as massive volumes of data become available for data mining applications. A typical drawback of incremental learners is their sensitivity to the order of presentation of training instances [Ripley, 1996 ; Guvenir and Sirin, 1996].

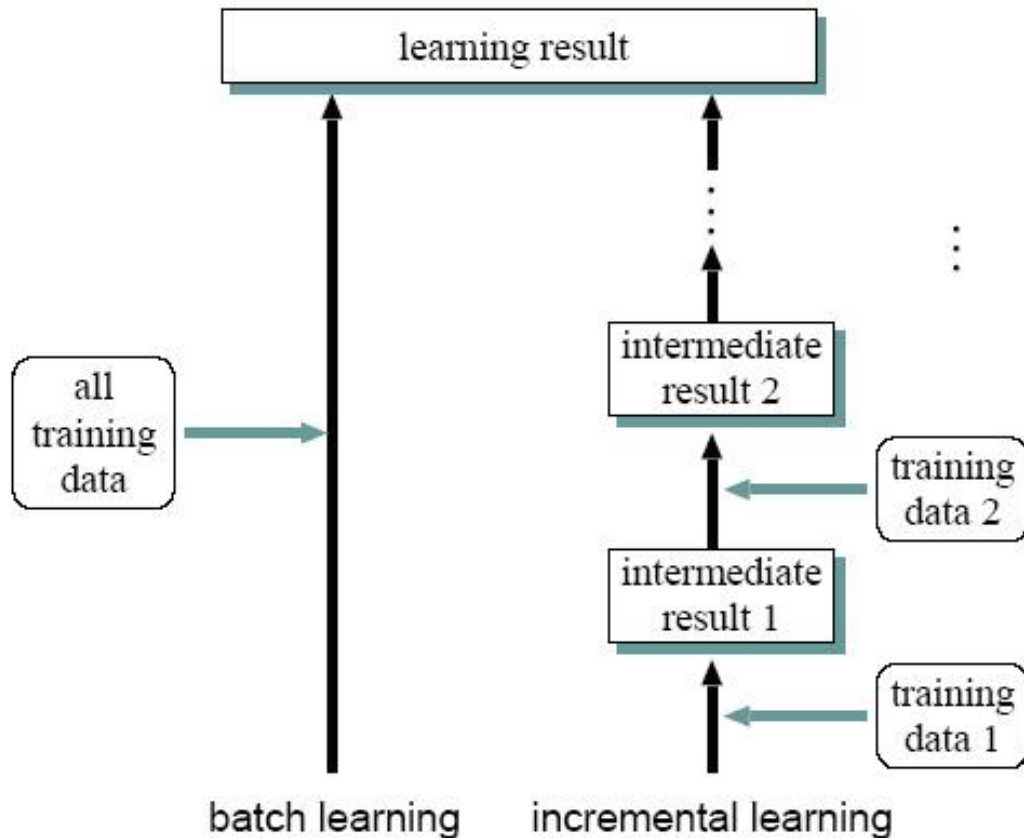


Figure 2.1: Incremental learning vs. non-incremental learning(Batch learning)

## 2.4 Incremental Learning

### 2.4.1 Incremental Learning Tasks

[2,30] The term incremental has been applied both to learning tasks and to learning algorithms, thus leading to some confusion. This section attempts to elucidate this confusion by providing formal definitions and examples of incrementality for tasks and for algorithms.

The following characterises the notion of incrementality as it applies to learning

tasks.

A learning task is incremental if the training examples used to solve it become available over time, usually one at a time.

Note that, if one is prepared to wait long enough, any incremental learning task can, in principle, become a non-incremental one. Hence, for incremental learning tasks, there is an implicit assumption that waiting is undesirable and/or impractical. In particular, the nature of the application may render unfeasible the timely generation of a sufficiently large number of representative examples, either because the environment changes in time (and thus learning becomes situated or context sensitive) or because the rate at which examples become available may be too slow.

## 2.4.2 Examples of incremental learning tasks

In the first three, incrementality is due to changes in the target over time. In the last one, the target does not change, but the acquisition of data is untimely.

### 2.4.2.1 User modelling/profiling

With humans, today's behaviour is not necessarily a good indication of tomorrow's. Experts argue that users' behaviour and interests vary radically within as little as six months. Hence, the useful task of learning user profiles requires on-line monitoring of individual users.

### 2.4.2.2 Robotics

In all but the simplest of cases, a robot's environment is changing and often unpredictable. Hence, in order to survive (e.g., by negotiating collision-free navigation) and to carry out its tasks successfully, a robot must be able to react and adapt incrementally to environmental cues.

### 2.4.2.3 Intelligent agents

Agents are software implements characterised by both reactivity and proactivity. Hence, as with robots, incrementality is inherent in agent learning. Traditional agent applications where learning is useful include network management (e.g., load balancing, routing) and intelligent user interfaces.

### 2.4.2.4 Software project estimation

Estimating the cost, effort and duration of software projects is largely a matter of experience. Due to the rather long timescales of such projects however, useful data becomes available in a piecemeal way, over time. It has been argued that the construction of an adequate baseline for estimation may take up to three years. Yet, even limited experience is better than no experience at all in improving the accuracy of estimates. In contrast, classification problems (e.g., discriminating between rocks and mines from a sample of labelled sonar readings) are standard examples of nonincremental tasks since all training data is usually available a priori.

### 2.4.3 The main characteristics of an incremental learning task

- Examples are not available a priori but become available over time, usually one at a time.
- Learning may need to go on (almost) indefinitely.

## 2.5 Incremental Learning Algorithms

[2,27] Incremental learning algorithms have been thoroughly studied within the machine learning community. During the second half of the eighties several systems were proposed in the field of clustering, Fisher's COBWEB being one of the most cited ones even nowadays. More recently, the field of Data Mining and Knowledge Discovery in Databases is concerned with very large databases available in streams which do not fit in main memory and has spawned new interest in incremental methods. Incremental learning algorithms have received less attention. There exists, as far as we know, the work of Buntine (1991), Lam and Bacchus (1994) and finally Friedman et. al (1997). More recently Hulten et al. (2002) have developed a general method for mining large databases which can also be used as an incremental method for mining data streams.

### 2.5.1 Purpose and definition

There are real-world environments with very strong constraints and requirements that may affect the learning process. We could summarize such constraints and requirements under three main categories:

- Resource limitation: in real-world applications there may be computing time and memory space limitations. Additionally, when databases are so huge that they must be stored in secondary memory, multiple inspection of such amount of data is unfeasible. Similarly, it may be unreasonable to keep in memory several alternative knowledge bases.
- Any-time availability: sometimes, given the nature of the real-world application, an intelligent agent needs to use a domain model in order to carry out its performance task even if the whole dataset is not available. Incremental methods can deal with such situations because they keep a domain model during the whole learning process.
- Changing worlds: when intelligent agents must survive in a changing world they should be able to make their model of the world evolve. Incremental algorithms are a natural solution to cope with such situations because they are able to incorporate into the model new samples from the changing world. In such environments, learning algorithms should also be provided with some mechanisms in order to forget old experiences (i.e. data instances) that do not represent the current state of the world. Incremental algorithms are a response to these requirements as we can see in the definitions found in the literature. Maybe the most widely accepted definition of the main properties of An incremental algorithm was stated by Langley.

**DEFINITION 1** *An Incremental algorithm*, a learner  $L$  is incremental if  $L$  inputs one training experience at a time, does not reprocess any previous experiences, and retains only one knowledge structure in memory.

This definition is rather strong, because it imposes three heavy constraints on an algorithm in order to be incremental. The first two constraints require learning algorithms to be able to use their knowledge at any time during the learning process. The second one rules out those systems that process new data together with old data in order to come up with a new model. The important idea of this constraint is maintaining reasonably low and constant the time required to process each data instance over the whole dataset. The third constraint aims at learning algorithms not making unreasonable memory demands. However, these three strong constraints can be lowered in different degrees. Incremental algorithms can be allowed to process a chunk of  $k$  data items at a time, to process at most  $k$  previous instances after encountering a new training instance, or to keep  $k$  alternative knowledge bases in memory. In this way we can relax each of these three constraints in a degree that can be specified with a parameter.

**DEFINITION 2** *An incremental algorithm* should meet the following constraints:

- *It must require small constant time per record.*
- *It must be able to build a model using at most one scan of the data.*
- *It must use only a fixed amount of main memory, irrespective of the total number of records it has seen.*
- *It must make a usable model available at any point in time, as opposed to only when it is done with processing the data.*
- *It should produce a model that is equivalent (or nearly identical) to the one that would be obtained by the corresponding batch algorithm.[2,15]*

This second definition is also concerned with the time and memory space that incremental algorithms spend when they process new data instances. It also makes explicit that the learned model must be available at any time, which actually is a consequence of keeping low the time required to process an incoming data instance. Definition 2 explicitly wants the models learned with incremental algorithms to be of similar quality than the ones that batch algorithms would obtain. As we will see in the next section, incremental algorithms spend less computing time and memory space, but may produce models of lower quality than batch approaches.

**The main reason for the importance of incremental learning** is that we may gather new data every day and that it would be interesting to revise the current model in the light of this new data without spending an unreasonable amount of time and memory. And this is desirable even if it may happen that the data gathered in one single day were enough to obtain a model of very high quality, and hence, there would be no need for incremental algorithms.

We believe that this situation is very unlikely to happen because of two reasons. Firstly, we may not be sure that the data already available are really representative

of the whole domain. Namely, the data may not be a fair sampling of the underlying probability distribution and thus, it cannot be accurately estimated from the currently available dataset. And secondly, in order to obtain complex models, where lots of variables are related to each other, it is needed large datasets. It is widely reported in the statistical pattern recognition literature (read Jain et al.) that the performance of a classifier depends on the interrelationship between sample sizes, number of features, and classifier complexity. It has been often observed in practice that adding variables to a classifier may actually degrade its performance if the number of data instances that are used to learn the classifier is small relative to the number of variables. This is known as the peaking phenomenon which is a consequence of the curse of dimensionality, usually stated as follows: in order to estimate a joint probability, the number of required data instances grows exponentially with the number of variables. This is due to the fact that the required number of parameters in order to estimate a joint probability distribution grows exponentially with the number of variables, i.e the number of counters of a contingency table.

This is also illustrated by T. Hastie et al. [1], when they state that the sampling density is proportional to  $N/p$ , where  $p$  is the number of variables and  $N$  is the sample size. Thus, if  $N/p = 100$  represents a dense sample for a single input problem, then  $N/p = 100/10$  is the sample size required for the same sampling density with 10 inputs. Thus in high dimensions all feasible training samples sparsely populate the input space.

We also want to note that some authors label as *incremental*, algorithms that do not learn samples one by one but they learn variables incrementally as they become available. If we see a database as a matrix where rows are samples and columns are variables describing samples, an algorithm could incrementally learn variables (columns) instead of samples (rows). In this way, an incremental algorithm grows a domain model incorporating variables to it as they are available.

## 2.6 Some incremental learning algorithms

The following is a small sample of incremental learning algorithms[2,8,12]:

### 2.6.1 CANDIDATE-ELIMINATION

This algorithm induces binary classifications. Given representation languages for examples and generalizations, both the set  $S$  of maximally specific generalizations and the set  $G$  of maximally general generalizations consistent with the training data are stored.  $G$  is initialized to the most general concept in the space and  $S$  to the first positive training instance.  $S$  keeps generalizing to cover new positive training instances and  $G$  keeps specializing to avoid covering negative training instances. Changes to  $S$  and  $G$  are effected as training instances are presented one at a time.

### 2.6.2 COBWEB

This algorithm induces taxonomies or categorisations. The number of clusters, depth of the hierarchy and category memberships are determined by a global probabilistic metric, called category utility. Any time a new training instance is presented, the

algorithm considers the overall quality of either placing it in an existing category or modifying the current hierarchy to accommodate it (e.g., create a new category, merge categories). Only probabilities are stored and updated.

### 2.6.3 ID5

This algorithm induces decision trees. It is an incremental version of ID3. Instead of building a tree from a batch of training examples, it incrementally updates a tree after each training instance is presented. All transformations are performed by simply maintaining appropriate counters at each node.

### 2.6.4 ILA

This algorithm induces classifications. Note that, if one relaxes slightly the above definition to allow the next hypothesis to depend on the previous one and a small subset of new training examples (rather than a single one), then an iterative implementation of the bayesian framework, where the previous iteration's posterior is the current iteration's prior, would also be suited to incremental learning. In contrast with the above, most well-known learning algorithms, such as ID3, CN2 and backpropagation, are not incremental. Essentially, they assume that the training set can be constructed a priori, that examples may be processed more than once and that learning stops once the training set has been duly processed.

## 2.7 Design Issues

[2] This section highlights some of the main issues raised in the design of incremental learners.

### 2.7.1 Ordering Effects

Chronology, or the order in which knowledge is acquired, is an inherent aspect of incrementality. Thus, it is possible to integrate time implicitly as a factor and another source of bias in learning. For example, the system could choose to give precedence to newly acquired knowledge, recognising the possibility that early guesses may be incorrect. Alternatively, a learning system's experience with the world could be appropriately ordered (e.g., general rules before exceptions) so as to increase its efficiency. Although chronology may be used as a bias, it presupposes that the ordering of the data carries some meaning that should be implicitly captured by the learning system. Such is not always the case. Techniques have been proposed to deal with ordering effects in incremental systems, but the question of whether order independence can be achieved (if desired) remains largely open. One can argue that for large (and rich) enough training sets, the effects of ordering become more limited, as the current representation of the system eventually matches reality. In other words, global statistical patterns ultimately override errors or misrepresentation due to local patterns of noise, lack of information, or plain errors. Indeed, incremental learning presupposes redundancy in the data, as is true in human learning. Similar situations have a tendency to reproduce themselves so that general rules (or analogies) can ultimately be drawn from them.

## 2.7.2 Learning Curve

An incremental system may start from scratch and gain knowledge from examples given one at a time over time. As a result, the system experiences a sort of learning curve, where the quality of its predictions improves (possibly slowly) over time. Inherent in this is the fact that the system is not very trustworthy early on. Hence, although the system can make predictions at any time, one must be cautious with how much value is placed on them. Furthermore, it is difficult to determine the point at which the system has learned *enough* to be trusted.

## 2.7.3 Open-world Assumption

Informally, non-incremental learning assumes that the world is closed. That is, the system essentially works under the premise that its experience of the world, i.e., its training set, is the world. Although this can be theoretically convenient, it certainly does not hold in everyday life, where information, however much of it is available, is generally uncertain and incomplete. Consider, for example, the complementary situations of having to accommodate exceptions and recovering from them, as illustrated in the case of learning whether birds fly.

*Accommodating Exceptions.* Suppose that the training sample is made out of common European birds. Then, birds will be predicted to fly and, if the world is assumed closed, this will apply to all birds, even the penguins and ostriches the system may later encounter.

*Recovering from Exceptions.* Suppose that the training sample is rather atypical, consisting of birds from Australia and the South Pole (i.e., mostly ostriches and penguins). Then, birds will be predicted not to fly and, if the world is assumed closed, this will apply again to all birds. In both situations, further experience must be allowed to alter the truth of previously accepted facts, if a solution consistent with the real world is to be found.

Humans use commonsense reasoning to deal with their partial representation of the open world. The need for an open-world assumption is clearly a consequence of incrementality. If all the data relevant to the problem at hand is indeed available a priori, then the world may be assumed closed. Otherwise, there is a need for special learning mechanisms that invalidate portions of knowledge, while not affecting the rest of it. In order to deal with ordering effects and the openworld assumption incremental learners often need to store more information than their non-incremental counterparts.

## 2.7.4 Incremental learning from examples

[41] Hunt, Martin, and Stone (1966) were among the first to study machine concept learning from examples. Their Concept Learning System (CLS) nonincrementally builds decision trees that discriminate observations of different classes. CLS first divides the observations by their values along the *best* descriptive attribute; it uses a primitive frequency measure to determine the attribute whose values are most uniquely associated with different classes. The values of this divisive attribute are used to label arcs from the decision tree root and segregate observations into disjoint subsets. Each



subset is treated as a child of the root, and CLS recursively builds a subtree for each. Decision tree expansion terminates when all observations at a (sub)node are members of the same class.

A new observation is classified by following the labeled arcs that correspond to the observation's values. When a leaf is reached, the class of the observations residing there is asserted as the class of the new observation. If this prediction is correct, the new observation is saved (along with the original set). If erroneous, the tree is reconstructed using previous observations and the misclassified one. This exemplifies a revolutionary approach to learning - a revolution occurs with each misclassification.

There is a sense in which the revolutionary procedure appears to be incremental, since observations are processed serially. However, each misclassification incurs subsequently larger amounts of processing. Hunt et al. explored variants of CLS that restricted the memory of past observations to a constant by randomly replacing prior observations with new ones. Limiting the number of saved observations also limits tree reconstruction costs, but experiments showed that it slowed learning rates as well; that is, more observations were required to form a decision tree that perfectly discriminated the observations.

In a similar vein, Michalski and Larson (1978) investigated the utility of restricting the observations used during learning. The basis of their strategy is AQ (Michalski, 1973), a nonincremental learning from examples system that does not build decision trees, but a *flat* set of logical (i.e., DNF) concept descriptions. Michalski and Larson adapt AQ to behave incrementally by limiting the number of observations used to reconstruct a faulty knowledge base. Unlike CLS, retained observations are not selected randomly, but on the basis of a Euclidean distance-like measure that identifies *good* concept representatives. In addition, incremental AQ limits reconstruction to those portions of the knowledge base (i.e., individual concept descriptions) that led to a misclassification. This *locality* constraint reduces computational costs and sets it further apart from CLS, which constructs the entire knowledge base (i.e., decision tree) after each incorrect prediction.

Reinke and Michalski (1986) carried the locality constraint a step further. Instead of recomputing a complete concept with each misclassification, their GEM system rederives only a small part of the concept. As in AQ, concepts are in DNF; if new observations are inconsistent with a concept, the faults are traced to individual conjunctive terms within the concept's description. Each faulty term is submitted to a generalization procedure along with the observations it currently covers and those that triggered inconsistency. However, unlike AQ, GEM saves and may reuse all previous observations.

Reinke and Michalski empirically compared their method with a nonincremental version of AQ in three domains. Each experiment measured concept description complexity, classification performance, and computational expense. The results tentatively indicate that: (a) the incremental method yields more complex concept descriptions than the nonincremental method, (b) the incrementally formed concept descriptions classify novel observations almost as well as the nonincremental ones, and (c) knowl-

edge base update is less expensive using the incremental method than with the non-incremental method.

CLS, incremental AQ, and GEM differ in the extent to which they repair a faulty knowledge base: the entire knowledge base, entire concept descriptions, or partial concept descriptions, respectively. By reducing the scope of repair, these systems gain computational advantages over their nonincremental counterparts. In addition, each of these systems forms logical concept descriptions (CLS decision trees are tree-structured DNF concepts), and not coincidentally they insist on perfect consistency between the knowledge base and the environment. To maintain flexibility during incremental learning, they retain observations so that inconsistent portions of the knowledge base can be recomputed following each misclassification. Similarly, a system by Michalski (1985) retains observations that are misclassified by a knowledge base of production rules. In contrast, Winston's (1975) well-known system incrementally learns conjunctive concepts without retaining observations. However, Mitchell (1982) and Vere (1980) point out that Winston's system cannot insure consistency between a learned concept and observations.

Schlimmer's (1987a) STAGGER system (also Schlimmer and Granger, 1986a, b) is a recent addition to the line of incremental learning from examples systems. Like these previous systems (except CLS), STAGGER builds a knowledge base of *flat* concept descriptions and makes local knowledge base repairs. However, STAGGER departs in significant ways from these earlier systems. In part, these differences are motivated by the fact that real-world systems must be resistant to *noise* (i.e., incorrectly described observations due to faulty perception). As such, the system does not insist on perfect consistency between the knowledge base and the environment, nor does it make abrupt repairs following each misclassification. Rather, repair is triggered by a variable amount of inconsistency. To implement this strategy, STAGGER represents concepts as a probabilistic summary of important concept subcomponents. Like GEM, it is these components that are subject to repair, but repairs are made conservatively - only after a number of misclassifications indicate that revision is appropriate. Repairs are made by chunking primitive components, an ability that adds new terms to the concept language and improves learning (Schlimmer, 1987b). Even after making a repair though, the revised knowledge competes with the previous representation and is retracted if new observations prove the repair unwarranted. STAGGER does not reuse observations to effect repairs as do earlier systems. Rather, probabilistic information summarizes the training set and guides repair.

Computer experiments demonstrate that probabilistic representations and a conservative revision strategy enables STAGGER to deal effectively with noise. Furthermore, these characteristics enable the system to discern and respond to long-term environmental changes. STAGGER is relatively novel in addressing the problem of tracking environmental drift. Not coincidentally, other systems that address this problem (Hampson Kibler, 1983; Holland, 1975; Langley, 1987a) also rely on the flexibility afforded by probabilistic representations.

A system that appears quite different than STAGGER at a cursory level, but that draws important principles from it, is Schlimmer and Fisher's (1986) ID4. ID4

descends from CLS by way of Quinlan's (1986) ID3, and like CLS, ID4 constructs decision trees. Its control structure is similar to CLS's, but it uses a more sophisticated evaluation measure for selecting the *best* divisive attribute: the *best* attribute maximizes the expected information gained from the attribute's values. Intuitively, this reflects how confidently one can predict an observation's class by knowing an attribute's value.

ID4 is incremental and updates a decision tree with each new observation. At the core of this incremental ability is the observation that the information-theoretic evaluation measure need not be computed directly from the set of observations, but a probabilistic summary of the observations is sufficient (i.e., co-occurrence counts for all attribute-value/class membership combinations). As with STAGGER, the use of a probabilistic representation frees ID4 from saving observations. For initial division at the root, the values of each new observation are used to update the counts necessary for computing the information measure. When a statistically significant comparison between the attributes can be made (based on the chi-square measure), a root attribute is chosen. The new subtrees are not constructed immediately because no observations have been saved. Instead, after each subsequent observation has updated counts at the root, it is routed to the appropriate subtree to update the counts there before being discarded. Subtrees are gradually grown in this manner. Over time a new attribute may come to be preferred at the root of a subtree. In this case the current root is supplanted by the attribute with the superior information gain, and the nodes under the original subtree are discarded and regrown. Subtree repair (versus full-tree reconstruction as in CLS) illustrates the same locality principles as GEM and STAGGER do for flat representations.

Schlimmer and Fisher used formal and empirical methods to characterize ID4 in terms of learning quality and cost. Their general findings are that ID4 converges on decision trees equivalent in quality to ID3's. The cost of updating a decision tree in ID4 is at worst logarithmic in the number of previous observations, while revolutionary application of ID3 (similar to full-memory CLS) requires polynomial time. Finally, ID4 may require more observations to converge on the same decision tree as the revolutionary application of ID3, but the *total work* (number of observations cost per observation) required to reach equivalent trees is considerably less for ID4.

Incremental learning systems exhibit great variety in the strategies they use for balancing the cost/quality tradeoff. CLS is a nonincremental system forced into frequently rebuilding a decision tree from scratch. Update costs are controlled by limiting the number of retained observations, but only at the expense of slowing learning speed. Incremental AQ and GEM lessen costs further by localizing knowledge base repairs, again based on saved observations. Work with AQ is novel in its use of good concept representatives in knowledge base update. Schlimmer's STAGGER localizes knowledge base revision as well, but drops the assumption that perfect consistency is desired or even possible. As a result, repairs are not triggered after each classification. ID4 is a descendant of CLS, but shares STAGGER's assumptions and strategies: probabilistic evidence and conservative knowledge base repair are the main sources of learning robustness.

Now we turn to conceptual clustering, which drops the requirement of a *teacher* that preclassifies observations. Instead, the system must identify useful classes. This new specification has some important implications for performance, knowledge base structure, and learning strategy.

### 2.7.5 Incremental conceptual clustering

[42] Michalski (1980) proposed conceptual clustering as a means of discovering *understandable* patterns in data. However, this definition does not specify a performance task that improves with learning. Fisher (1987b, c) proposes that one such task is the prediction of unobserved properties. As such, Fisher's COBWEB system forms classification trees that are intended to yield *good* prediction along many attributes, rather than optimal prediction along a single *teacher*-defined attribute as in learning from examples. Despite COBWEB's reduced expectations, Fisher (1987a, b) shows that in many cases prediction with a single COBWEB classification tree approximates the accuracy obtained from multiple, special-purpose ID3 decision trees.

An important difference between COBWEB and earlier conceptual clustering systems is that it is incremental - COBWEB integrates an observation into an existing classification tree by classifying the observation along a path of *best* matching nodes. Like ID4, probabilistic summaries of previous observations are stored at each node, but the matching functions and the criteria used for subtree revision differ considerably. COBWEB uses the category utility function (Gluck Corter, 1985) to guide classification and tree formation. Category utility bases its evaluation on all of the observation's attribute-values rather than a single one, making COBWEB a polythetic classifier as opposed to a monothetic classifier (e.g., CLS and ID4). Similarly, COBWEB's subtree revisions are triggered by considering prediction ability over all attributes, but concern for multiple attributes complicates subtree revision. In ID4 a subtree is simply deleted, but in COBWEB a deletion that benefits one attribute may be inappropriate for others. In response, the system identifies points in the tree for cost-effective prediction of individual attributes. These points are marked by normative (Kolodner, 1983) or default values that COBWEB dynamically maintains during incremental clustering.

COBWEB's main contribution to the present discussion is that it maintains a knowledge base that coordinates many prediction tasks, one for each attribute. This is in sharp contrast to learning from examples systems where a knowledge base need only support one task. However, despite COBWEB's greater knowledge base complexity, Fisher (1987c) argues that its tree structure is still too restrictive; more general structures like directed acyclic graphs (DAGs) would yield better prediction. These structures are used by COBWEB's main precursors, UNIMEM (Lebowitz, 1982) and CYRUS (Kolodner, 1983). Regrettably, neither UNIMEM and CYRUS is characterized in terms of prediction accuracy, the increased costs that maintaining a DAG is likely to incur, or the fundamental tradeoff that must exist between these dimensions.

## 2.8 Conclusion

The idea of incremental learning arose from the observation that most part of human learning can be viewed as a gradual process of concept formation or as the human

ability for incorporating knowledge from new experiences into already learned concept structures .The incremental learning approach was firstly motivated as a human capability worth being incorporated into artificial agents. However, nowadays there exist other practical (i.e. industrial) reasons which increase the interest in incremental algorithms. Everyday, firms and companies store millions of new records. For example, banks store millions of transaction records, internet search engines store millions of searches, ... and so on. Batch algorithms are not easily able to process and incorporate to a knowledge base this great amount of continuously incoming instances in a reasonable amount of time and memory space.

# NEURAL NETWORKS

## 3.1 Introduction

TRADITIONALLY, the term Neural Network has been used to refer to a network of biological neurons. In modern usage, the term is often used to refer Artificial Neural Networks (also known as 'connectionist models' or 'parallel distributed processing') are made up of interconnecting artificial neurons designed to model (or mimic) some properties of biological neural networks.

The purpose of this chapter is to give an Overview of Artificial Neural Network as one of successful approaches in machine learning.

## 3.2 Biologically, How Human Brain Learns?

In the human brain, a typical neuron collects signals from others through a host of fine structures called dendrites. The neuron sends out spikes of electrical activity through a long, thin strand known as an axon, which splits into thousands of branches. At the end of each branch, a structure called a synapse converts the activity from the axon into electrical effects that inhibit or excite activity from the axon into electrical effects that inhibit or excite activity in the connected neurons. When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes.[39] Figure[3.1] shows the biological Neuron.

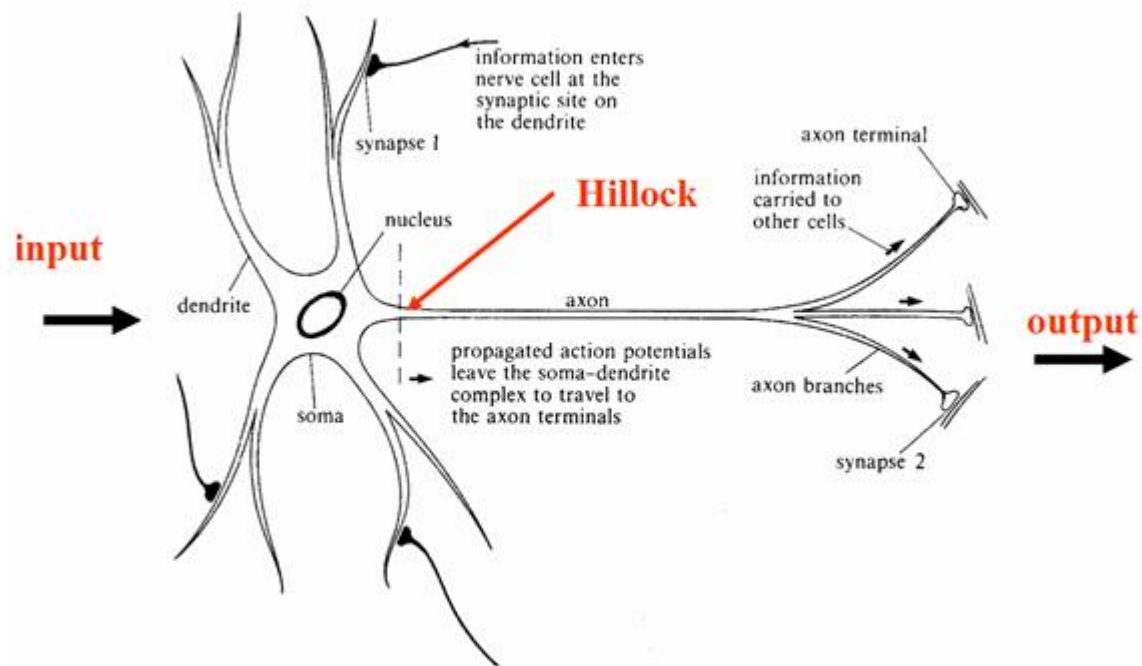


Figure 3.1: Biological Neuron illustration

### 3.3 From Human Neuron to Artificial Neuron

Many definition have proposed to Neural Netowrk in the literature some of them:

**DEFINITION 3** According to Haykin, S. (1994), *Neural Networks: A Comprehensive Foundation*, A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

1. Knowledge is acquired by the network through a learning process.
2. Interneuron connection strengths known as synaptic weights are used to store the knowledge.

**DEFINITION 4** According to the DARPA Neural Network Study (1988, AFCEA International Press, p. 60): A Neural Network is a system composed of many simple processing elements operating in parallel whose function is determined by network structure, connection strengths, and the processing performed at computing elements or nodes.

### 3.4 A simple Artificial neuron

An artificial neuron is a device with many inputs and one output. The neuron has two modes of operation; the training mode and the using mode. In the training mode, the neuron can be trained to fire (or not), for particular input patterns. In the using mode, when a taught input pattern is detected at the input, its associated output

becomes the current output. If the input pattern does not belong in the taught list of input patterns, the firing rule is used to determine whether to fire or not. Follows by Figure[3.2].[36]

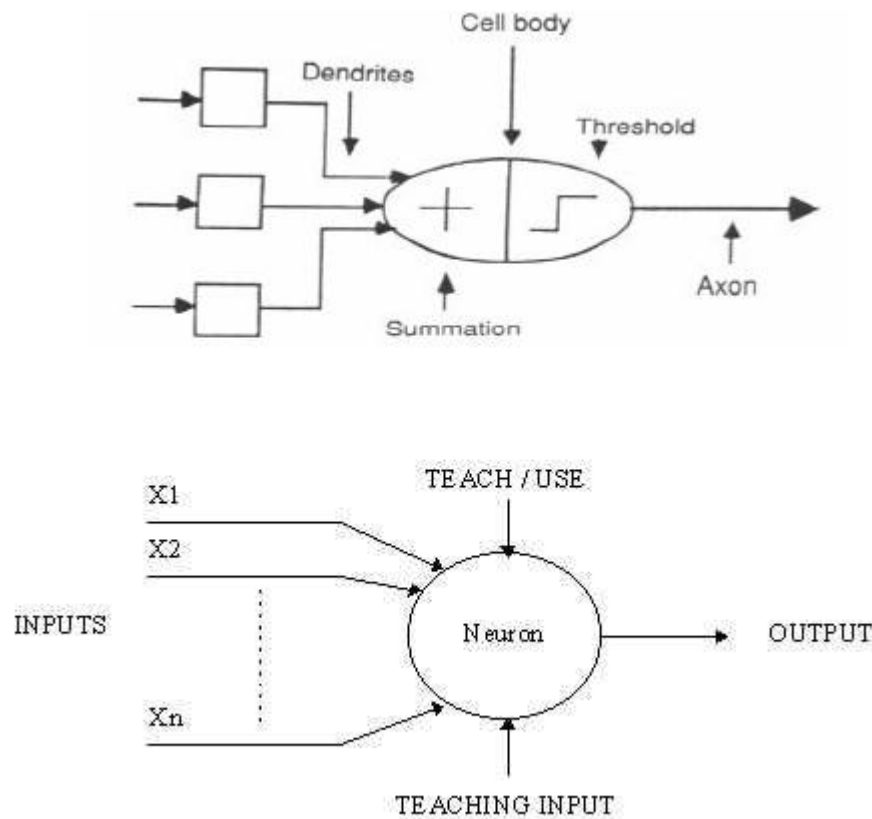


Figure 3.2: Artificial Neuron Model

## 3.5 Why we use Neural Networks

Neural network simulations appear to be a recent development. However, this field was established before the advent of computers, and has survived at least one major setback and several eras. the neural network field enjoys a resurgence of interest and a corresponding increase in funding. [8]

### 3.5.1 Motivations

Initial motivation: Recognition of difference in how computation is done in current technology and in biology.

Current motivations:

- Very effective way to solve certain problems.
- Solve problems in a more human-like way.
- Understand high-level aspects of computation in real neural networks.



- A novel architecture for computers.

### 3.5.2 Artificial Neural networks versus conventional computers

Neural networks take a different approach to problem solving than that of conventional computers. Conventional computers use an algorithmic approach i.e. the computer follows a set of instructions in order to solve a problem. Unless the specific steps that the computer needs to follow are known the computer cannot solve the problem. That restricts the problem solving capability of conventional computers to problems that we already understand and know how to solve. [6,8]

Neural networks process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements (neurons) working in parallel to solve a specific problem. Neural networks learn by example. They cannot be programmed to perform a specific task. The examples must be selected carefully otherwise useful time is wasted or even worse the network might be functioning incorrectly. The disadvantage is that because the network finds out how to solve the problem by itself, its operation can be unpredictable. Table[3.1] describe the difference between the conventional computers and Neural Networks.

CHARACTERISTICS	TRADITIONAL COMPUTING(including Expert Systems)	ARTIFICIAL NEURAL NETWORKS
<b>Processing style Functions</b>	Sequential Logically (left brained) via Rules Concepts Calculations	Parallel Gestalt (right brained) via Images Pictures Controls
<b>Learning Method Applications</b>	by rules (didactically) Accounting word processing math inventory digital communications	by example (Socratically) Sensor processing speech recognition pattern recognition text recognition

Table 3.1: NN vs. conventional computers.

We conclude that Artificial neural networks, parallel distributed processing (PDP) or connectionist architectures developed to solve Classical problems.[7]

***N.B:** Neural networks cannot do anything that cannot be done using traditional computing techniques, BUT they can do some things which would otherwise be very difficult. [8]*

### 3.5.3 The Ability of ANNs

Neural networks are a form of multiprocessor computer system, ANNs have been applied to an increasing number of real-world problems of considerable complexity. Their most important advantage is in solving problems that are too complex for conventional technologies problems that do not have an algorithmic solution or for which an algorithmic solution is too complex to be found. Ann can also help where we can get lots

of examples of the behavior we require or where we need to pick out the structure from existing data. In general, because of their abstraction from the biological brain, ANNs are well suited to problems that people are good at solving, but for which computers are not. These problems include pattern recognition and forecasting (which requires the recognition of trends in data). [14]

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyse. This expert can then be used to provide projections given new situations of interest and answer *what if* questions. Other advantages include:

- Adaptive learning: "generalization ability" An ability to learn how to do tasks based on the data given for training or initial experience.
- Self-Organisation(distributed representation and computation): An ANN can create its own organisation or representation of the information it receives during learning time.
- Real Time Operation(massive parallelism) ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
- Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage. [23]

### 3.6 Representing 'knowledge' in a Neural Network :

A major task for a neural network is to learn a model of the world or the environment in which it is embedded. Knowledge of any domain can be divided into two groups: *prior information*, facts and what has been known, and *observations* (measurements) of the world, usually in a noisy environment. These observations, usually form a observations, referred sometimes to training data set. Each example is an *input-output* pair, an input signal and the corresponding desired response for the neural network. Training a neural network: this involves the execution of a two phased strategy, the learning phase and the generalisation phase. During the learning phase an appropriate architectures (learning algorithm) for the network such that the input, (hidden), and output layers have some synergy with its environment. A subset of examples, both positive and negative, is then used to train the network.

The second phase is an evaluation of a trained neural networks by exposing them to data the networks have seen before. The network is expected to pick regularities in the observations. Thus, unlike classical information processing paradigms, where in a mathematical model environmental observations is formulated and validated with the help of real-world data, and then building the design on the basis of the model.

The design of a neural network is based on real data: The neural network not only provides an implicit model of the environment in which it is embedded, but also performs some information-processing function of interest.

Neural Networks 'learn' by adapting in accordance with a training regimen: The network is subjected to particular information environments on a particular *schedule* to achieve the desired *end-result*[39]

### 3.7 ANN Components

[6,33,39] Artificial Neural Network is characterized by:

- network architecture (topology);
- network node properties;
- connections between the neurons (weights);
- updating (learning) rules for the weights and the states of the neurons.

Figure[3.3] illustrates ANN components.

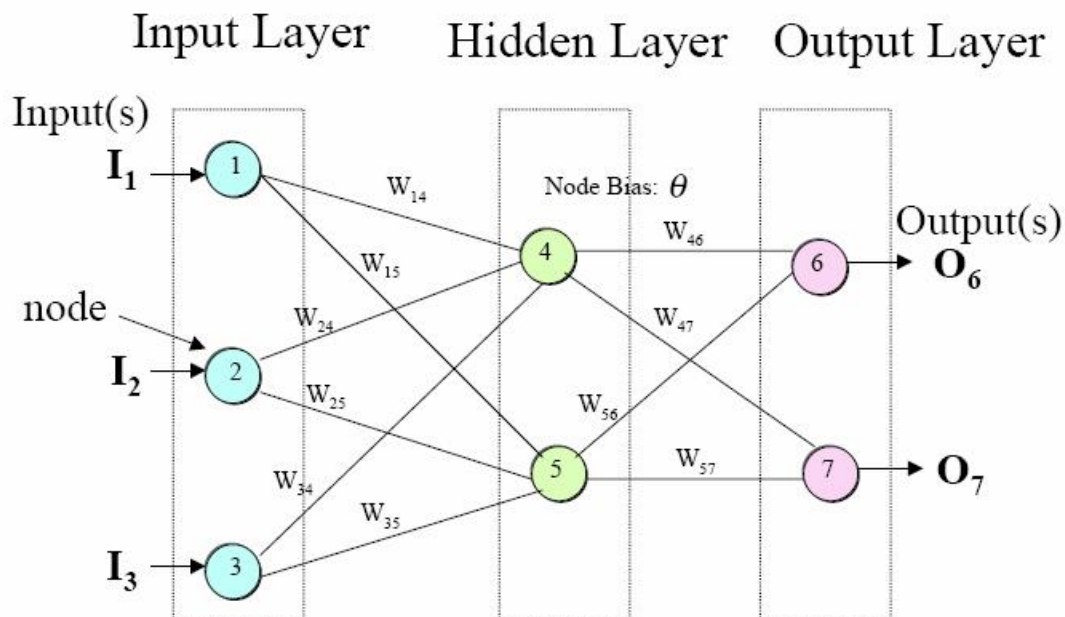


Figure 3.3: Neural Network Components

#### 3.7.1 Input Layer

Introduces input values into the network. No activation function or other processing.

### 3.7.2 Hidden Layer (s)

Traduction phase..Two hidden layers are sufficient to solve any problem.

### 3.7.3 Output layer

Functionally just like the hidden layers. Outputs are passed on to the world outside the neural network.

### 3.7.4 Connections /Arcs

The connections (arcs) are from input nodes to hidden layer nodes and from hidden layer nodes to output nodes. No specific topology.

### 3.7.5 Weights

Each arc is assigned an initial random weight, usually between  $[-0.5..0.5]$ , used in training and may be modified in the learning process. An initial weight (bias) is assigned to nodes in hidden and output layer.

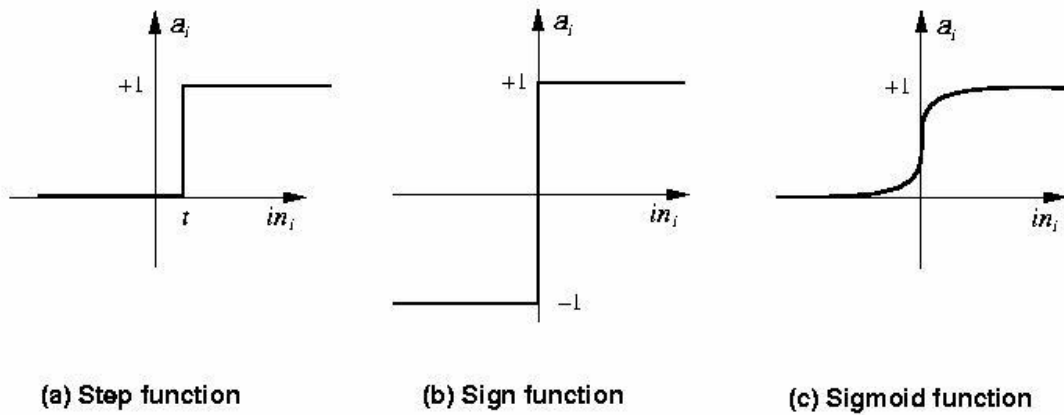
### 3.7.6 Summation Function

The first step in a processing element's operation is to compute the weighted sum of all of the inputs. Mathematically, the inputs and the corresponding weights are vectors which can be represented as  $(i_1, i_2...i_n)$  and  $(w_1, w_2...w_n)$ . The total input signal is the dot, or inner, product of these two vectors. This simplistic summation function is found by multiplying each component of the  $i$  vector by the corresponding component of the  $w$  vector and then adding up all the products.  $input_1 = i_1 * w_1$ ,  $input_2 = i_2 * w_2$ , etc., are added as  $input_1 + input_2 + ... + input_n$ . The result is a single number, not a multi-element vector .

### 3.7.7 Activation (transfer) functions

: Transforms neuron's input into output. The result of the summation function was mentioned above, almost always the weighted sum, is transformed to a working output through an algorithmic process known as the transfer function. Features of activation functions: A squashing effect is required Prevents accelerating growth of activation levels through the network. Some activation functions:

- The hard-limiting threshold function (step function), Corresponds to the biological paradigm, either fires or not
- Sigmoid functions ('S'-shaped curves), The logistic function, The hyperbolic tangent (symmetrical).show Figure[3.4].



$$f(x) = \begin{cases} 0 & \text{if } 0 > x \\ 1 & \text{if } x \geq 0 \end{cases}$$

$$f(x) = \frac{1}{1+e^{-\beta x}}$$

Figure 3.4: Sample Activation functions

### 3.7.8 Bias, offset, threshold

These terms all refer to a constant (i.e., independent of the network input but adapted by the learning rule) term which is input to a unit. They may be used interchangeably, although the latter two terms are often envisaged as a property of the activation function. Furthermore, this external input is usually implemented (and can be written) as a weight from a unit with activation value 1.

## 3.8 Learning Process

One of the most important aspects of Neural Network is the learning process. Learning can be done in supervised or unsupervised manner.

In **supervised training**, both the inputs and the outputs are provided. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are then calculated, causing the system to adjust the weights which control the network. This process occurs over and over as the weights are continually tweaked.

In **unsupervised training**, the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data. This is often referred to as self-organization or adaption.[36]

### 3.8.1 Connectionist Learning Algorithms

[13] Learning involves improvement in performance. ANNs ability to learn from examples makes them attractive and exciting.

In order to understand the learning process, we need a model of the computation according to which the network operates and we must know what information is available to the network. The model of the computation is inductive learning from available data examples.

Connectionist learning typically involves the manipulation of connection weights in a single network of units.

The aim of the learning is to reach a point where the network produces certain types of input/output behaviour. This normally involves systematically updating the weights on possible connections between units.

The network weights are updated by learning rules, which learning rules govern the learning process. A learning algorithm refers to the procedure in which learning rules are used for adjusting the network weights.

In the connectionist learning scenario, the learner is the weight updating procedure and the target representation is the network with a certain configuration of weights.

If the architecture of the network is fixed the hypothesis space is the space of possible weight configurations and a single hypothesis is a particular configuration of weights. If the architecture is not fixed, the hypothesis space is made up of all possible architecture/weight configuration combinations. Figure[3.5]. There are two learning approaches:

- Batch learning- when a large set of examples are processed at once.
- Incremental learning- when the examples are processed one at a time;

Different network architectures require appropriate learning algorithms. There are four basic types of learning algorithms depending on the rules:

- Error-correction rules
- Boltzmann rule
- Hebbian rule
- Competitive rule

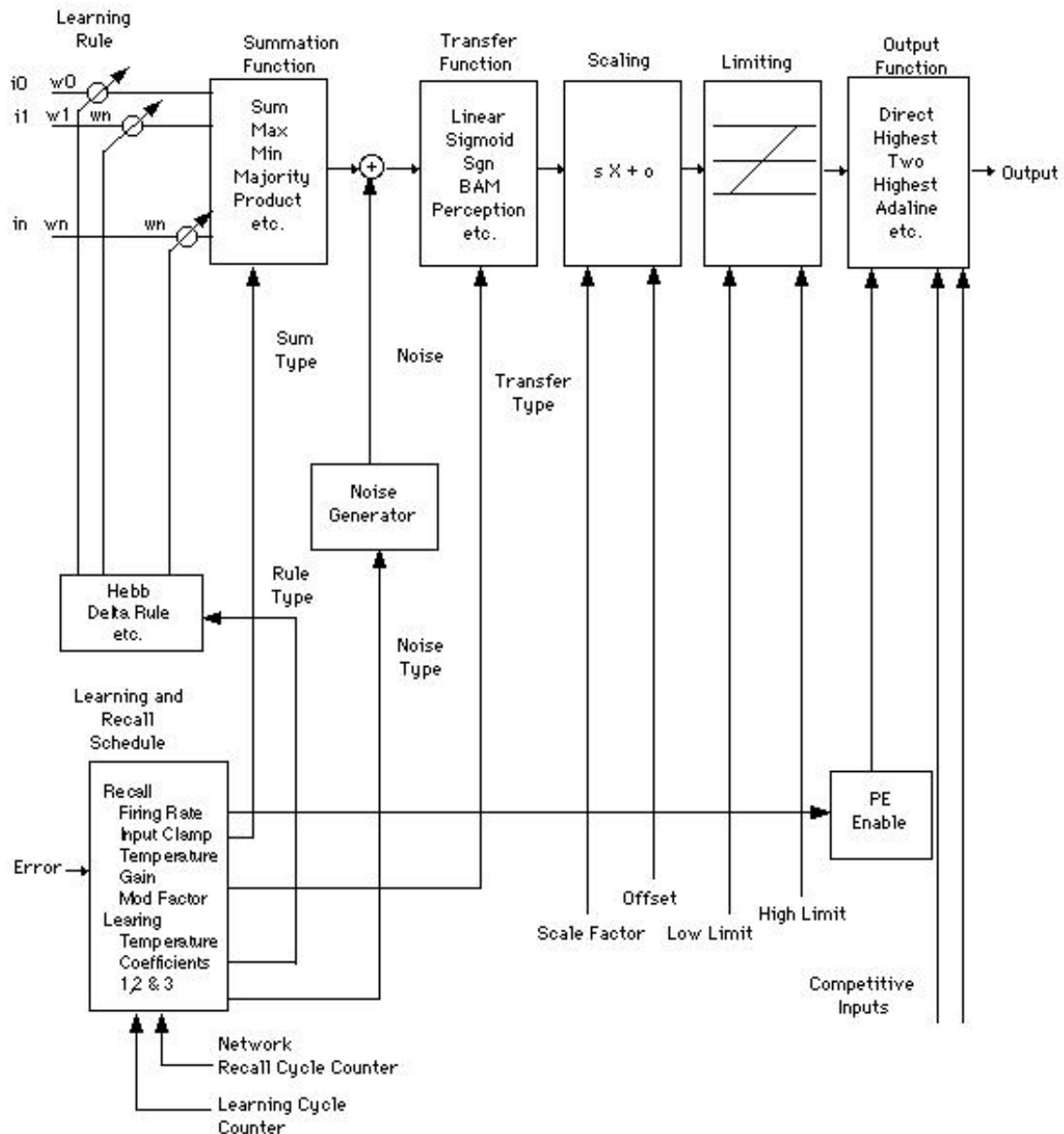


Figure 3.5: Processing Element. This figure is adapted from NeuralWare's simulation model used in NeuralWorks Profession II/Plus

## 3.8.2 Learning Rules

Many learning laws are in common use. Most of these laws are some sort of variation of the best known and oldest learning law, Hebb's Rule. Some researchers have the modeling of biological learning as their main objective. A few of the major laws are presented as examples. [6]

### 3.8.2.1 Hebb's Rule

The first, and undoubtedly the best known, learning rule was introduced by Donald Hebb. basic rule is: If a neuron receives an input from another neuron, and if both

are highly active (mathematically have the same sign), the weight between the neurons should be strengthened.

### 3.8.2.2 Hopfield Law

It is similar to Hebb's rule with the exception that it specifies the magnitude of the strengthening or weakening. It states, "if the desired output and the input are both active or both inactive, increment the connection weight by the learning rate, otherwise decrement the weight by the learning rate."

### 3.8.2.3 The Delta Rule

This rule is a further variation of Hebb's Rule. It is one of the most commonly used. This rule is based on the simple idea of continuously modifying the strengths of the input connections to reduce the difference (the delta) between the desired output value and the actual output of a processing element. This rule changes the synaptic weights in the way that minimizes the mean squared error of the network. This rule is also referred to as the Widrow-Hoff Learning Rule and the Least Mean Square (LMS) Learning Rule.

The way that the Delta Rule works is that the delta error in the output layer is transformed by the derivative of the transfer function and is then used in the previous neural layer to adjust input connection weights. In other words, this error is back-propagated into previous layers one layer at a time. The process of back-propagating the network errors continues until the first layer is reached. The network type called Feedforward, Back-propagation derives its name from this method of computing the error term.

When using the delta rule, it is important to ensure that the input data set is well randomized. Well ordered or structured presentation of the training set can lead to a network which can not converge to the desired accuracy. If that happens, then the network is incapable of learning the problem.

### 3.8.2.4 The Gradient Descent Rule

This rule is similar to the Delta Rule in that the derivative of the transfer function is still used to modify the delta error before it is applied to the connection weights. Here, however, an additional proportional constant tied to the learning rate is appended to the final modifying factor acting upon the weight. This rule is commonly used, even though it converges to a point of stability very slowly.

It has been shown that different learning rates for different layers of a network help the learning process converge faster. In these tests, the learning rates for those layers close to the output were set lower than those layers near the input. This is especially important for applications where the input data is not derived from a strong underlying model.



### 3.8.2.5 Kohonen's Learning Law

This procedure, developed by Teuvo Kohonen, was inspired by learning in biological systems. In this procedure, the processing elements compete for the opportunity to learn, or update their weights. The processing element with the largest output is declared the winner and has the capability of inhibiting its competitors as well as exciting its neighbors. Only the winner is permitted an output, and only the winner plus its neighbors are allowed to adjust their connection weights.

Further, the size of the neighborhood can vary during the training period. The usual paradigm is to start with a larger definition of the neighborhood, and narrow in as the training process proceeds. Because the winning element is defined as the one that has the closest match to the input pattern, Kohonen networks model the distribution of the inputs. This is good for statistical or topological modeling of the data and is sometimes referred to as self-organizing maps or self-organizing topologies.

## 3.9 Types of connectionist Models

[3,34] Artificial Neural Network types can be classified based on following attributes:

- **Input and output (Topology)**
  - Matrix-memory models (Single layer): all units participate potentially in input and output
  - Others (Multilayer): separate input, output, and sometimes hidden units
- **Representations**
  - Localist
  - Distributed: input, output, hidden
  - Multiple units participate in the representation of each concept.
  - Multiple concepts are represented by each unit.
- **Continuous and discrete time**
- **Static and sequential networks**
- **Learning**
  - Supervised
  - Reinforcement
  - Unsupervised
- **Connectivity**
  - Feedforward
  - Partially recurrent
  - Completely recurrent; constraint satisfaction

- **Applications**
  - Classification
  - Clustering
  - Function approximation
  - Prediction

## 3.10 ANN Architectures

[13,39,Wekipidia.com] Based on the connection pattern (architecture) ANNs can be grouped into two categories

### 3.10.1 Feed-forward networks

In which graphs have no loops. Gnerally speaking feed-forward networks are static because they produce only one set of output values rather than a sequence of values from a given input; Taxonomy of Feed-forward:

- Single-layer perceptron
- Multilayer perceptron
- Radial-basis function networks
- Higher-order networks
- Polynomial learning networks

#### 3.10.1.1 Single-layer perceptron

The earliest kind of neural network is a single-layer perceptron network, which consists of a single layer of output nodes; the inputs are fed directly to the outputs via a series of weights. In this way it can be considered the simplest kind of feed-forward network. The sum of the products of the weights and the inputs is calculated in each node, and if the value is above some threshold (typically 0) the neuron fires and takes the activated value (typically 1); otherwise it takes the deactivated value (typically -1). Neurons with this kind of activation function are also called McCulloch-Pitts neurons or threshold neurons. In the literature the term perceptron often refers to networks consisting of just one of these units. Follows,Figure[3.6].

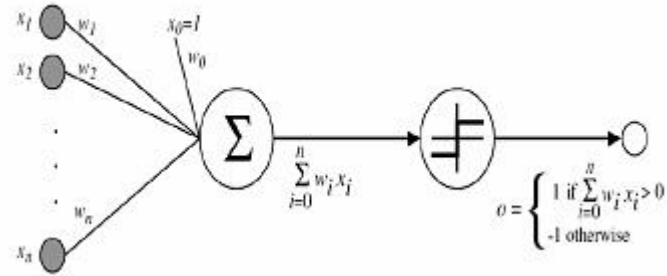


Figure 3.6: perceptron

Perceptrons can be trained by a simple learning algorithm that is usually called the delta rule. It calculates the errors between calculated output and sample output data, and uses this to create an adjustment to the weights, thus implementing a form of gradient descent. Single-unit perceptrons are only capable of learning linearly separable patterns, for that Multi-layer perceptrons have arisen, table () illustrates Different Non-Linearly Separable Problems. Follows, Figure [3.7].

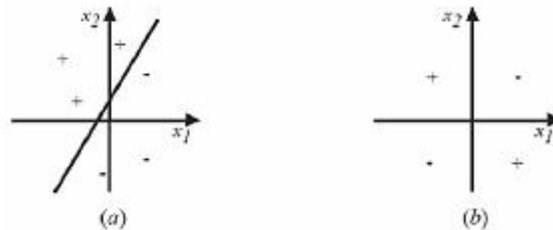


Figure 3.7: (a) Linearly separable (b) Non-Linearly separable

### 3.10.1.2 Multi-layer perceptrons

MLPs: Feed-forward neural network with multiple layers of processing neurons. This class of networks consists of multiple layers of computational units, usually interconnected in a feed-forward way. Each neuron in one layer has directed connections to the neurons of the subsequent layer. In many applications the units of these networks apply a sigmoid function as an activation function. Follows by, Figure [3.8][3.9].

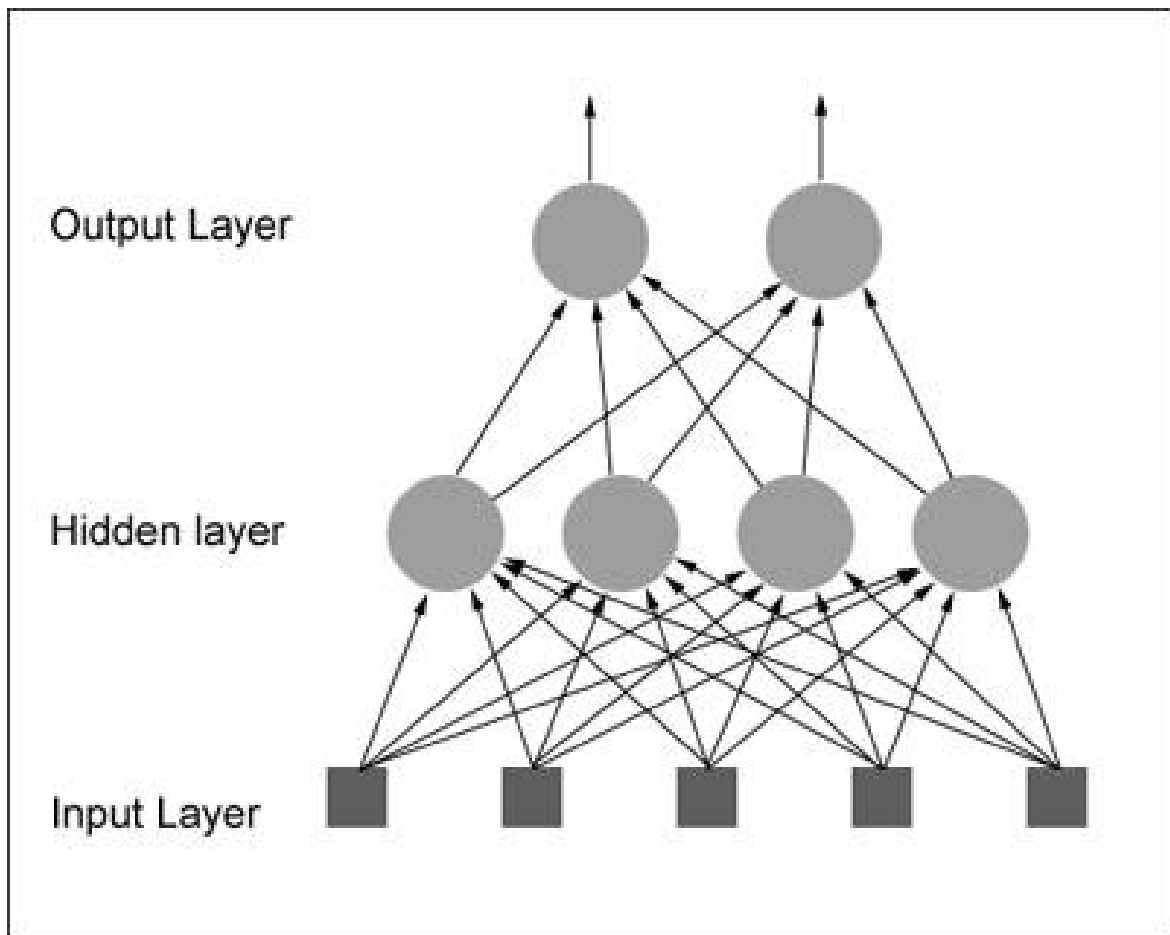


Figure 3.8: Multi-layer perceptrons


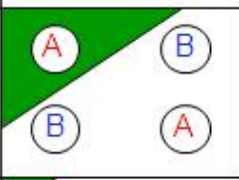
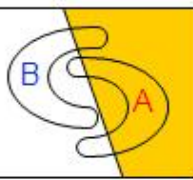
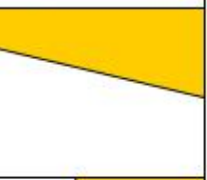
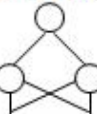
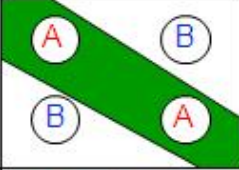
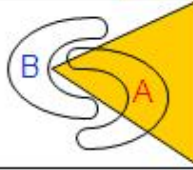
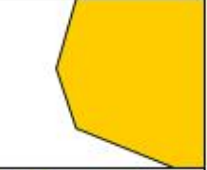

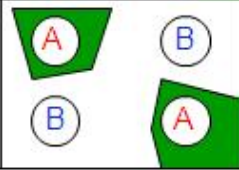

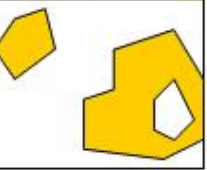
Structure	Types of Decision Regions	Exclusive-OR Problem	Classes with Meshed regions	Most General Region Shapes
Single-Layer 	Half Plane Bounded By Hyperplane			
Two-Layer 	Convex Open Or Closed Regions			
Three-Layer 	Arbitrary (Complexity Limited by No. of Nodes)			

Figure 3.9: Different Non-Linearly Separable Problems

### 3.10.1.3 ADALINE

ADaptive LInear NEuron or later called Adapter Linear Element. It was developed by Widrow and Hoff from Stanford university in 1960. It's based on McCulloch-Pitts model. It consists of a weight, a bias and a summation function.

While the Adaline is through this capable of simple linear regression, it has limited practical use.

There is an extension of the Adaline, called the Multiple Adaline (MADALINE) that consists of two or more adalines serially connected. Figure[3.10]

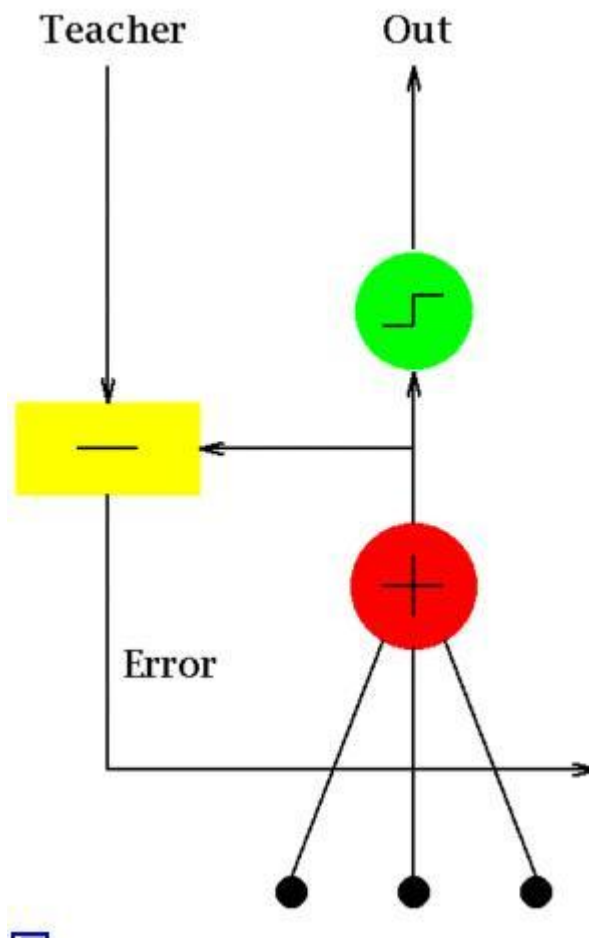


Figure 3.10: ADLINE

### 3.10.1.4 Radial basis function (RBF)

Radial Basis Functions are powerful techniques for interpolation in multidimensional space. A RBF is a function which has built into a distance criterion with respect to a centre. Radial basis functions have been applied in the area of neural networks where they may be used as a replacement for the sigmoidal hidden layer transfer function in multilayer perceptrons. RBF networks have 2 layers of processing: In the first, input is mapped onto each RBF in the 'hidden' layer. The RBF chosen is usually a Gaussian. In regression problems the output layer is then a linear combination of

hidden layer values representing mean predicted output. The interpretation of this output layer value is the same as a regression model in statistics. In classification problems the output layer is typically a sigmoid function of a linear combination of hidden layer values, representing a posterior probability. Performance in both cases is often improved by shrinkage techniques, known as ridge regression in classical statistics and known to correspond to a prior belief in small parameter values (and therefore smooth output functions) in a Bayesian framework.

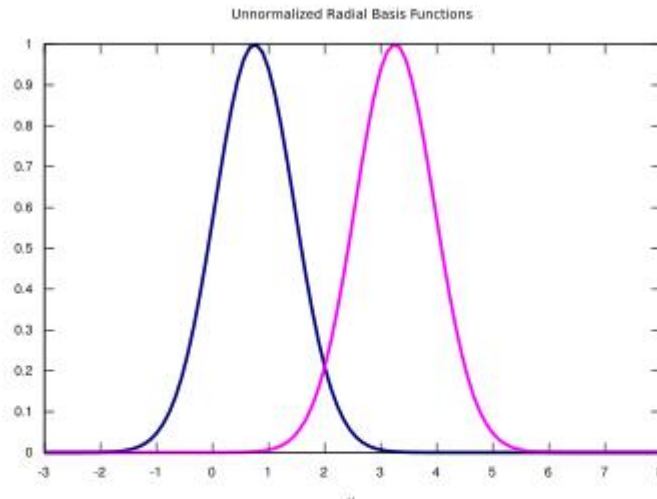


Figure 3.11: Two unnormalized Gaussian radial basis functions in one input dimension. The basis function centers are located at  $c_1=0.75$  and  $c_2=3.25$ .

**RBF types :** Commonly used types of radial basis functions include

- Gaussian.
- Multiquadric.
- Thin plate spline .

### 3.10.2 Recurrent (feedback) networks

In which loops occur because of feedback connections. Recurrent networks are dynamic. Taxonomy of Recurrent network architectures:

- Competitive networks
- Self-organizing maps
- Hopfield networks
- Adaptive-resonance theory models (ART)

### 3.10.2.1 Simple recurrent network

A simple recurrent network (SRN) is a variation on the multi-layer perceptron, sometimes called an "Elman network" due to its invention by Jeff Elman. A three-layer network is used, with the addition of a set of "context units" in the input layer. There are connections from the middle (hidden) layer to these context units fixed with a weight of one. At each time step, the input is propagated in a standard feed-forward fashion, and then a learning rule (usually back-propagation) is applied. The fixed back connections result in the context units always maintaining a copy of the previous values of the hidden units (since they propagate over the connections before the learning rule is applied). Thus the network can maintain a sort of state, allowing it to perform such tasks as sequence-prediction that are beyond the power of a standard multi-layer perceptron.

In a fully recurrent network, every neuron receives inputs from every other neuron in the network. These networks are not arranged in layers. Usually only a subset of the neurons receive external inputs in addition to the inputs from all the other neurons, and another disjunct subset of neurons report their output externally as well as sending it to all the neurons. These distinctive inputs and outputs perform the function of the input and output layers of a feed-forward or simple recurrent network, and also join all the other neurons in the recurrent processing.

### 3.10.2.2 Hopfield network

The Hopfield network is a recurrent neural network in which all connections are symmetric. Invented by John Hopfield in 1982, this network guarantees that its dynamics will converge. If the connections are trained using Hebbian learning then the Hopfield network can perform robust content-addressable memory, robust to connection alteration. Figure[3.12]

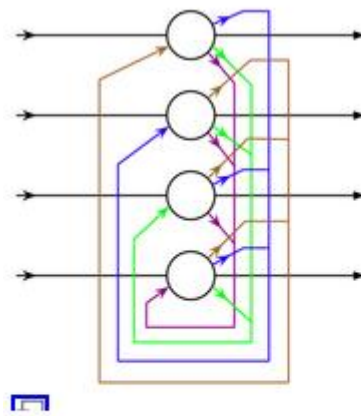


Figure 3.12: Hopfield Network

### 3.10.2.3 Kohonen self-organizing network

The self-organizing map (SOM) invented by Teuvo Kohonen uses a form of unsupervised learning. A set of artificial neurons learn to map points in an input space to coordinates in an output space. The input space can have different dimensions and topology from the output space, and the SOM will attempt to preserve these.

### 3.10.3 Stochastic neural networks

A stochastic neural network differs from a regular neural network in the fact that it introduces random variations into the network. In a probabilistic view of neural networks, such random variations can be viewed as a form of statistical sampling, such as Monte Carlo sampling.

### 3.10.4 Probabilistic neural networks(PNN)

In 1990, Donald F. Specht [5] proposed a method to formulate the weighted-neighbor method in the form of a neural network. He called this a Probabilistic Neural Network, Chapter 5 discusses PNN in more details . Figure[5.8] demonstrates a diagram of a PNN network:

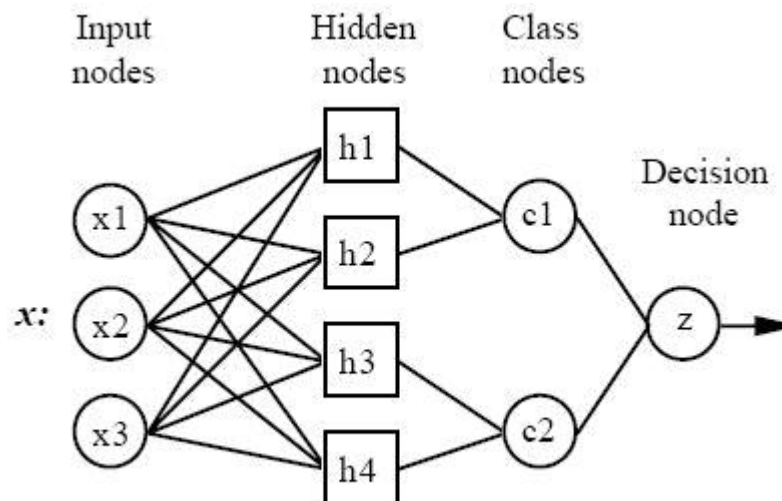


Figure 3.13: PNN Architecture

### 3.10.5 Backpropagation Algorithm

One of the most popular training algorithms is used to train neural network . the output values are compared with the correct answer to compute the value of some predefined error-function. By various techniques the error is then fed back through the network. Using this information, the algorithm adjusts the weights of each connection in order to reduce the value of the error function by some small amount. After repeating this process for a sufficiently large number of training cycles the network will usually converge to some state where the error of the calculations is small. In this case one says that the network has learned a certain target function.



Initialize each  $w_i$  to some small random value  
 Until the termination condition is met, Do  
 For each training example  $\langle (x_1, \dots, x_n), t \rangle$  Do

1. Input the instance  $(x_1, \dots, x_n)$  to the network and compute the network outputs  $o_k$
2. For each output unit  $k$   $\delta_k = o_k(1 - o_k)(t_k - o_k)$
3. For each hidden unit  $h$   $\delta_h = o_h(1 - o_h) \sum_k w_{h,k} \delta_k$
4. For each network weight  $w_{i,j}$  Do  $w_{i,j} = w_{i,j} + \Delta w_{i,j}$  where  $\Delta w_{i,j} = \eta \delta_j x_{i,j}$

### 3.10.6 The Gradient Descent Algorithm

Initialize all weights to small random values. REPEAT until done  $\Delta w_{i,j} := 0$

For each weight  $w_{i,j}$  set

For each data point  $(x, t)^p$

set input units to  $x$

compute value of output units  $\Delta w_{i,j} := \Delta w_{i,j} + (t_i - y_i)y_i$

For each weight  $w_{ij}$  set

For each weight  $w_{ij}$  set  $w_{i,j} := w_{i,j} + u \Delta w_{i,j}$

The algorithm terminates once we are at, or sufficiently near to, the minimum of the error function, where  $G = 0$ . We say then that the algorithm has converged.

## 3.11 Overtraining , how can we avoid it !

The critical issue in developing a neural network is generalization: how well will the network make predictions for cases that are not in the training set? NNs, like other flexible nonlinear estimation methods such as kernel regression and smoothing splines, can suffer from either underfitting or overfitting. A network that is not sufficiently complex can fail to detect fully the signal in a complicated data set, leading to underfitting. A network that is too complex may fit the noise, not just the signal, leading to overfitting. Figure[3.14], Overfitting is especially dangerous because it can easily lead to predictions that are far beyond the range of the training data with many of the common types of NNs. Overfitting can also produce wild predictions in multilayer perceptrons even with noise-free data.

The best way to avoid overfitting is to use lots of training data. If you have at least 30 times as many training cases as there are weights in the network, you are unlikely to suffer from much overfitting, although you may get some slight overfitting no matter how large the training set is. For noise-free data, 5 times as many training cases as weights may be sufficient. But you can't arbitrarily reduce the number of weights for fear of underfitting. [33]

Given a fixed amount of training data, there are at least six approaches to avoiding underfitting and overfitting, and hence getting good generalization:

- Model selection
- Jittering

- Early stopping
- Weight decay
- Bayesian learning
- Combining networks

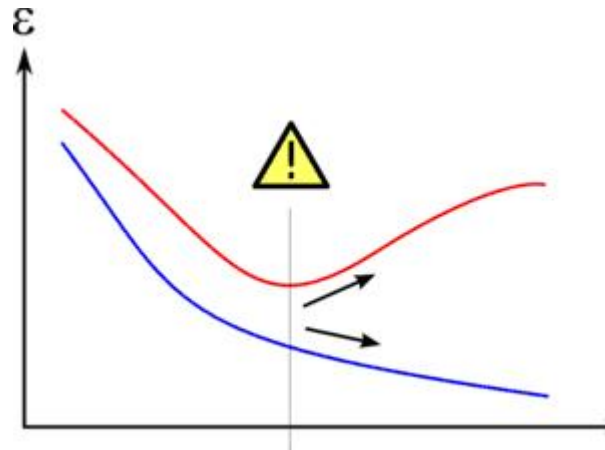


Figure 3.14: Overfitting/Overtraining in supervised learning; Training error is shown in blue, validation error in red. If the validation error increases while the training error steadily decreases then a situation of overfitting may have occurred

## 3.12 Strengths and Weaknesses of Neural Network Models

Philosophers are interested in neural networks because they may provide a new framework for understanding the nature of the mind and its relation to the brain (Rumelhart and McClelland, 1986, Chapter 1). Connectionist models seem particularly well matched to what we know about neurology. The brain is indeed a neural net, formed from massively many units (neurons) and their connections (synapses). Furthermore, several properties of neural network models suggest that connectionism may offer an especially faithful picture of the nature of cognitive processing. Neural networks exhibit robust flexibility in the face of the challenges posed by the real world. Noisy input or destruction of units causes graceful degradation of function. The net's response is still appropriate, though somewhat less accurate. In contrast, noise and loss of circuitry in classical computers typically result in catastrophic failure. Neural networks are also particularly well adapted for problems that require the resolution of many conflicting constraints in parallel. There is ample evidence from research in artificial intelligence that cognitive tasks such as object recognition, planning, and even coordinated motion present problems of this kind. Although classical systems are capable of multiple constraint satisfaction, connectionists argue that neural network models provide much more natural mechanisms for dealing with such problems.

Over the centuries, philosophers have struggled to understand how our concepts are defined. It is now widely acknowledged that trying to characterize ordinary notions

with necessary and sufficient conditions is doomed to failure. Exceptions to almost any proposed definition are always waiting in the wings. For example, one might propose that a tiger is a large black and orange feline. But then what about albino tigers? Philosophers and cognitive psychologists have argued that categories are delimited in more flexible ways, for example via a notion of family resemblance or similarity to a prototype. Connectionist models seem especially well suited to accommodating graded notions of category membership of this kind. Nets can learn to appreciate subtle statistical patterns that would be very hard to express as hard and fast rules. Connectionism promises to explain flexibility and insight found in human intelligence using methods that cannot be easily expressed in the form of exception free principles (Horgan and Tienson, 1989, 1990), thus avoiding the brittleness that arises from standard forms of symbolic representation.

Despite these intriguing features, there are some weaknesses in connectionist models that bear mentioning. First, most neural network research abstracts away from many interesting and possibly important features of the brain. For example, connectionists usually do not attempt to explicitly model the variety of different kinds of brain neurons, nor the effects of neurotransmitters and hormones. Furthermore, it is far from clear that the brain contains the kind of reverse connections that would be needed if the brain were to learn by a process like backpropagation, and the immense number of repetitions needed for such training methods seems far from realistic. Attention to these matters will probably be necessary if convincing connectionist models of human cognitive processing are to be constructed. A more serious objection must also be met. It is widely felt, especially among classicists, that neural networks are not particularly good at the kind of rule based processing that is thought to undergird language, reasoning, and higher forms of thought. [3]

## 3.13 Applications

Neural Network Applications can be grouped in following categories [13]:

### 3.13.1 Clustering

A clustering algorithm explores the similarity between patterns and places similar patterns in a cluster. Best known applications include data compression and data mining.

### 3.13.2 Classification/Pattern recognition

The task of pattern recognition is to assign an input pattern (like handwritten symbol) to one of many classes. This category includes algorithmic implementations such as associative memory.

### 3.13.3 Function approximation

The tasks of function approximation is to find an estimate of the unknown function  $f()$  subject to noise. Various engineering and scientific disciplines require function approximation.

### 3.13.4 Prediction/Dynamical Systems

The task is to forecast some future values of a time-sequenced data. Prediction has a significant impact on decision support systems. Prediction differs from Function approximation by considering time factor. Here the system is dynamic and may produce different results for the same input data based on system state (time).

## 3.14 Conclusion

Artificial Neural Networks have contribute in many effecient ways to solve classical approaches problems. This chapter introduced the importance of ANN , differnt architectures , this chapter also deeply discussed components of ANN and how ANN are learneing and training . some successful applications with ANN also are presented .Finally we recognize to weakness an strength in ANN architecture.

# INCREMENTAL LEARNING BASED ON NEURAL NETWORKS

## 4.1 Introduction

**M**OST neural network learning algorithms (Bishop, 1995) involve the network being trained with all the available data during a single training session, learning all the data concurrently. Once that training is finished, the network acquires no further information. Such concurrent training can make it difficult to update the network if additional information becomes available later, and needs to be incorporated into the neural network's performance.

An incremental learning algorithm gives a system the ability to learn from new information as it becomes available. Incremental learning is particularly important and relevant since in many real world applications the complete set of data is not all available at once, and learning really does need to be an ongoing process (Giraud-Carrier, 2000). A neural network should be able to use any new training data to improve its performance, without requiring access to the previous data. This could involve the network having to accommodate new classes of data that are introduced with the new data[14].

Many Neural learning algorithms are not incremental. One of the greatest impediments in building large, scalable learning systems based on neural networks is that when a network trained to solve task A is subsequently trained to solve task B, it "forgets" the solution to task A.[3]

## 4.2 Assessing The quality of a Neural Network solution

There are three factors that affect the quality of a neural network solution:

*Success achieved* on test data indicates how well the network generalizes to data unseen during training which one wants to maximize. This generally is taken as the only performance criterion.

**Network complexity** by itself can be very difficult to assess but two important factors are network size and processing complexity of each unit. Network size gives the memory required which is the product of the number of connections and the number of bits required to store each connection weight. Processing complexity depends on how costly it is to implement processing occurring in each unit, e.g., sigmoid vs. threshold nonlinearity, fanin, fanout properties, precision in storage and computation, etc. This has a negative effect on the quality as one prefers smaller and cheaper networks.

**Learning time** is the time required to learn the given training data till one gets a reasonable amount of performance. This is to be minimized also. [7]

### 4.3 Batch vs. Incremental Learning

Batch learning proceeds as follows: [32]

- Initialize the weights.
- Repeat the following steps:
- Process all the training data.
- Update the weights.

Incremental learning proceeds as follows:

- Initialize the weights.
- Repeat the following steps:
- Process one training case.
- Update the weights.

### 4.4 learning algorithm criteria

- It should be able to learn additional information from new data.
- It should not require access to the original data, used to train the existing classifier.
- It should preserve previously acquired knowledge (that is, it should not suffer from catastrophic forgetting).
- It should be able to accommodate new classes that may be introduced with new data.[27]

## 4.5 Incremental Neural learning framework

In general , we can adopt Figure[4.1] as a kind of processing incremental learning.[13]

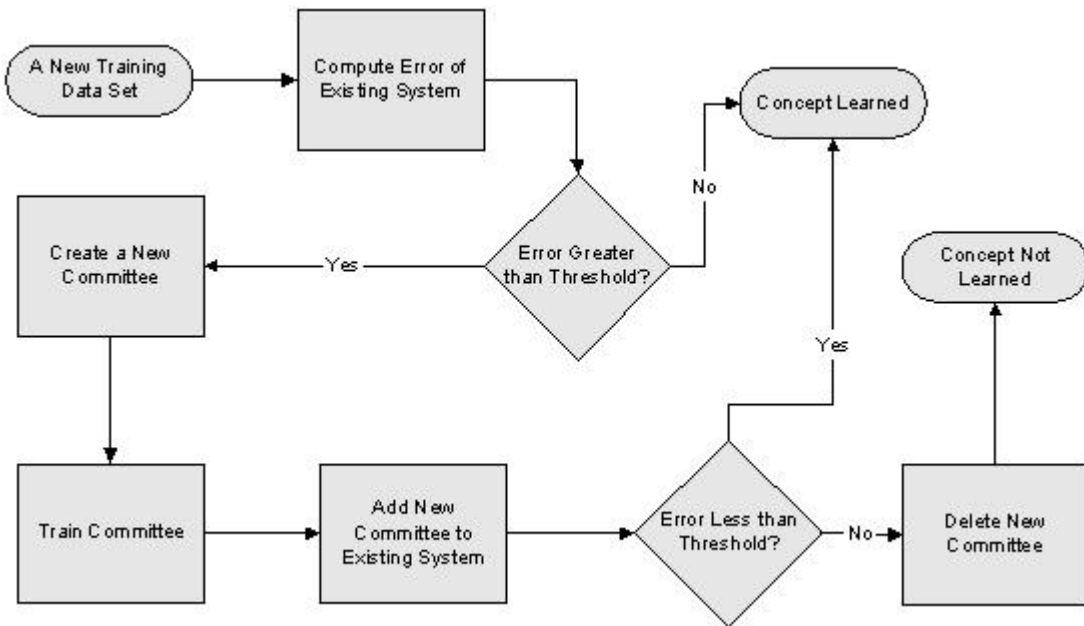


Figure 4.1: INL: an Incremental Neural Learning framework

## 4.6 Catastrophic Interference

Disruption in neural networks, called "forgetting" or "catastrophic interference", often occurs during incremental learning. It is caused by the excessive adaptation of connection weights to new data. One way of overcoming this problem is that only representative training data are kept in memory and some of them are trained with newly given training data.

The problem of catastrophic interference in connectionist networks has been known and studied since the early 1990's. The problem is of particular importance because sequential learning of the kind done by humans cannot be achieved unless a solution is found to this problem. In other words, network models of cognition must, as Grossberg has stressed, be sensitive to new input but not so sensitive that the new input destroys previously learned information. Certain types of patterns, such as those found in highly structured domains, are less susceptible to catastrophic interference than patterns from less well structured domains. Nature seems to have evolved a way of keeping new learning (hippocampal learning) at arms' length from previously learned information stored in the neo-cortex (neo-cortical consolidation), thus physically preventing new learning from interfering with previously learned information. Connectionist models have been developed that simulate this cerebral separation. This may not be - in fact, is certainly not - the only way to route to solving the problem of catastrophic interference, but its close relationship with the way in which the brain may have solved the problem, makes further exploration of these dual-memory models of particular interest. Catastrophic interference is the "stability-plasticity dilemma [S. Grossberg, "Nonlinear neural networks: principles, mechanisms and architectures," *Neural Netw.*, vol. 1, no. 1, pp.

17-61, 1988]" problem in spades. It occurs when a network has learned to recognize a particular set of patterns and then is called upon to learn a new set of patterns. The learning of the new patterns modifies the weights of the network in such a way that the originally learned set of patterns is forgotten. In other words, The dilemma points out the fact that a completely stable classifier will preserve existing knowledge, but will not accommodate any new information, whereas a completely plastic classifier will learn new information but will not conserve prior knowledge , that's mean the newly learned patterns suddenly and completely - "catastrophically" - erase the network's memory of the previously learned patterns. To avoid the catastrophic interference, several approaches have been proposed so far will be mention later on this chapter.[26,36]

## 4.7 Incremental Learning and Neural Network:

The phrase "incremental learning" has been used to refer in the literature to as diverse concepts as incremental network growing and pruning of classifier architectures demonstrated in fig [4.2] , on-line learning to selection of most informative training samples , In other cases, some form of controlled modification of classifier weights has been suggested, typically by retraining with misclassified signals . These algorithms are capable of learning new information. however, they do not simultaneously satisfy all of the above-mentioned criteria for incremental learning: they either require access to old data, forget prior knowledge along the way, or unable to accommodate new classes . various other terms, such as constructive learning, lifelong learning, and evolutionary learning have also been used to imply learning new information.[28]

Researchers have presented various incremental learning methods for neural networks to overcome Catastrophic interference . These methods can be divided into two groups according to the way the old memories are kept in the neural network. One group makes the neural network relearn previously input patterns, while the second group makes the network restrict the modification of predetermined parameters.From other viewpoint , They are roughly categorized into three approaches. In the first approach, the connection weights trained formerly are not modified with a new training sample as much as possible; that is, the connection weights adapted for a new sample are separated from those adapted for previous samples . In the second approach, neural networks with spatially localized basis functions are adopted. As such a basis function, tile-like receptive fields and radial-basis functions are often used. In the third approach, some past training samples as well as a new sample are simultaneously trained to suppress the interference. That is to say, all (or a part) of training samples are accumulated in memory, and they are utilized for learning at every step. This approach is called memory-based learning, and Locally Weighted Regression is one of the successful examples. Although this approach is very useful, the problem is that large memory capacity is often needed to store past training samples. To alleviate this problem, another approach has been proposed in which representative samples are selected to keep and some of them are trained with a current training sample [17]

Int the context of selection of most informative training samples The other difficulty in classification problems lies in the uncertainty of data distribution; that is, we cannot infer what training samples will appear in future. Hence, it is almost impossible



to select essential features only from an initial training dataset. To solve this problem, we can select appropriate features on-line based on the property of data streams. In this context, there have been proposed several learning algorithms for on-line feature selection such as Incremental Principal Component Analysis (IPCA) and Incremental Linear Discriminant Analysis (ILDA). As for IPCA, the approaches are divided into two large groups: covariance-free approach and an approach to updating eigenvalue decomposition problems. In the first approach, several principal components are estimated without keeping a covariance matrix through iterative computations. Although this approach is very efficient in the memory costs, they often suffer from convergence problems especially when high-dimensional inputs are given. To alleviate this problem, Weng et al. have proposed a new covariance-free IPCA method which has good convergence properties based on an efficient estimate [22,7]. In the first approach, however, we can obtain principal components one by one; hence, the approximation error for high-order components could be large depending on initial conditions. On the other hand, in the second approach, we can obtain eigenvectors and eigenvalues more correctly because an eigenvalue problem to be solved is successively updated, and then it is exactly solved every time a new sample is given [Fast incremental learning]

The idea of incremental learning implies starting from the simplest possible network and adding units and/or connections whenever necessary to decrease error (Alpaydin, 1990a). To be able to decrease network size and increase generalization ability, one also wants to be able to get rid of units/connections whose absence will not degrade significantly system's performance. In both cases, as opposed to a static network structure, small modifications to a dynamic network structure during learning is envisaged. Determination of the network structure and computation of connection weights are not done separately but together, both by the learning algorithm. Approaches given in the connectionist literature leading to network structure modification can be divided into two classes. There are those that start with a big network and eliminate the unnecessary and there are others that start from small and add whatever is necessary. Note that there are also incremental unsupervised learning algorithms like ART (Carpenter Grossberg, 1987) and GAR (Alpaydin, 1990a). In unsupervised incremental learning, one adds a new cluster index whenever the current input is not similar to any of the existing clusters. The similarity measure is thus done in the input space regardless of the class to which the input patterns belong. [8]

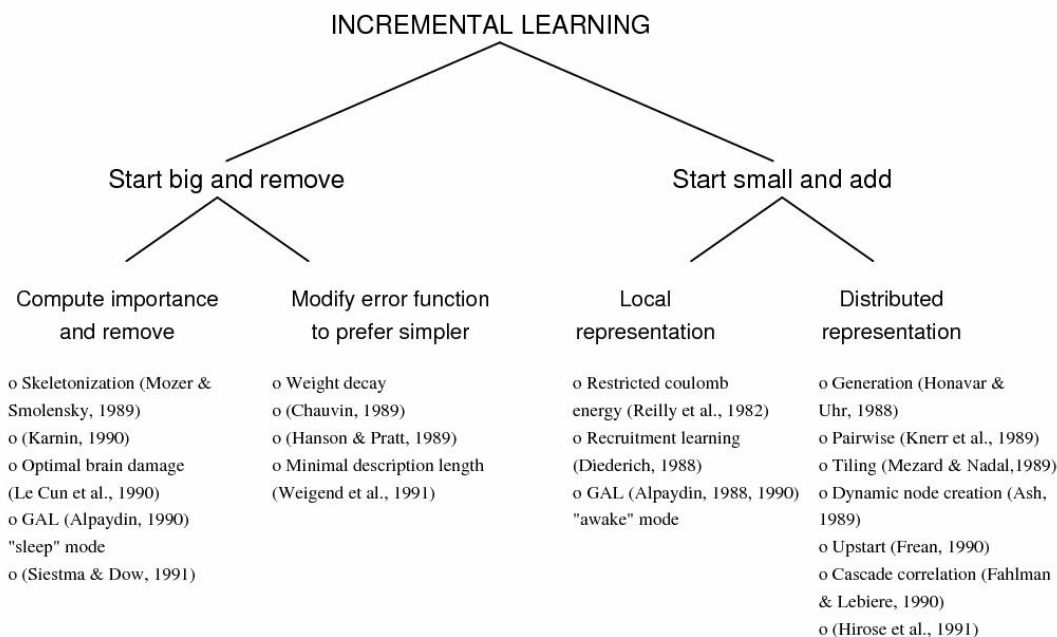


Figure 1. Taxonomy of incremental learning.

Figure 4.2: Taxonomy of Incremental Neural Learning framework

## 4.8 Start big and remove

In the context of polynomial curve fitting the “start big and remove” approach implies starting from a high order polynomial and eliminating those high order terms which do not contribute significantly to success. Such methods are also called pruning or destructive. If one starts with a large network and if the problem in fact requires a simpler network, one likes to have the weights of all unnecessary connections and the output of all unnecessary units equal to zero. There are two approaches in achieving this:

[1] One may explicitly try to compute how important is the existence of a connection/unit in keeping the error low after the network has been trained and a number of the least important may then be deleted. The remaining network needs to continue to be trained. In the ideal case, understanding the importance of a connection/unit requires training two networks one with the connection/unit and one without. As this is not practical for large networks, heuristical approaches have been proposed with the backpropagation algorithm where the sensitivity of the error function to the elimination of a connection/unit is estimated.

[2] Instead of approximating how much the error will change if the unit/connection is eliminated, one may also modify the learning algorithm so that after training, the unnecessary connections/units will have zero weight/output.

## 4.9 Start small and add

The other approach in dynamic modification of network structure during learning, which can be named “start small and add,” implies starting from a simple network and adding units and/or connections to decrease error. These methods are also called growth or constructive. In the context of curve fitting, it implies starting with a low order polynomial and adding higher order terms whenever the polynomial of current order cannot give a good fit for any set of coefficients. Note that this cannot be done in a straightforward manner especially in networks where associations are distributed over a number of shared connections; the whole training should be redone in such a case. One needs a certain mechanism whereby addition of a new unit improves success instead of corrupting the harmony as one would normally expect. There are two possibilities:

[1] If one can make sure that when the new unit gets activated, none of the ancient units get activated, there will be no problem. The units should thus somehow be able to suppress other units when they get control. This implies a competitive strategy and a local representation.

[2] Another possibility is to divide the network into separately trained subnetworks where such subnetworks can be added in an incremental manner. One approach is to have subnets that have competition between subnets, another is to have each subnet as another hidden layer, it is repeated in a recursive manner to lead to a binary tree which can then be “squashed” into one hidden layer. In the “cascade correlation” algorithm (Fahlman Lebiere, 1990), if the required mapping cannot be learned by one layer, a hidden unit is added and trained while the previously trained

## 4.10 Overview of Incremental Neural Network Algorithms :

" The first incremental neural learning algorithm is the Restricted Coulomb Energy (RCE) model (Reilly et al., 1982) which is an incremental version of Parzen windows. Associated with each unit is a number of prototypes where a prototype gets activated only if the input falls into its domination region, determined by a distance computation followed by a thresholding. If an input does not activate any prototype, a new prototype unit is created at that position with an initially large domination region. Prototypes that get activated for inputs that belong to different classes are penalized by having their regions decreased which is done by modifying the threshold. The input space is thus divided into zones dominated by prototype units. A number of sweeps is necessary to finetune the thresholds where units closer to class boundaries have small zones and units interior have larger domination zones.

Recruitment learning (Diederich, 1988) is used in the case of structured connectionist networks where a previously free unit is committed to represent a new concept and required connections built up dynamically (Feldman, 1982). This is a oneshot learning algorithm, i.e., one iteration is sufficient to learn a new concept.

In the first version of GrowandLearn (Alpaydin, 1988), weights in a single layer were learned by Hebbian learning at one shot. However if an association could not be learned or if addition of this association corrupted the previously learned associations, a new hidden unit was added with input weights equal to the input vector. The output weight was computed in such a manner to compensate for the effect of the input layer and thus impose any output. The problem was that as Hebbian learning was used, orthogonality of input patterns were necessary and as this is rarely the case, many units were allocated. However Hebbian learning made the algorithm a oneshot learning one. GrowandLearn (GAL) algorithm (Alpaydin, 1990a, 1990b), uses also a local representation by having a number of exemplars associated with each class. It learns at oneshot but orthogonality of patterns is no longer required.

The “generation” method proposed by Honavar and Uhr (1988) enables a “recognition cone” to modify its own topology by growing links and recruiting units whenever performance ceases to improve during learning by weight adjustment using back-propagation.

The “stepwise procedure” uses subnets of different conceptual interpretations (Knerr et al., 1989). In this method, one first trains a one layer network with the Perceptron learning algorithm assuming that classes are linearly separable. For a class where this is not satisfied, one adds a subnet to separate classes in a pairwise manner. For cases where this does not work either, one performs a piecewise approximation of boundaries using logical functions by additional subnets. As linear separability is rarely the case, one generally is obliged to separate classes in a pairwise manner two by two. The major drawback of this is that the number of hidden units increase exponentially with the number of class units.

Another approach named the “tiling” algorithm adds a new hidden layer whenever the required mapping cannot be done with the existing network (M'ezard Nadal, 1989; explained also in Hertz et al., 1991). There is a “master” unit which is trained to be the output unit by the pocket algorithm—a variant of the Perceptron learning algorithm. If this unit cannot learn all the required associations, additional “ancillary” units are added to learn the rest and another layer is created with a master unit and learning proceeds till the master unit can learn to behave like the output unit.

The “dynamic node creation” method (Ash, 1989; explained also in M“uller Reinhardt, 1990) trains networks with one hidden layer only. Given a certain net that is being trained, if the rate of decrease of error falls down a certain value, a new hidden unit is added and training is resumed when all connections are continued to be modified.

The “upstart” algorithm (Frean, 1990) uses binary units. Like the “tiling” algorithm, first one unit is trained to learn the required associations using the pocket algorithm. If this is not successful, “daughter” units are created to correct the output of this “parent” unit, for “wrongly on” and “wrongly off” cases. This weights are “frozen.” If this does not work either, another hidden unit is added as another hidden layer and so on. A hidden layer has only one hidden unit but connections skip layers, i.e., a unit has connections to all the following layers.

Method proposed by (Hirose et al., 1991) is quite similar to that proposed by Ash (1989), namely, using a network with only one hidden layer, if the rate of decrease for error becomes small, additional hidden units are added. Their contribution is that, once the network converges, the most recently added hidden unit is removed and the network is checked to determine whether the same function can be achieved by fewer hidden units. If the network cannot converge when a hidden unit is removed, the last network that converged is chosen as the final network.[7,26,23]

#### 4.10.1 Evolving Neural Networks:

The idea of applying the basic principles and ideas of natural evolution to optimize the performance of neural network systems is now widely used (e.g. Yao, 1999; Bullinaria, 2003). A distinctive feature of evolving neural networks is their adaptability to a dynamic environment, their ability to change their architecture and learning rules appropriately with limited or no human intervention. Results obtained from evolving neural networks have been reported to be significantly better than traditional hand built neural networks (Yao 1999, Bullinaria 2003).

The standard approach is to start the evolutionary process with an initial population of randomly created artificial genotypes, each encoding some or all of the free parameters of a neural network, or the initial values of an adaptable parameters (such as connection weights). Each network is then trained and evaluated to determine its performance on the task at hand, and the fittest networks are allowed to reproduce by generating copies of their genotypes, with changes introduced by genetic operators such as crossover and mutations. This process is repeated for a number of generations until a network, or group of networks, that best satisfy the performance criteria is obtained. The major difficulties are to determine: which innate parameters to include in the genotype and how to represent them, how exactly to specify the fitness and choose the parents, and what are the most appropriate cross-over and mutation operator.[31]

#### 4.10.2 Resource Allocating Network with Long-Term Memory:

The learning algorithm of RAN-LTM based on the linear method is divided into two phases: the dynamic allocation of hidden units (i.e., the selection of RBF centers in an incremental fashion) and the calculation of connection weights between hidden and output units. The procedure in the former phase is almost the same as that in the original RAN [5], except that hidden units can be added after the update of connection weights (see Step 5 in the learning algorithm) and that memory items are generated at the same time. Once hidden units are allocated, the centers are fixed afterwards. Therefore, the connection weights are only parameters that are updated based on the output errors. To minimize the errors based on the least squares method. [14]

#### 4.10.3 Evolving connectionist systems

Evolving connectionist system (ECoS) are systems that evolve their structure and functionality over time through interaction with the environment .

Evolving Growing Cluster Classifier (EGCC): The EGCC is a knowledge-based

neural network model for classification and is modification of the Radial Basis Function (RBF) type networks. EGCC is similar to the Evolving Classifier Function (ECF) network . The implementation of the concept of "growing" in EGCC is different from that of other growing neural networks, such as growing neural gas (GNG) and growing cell structures (GCS). EGCC has simple training and test procedures that require only two pass training (no further iterations are needed). There is no need for a parameter setting. A cluster center (CC) grows gradually (according to a predefined growth speed) till its influence field reaches the maximum influence field or the influence field of cluster center from a different class. The main factor that affects the speed of the training process of EGCC is the growing speed of the CCs that can be set by users. A CC is learned by a neuron.[21]

#### **4.10.4 Sleep and Awake methods**

In [16] presented two types of incremental learning method designed to achieve quick adaptation with low resources. One approach is to use a sleep phase to provide time for learning. The other one involves a "meta-learning module" that acquires learning skills through experience. The system carries out "reactive modification" of parameters not only to memorize new instances, but also to avoid forgetting old memories using a meta-learning module.

#### **4.10.5 The ART (adaptive resonance theory) net**

If the current instance does not match the associated weight vector of any stored category neuron to a predefined degree, then a new category neuron is allocated for that instance (neuron generation).[10]

#### **4.10.6 The PNN (probilistic neural network)**

A new hidden unit is added for each new instance (neuron generation). [5]

#### **4.10.7 The cascade correlation net**

A new hidden unit is added to the net if its output error exceeds a predefined level (neuron generation). The new hidden unit receives trainable input connections from all the input units and from all preexisting hidden units, and is connected to the output units. For each new hidden unit, the magnitude.[22]

#### **4.10.8 Incremental backpropagation network**

In [17] presented an incremental learning method for pattern recognition, called the "incremental backpropagation learning network," which employs bounded weight modification and structural adaptation learning rules and applies initial knowledge to constrain the learning process. A hidden unit should be added if the neural network cannot accommodate the current instance through weight adaptation (neuron generation). A previously added hidden unit should be deleted if its output weight is decayed to a predefined threshold value (neuron elimination).

### 4.10.9 Learn++

Learn++, an algorithm for incremental training of neural network (NN) pattern classifiers, Learn++ utilizes ensemble of classifiers by generating multiple hypotheses using training data sampled according to carefully tailored distributions.

Learn++ is based on the following intuition: Each new classifier added to the ensemble is trained using a set of examples drawn according to a distribution, which ensures that examples that are misclassified by the current ensemble have a high probability of being sampled. In an incremental learning setting, the examples that have a high probability of error are precisely those that are unknown or that have not yet been used to train the classifier.[27]

## 4.11 Conclusion

An incremental learning algorithm gives a system the ability to learn from new information as it becomes available.

Incremental learning aims to improve network capacity by consuming Spatial And Time components .

The phrase "incremental learning" has been used to refer in the literature to as diverse concepts as incremental network growing and pruning of classifier architectures, to selection of most informative training samples , In other cases, some form of controlled modification of classifier weights has been suggested, typically by retraining with misclassified signals .

# AN INCREMENTAL LEARNING MODEL USING PROBABILISTIC NEURAL NETWORK FOR FACE RECOGNITION

## 5.1 Facial recognition system

### 5.1.1 Introduction

**I**DENTIFYING an individual from his or her face is one of the most nonintrusive modalities in biometrics. However, it is also one of the most challenging ones. A facial recognition system is a computer application for automatically identifying or verifying a person from a digital image or a video frame from a video source. One of the ways to do this is by comparing selected facial features from the image and a facial database. It is typically used in security systems and can be compared to other biometrics such as fingerprint or eye iris recognition systems. Popular recognition algorithms include eigenface, fisherface, the Hidden Markov model, and the neuronal motivated dynamic link matching. A newly emerging trend, claimed to achieve previously unseen accuracies, is three-dimensional face recognition. Another emerging trend uses the visual details of the skin, as captured in standard digital or scanned images. Tests on the FERET database, the widely used industry benchmark, showed that this approach is substantially more reliable than previous algorithms.[11]

### 5.1.2 Notable users and deployments

The London Borough of Newham, in the UK, has a facial recognition system built into their borough-wide CCTV system. The German Federal Police use a facial recognition system to allow voluntary subscribers to pass fully automated border controls at Frankfurt Rhein-Main international airport. Subscribers need to be European Union or Swiss citizens. Griffin Investigations is famous for its recognition system used by casinos to catch card counters and other blacklisted individuals. The Australian Customs Service has an automated border processing system called SmartGate that uses facial recognition. The system compares the face of the individual with the image in the e-passport microchip, certifying that the holder of the passport is the rightful owner. Pennsylvania Justice Network searches crime scene photographs and CCTV footage in the mugshot database of previous arrests. A number of cold cases have been resolved since the system became operational in 2005. Other law enforcement agencies in the USA and abroad use arrest mugshot databases in their forensic investigative work.



U.S. Department of State operates one of the largest face recognition systems in the world with over 75 million photographs that is actively used for visa processing.[11]

### **5.1.3 Early development**

Pioneers of Automated Facial Recognition include: Woody Bledsoe, Helen Chan Wolf, and Charles Bisson.

During 1964 and 1965, Bledsoe, along with Helen Chan and Charles Bisson, worked on using the computer to recognize human faces (Bledsoe 1966a, 1966b; Bledsoe and Chan 1965). He was proud of this work, but because the funding was provided by an unnamed intelligence agency that did not allow much publicity, little of the work was published. Given a large database of images (in effect, a book of mug shots) and a photograph, the problem was to select from the database a small set of records such that one of the image records matched the photograph. The success of the method could be measured in terms of the ratio of the answer list to the number of records in the database. Bledsoe (1966a) described the following difficulties:

This recognition problem is made difficult by the great variability in head rotation and tilt, lighting intensity and angle, facial expression, aging, etc. Some other attempts at facial recognition by machine have allowed for little or no variability in these quantities. Yet the method of correlation (or pattern matching) of unprocessed optical data, which is often used by some researchers, is certain to fail in cases where the variability is great. In particular, the correlation is very low between two pictures of the same person with two different head rotations. -Woody Bledsoe, 1966

This project was labeled man-machine because the human extracted the coordinates of a set of features from the photographs, which were then used by the computer for recognition. Using a graphics tablet (GRAFACON or RAND TABLET), the operator would extract the coordinates of features such as the center of pupils, the inside corner of eyes, the outside corner of eyes, point of widows peak, and so on. From these coordinates, a list of 20 distances, such as width of mouth and width of eyes, pupil to pupil, were computed. These operators could process about 40 pictures an hour. When building the database, the name of the person in the photograph was associated with the list of computed distances and stored in the computer. In the recognition phase, the set of distances was compared with the corresponding distance for each photograph, yielding a distance between the photograph and the database record. The closest records are returned.

This brief description is an oversimplification that fails in general because it is unlikely that any two pictures would match in head rotation, lean, tilt, and scale (distance from the camera). Thus, each set of distances is normalized to represent the face in a frontal orientation. To accomplish this normalization, the program first tries to determine the tilt, the lean, and the rotation. Then, using these angles, the computer undoes the effect of these transformations on the computed distances. To compute these angles, the computer must know the three-dimensional geometry of the head. Because the actual heads were unavailable, Bledsoe (1964) used a standard head derived from measurements on seven heads.

After Bledsoe left PRI in 1966, this work was continued at the Stanford Research Institute, primarily by Peter Hart. In experiments performed on a database of over 2000 photographs, the computer consistently outperformed humans when presented with the same recognition tasks (Bledsoe 1968). Peter Hart (1996) enthusiastically recalled the project with the exclamation, It really worked!

By about 1997, the system developed by Christoph von der Malsburg and graduate students of the University of Bochum in Germany and the University of Southern California in the United States outperformed most systems with those of Massachusetts Institute of Technology and the University of Maryland rated next. The Bochum system was developed through funding by the United States Army Research Laboratory. The software was sold as ZN-Face and used by customers such as Deutsche Bank and operators of airports and other busy locations. The software was robust enough to make identifications from less-than-perfect face views. It can also often see through such impediments to identification as mustaches, beards, changed hair styles and glasses-even sunglasses.[10]

In about January of 2007, image searches were based on the text surrounding a photo, for example, if text nearby mentions the image content. Polar Rose technology can guess from a photograph, in about 1.5 seconds, what any individual may look like in three dimensions, and thought they will ask users to input the names of people they recognize in photos online to help build a database.

#### 5.1.4 General Framework

In most cases, a face recognition algorithm can be divided into the following functional modules: a face image detector finds the locations of human faces from a normal picture against simple or complex background, and a face recognizer determines who this person is. Both the face detector and the face recognizer follow the same framework; they both have a feature extractor that transforms the pixels of the facial image into a useful vector representation, and a pattern recognizer that searches the database to find the best match to the incoming face image. The difference between the two is the following; in the face detection scenario, the pattern recognizer categorizes the incoming feature vector to one of the two image classes: *face* images and *non-face* images. In the face recognition scenario, on the other hand, the recognizer classifies the feature vector (assuming it is from a *face* image) as Smith's face, Jane's face, or some other person's face that is already registered in the database. [13]

#### 5.1.5 Variations in Facial Images

Face recognition is one of the most difficult problems in the research area of image recognition. A human face is not only a 3-D object, it is also a non-rigid body. Moreover, facial images are often taken under natural environment. That is, the image background could be very complex and the illumination condition could be drastic. Figure 2 is an example of an image with a complex background.

The variations in facial images could be categorized as follows:

- Camera distortion and noise
- Complex background
- Illumination
- Translation, rotation, scaling, and occlusion
- Facial expression
- Makeup and hair style

Camera distortion and noise are standard variations in image recognition problems. Previous researchers have developed numerous tools to increase the signal-to-noise ratio. To deal with complex image background, the recognizer requires a good face detector to isolate the real faces from other parts of the image. Illumination is often a major factor in the obstruction of the recognition process. To alleviate the influence of the illumination effect, people may take conventional image enhancement techniques (dynamic thresholding, histogram equalization), or train a neural network for feature extraction (Brunelli, 1993)(Lin, 1997).

Translation, scaling, and rotational variations should also be dealt with in the face detection phase. Among the three variations, translation is the easiest one to solve. A simple windowing approach can do the job. Scaling problem (different face sizes) is also easy to solve if we create an image pyramid to represent the input image (image pyramid is a collection of the same image with different resolutions). Rotation along the Z axis (the axis that is perpendicular to the image plane) is harder. A brute force solution is time-consuming.

Another way to deal with facial expression changes is, instead of using the whole facial area to perform recognition task, using only the significant facial region. The significant facial region is a square area close to the center of the human face. It contains both eyes and the nose, but excludes the mouth and ears. Study shows that facial expressions and hair style changes have less influence on the significant facial region, and yet the face is still recognizable by viewing only the significant facial region.[13]

### **5.1.6 Comparative study**

Among the different biometric techniques facial recognition may not be the most reliable and efficient but its great advantage is that it does not require aid from the test subject. Properly designed systems installed in airports, multiplexes, and other public places can detect presence of criminals among the crowd. Other biometrics like fingerprints, iris, and speech recognition cannot perform this kind of mass scanning. However, questions have been raised on the effectiveness of facial recognition software in cases of railway and airport security.

### 5.1.7 Incremental Learning vs. Batch Learning in the context of Face recognition:

Learning methods fall into two categories, batch and incremental. A batch learning method requires that all the training face images are available at a fixed training time. It is difficult to determine a priori how many and what kinds of training images are needed in order to reach a required performance level. Thus, a batch learning method requires multiple cycles of collecting data, training, and testing. The limited space available to store training images and the need for more images for better performance are two conflicting factors. Therefore, if a batch learning method is used, the task of collecting a sufficiently good set of training samples is very tedious in practice. Further, each batch training session takes a significant amount of time to learn the entire batch of the training data.

With an incremental learning method, training samples are available only one (or a small set) at a time. Each training sample is discarded as soon as it has been incorporated into the system. If the output result from the current system is not correct (or with a large error), the current sample is used to update the system [54]. Otherwise, the current training image is rejected. This selective learning mechanism effectively prevents redundant learning in order to keep the size of the face-image database relatively small. Using this incremental learning mode, updating the system is convenient. We do not need to load all the old images to re-learn when new images are added. All we need to do is to run an update algorithm using only the new images.[11]

## 5.2 Probabilistic models in Machine Learning :

Unlike other methods probabilistic machine learning is based on one consistent principle which is used throughout the entire inference procedure. Probabilistic methods approach inference of latent variables, model coefficients, nuisance parameters and model order essentially by applying Bayesian Theory. Hence we may treat all unknown variables identically which is mathematically nice. For computational reasons a fully probabilistic model might not be feasible. In such situations we have to use approximations. Obviously for an empirical methodology, a mathematical consistency argument is not too convincing.

### Advantages of using probabilistic models:

Fully probabilistic models avoid "black box" characteristics. We may instantiate arbitrary sets of variables and for diagnosis purposes infer the distributions over (or expectations of) other variables of interest and thus obtain some insight how we obtain a particular decision. Using a probabilistic model is relatively easy. Inference (if properly implemented) should be insensitive to the setting of all "fiddle parameters" and will thus provide results that are close to optimal. Probabilistic models provide means for intelligent sensor fusion which allows e.g. to combine information that is known with different certainty. [26]

**Go Ahead !** with some principles of Statistics:

**DEFINITION 5** A **Population** is the set of all possible states of a random variable. The size of the population may be either infinite or finite.

**DEFINITION 6** A **Sample** is a subset of the population; its size is always finite.

**DEFINITION 7** A **parameter** is a numerical quantity measuring some aspect of a population of scores. For example, the mean is a measure of central tendency.

**DEFINITION 8** A **probability distribution** describes the values and probabilities that a random event can take place. The values must cover all of the possible outcomes of the event, while the total probabilities must sum to exactly 1, or 100%.

**DEFINITION 9** A probability distribution is called **discrete** if its cumulative distribution function only increases in jumps. The set of all values that a discrete random variable can assume with non-zero probability is either finite or countable infinite because the sum of uncountable many positive real numbers (which is the smallest upper bound of the set of all finite partial sums) always diverges to infinity. Typically, the set of possible values is topologically discrete in the sense that all its points are isolated points. But, there are discrete random variables for which this countable set is dense on the real line. Discrete distributions are characterized by a probability mass function,  $p$  such that:

$$F(x) = Pr[X \leq x] = \sum_{x_i \leq x} p(x_i)$$

**DEFINITION 10** By one convention, a probability distribution is called **continuous** if its cumulative distribution function is continuous, which means that it belongs to a random variable  $X$  for which  $Pr[X = x] = 0$  for all  $x$  in  $R$ . Another convention reserves the term continuous probability distribution for absolutely continuous distributions. These distributions can be characterized by a probability density function: a non-negative Lebesgue integrable function  $f$  defined on the real numbers such that:

$$F(x) = Pr[X \leq x] = \int_{-\infty}^x p(x_i)$$

**DEFINITION 11** A **statistical parameter** is a parameter that indexes a family of probability distributions.

### 5.3 statistical parameters Measurements

This section introduces a description for Univariate data that describes individual variables [18,34]

### 5.3.1 Measures of Location (central tendency)

When describing data with a few parameters, one can specify characteristic properties of the data distribution. Let us, for example, study a data set obtained from the measurement of the body weight of newly born babies. We could observe that:

- the average weight is 3.4 kg, or that
- 50% of the babies weigh less than 3.3 kg, or that
- 95% of the babies have a weight between 2.8 and 4.3 kg.

#### 5.3.1.1 Mean

The mean is commonly called the average. It is calculated by adding all the values and dividing the sum by the number of values. Let  $x_i$  represent the values of a variable  $X$ , with  $i = 1, 2, \dots, n$ . The mean is then defined as:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

#### 5.3.1.2 Median

When considering the distribution curve (or the histogram) of a sample, the median is the location which divides the area under the curve (or the area of the histogram) into two equal halves. The relative position of the mode, the median, and the mean provides an indication of the skewness of a distribution. The median is calculated as follows:

- Sort all values in ascending order.
- If the number of values is odd, take the middle number.
- If the number of values is even, take the average of the middle two numbers.

Example: Calculate the median of the following values: 4.4, 5.1, 4.1, 6.2, 5.7, 5.6, 7.0

1. Sort the seven values : 4.1, 4.4, 5.1, 5.6, 5.7, 6.2, 7.0
2. Pick the middle value (since the number of values is odd) as the median: 5.6

#### 5.3.1.3 Mode

The mode is the value which is most frequent in the data set. The mode is only of interest for large data sets, since the mode may be meaningless for a small sample. The relative position of the mode, the median, and the mean provides an indication of the skewness of a distribution.

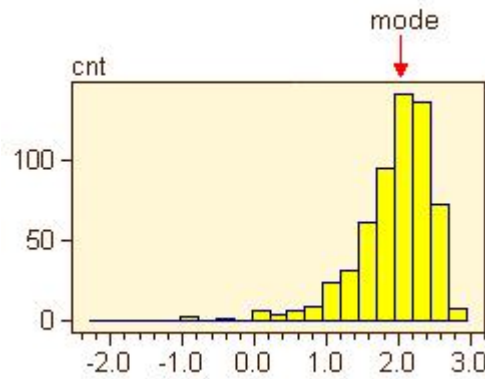


Figure 5.1: Mode.

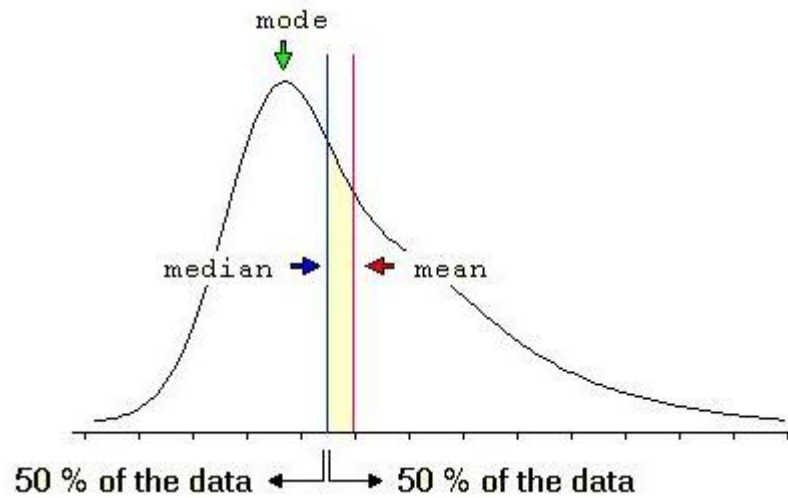


Figure 5.2: Mean, Median, Mode

#### 5.3.1.4 Quartile (Fractiles)

Quartiles partition - as the name suggests - the corresponding distribution into four quarters each containing 25% of the data. A particular quartile is therefore the border between two neighboring quarters of the distribution. The calculation of the quartiles is sometimes not quite clear (especially if the number of observations of a sample is not divisible by four). We therefore provide exact instructions how to calculate the quartiles. Assuming a sample of  $N$  observations the quartiles are defined as follows (round stands for the rounding to the nearest integer):

1. quartile: the value of the sorted series of observations having the position  $x = \text{round}(0.25 * (N + 1))$
2. quartile (median): if  $N$  is even,  $Q2$  is the mean of the two values at the positions  $\frac{N}{2}$  and  $\frac{N}{2} + 1$ ; if  $N$  is odd,  $Q2$  is the value at the position  $\frac{N}{2} + 1$ .
3. quartile: the value of the sorted series having the position  $x = \text{round}(0.75 * (N + 1))$

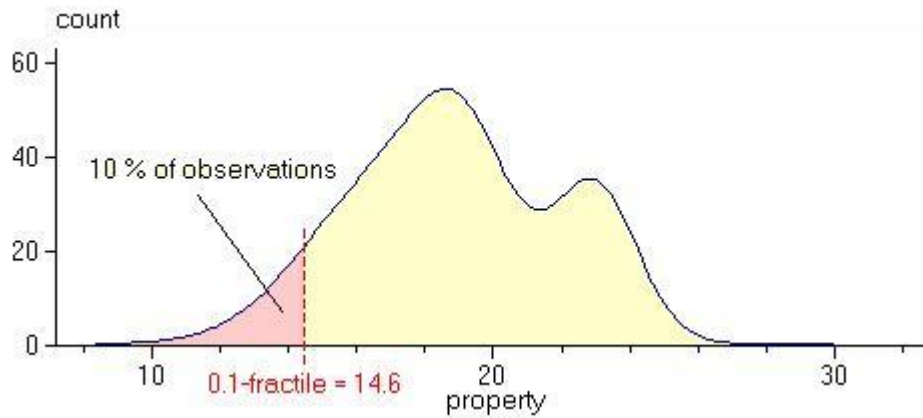


Figure 5.3: Fractiles

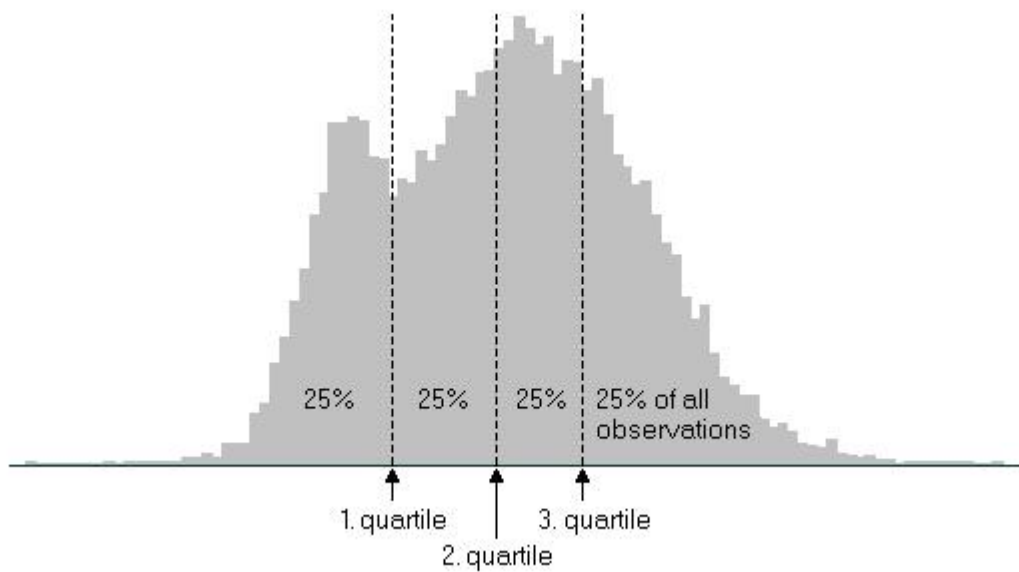


Figure 5.4: Quartile

Example : Assume that we have obtained the following 20 observations:

2, 4, 7, -20, 22, -1, 0, -1, 7, 15, 8, 4, -4, 11, 11, 12, 3, 12, 18, 1

In order to calculate the quartiles we first have to sort the observations:

-20, -4, -1, -1, 0, 1, 2, 3, 4, 4, 7, 7, 8, 11, 11, 12, 12, 15, 18, 22

The position of the first quartile is  $x = \text{round}(0.25 * (20 + 1)) = \text{round}(5.25) = 5$ , which means that  $Q1$  is the 5th value of the sorted series, namely  $Q1 = 0$ . The other quartiles are calculated in the same way resulting in  $Q2 = 5.5$  and  $Q3 = 12$ .

### 5.3.2 Measures of Variation

If you are collecting data on a process, it is important to determine not only the location of the mean, but also to look at the variation within the data. If you are,



for example, interpreting the results of a chemical analysis, you may put much more emphasis on the obtained average value if you know that the individual samples vary only very little in comparison to the mean.

### 5.3.2.1 Range

Range is the difference between the highest and lowest data element.

### 5.3.2.2 Variance

In addition to the measures of location for describing the position of the distribution of a variable, one has to know the spread of the distribution (and, of course, about its form). The spread of a distribution may be described using various parameters, of which variance is the most common one. Mathematically speaking, the variance  $v$  is the sum of the squared deviations from the mean divided by the number of samples less 1:

$$v = \sigma^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

Examination of this formula should lead to at least three questions:

1. Why take the sum of squares and not, for example, the sum of absolute deviations from the mean? The answer to this is quite simple: the mathematical analysis is simpler, if the sum of squares is used.
2. Why is the sum divided by  $n - 1$ ; wouldn't it be more logical to take just  $n$ ? Here again, the answer is simple: the concept of the degree of freedom.
3. What about the  $\sigma^2$  in the formula? The parameter  $\sigma$  which is apparently the square root of the variance is called the standard deviation.

### 5.3.2.3 Standard Deviation

The standard deviation is the positive square root of the variance, and is depicted by  $s$  for samples, or by  $\sigma$  for populations. The standard deviation is a useful measure of variability because of its mathematical tractability. There is often some confusion about the standard deviation and its interpretation. One should carefully distinguish between the formal definition of the standard deviation and the interpretation of it. The standard deviation as a numerical value can always be calculated provided that there are enough samples available. In contrast to this, the interpretation of the standard deviation as a measure of spread can be fully utilized only if the type of the distribution is known.

### 5.3.2.4 Interquartile Range

As with the mean and the standard deviation, distributions can also be described by the median and a range of fractiles around the median. The interquartile range (IQR) in particular is used to describe the dispersion of the data.

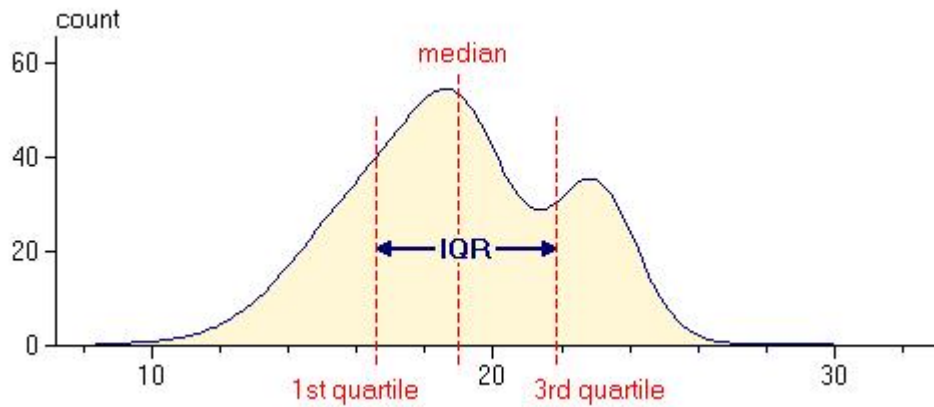


Figure 5.5: Interquartile Range

The interquartile range (IQR) is defined as the range between the first and the third quartile. Please note that the IQR contains exactly 50% of the data within the distribution.

### 5.3.3 Moments of a Distribution

Moments can be used to describe several properties of a distribution. There are two different methods to define moments; one kind of moment is called *moments about zero* and is calculated according the following equation:

$$\frac{1}{n} \sum_{i=1}^n (x_i)^r$$

Another kind of moment is called "moments about the mean"; it is calculated according to:

$$\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^r$$

The exponent  $r$  defines the  $r$ th moment. As you can easily see, the first moment about zero is equal to the mean value, the second moment about the mean is the variance. The third moment is related to the *skewness*, and the fourth moment is related to the *kurtosis* of a distribution.

#### 5.3.3.1 Skewness

A distribution is said to be skewed to the right (left) if it shows a tailing off at the right (left). The amount of skewing can be determined by the third moment of the distribution, which is usually called skewness:

$$\frac{1}{n * \sigma} \sum_{i=1}^n (x_i - \bar{x})^3$$

Note that the skewness is occasionally defined by a somewhat different formula, leading to different values.

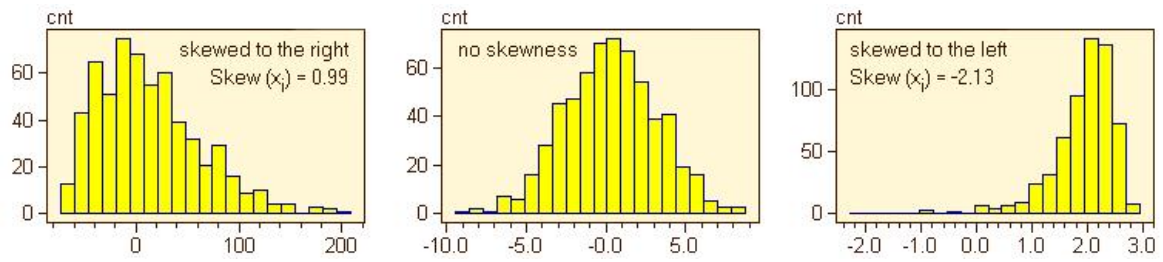


Figure 5.6: Skewness

### 5.3.3.2 Kurtosis

The kurtosis (or excess) measures the relative flatness of a distribution (as compared to the normal distribution, which shows a kurtosis of zero). A positive kurtosis indicates a tapering distribution (also called leptokurtic distribution), whereas a negative kurtosis indicates a flat distribution (platykurtic distribution). The kurtosis is defined by the following formula:

$$\left( \frac{1}{n * \sigma} \sum_{i=1}^n (x_i - \bar{x})^4 \right) - 3$$

Below you find two examples of distributions with different kurtosis.

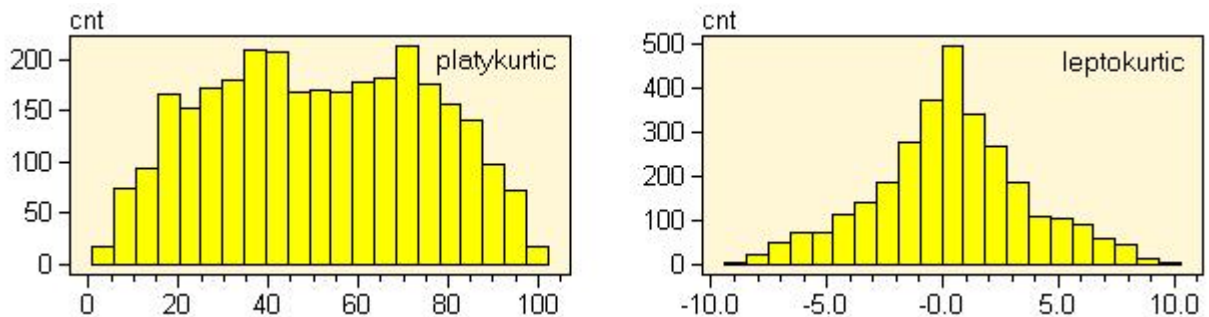


Figure 5.7: Kurtosis

Note that the kurtosis is sometimes defined by another formula, omitting the term "-3" in the formula above. In this case a normal distribution would yield a kurtosis of 3.

## 5.4 Incrementality notion and statistical parameters

we note that we can show the principles of incrementality notion inside statistical parameters itself like ( average, variance, momentum , etc..). Let us take this Example to understand how statistical parameters can be an efficient incremental way to estimate the problem.. for example : Average . If we have a class compose of 5 students and the teacher wants to make an evaluation of the class level using a statistical way depending on their marks to calculate the class mean .. Let  $x_i$  denotes to the student marks: {15, 20, 10, 6, 16}, and  $n$  the number of students, and start calculating the mean

equation as following :

$$\bar{x}_1 = \frac{\sum_{i=1}^n x_i}{n} \quad (5.1)$$

$$\bar{x}_1 = \frac{15 + 20 + 10 + 6 + 16}{5} = 13 \quad (5.2)$$

Assume that I forgot to calculate a student's mark. It's not logical to start over and calculate overall data. In this case we will add this student's mark to the average in incremental way as following :

$$\bar{x}_2 = \frac{\bar{x}_1 * n}{n + 1} \quad (5.3)$$

Assuming that new mark = 16. That's mean :

$$\bar{x}_2 = \frac{13 * 5 + 16}{5 + 1} = 13.5 \quad (5.4)$$

This is the same if we start over :

$$\bar{x}_1 = \frac{15 + 20 + 10 + 6 + 16 + 16}{6} = 13.5 \quad (5.5)$$

In the same way we can calculate the rest of statistical parameters , For this reason we conclude that all statistical parameters are incremental in their funtion .Due to that we can conclude that every method that uses statistical parameters can be eaisly worked incrementally .

We noted that there is some neural networks methods utilize probabilities and statstic like PNN,GNN and APPN.

In the next section we will introduce the PNN and APNN concepts and we will explain later how to use APNN to bulid a robust incremental learning NN model .

## 5.5 Architecture of PNN Network

In 1990, Donald F. Specht proposed a method to formulate the weighted-neighbor method in the form of a neural network. He called this a *Probabilistic Neural Network*. PNN has gained interest because it offers a way to interpret the network's structure in the form of a probability density function and it is easy to implement. An accepted norm for decision rules or strategies used to classify patterns is that they do so in a way that minimizes the *expected risk*. Such strategies are called *Bayes strategies* and

can be applied to problems containing any number of classes. Figure[5.8] is a diagram of a PNN network.[5]

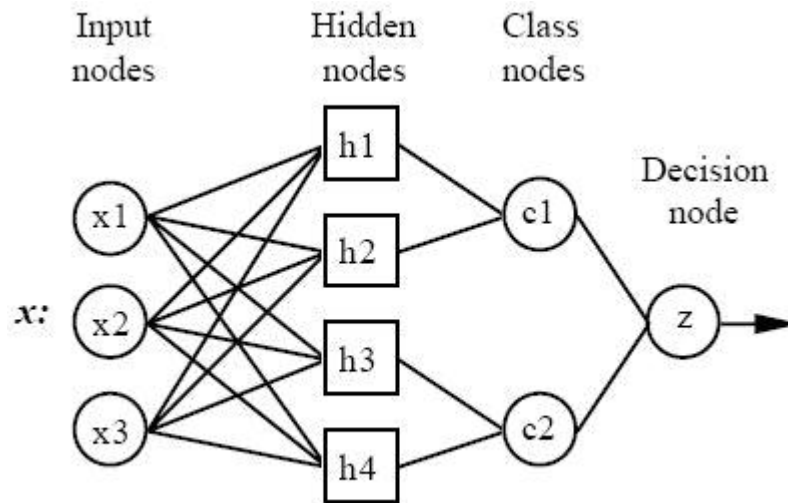


Figure 5.8: Diagram of a PNN network

All PNN networks have four layers:

1. **Input layer** There is one neuron in the input layer for each predictor variable. The input neurons (or processing before the input layer) standardize the range of the values by subtracting the median and dividing by the interquartile range. The input neurons then feed the values to each of the neurons in the hidden layer.
2. **Hidden layer** This layer has one neuron for each case in the training data set. The neuron stores the values of the predictor variables for the case along with the target value. When presented with the  $x$  vector of input values from the input layer, a hidden neuron computes the Euclidean distance of the test case from the neuron's center point and then applies the RBF kernel function using the sigma value(s). The resulting value is passed to the neurons in the pattern layer.
3. **Pattern layer / Summation layer** The next layer in the network there is one pattern neuron for each category of the target variable. The actual target category of each training case is stored with each hidden neuron; the weighted value coming out of a hidden neuron is fed only to the pattern neuron that corresponds to the hidden neuron's category. The pattern neurons add the values for the class they represent (hence, it is a weighted vote for that category).
4. **Decision layer** The decision layer is different for PNN, the decision layer compares the weighted votes for each target category accumulated in the pattern layer and uses the largest vote to predict the target category.

## 5.6 Advanced PNN

[4] Figure[5.9] shows the neural network organization for classification of input patterns  $X$  into two categories. In this figure, the input layer is the merely distribution units that supply the same input values to all of the pattern units. The second layer Advanced Probabilistic Neural Network [31] consists of a number of pattern units. In PNN, each pattern unit (shown in more detail in figure[5.10]) forms a dot product of the input pattern vector  $X$  with a weight vector  $W_i$ ,  $Z_i = X * W_i$ , and then performs a nonlinear operation on  $Z_i$  before outputting its activation level to the summation unit. Instead of the sigmoid activation function commonly used for back-propagation neural network, the nonlinear operation used here is  $exp[(Z_i - 1)/\sigma^2]$ . Assuming that both  $X$  and  $W_i$  are normalized to unit length, this is equivalent to using:

$$\exp \left[ -\frac{(X - W_i)^T(X - W_i)}{2\sigma^2} \right] \quad (5.6)$$

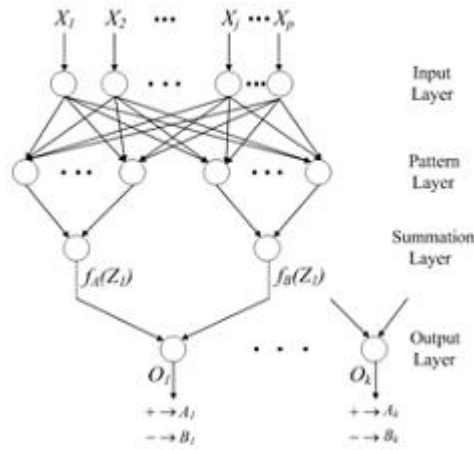


Figure 5.9: Structur of a PNN

However, the probability density function did not consider the individual probabilistic property of variables in PNN because only one global smoothing parameter was used. Therefore, in [31] paper, the advanced probabilistic neural network (APNN) was proposed to reflect the global probability density function by summing the heterogeneous local probability density function automatically determined to use the individual standard deviation of variables. The basic idea is to individually use the heterogeneous local probability density function in a variable because the probabilistic property of variables is not homogenous but heterogeneous. The individual probability density function was derived from the standard deviation of variables. The probability density function for  $i^{th}$  sample is determined to sum different standard deviations of the training vector with  $j^{th}$  variables (figure[5.11]). Therefore, the nonlinear operation of APNN can be expressed as

$$\exp \left[ -\sum_{j=1}^p \frac{(X - W_{i,j})^2}{2\sigma^2} \right] \quad (5.7)$$

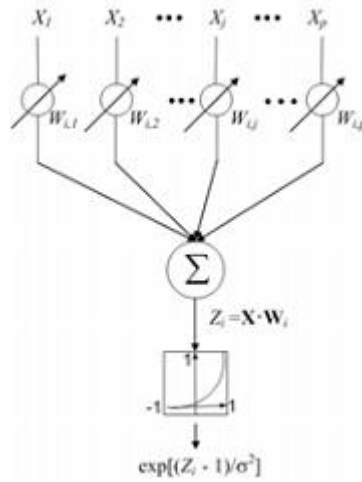


Figure 5.10: Pattern layer of PNN

Where  $i$  and  $j$  are indices for the  $i^{th}$  training pattern and  $j^{th}$  variable;  $p$  is the number of variables;  $X_j$  is the  $j^{th}$  variable of input data;  $W_{i,j}$  is the  $j^{th}$  variable of the  $i^{th}$  training vector.

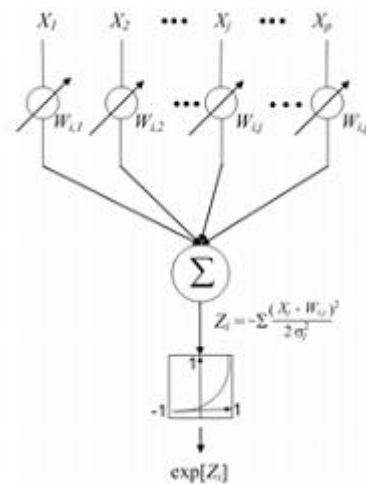


Figure 5.11: Pattern layer of APNN

## 5.7 Incremental APNN

Our architecture of Incremental Neural network for face recognition depended on :

1. The architecture based on Probablstic Neural network [5].
2. The transfer function in hidden layer based on APNN [4].
3. The architecture based on constructive learning (start small and add)
4. The incrementality notion is not limited only in adding neurons, but also in updating weights using the incremental principle in statistical parameters.

### 5.7.1 Representation of Incremental APNN

The Architecture of PNN is depending on a constructive networks principles (start small and add) .The network start with neither hidden neurons nor pattern neurons , when input data come to the network learns incrementally and start Add one by one .

N.B . : No training set is preserved . just some information about every training set (images) is preserved in the weights of hidden layer and they used to learn new images.

To understand the representation of our neural network architecture we introduce here some concepts which can be used .

#### 5.7.1.1 Input data representation

As we know that input data to our architecture are images of faces in gray scale , but for simplifying of description here we will use images of handwritten characters which assume size 16\*16 pixels . In this case the number of pixels in each image = 256 .

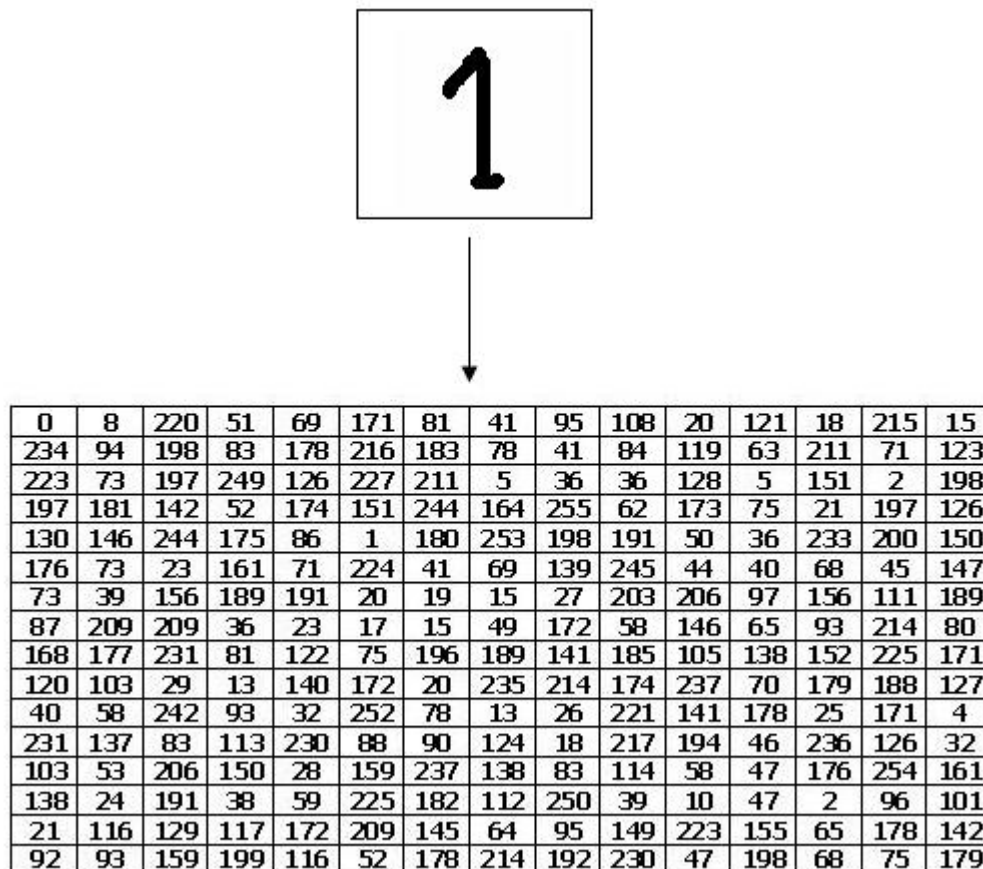


Figure 5.12: Input data representation

Let us Assume that we have a set of images represent the characters handwritten of numbers 1 and 8 , as following :

- $U = \{C_1, C_2\}$ .



- $C_1$  = a set of images that represents number 1 .
- $C_2$  = a set of images that represents number 8 .

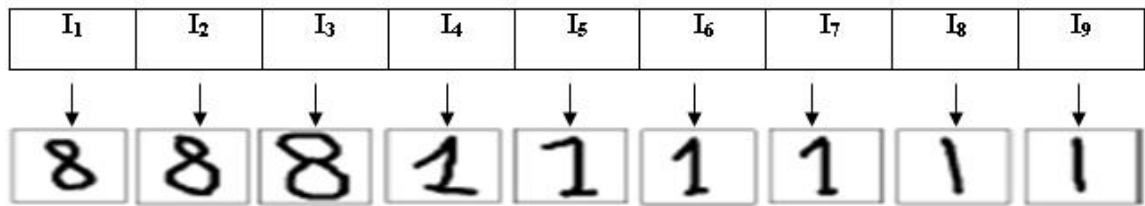
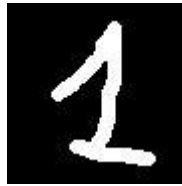


Figure 5.13:  $U = \{C_1, C_2\}$

That means :

- $C_1 = \{I_4, I_5, I_6, I_7, I_8, I_9\}$ .
- $C_2 = \{I_1, I_2, I_3\}$ .

In figure[5.13]we notice there are many handwritten ways to write a character for example : we can write character one in different ways.  
Some people write number one just like this :



Others may write with this way :



Someone else can write like :



And all those represent number one ,Therefore , we can classify more specific subsets by collecting the most similar images in shape in one subset as following :

$$C_1 = \{I_4, I_5, I_6, I_7, I_8, I_9\}$$

$$H_2 = \{I_4, I_5, I_6, I_7\}$$

$$H_3 = \{I_8, I_9\}$$

$H_2, H_3$  are subsets of  $C_1$ .

Note :  $H_2 \cap H_3 = \phi$ .

Then we conclude that  $H_i$  = a group of Characters images belongs to the same character set  $C_j$  (the Characters images have convergence in their features).

By the same way for number 8 but in our case we have one subset for number 8 because all of them are resembled and have the same features :

$$C_2 = \{I_1, I_2, I_3\}$$

$$H1=\{I_1, I_2, I_3\}$$

$H_1$  is a subset of  $C_2$ .

Then , each set has one or more subset which contains set of images which are most correspondence in their features.

Notion of classes and subclasses in our architecture:

- Class= set (C)
- subclass= subset (H)

### 5.7.2 Representation of input layer

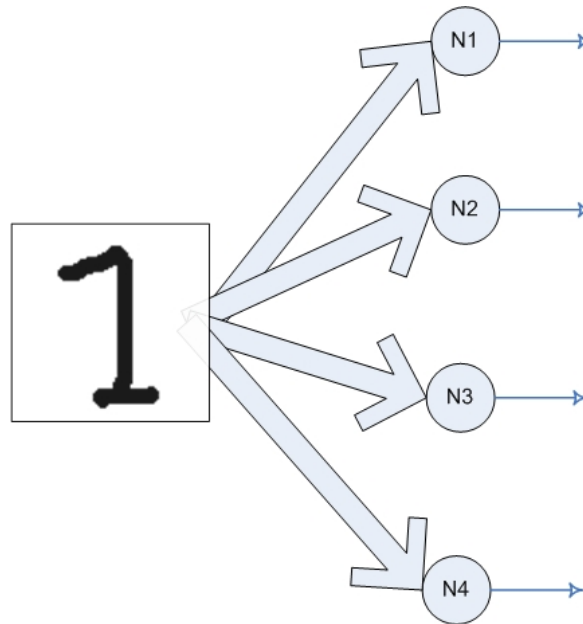


Figure 5.14: Input layer

Each neuron in this layer represents a value of some statistical parameter calculated upon the input image. In input layer for the input image we can calculate one or more statistical parameters as we need in our work, For example in this case we will calculate four statistical parameters to the population as follow:

- Population size = Number of pixels in one image .

- $S = \{0, 8, 220, 51, 69, \dots, 198, 68, 75, 179, 116\}$ . Population Set .
- Each neuron will apply the function of statistical parameters instead of simple summation function upon the Population set.
- N1 calculates the Mean upon S denoted by M.
- $M = \frac{0+8+220+51+69+\dots+198+68+75+179+116}{16*16}$  .
- N2 calculates *Variance* upon S Denoted by V.
- N3 calculates *Kurtosis* upon S denoted by Ku.
- N4 calculates *Skewness* upon S denoted by Sk. .
- Nb : The number of neurons in the input layer equals the number of statistical parameters calculated upon the image, in this case Number of neurons = 4 .
- Then the representation of the input layer will appear as following :

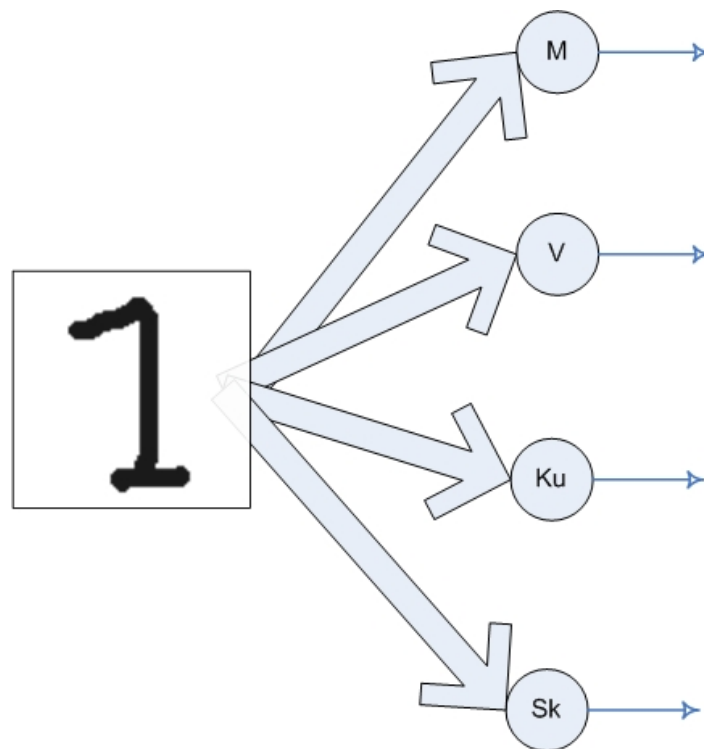


Figure 5.15: Input layer

Then each neuron will fire its value to all neurons in the next layer (hidden layer) as shown in the figure.

### 5.7.3 Representation of hidden layer

Neuron in this layer = subclass.

Each neuron in this layer represents a subclass, referring to the characters images example, we have three subclasses two subclasses represent character one and one subclass represents character 8 as shown in figure[5.16].

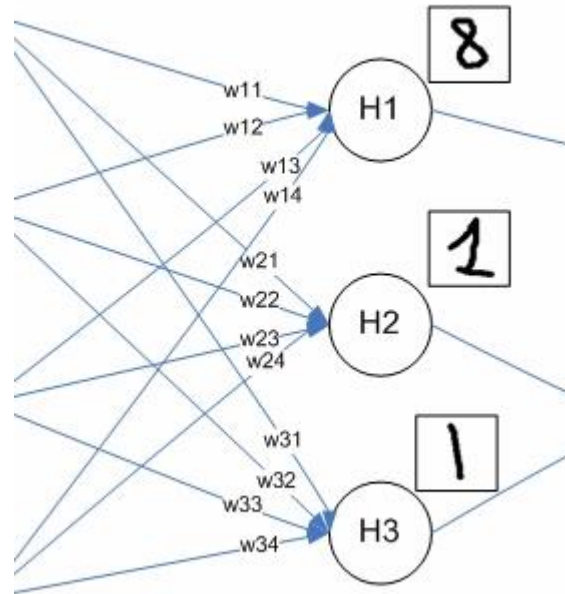


Figure 5.16: Hidden layer

In this layer each weight represents one kind of statistical parameters of union of all images in the subclass. These parameters (weights) how they are calculated for all images in the subclass !

Lets take the simple example of a statistical parameters (Mean) and we take neuron H1 to exemplify to answer this question. Referring to figure[5.17], the subclass  $H_1 = \{I_1, I_2, I_3\}$

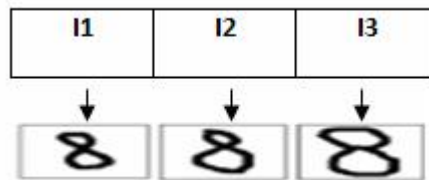


Figure 5.17:  $H_1 = \{I_1, I_2, I_3\}$

As we know that the image sizes 16\*16 then the representation of each image above will be as follow :

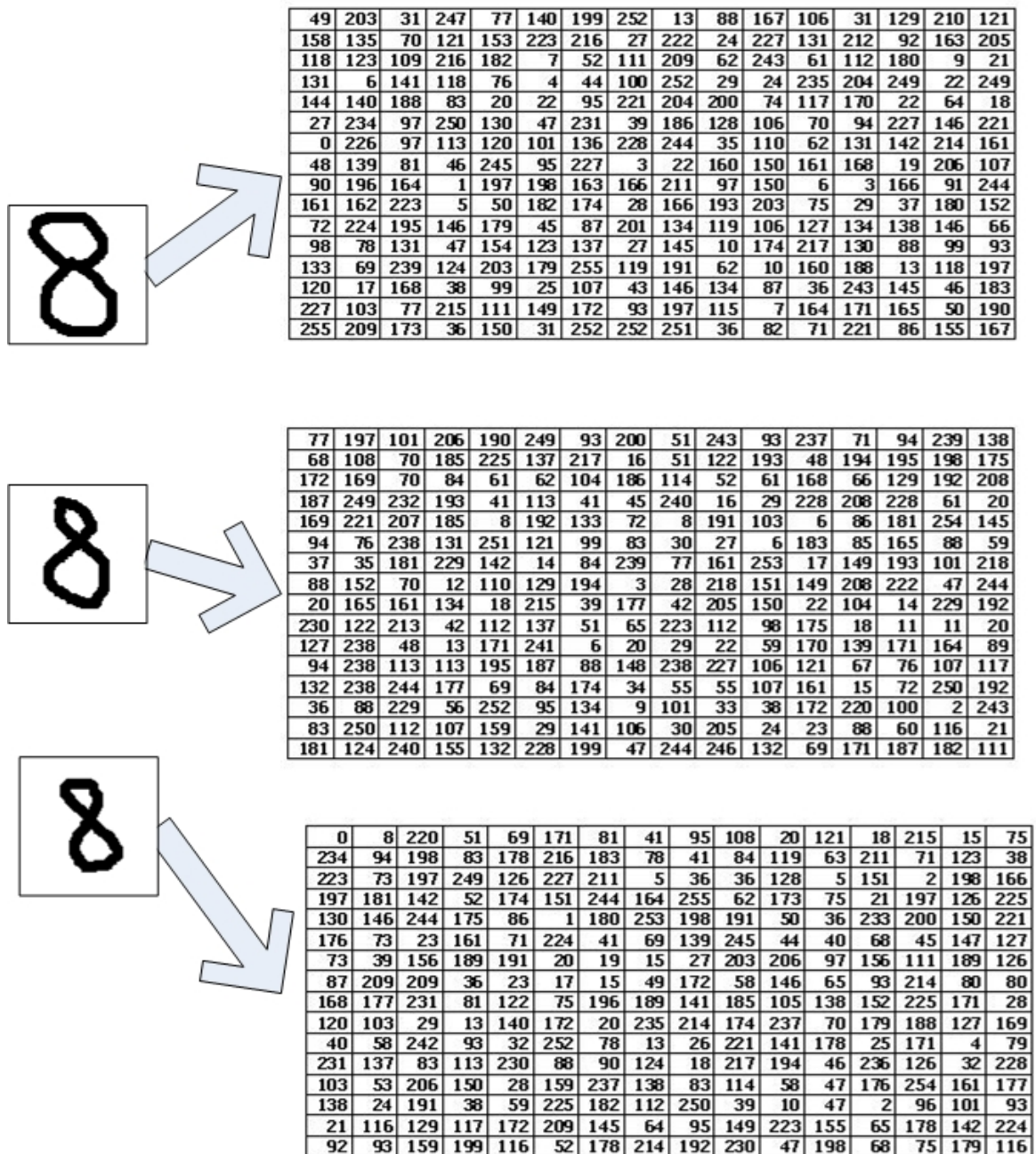


Figure 5.18:  $H_1 = \{I_1, I_2, I_3\}$

When we want to calculate the mean for all images in the subset we will use as follow:

Population =  $H_1$  = the union of pixels of all images in a subset.  $H_1 = \{I_1, I_2, I_3\}$ .

3 images and Each image has  $16 \times 16$  pixels then total Number of pixels =  $3 \times 16 \times 16 = 2028$ . Population size = Number of pixels in subset = 18.

$H_1 = \{0, 8, 220, \dots, 75, 179, 116, 77, 197, 101, \dots, 187, 182, 111, 49, 203, 31, \dots, 86, 155, 167\}$ .

$$Mean(H_1) = \frac{(0+8+220+\dots+75+179+116)+(77+197+101+\dots+187+182+111)+(49+203+31+\dots+86+155+167)}{2028} = 77.6.$$

NB : we have to emphasis that the general mean of a subclass is not the mean of the mean of all images separately.

Ex :  $Mean(H_1) = \frac{Mean(I_1)+Mean(I_2)+Mean(I_3)}{3}$  this is not the general mean .

By the same way can calculate the other statistical parameters for this subclass .

As we discussed in previous section of *representation of input layer* , each neourn in input layer fires to the all neourns in next layer (hidden layer), then each neourn in hidden layer has weights equals to the number of neourns in input layers in our case we have 4 weight for each neourn in the hidden layer .

The number of neourns in hidden layer depends on how many subclasses we have for each class in our set .

### 5.7.3.1 The notion of incntrumentality of statistical parameters in a neourn in hidden layer

In real, our approach uses the incremental way to add any new image to the subclass in the hidden layer using the statistical parameters manipulation . by referring to above example of students, if we want to add a new image of number 8 to the subclass  $H_1$  as follow :

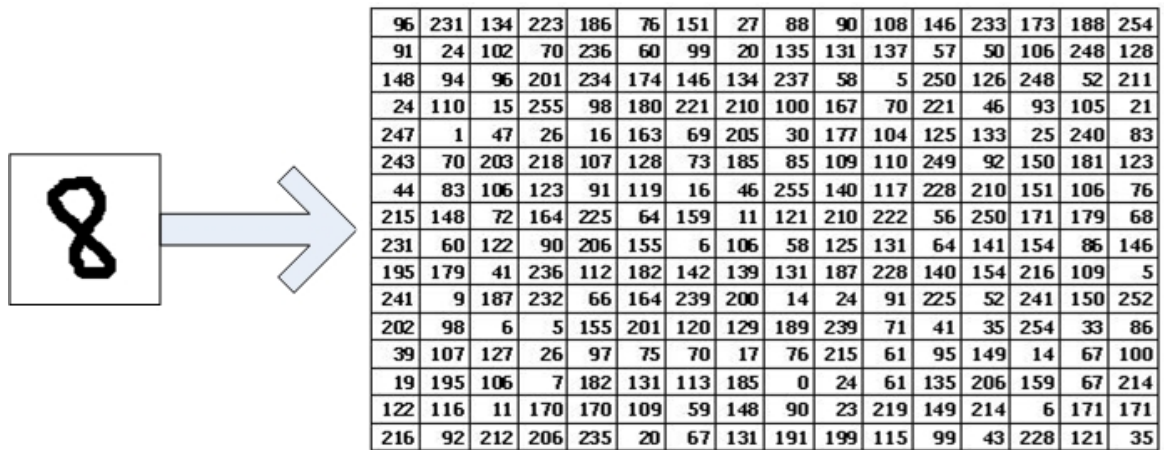


Figure 5.19: Add Another 8.

We have already information saved in  $H_1$  that the preserved value of pixels number was = 2028 because we have 3 images in this subclass , and weights values were se respectively as  $W_1 = 77.6$  ,  $W_2 = ..$  ,  $W_3 = ..$  ,  $W_4 = ...$  Then we start add the new information of the new image to the subset  $H_1$  as follow :

$$\text{New } W_1 = \frac{77.6*2028+(96+231+134+..228+121+35)}{2028+16*16} = 130.2.$$

And we do the same thing for the rest of weights of subclass  $H_1$ . By this way we added new image to  $H_1$  in incremental way that is grace of the incremental statistical parameters way to add a new information without need to start manipulation over the whole set.

Now :  $H_1 = \{I_1, I_2, I_3, I_{10}\}$  and the new Number of pixels = 24 ,  $W_1 = M = 130.2$  ,  $W_2 = ..$  ,  $W_3 = ..$  ,  $W_4 = ...$

### 5.7.3.2 Summation and Activation Function (Transfert Function)

The activation function returns the distance value between weights and the coming values from input layer. Each subclass( $H_i$ ) in hidden layer belongs to a main class ( $C_i$ ) in next layer (pattern layer), in our example means :

$$C_1 = \{H_1\} \quad C_2 = \{H_2, H_3\}$$

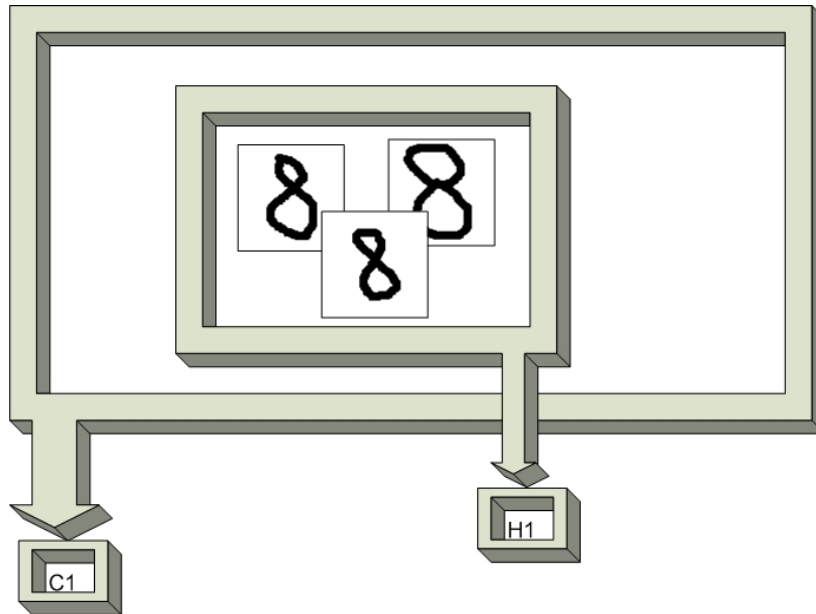


Figure 5.20: Class of 8.

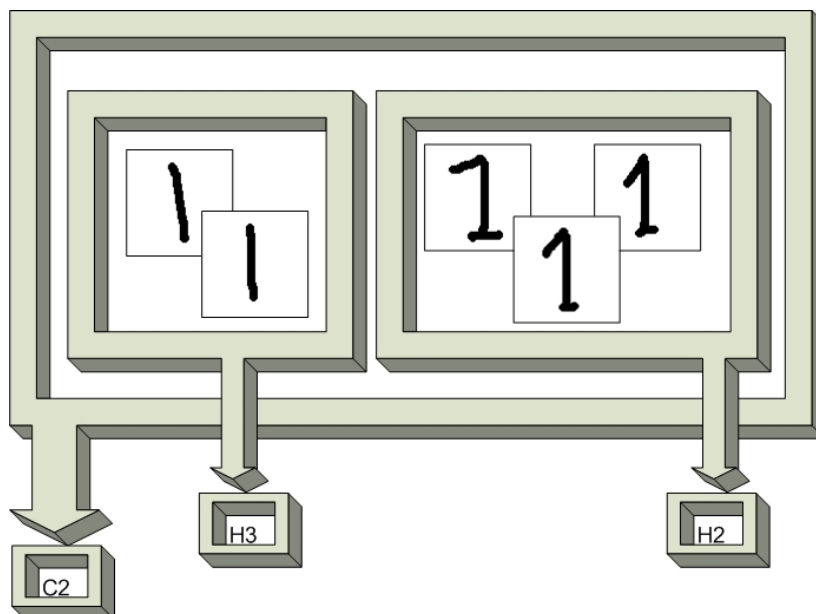


Figure 5.21: Class of 1.

Then each neuron in hidden layer (subclass ) fires just to its main class in pattern layer as it demonstrated in figure[5.22].

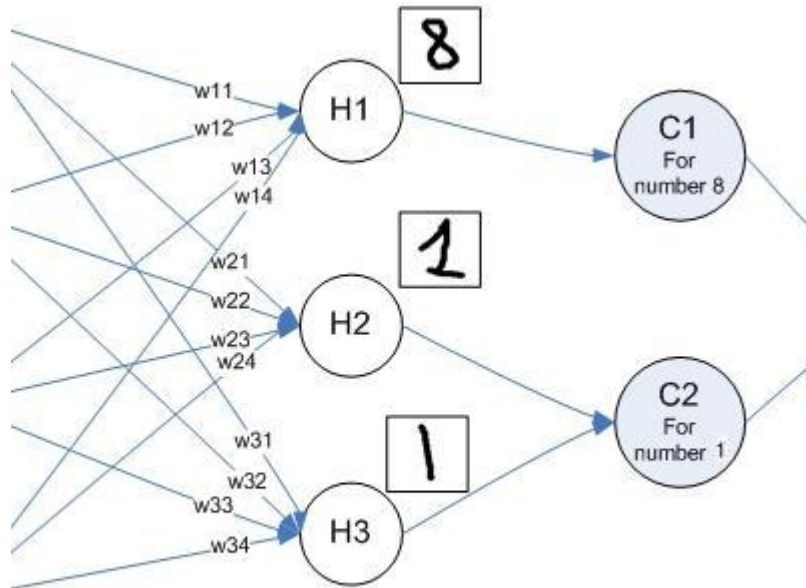


Figure 5.22: Neuron in hidden layer fires just to its main class in pattern layer.

N.B. : The notion of subclasses in the hidden layer helps to consume the space memory by decrease the size of the network.

The idea of subclasses arose to decrease the number of hidden layer neurons that leads to decrease consuming of memory , that is accomplished by grouping the most convergent Characters samples belong to one character in one subclass , insted of representing each character sample by a single neuorn.

#### 5.7.4 Pattern layer (Summation layer)

Neourn in this layer = class. Each neuron in this layer represents a Class as shown in figure[5.23]. Summation and activation function, for each neuron in this layer return the minimum value of the incoming values from the hidden layer . (The minimum value represents the nearest subclass to the input image ) .

Each neuron fires the minimum value to the next layer (output layer).

NB: Number of neurons in this layer depends on how much we have classes.



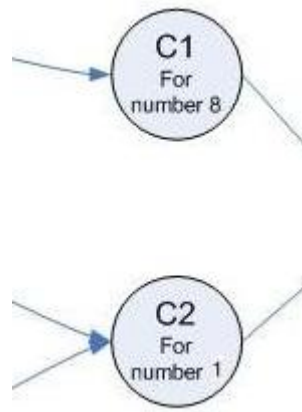


Figure 5.23: Pattern layer.

### 5.7.5 Output layer (Decision Layer )

This layer contains one neuron that the summation Activation function returns the minimum value of the incoming values from the pattern layer(The minimum value represents the class which the input image is belonging to) .

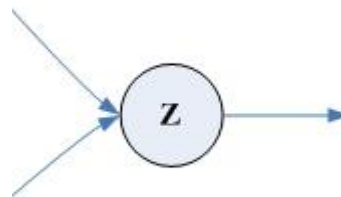


Figure 5.24: Output layer.

### 5.7.6 Learning phase (How does it work ! )

As noticed in the learning algorithm above, our NN uses Non supervised learning, that means, when we introduce a new image to the NN to learn it , at first we have to know this image for any class and subclass is belonging to .

Then the learning phase will start by recognition phase to determine the class and the subclass that the input image is belonged to.

### 5.7.7 Demonstration Example For our Network

Assuming that we have already a constructive NN with two classes in the pattern layer and three subclasses in the hidden layer . the example of the characters we mentioned above :

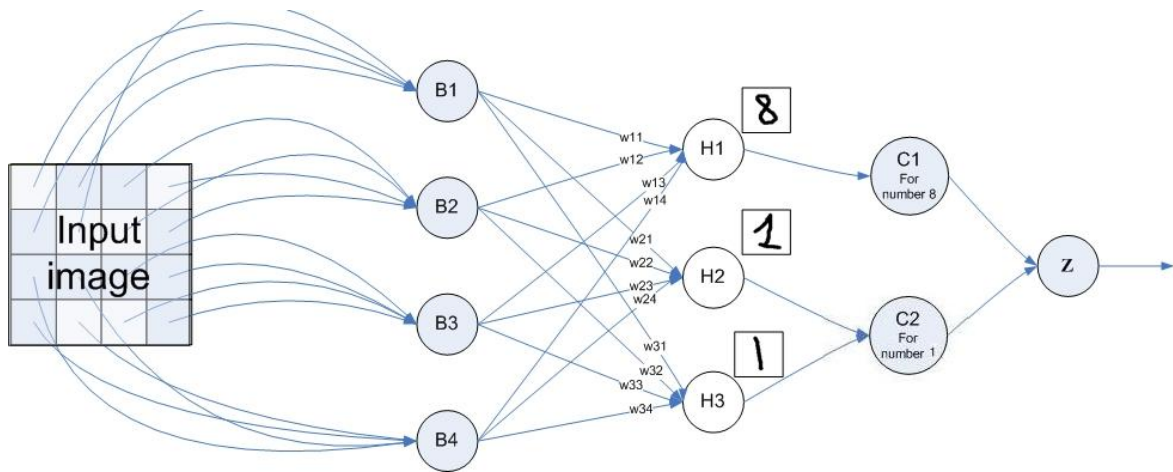


Figure 5.25: Demonstration Example For our Network.

### 5.7.8 Recognition phase

The usual recognition method will follow in to recognize which class and subclass the input image is belonged to . That means , the question posed here : *Is there any class corresponded with the new image !! , in other words Is there any class converge to the input image in their features , if yes , then find the nearest subclass to this image with the same principle of convergence.*

Input layer will compute the statistical parameters of the input image and fire to the hidden layer , then the hidden layer calculate the Euclidian distance between weights and coming value from the input layer using the transfer function of APNN In [] as follows :

Next, the hidden layer fires these values of distance to the next layer (pattern layer) which selects some values (representative distances) to come out to the next layer. Output layer select the minimum value of the representative distances. The output layer returns the nearest class and subclass to the input image.

### 5.7.9 Learning phase

There are three levels of learning in our architecture, Remember that we have not training step because we add the information sequentially without need to repeat learning all training set to learn the new information.

N.B : it is important to know that the neural network does not preserved the images themselves in the hidden layer ,but it preserves the information about the training set and just it updates the structure .

#### 5.7.9.1 Level One: Add new image to existening subclass

**Description:** Just updating weights!

Input image =  $I$ .

After Recognition process we realize that the input image is belonging to class  $C_2$  and the nearest subclass is  $H_2$  and the distance between the  $H_2$  and  $H_2 \cup I$  is small. Then we don't need to add a new subclass for this image because as we have mentioned the features of this image are resembled to those features of the images in that subclass and when I add this new information of this image will not affect upon the learning process.

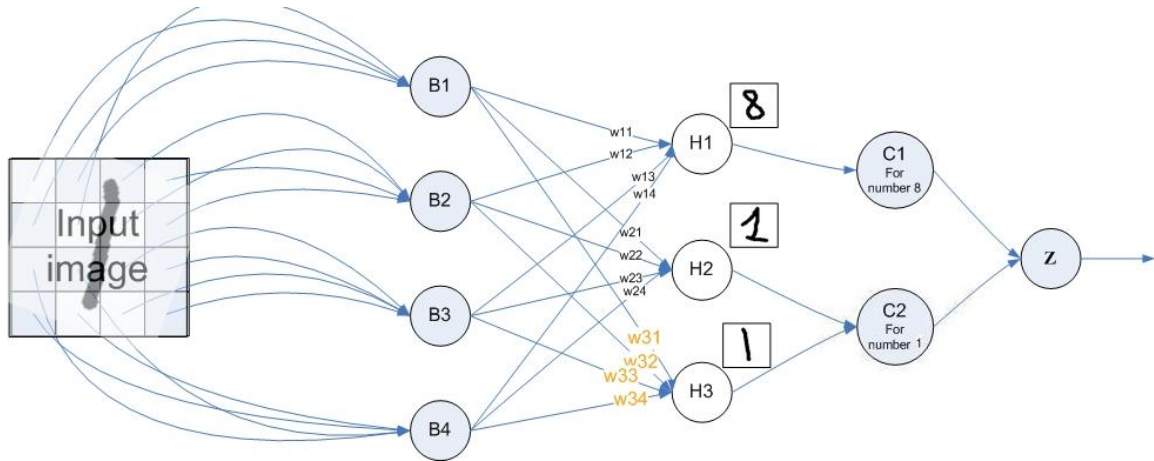


Figure 5.26: Add new image to existence subclass.

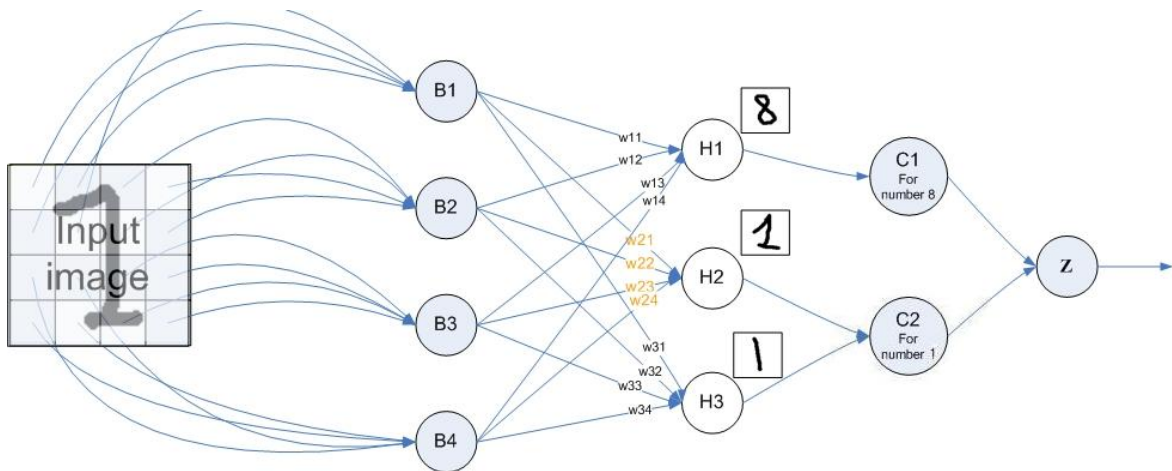


Figure 5.27: Add new image to existence subclass.

### 5.7.9.2 Level two: Add new image as a new subclass to existing class

**Description:** Create a new neuron in the hidden layer and match with the correspondence neuron in the pattern layer.

After recognition process we realize that the input image is belonging to class  $C_2$  and the nearest subclass is  $H_2$  but the distance between the  $H_2$  and  $(H_2 \cup I)$  is large. Then if we add the information of  $I$  to the existence information of  $H_2$  may

affect upon learning and it may lead to overfitting. Therefore, we have to add a new subclass  $H_4$  for this image in the hidden layer and link it to  $C_2$  in the pattern layer.

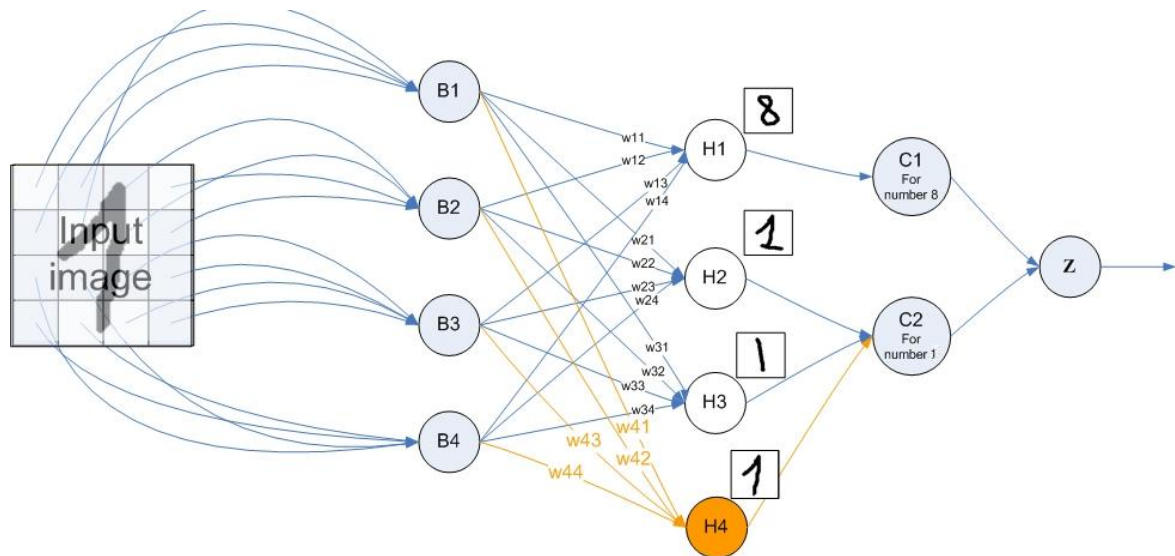


Figure 5.28: Add new image as a new subclass to existence class.

### 5.7.9.3 Level Three: Add new image as a new class and a new subclass as well

**Description:** Create a new neuron (class) into the pattern layer and add new neuron (subclass) to the hidden layer, then match a subclass to the class.

This image is completely new to the network and the network cant match any class resemble to the new image, that's mean we have to add a new class  $C_3$  in the pattern layer for the input image and new subclass  $H_4$  for the hidden layer.

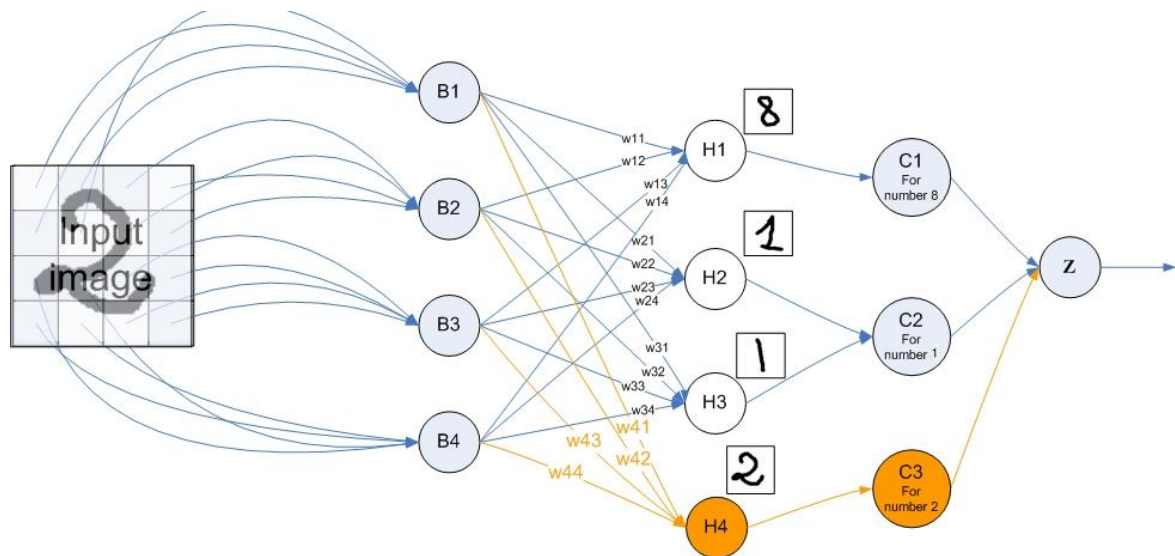


Figure 5.29: Add new image as a new class and a new subclass as well.

## 5.8 Incremental Learning scheme

Assume,

$D$  new data .

$C = \{C_1, C_2, C_3, \dots, C_k\}$ , classes set.

$H = \{H_1, H_2, H_3, \dots, H_p\}$ , subclasses set.

$\epsilon$  small value is used as a threshold.

Add New data  $D$ :

1. Compare and Find the nearest class  $C_i$  to  $D$ .
2. *if*  $C_i$  is far from  $D$  , the distance between  $D$  and  $C_i$  greater than  $\epsilon$  then.
  - Add new class  $C_{k+1}$ , Add neourn into pattern layer
  - Add new sub class  $H_{p+1}$ , Add neourn into hidden layer
  - Link  $H_{p+1}$  to  $C_{k+1}$  .

*else*:  $C_i$  is near to  $D$  , the distance between  $D$  and  $C_i$  less than  $\epsilon$  then Add  $D$  to  $C_i$  :

- (a) Compare and Find in subclasses set of  $C_i$  the nearest subclass  $H_j$  to  $D$ .
- (b) *if* the distance between  $H_j$  and  $(H_j \cup D)$  greater than  $\epsilon$  then :
  - Add a new subclass  $H_{p+1}$  which represente only  $D$ , Add neourn into hidden layer
  - Link  $H_{p+1}$  to  $C_i$  .

*else*: the distance between  $H_j$  and  $(H_j \cup D)$  less than  $\epsilon$  then add  $D$  to  $H_j$ , update the weights of the neourn that represent  $H_j$  in hidden layer.

Follows by schem[5.30].

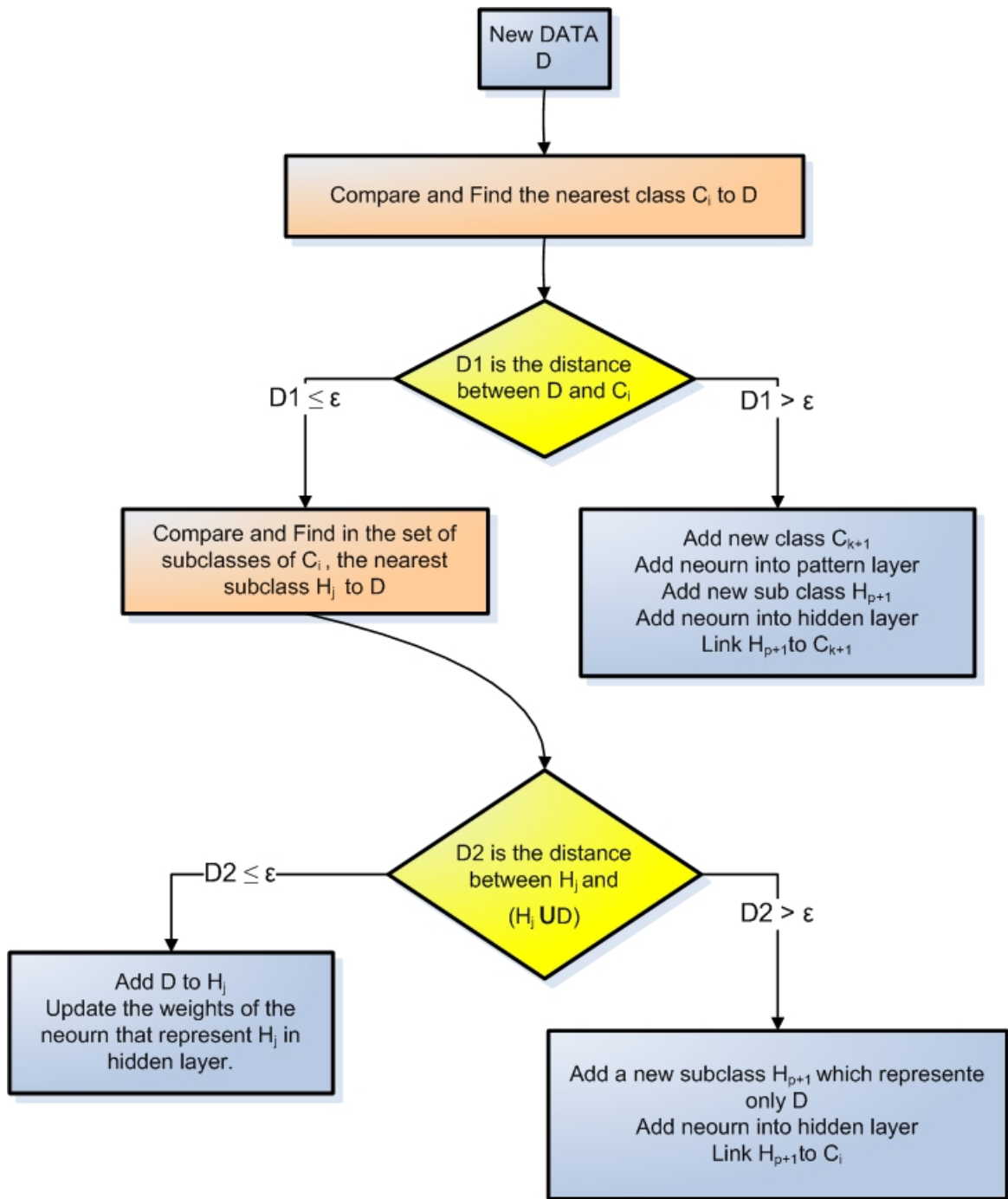


Figure 5.30: Diagram of Incremental Learning scheme

**N.B:** This Algorithm is considered as unsupervised learning, Because it researches firstly about the nearest class and subclass to the input data . This model also could be a supervised learning with determine from the begning to which class this input data belongs to , and this is a special case of the scheme described above.

## 5.9 Feature extraction

The goal of feature extraction is to find preferably small number of features that are particularly distinguishing or informative for the classification process, and that are invariant to irrelevant transformations of the data. our feature extraction works based on dividing the image into Blocks and making up some calculation using any statistical parameter on every block, 'mean' in the simplest sense.

### 5.9.1 Block extraction

The image Blocks are extracted by sliding a rectangular window from left to right and from top to bottom across the image. The blocks would be overlapped, i.e., each block will overlap the others blocks by some way \* more details in next section. The blocks have the same height  $T_v$ , width  $T_h$ , horizontal and vertical overlap  $P_v, P_h$ .

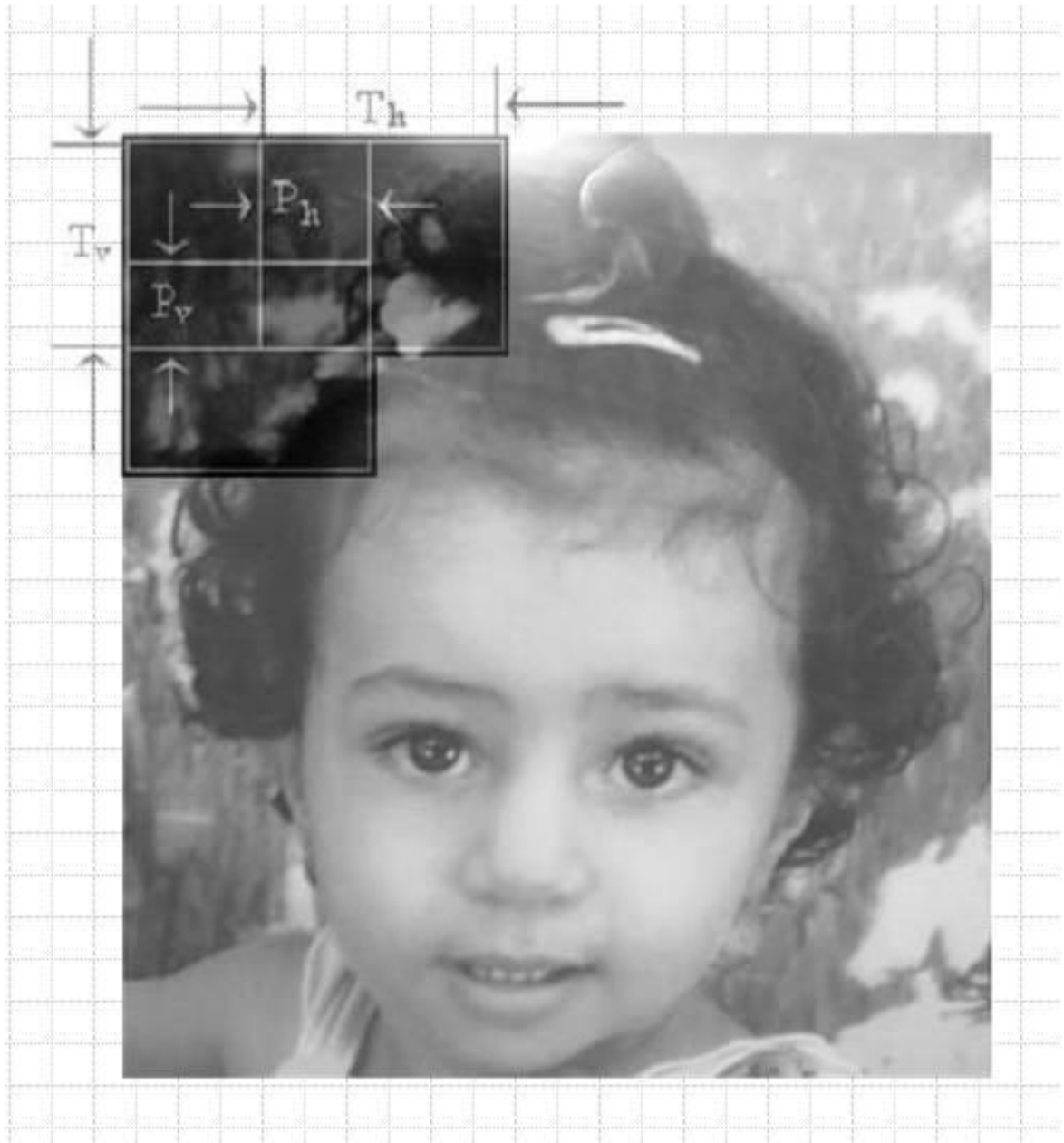


Figure 5.31: Block extraction

Assuming that

- $T_0$  = number of vertical blocks ,  $T_1$  = number of horizontal blocks
- $T_v$  = Hight of vertical blocks ,  $T_h$  = width of horizontal blocks
- $P_v$  = vertical overlap ,  $P_h$  = horizontal overlap
- $H$  = Hight of the image ,  $W$  = width of the image then and following by the equation :

$$T_0 = (H - T_v)/(T_v - P_v) + 1 \quad (5.8)$$

$$T_1 = (W - T_h)/(T_h - P_h) + 1 \quad (5.9)$$



Follows by ,Figure[5.31]. **The Advantage of dividing the image into blocks :**

Let assume that we have an image sizes 100x100 dimentions that leads to have 10000 pixels for the whole image. in the simple case each pixel would represent a node to come into input layer , in this case it would be 10000 nodes are going to enter to the input layer which will cause to have a big neural network that it will drive reserving more space in the memory . in contrast, if we divid the image into blocks the number of nodes would be less because each block will represent number of pixels , i.e, if we divide the image sizes 100x100 into 100 blocks of 10x10 . the second Advantage of using blocks concept is the importance of taking the considerartion of the global view in data representation ,i.e , one pixel may not give a useful representation to the image and make a local view.

### **The Advantage of overlapping Blocks:**

Depending on the same example have mentioned above , assuming that we divide the image into 100 blocks every block has 10x10 dimention , i.e , each block has 100 pixels , this way of dividing is not wrong and we will get less number of nodes , but in the other hand it would be less globalitization for data representation , due to this reality we can take a medium case and make overlapping when we divid the image , take an overlap of 5x5 dimentions it would be have 400 blocks .

## **5.10 Testes**

In all fields and in all disciplines, each new product must To be tried and tested to know the advantages and disadvantages if any, but also and especially for a position in relation to its competitors. Our system is no exception. The goal is to obtenirles best parameters to guarantee a rate of recognition up to a minimum, this is to find a compromise between rates recognition and time of recognition. We present first base faces on which were test, then we will see what are the relevant parameters and their influences on system. Finally, we will give the results of experiments and we draw conclusions.

### **5.10.1 The database of faces ORL**

The database of faces ORL (Olivetti Research Laboratory) is the basis of base line data for the system of automatic recognition of faces. Indeed, all systems face recognition found in the literature were tested on this basis which was conducted by ATandT Laboratories of the University Cambridge in England. This database contains 400 images of faces corresponding to 40 individuals, therefore, 10 different images of each individual. The images for a single person were taken at intervals of time different up to three months.

The extraction of faces from the images was done manually, all possible changes and predictable face were taken into account. There is change due to the conditions of accession change of illumination , change scale because of the distance between the acquisition and the individual. Also, the base ORL takes into account changes in facial expressions of a individual , and changes in the orientation of the face . The database also includes individuals Diffrent ages, races and sexes . An individual may

wear glasses , have a beard or moustache and change hairdressing. Cases such are represented in the database.



Figure 5.32: Example of a change of lighting



Figure 5.33: Example of a change of scale



Figure 5.34: Example of a change of facial expressions



Figure 5.35: Example shift face



Figure 5.36: Example of individuals carrying or without glasses



Figure 5.37: Example change hair and beard presence



Figure 5.38: Example of individuals of different ages, gender and skin color

## 5.11 Result

### 5.11.1 Rate

Approche	Recognition Rate
Auto Association and Classification NN [7]	20%
Dynamic Link Matching [8]	80%
Eigenface [9]	80%
HMM [2]	85%
VFR Model [10]	92.5%
Convolutional NN [12]	96.2%
Our APNN	96.5%
EHMM [2]	98.5%

Table 5.1: Approachs comparison

### 5.11.2 Number of blocks

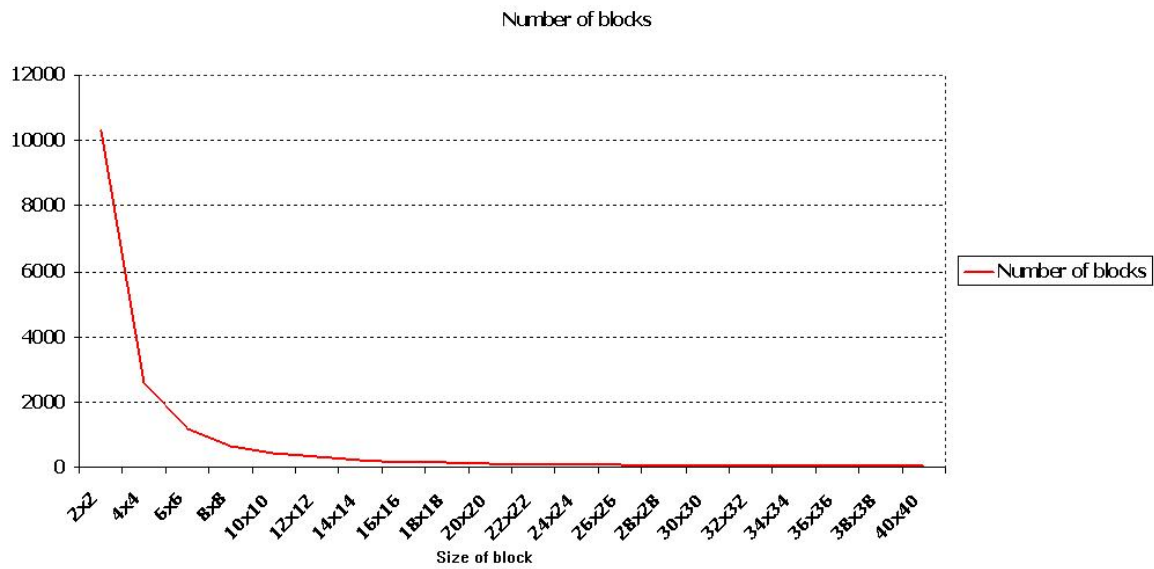


Figure 5.39: Whenever size of extract blocks increases , Number of extracted blocks decreases

### 5.11.3 Size of NN

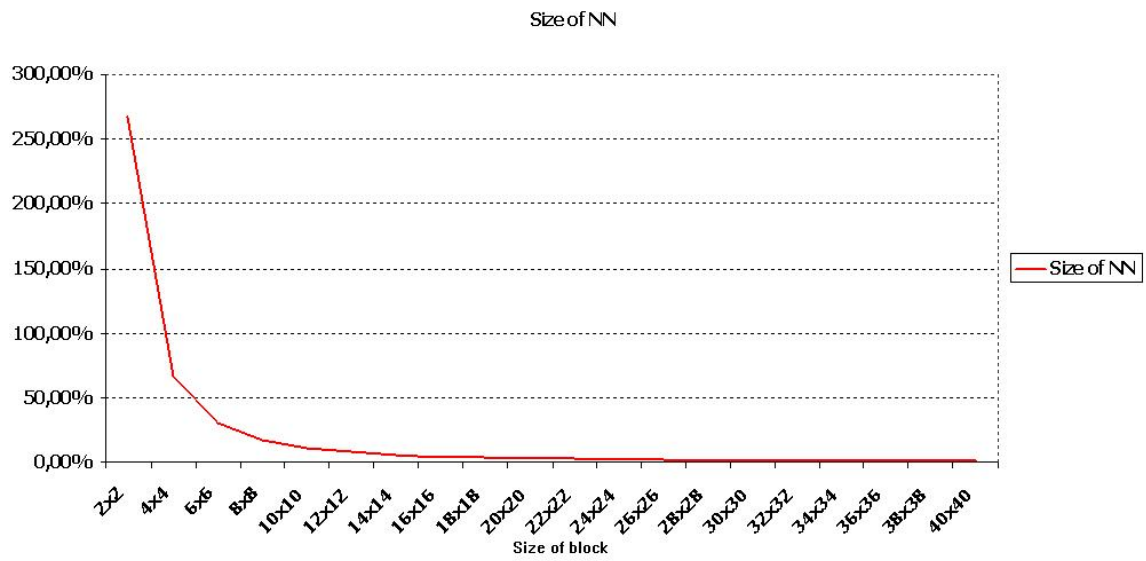


Figure 5.40: Whenever the extracted block size increases, the NN size decreases.

### 5.11.4 Time of learning (ms) of one image

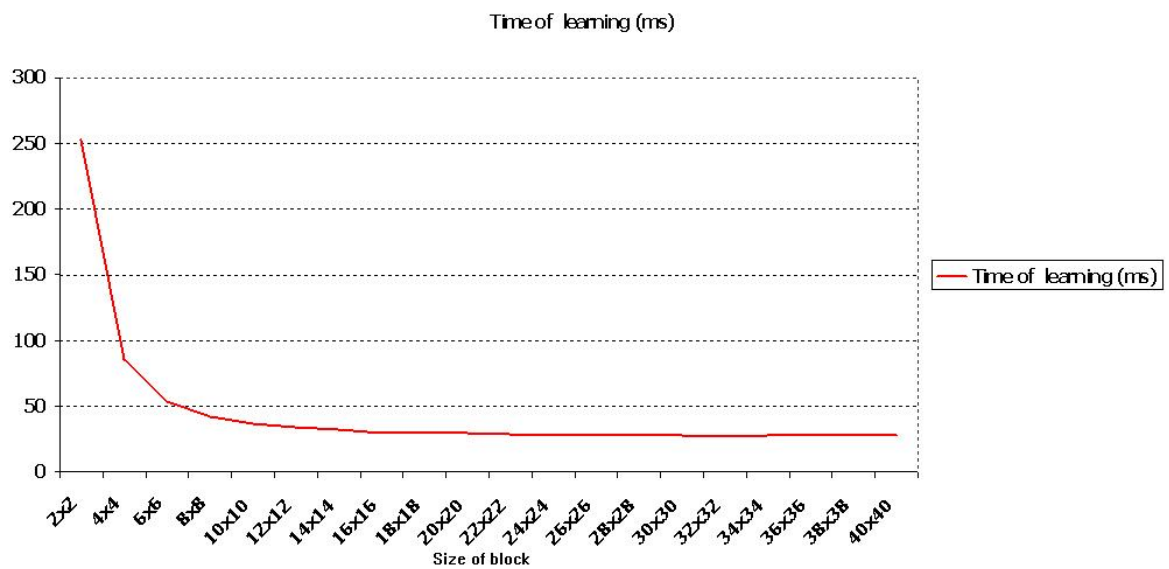


Figure 5.41: Whenever the extracted block size increases, Time of learning decreases.

### 5.11.5 Time of recognition (ms) of one image

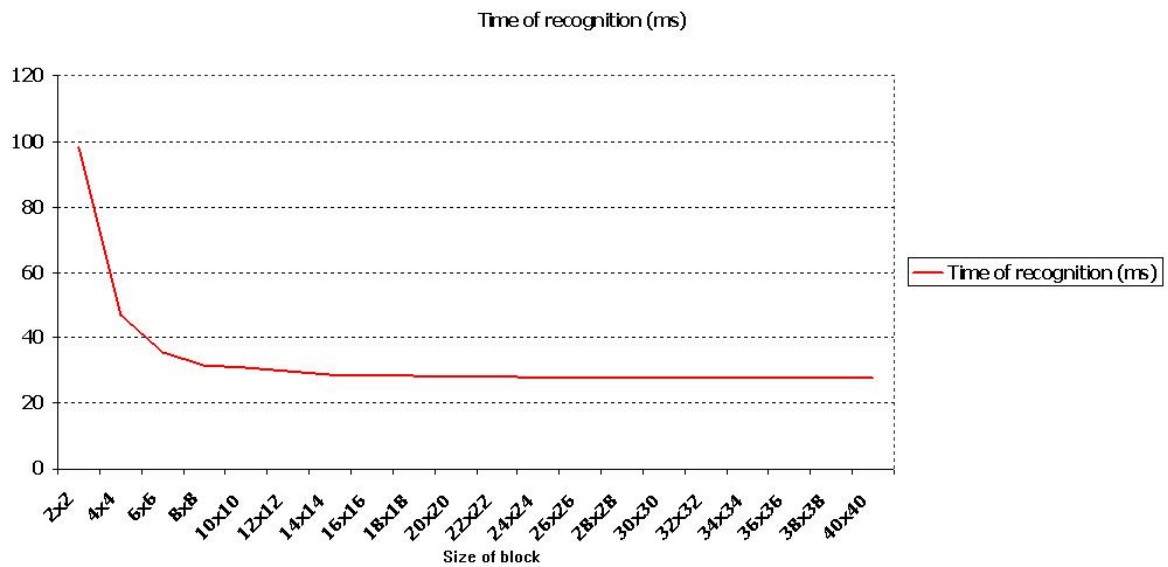


Figure 5.42: Whenever the extracted block size increases, Time of recognition decreases.

### 5.11.6 Rate with the Size of block

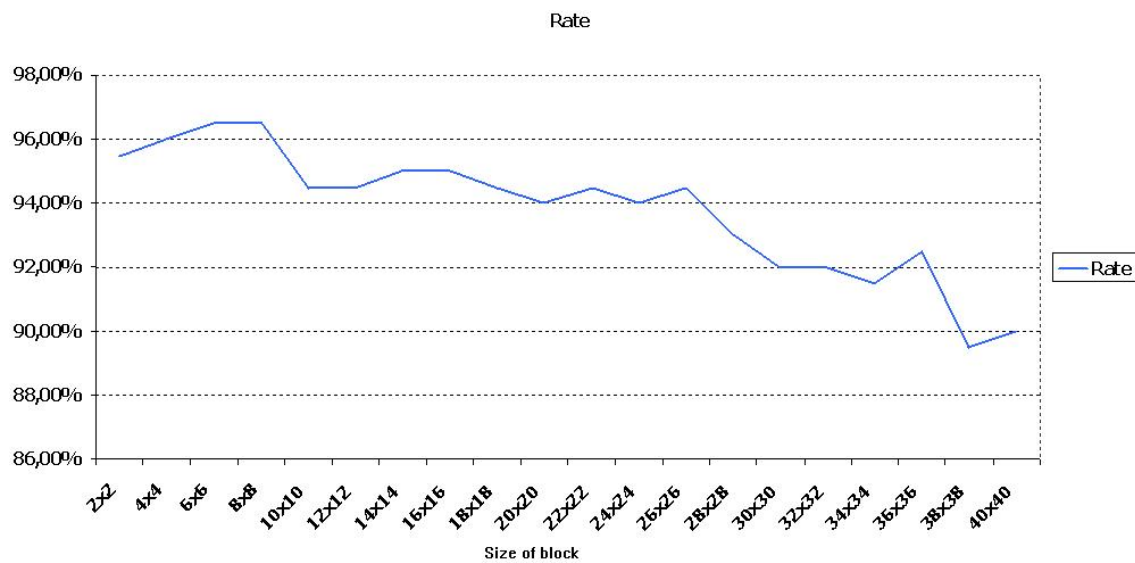


Figure 5.43: Whenever the extracted block size increases too much ex.(40x40), The rate of recognition starts decrease. Due to the extream general viewpoint . Likewise, Whenever the extracted block size decreases too much ex.(2x2) , The rate of recognition starts decreases as well . Due to the extream local viewpoint

### 5.11.7 Time of learning/Time of recognition

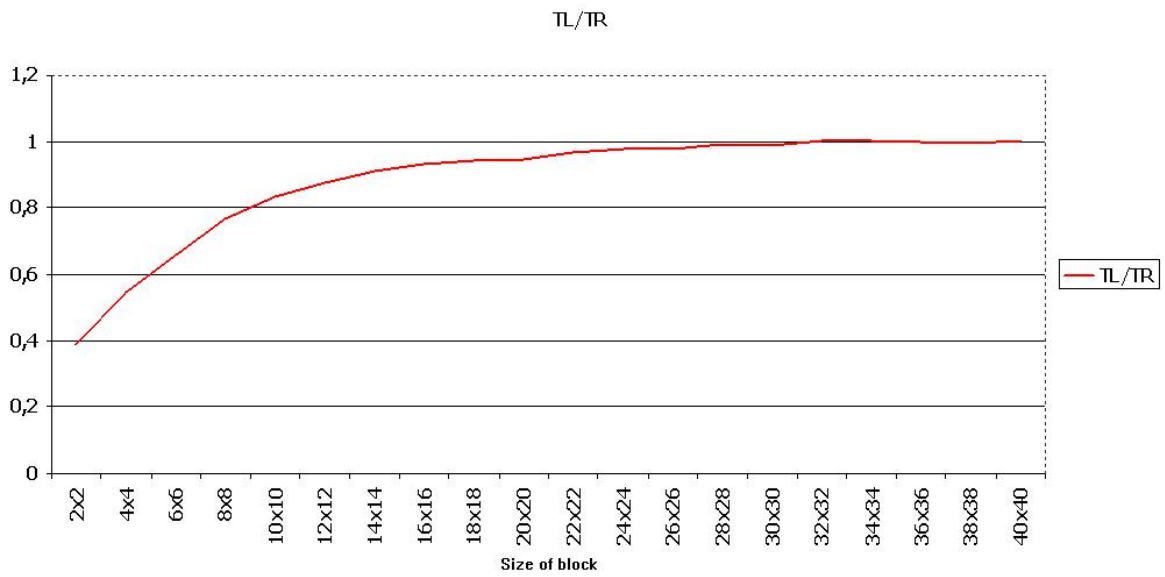


Figure 5.44: Whenever the extracted block size increases, The time of learning converges to the time of recognition.

### 5.11.8 Number of training image

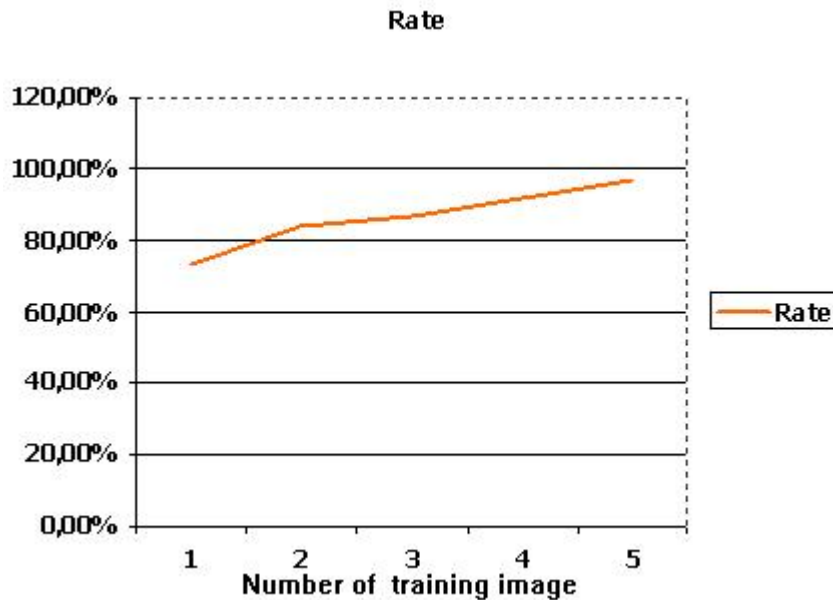


Figure 5.45: Whenever Number of training set of the image increases, NN recognition rate increases. and this is similar to the human learning



### 5.11.9 Rate with Sp

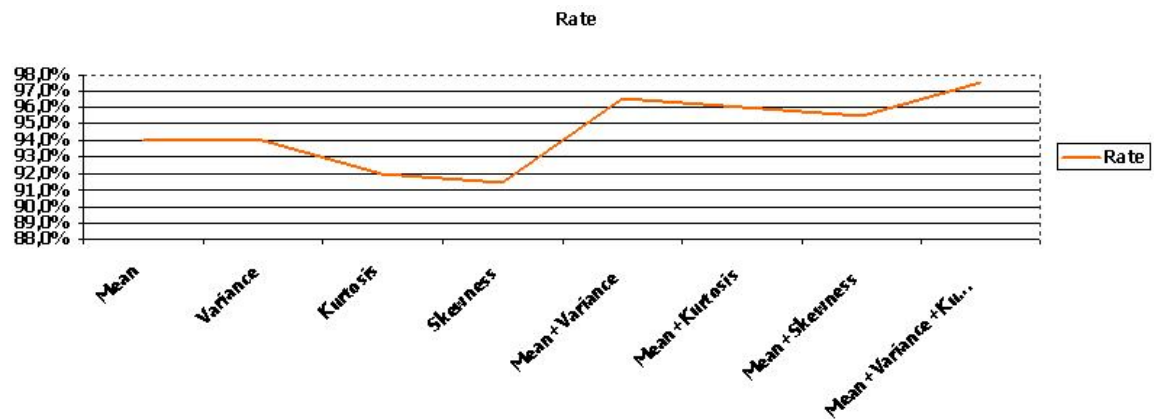


Figure 5.46: Whenever number of using of SP increases (ex. mean , median,... are calculated together upon the image), Rate recognition increases.

### 5.11.10 Number of blocks

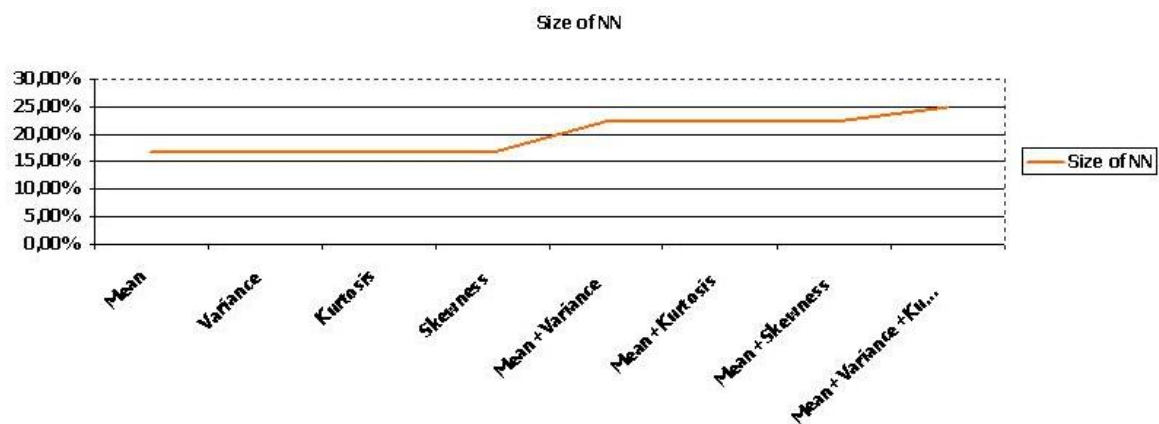


Figure 5.47: Whenever number of SP increases ,Size of NN increases

## 5.12 Conclusion

This chapter focused on two main points , the first one is giving a general view of facial recognition system , with a peek of probabilistic models in machine learning , then it gives an overview of importance of using of Statistical Parameters and their different measurments. The second point is focused on our Incremental Probabilistic Neural Network model, it discussed the architecture, how the data is represented in each layer, How does it work and learn. it also explains when we are using APNN in our architecture , how statistical parameters help strongly the network to learn incrementally,in addition to the constructive learning (adding neourns).

# Conclusion

**M**ACHINE Learning is concerned with the development of algorithms and techniques that allow computers to learn, in the sense of things learn when they change their behavior in a way that makes them perform better in the future.

In machine learning we say An algorithm is nonincremental if it reprocesses all earlier training instances in order to learn from each new instance. It is incremental if it can build and refine a concept gradually as new training instances come in, without reexamining all instances seen in the past.

Artificial Neural network is one of the successful approaches in machine learning. Neural networks process information in a similar way the human brain does. Neural Networks learn by acquiring and representing the knowledge of their environment autonomously with the help of learning algorithms.

Incremental learning aims to improve network capacity by consuming Spatial And Time components.

In this thesis, We produced a model of incremental learning depending on Probabilistic Neural Networks [5], which uses statistical parameters, it also depends on constructive learning (starts small and increases) . We have applied the idea for face recognition and we succeeded in obtaining 96 % as a rate of recognition on ORL database . The power of our approach is represented in three points the first one that is fully incremental function by adding images simply as a new neuron (new subclass in hidden layer to existing class in the pattern layer or new subclass in hidden layer and new class in the pattern layer ) . The second one is the Incrementality is not only limited in adding neurons , it also appears strongly in updating weights in hidden layer grace of incremental statistical parameters processing . The third advantage is the notion of subclasses that helps to consume more space in the memory in addition to the incremental learning advantage in regard to spacial and temporal consuming .

Future work : We hope to apply the notion of destructive learning (start big and delete) to cluster most informative features and put it in one neuron (subclass) ,and delete neurons (subclasses) that are not needed. and we hope to apply this approach for image indexing.

# Bibliography

- [1] Alex Smola. *Introduction to Machine Learning*. Lecture 1 to Lecture 5, The Australian National University. <http://axiom.anu.edu.au/~smola/engn4520/week1114.pdf>.
- [2] Christophe Giraud-Carrier. *Note on the Utility of Incremental Learning*. University of Bristol ,AI Communications ISSN 0921-7126, IOS Press
- [3] Christos Stergiou and Dimitrios Siganos. *Neural Network* . [http://www.doc.ic.ac.uk/~nd/surprise\\_96/journal/vol4/cs11/report.html#Contents](http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html#Contents) .
- [4] Doo Kie Kim, Seong Kyu Chang and Sang Kil Chang. *Advanced Probabilistic Neural Network for the Prediction of Concrete Strength*. ICCES, vol.2, no.1, pp.29-34, 2007.
- [5] Donald F. Specht. *Probabilistic Neural Network* . In *Neural Networks* 3, pages 109-118, 1990
- [6] DTREG. *Probabilistic and General Regression Neural Networks*. DREG homepage.
- [7] DACs. *Artificial Neural Networks Technology*. DoD DACs homepage. <https://www.dacs.dtic.mil/techs/neural/neural6.php>.
- [8] Ethern Alpaydin. *GAL : Networks that grow when they learn and shrinks when they forget*. International Computer institute 1991.
- [9] Frank Keller. *Introduction to Machine Learning, Connectionist and Statistical Language Processing*.. Universit"at des Saarlandes.
- [10] Goharian , Grossman. *Neural Network Classification*. 2003.
- [11] J.Mario and William D.Ross. *Incremental Learning ART : A Neural Network System for Recognition By Incremental Feature Extraction*. Technical Report CAS-CNS-94-006, 1994.
- [12] John J. Weng , Daniel L. Swets. *FACE RECOGNITION*. Michigan State University, Augustana College.
- [13] Kieron Messer Thirimachos Bourlai and Josef Kittler. *Face Verification System. Architecture Using Smart Cards*, 2004.
- [14] Krose , van der Smagt. *An introduction to Neural Network*. The University of Amsterdam , 1996.

- 
- [15] Keisuke Okamoto, Seiichi Ozawa, and Shigeo Abe . *A Fast Incremental Learning Algorithm of RBF Networks with Long-Term Memory*. Graduate School of Science and Technology, Kobe University 1-1 Rokko-dai, Nada, Kobe 657-8501, JAPAN.
- [16] Klaus P. Jantke. *Types Of Incremental Learning*. FB Informatik , HTWK Leipzig (FH), Post fach 66.
- [17] Koichiro Yamauchi · Takayuki Oohira · Takashi Omori. *Fast incremental learning methods inspired by biological learning behavior*. *Artif Life Robotics* (2005) 9:128-134 © ISAROB 2005 DOI 10.1007/s10015-004-0325-5.
- [18] Keith G. Calkins. *An Introduction to Statistics*. Andrews University, Berrien Springs ,2005. <http://www.andrews.edu/~calkins/math/webtexts/stattoc.htm>
- [19] LiMin Fu, Hui-Huang Hsu, and Jose C. Principe. *Incremental Backpropagation Learning Networks* . IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. I, NO. 3, MAY 1996.
- [20] M. Kubat , I. Bratco, R.S Michalski. *A Review of Machine Learning Methods*. Department of Computer Engineering, Boğaziçi University TR-34342 Istanbul, Turkey.
- [21] Nils J. Nilsson. *Introduction to Machine Learning*. 1996.
- [22] Nick Dunkin, John Shawe-Taylor, Pascal Koiran. *A New Incremental Learning Technique*. WIRN Viteri-96, Proceedings of the 8th Italian Workshop on Neural Nets. Springer, 1997. pp. 112 - 118.
- [23] Nikola Kasabov, David Zhang, Paul S. Fang. *Incremental Learning in Autonomous Systems: Evolving Connectionist Systems for On-line Image and Speech Recognition*. 1. T2005 IEEE Workshop on Advanced Robotics and its Social Impacts.
- [24] Oya Aran, Ethem Alpaydin. *An Incremental Neural Network Construction Algorithm for Training Multilayer Perceptrons*. Department of Computer Engineering, Boğaziçi University TR-34342 Istanbul, Turkey.
- [25] Plamen Angelov. *Nature-Inspired Methods for Knowledge Generation from Data in Real-Time* Department of Communication Systems , InfoLab21, Lancaster University Lancaster, LA1 4WA, UK.
- [26] Pemser, k. *How Facial Recognition system work* Facial Recognition Department of pronspotation organ.
- [27] Peter Sykacek. *Probabilistic machine learning*. University of Oxford ,2002. <http://www.robots.ox.ac.uk/~psyk/research.html>
- [28] Rajesh, Jihoon, and Vasant . *Constructive Neural-Network Learning Algorithms for Pattern Classification*. IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 11, NO. 2, MARCH 2000.

- 
- [29] Robert M. *Catastrophic Forgetting in Connectionist Networks: Causes, Consequences and Solutions*. Trends in Cognitive Sciences, 3(4), 128-135, French, 1999.
- [30] R. Polikar, L. Udpa, S.S. Udpa, V. Honavar. *LEARN++: AN INCREMENTAL LEARNING ALGORITHM FOR MULTILAYER PERCEPTRON NETWORKS*. , 0-7 803-6293-4/00/2000 IEEE.
- [31] ROBI POLIKAR. *PATTERN RECOGNITION*. Rowan University, Glassboro, New Jersey.
- [32] Seiichi Ozawa, Soon Lee Toh, and Shigeo Abe, Shaoning Pang and Nikola Kasabov. *Incremental Learning for Online Face Recognition*. Proceedings of International Joint Conference on Neural Networks, Montreal, Canada, July 31 - August 4, 2005.
- [33] STEPHAN K. CHALUP. *INCREMENTAL LEARNING IN BIOLOGICAL AND MACHINE LEARNING SYSTEMS*. International Journal of Neural Systems, Vol. 12, No. 6 (2002) 447465.
- [34] Statistic4u. *Fundamentals of Statistics*. lectures and courses of H. Lohninger on statistics, data analysis and chemometrics, 2006. [http://www.statistics4u.info/fundstat\\_eng/index.html](http://www.statistics4u.info/fundstat_eng/index.html)
- [35] Tebogo Seipone and John A. Bullinaria. *Evolving Improved Incremental Learning Schemes for Neural Network Systems*. School of Computer Science The University of Birmingham Birmingham, B15 2TT, UK.
- [36] Sarle, W.S., ed. *Introduction to Neural Network*. Neural Network FAQ, part 1 of 7: Introduction, periodic posting to the Usenet newsgroup comp.ai.neural-nets, 1997. <ftp://ftp.sas.com/pub/neural/FAQ.html>.
- [37] Shapiro. *Perceptrons and multi-layer perceptrons*. School of Computer Science, University of Manchester , 2006.
- [38] Shapiro. *Introduction and Motivation*. Lecture 1, School of Computer Science, University of Manchester , 2006.
- [39] Tom M. Mitchel . *The Discipline of Machine Learning*. CMU-ML-06-108, July 2006.
- [40] Unis . *Learning In Neural Networks*. School of ECM , University of Surry, UK. [http://www.computing.surrey.ac.uk/ai/PROFILE/learn\\_net.html#learn](http://www.computing.surrey.ac.uk/ai/PROFILE/learn_net.html#learn)
- [41] XIN YAO. *Evolving Artificial Neural Networks*. PROCEEDINGS OF THE IEEE, VOL. 87, NO. 9, SEPTEMBER 1999.
- [42] *Neural Networks Classification*. Illinois Institute of Technology. <http://www.ir.iit.edu/~nazli/cs422/CS422-Slides/DM-NeuralNetwork.pdf>
- [43] *Introduction to Neural Network*. World of Knowledge Modeling, 2004 . <http://www.makhfi.com/tutorial/introduction.htm>
- [44] *Connectionist vs. Symbolic Models* . Copyright 1996, The Trustees of Indiana University. <http://www.indiana.edu/~gasser/Q351/connectionism2.html>

- [45] *Incremental Learning from example* . <http://www.vuse.vanderbilt.edu/~dfisher/tech-reports/tr-88-05/node5.html>
- [46] *Incremental Conceptual Clustering* . <http://www.vuse.vanderbilt.edu/~dfisher/tech-reports/tr-88-05/node6.html>

# Glossary

<b>OCR</b>	Optical Character Recognition
<b>ORL</b>	Olivetti Research Laboratory
<b>ML</b>	Machine Learning
<b>SP</b>	Statistical Parameters
<b>PNN</b>	Probabilistic Neural Network
<b>APNN</b>	Advanced Probabilistic Neural Network