B.Sc. in Computer Science and Engineering Thesis

A Real-time approach to Identify Digital Users using Artificial Neural Network

Submitted by

Fahmid Morshed Fahid
1105021

Supervised by

Mohammad Mahfuzul Islam

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka, Bangladesh

February 2017

CANDIDATES' DECLARATION

This is to certify that the work presented in this thesis, titled, "A Real-time approach to Identify Digital Users using Artificial Neural Network", is the outcome of the investigation and research carried out by us under the supervision of Mohammad Mahfuzul Islam.

It is also declared that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Fahmid Morshed Fahid
1105021

CERTIFICATION

This thesis titled, **"A Real-time approach to Identify Digital Users using Artificial Neural Network"**, submitted by the group as mentioned below has been accepted as satisfactory in partial fulfillment of the requirements for the degree B.Sc. in Computer Science and Engineering in February 2017.

Group Members:

Fahmid Morshed Fahid

Supervisor:

Mohammad Mahfuzul Islam
Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

ACKNOWLEDGEMENT

I would like to express my sincere gratitude and profound indebtedness to my supervisor Dr. Mahammad Mahfuzul Islam. He not only introduced me to this wonderful and fascinating world of machine learning but also provided me with numerous resources and helpful direction. His constant guidance, insightful advice, helpful criticism, valuable suggestions, commendable support and endless patience paved the way towards the completion of this thesis. I feel very proud to have worked with him. Without his enthusiasm and encouragement, this work could not have been completed.

I wish to thank Bangladesh University of Engineering and Technology (BUET) for providing an excellent environment for research and financial support in the form of scholarships and teaching facilities.

In this regard, I would like to give thanks to numerous researchers around the world who build the stepstone so that more can be achieved. Also I would also like to express my gratitude to all those people who created great tools - such as Latex, Matlab, Visual Studio etc - that simplify people lives and make research a pleasure.

I am ever grateful to my beloved parents, who always exist as the sources of inspiration behind every success of mine I have ever made.

Last but not the least, I thank Allah for the talents and abilities I was given that made it possible to undertake this research.

Dhaka
February 2017

Fahmid Morshed Fahid

Contents

CANDIDATES' DECLARATION **i**

CERTIFICATION **ii**

ACKNOWLEDGEMENT **iii**

List of Figures **vii**

List of Tables **viii**

List of Algorithms **ix**

ABSTRACT **x**

1 Introduction **1**

2 Artificial Neural Network **4**

2.1 Machine Learning 4
2.2 Supervised Learning 5
2.3 Perceptron 5
2.4 Biological Neural Network 7
2.5 History of Artificial Neural Network 8
2.6 Mathematical Model of the Neuron 9
2.7 Multi-Layer Perceptron 10
2.7.1 Back Propagation Network Training Algorithm 12
2.7.2 Training Algorithm - Practical Aspects 15
2.8 RBF Networks 16
2.8.1 RBF Network Architecture 17
2.8.2 Learning of the RBF Networks 18
2.9 Conclusion 20

3 Kohonen's Self-Organizing Maps **22**

3.1 Self-Organizing Maps General Overview 23
3.2 Self-Organizing Maps Training Algorithm 24

3.2.1 Initialization Process 25
3.2.2 Competition Process 25
3.2.3 Cooperation Process 26
3.2.4 Adaption Process 27
3.2.5 Parameters Selection 28
3.2.6 Training Algorithm 29
3.3 Other Clustering Methods 29
3.3.1 K-means Clustering 30
3.3.2 Fuzzy C-means Clustering 31
3.4 Conclusions 31

4 Stylometry **33**
4.1 Stylometry in Writting Language 33
4.2 History of Stylometry 34
4.3 Objectives of Stylometry 35
4.4 Methodologies in Practice 36
4.4.1 Statistic Based Analysis 36
4.4.2 Artificial Intelligence Techniques 37
4.5 Authorship Attribution in Instant Messaging 38
4.6 Conclusions 39

5 Real-Time User Identification **40**
5.1 Data set 40
5.2 Feature Extraction 42
5.3 Experiment and Results 45
5.4 Conclusions 47

6 Concluding Remarks **48**

References **50**

Index **53**

A Algorithms **55**
A.1 Multi-Layer Perceptron 55
A.2 RBF Gradient Training Algorithm 56
A.3 SOM Training Algorithm 57
A.4 Fuzzy C-means Clustering Algorithm 57
A.5 K-means Clustering Algorithm 58

B Codes **59**

B.1 Real-Time User Identification 59

List of Figures

2.1 Perceptron 6
2.2 Perceptron Classification 6
2.3 Biological Neurons 8
2.4 Artificial Neuron 9
2.5 Multi-Layer Perceptron 11
2.6 RBF Network 17

3.1 Willshaw-von der Malsburg Model 22
3.2 SOM Architecture 24

List of Tables

2.1 Activation Functions 11
2.2 Different $\phi(.)$ Functions 18

5.1 Data set 41
5.2 Function Words 42
5.3 Feature Vectors 45
5.4 Training Data Results 46
5.5 Validation Data Results 46
5.6 Test Data Results 47

List of Algorithms

1 Multi-Layer Perceptron Algorithm 55
2 RBF Gradient Training Algorithm 56
3 SOM Training Algorithm 57
4 FCM Algorithm 57
5 K-means Clustering Algorithm 58

ABSTRACT

In this global era, digital signature is the only key to identify a person virtually. Privacy concern thus plays a vital role. But almost all the privacy schemes that are used publicly are limited to static measures and once the initial barrier of verification is crossed, there is no way to identify the person. But every person has a unique pattern in the digital world, a data stream that he or she maintains unintentionally in a particular website, which can be approximated with proper learning algorithm. The data stream that is discussed in this book is solely confined to written natural languages that is used in communication. Although natural language is very abrupt and bizarre in its formation and use, every individual has his or her own preferences and uniqueness which can be approximated using the technique called Stylomerty, the statistical analysis of variations in literary style between one individual to another. Artificial Neural Network, which is globally used to predict patterns is a feasible tool that can use features extracted using Stylometry. In this regard, I am proposing a procedure that uses Artificial Neural Network as a Machine Learning tool to predict patterns on dynamic data generated by an individual in a particular website. This thesis book will first explore the several different types of Artificial Neural Network and the parameters of Stylometry. Then describe an algorithm to determine the different steps in dynamic detection of a digital identity using natural languages. Finally the book will contain difference matrices as a quantitative analysis to the procedure and end with possible scopes for future researchers.

Chapter 1

Introduction

With the emergence of World Wide Web (WWW[1]), digital identity is becoming more and more of a primary identification for a person. A digital identity now-a-days consists of thousands of personal data. These data are getting more valuable every day as we are getting more and more involved in the virtual world. In this global era of Internet, virtual identity is the norm of communication, wither it is a job application or a simple social chit-chat. Every interaction in the virtual world thus plays a vital role in our everyday life. For example, a wrong kind of text from a social networking site can hurt a persons' image or career badly. Thus different measures are on the play to overcome this kind of mischief.

Contemporary privacy policies are mostly different Password protection scheme. In this scheme, a user has a set of characters or numbers or a mix of both that is presumed to be unknown to other individuals, which is called a (or in some cases PIN[2]). A digital system identifies an individual by a unique user-name or email or phone number along with that set of characters or numbers or a mix of both in its exact form. This is known as a verification process. A more recent advancement in this privacy scheme is the **Two-Step Verification** that requires not only a password and user-name (or phone number or email) but also something that only, and only, that user has on them on that very moment of verification. In most cases that "something" is another temporary password that is sent to a trusted device (phone).

An Identity Thief can easily jeopardize a user if somehow the initial verification process is crossed, wither it is a single step verification or a two step verification. Different two step verification schemes have been developed over the years. For example in [1], Van proposed a two step verification where a combination of graphical and text password. A typical case is if a person forgets to log out of a system and someone else, an Identity Thief, starts using the system. In that scenario, the system still considers all the interactions on the digital world as

[1]An information system on the Internet that allows documents to be connected to other documents by hypertext links, enabling the user to search for information by moving from one document to another.

[2]Personal Identification Number

that person without noticing that the user is no longer the owner of the account.

To overcome such scenario, some system implements a scheme where a period of no-interaction automatically makes the user log out of the system. An even better solution that has been tried such as log out of the system anyway after a period of time irrespective of any interaction and ask the user to verify his authenticity. These schemes again fall short in case of immediate identity theft and besides, these schemes are not in practice widely as they ensure a small increase in authentication but tends the annoy the user.

All these different schemes are static in nature. A collection of characters is all the system has to identify a user. But every single user has a unique pattern of data streams that he or she produce digitally. For example, a user may tend to use more special characters when he or she interacts in a conversation. These pattern might not be visible in plain eyes, but using appropriate attributes for these data stream and a machine, these patterns can be learnt.

A data stream that reflects a users' behaviour rigorously is the conversation the user makes in the virtual world, wither it is Instant Messaging or a Blog or an Mail. According to **Stylistics**[3], every individual is biased towards a linguistic style when writing. For example, a user has a bias towards sentence structure, use of special characters, function words and many others. **Stylometry**(See Chapter 4), an applied branch of linguistic style (Stylistics), usually to written language, offers many different set of attributes that can be used to identify a user. Many work on Stylometry has been done in recent years. But almost all of them focus on Authorship Attribution[4]. For example in [2] Can proposed a system that identifies authors based on their writing style. Another example is Ramayaa's work where he used Machine learning to identify different genre of writing [3]. These research managed to distinguish the famous the **Federalist Papers**[5] and much more in the field of authorship attribution. But unfortunately, there is almost no work on digital user identification using Stylometry. In this book, we are proposing a Stylometric approach that can identify digital user on real-time.

An intelligent system can learn the pattern of an individual by using the attributes of Stylometry quantitatively. Such system can easily verify a user on real-time by observing the users' current data stream to his or her previous authentic data streams. In such scheme, after the initial verification, the system will only trigger a verification if a user behaves significantly different from his or her previous behavior rather than triggering after a fixed period of time. An intelligent system like this will learn from its mistake every time it predicts wrong and the initial learning will be done through a set of previous streams for an individual user.

[3]A branch of applied linguistics, is the study and interpretation of texts in regard to their linguistic and tonal style. As a discipline, it links literary criticism to linguistics.

[4]Authorship attribution is the task of identifying the author of a given text. The main concern of this task is to define an appropriate characterization of documents that captures the writing style of authors.

[5]During 1787 and 1788, 77 articles were anonymously published to persuade New Yorkers to support ratification of the new US constitution. Most of the articles were later attributed credibly to their authors, except for 12 of them which were claimed to be written by both General Alexander Hamilton and President James Madison.

An Artificial Neural Network (See Chapter 2) is a tool that is used in classification. Designing a system using Artificial Neural Network using the attributes of Stylometry thus offers feasible solution towards the authentication problems. Such a system will predict in real-time if a user is authentic to the current **Login Session**[6] and if not it may trigger a verification process either to learn from mistake or to catch the digital identify thief.

In this thesis, we have proposed and studied a system that utilizes the attributes of attributes of Stylometry and using the model of Artificial Neural Network can predict a digital users' behavior based on his or her conversations. To do so, we first discussed about Artificial Neural Network at length in Chapter 2. Then we discussed Kohonen's Self-Organizing Maps in Chapter 3 what we will use to identify the mean of the data streams. In Chapter 4, we discussed the different approaches of stylometry and its attributes so that the reader may get familiar. In our core chapter, Chapter 5 we proposed our system of real-time identification of digital users along with our experiment results. Finally, Chapter 6 ends our work by evaluating our system and proposes scopes for further research.

[6]A **Session** is a semi-permanent interactive information interchange, also known as a dialogue, a conversation or a meeting, between two or more communicating devices, or between a computer and user. A **Login Session** is the period of activity between a user logging in and logging out.

Chapter 2

Artificial Neural Network

In this chapter we define some basic terminologies of **Machine Learning**, more specifically Artificial Neural Network which will be used throughout the rest of this thesis book. We began our discussion on **Perceptrons** and their related terminologies. Then some definitions of Artificial Neural Network in general is given. Finally we discussed different types of Artificial Neural Networks.

2.1 Machine Learning

To begin our discussion, we first need to understand the concept of Machine Learning. Machine Learning is the sub-field of computer science that gives computers the ability to learn without being explicitly programmed. Evolved from the study of pattern recognition and computational learning theory in **Artificial Intelligence**[1], Machine Learning explores the study and construction of algorithms that can learn from and make predictions on data such algorithms overcome following strictly static program instructions by making data driven predictions or decisions, through building a model from sample inputs. Machine Learning is employed in a range of computing tasks where designing and programming explicit algorithms is infeasible; example applications include spam filtering, detection of network intruders or malicious insiders working towards a data breach, optical character recognition (OCR), search engines and computer vision.

There are different approaches to Machine Learning such as **Perceptron Learning**, **Artificial Neural Networks** (ANN), **Deep Learning**, **Support Vector Machines** (SVM), **Decision Tree Learning** etc.

[1]The theory and development of computer systems able to perform tasks that normally require human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages.

2.2 Supervised Learning

Supervised Learning is the Machine Learning task of inferring a function from labeled training data. The training data consist of a set of training examples. In Supervised Learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

Given a set of N training examples of the form $\{(x_1, y_1), ..., (x_N,\ y_N)\}$ such that x_i is the feature vector of the i-th example and y_i is its label (i.e., class), a learning algorithm seeks a function $g : X \to Y$, where X is the input space and Y is the output space. The function g is an element of some space of possible functions G, usually called the hypothesis space. It is sometimes convenient to represent g using a scoring function $f : X \times Y \to \mathcal{R}$ such that g is defined as returning the y value that gives the highest score: $g(x) = \arg\max_y\ f(x, y)$.

2.3 Perceptron

Perceptron is an algorithm for supervised learning of Binary Classifiers[2]. It is a type of linear classifier, i.e. a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector. The algorithm allows for online learning, in that it processes elements in the training set one at a time.

In the modern sense, the Perceptron is an algorithm for learning a binary classifier: a function that maps its input $\mathbf{x}$ (a real-valued vector) to an output value $f(\mathbf{x})$ (a single binary value):

$$[f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0 & \text{otherwise} \end{cases}] \tag{2.1}$$

where $\mathbf{w}$ is a vector of real-valued weights, $\mathbf{w} \cdot \mathbf{x}$ is the dot product $\sum_{i=1}^{m} w_i x_i$, where m is the number of inputs to the perceptron and b is the bias. The bias shifts the decision boundary away from the origin and does not depend on any input value.

The value of $f(x)$ (0 or 1) is used to classify $\mathbf{x}$ as either a positive or a negative instance, in the case of a binary classification problem. b is negative, then the weighted combination of

[2]Binary Classifiers are functions that can decide whether an input, represented by a vector of numbers, belongs to some specific class or not

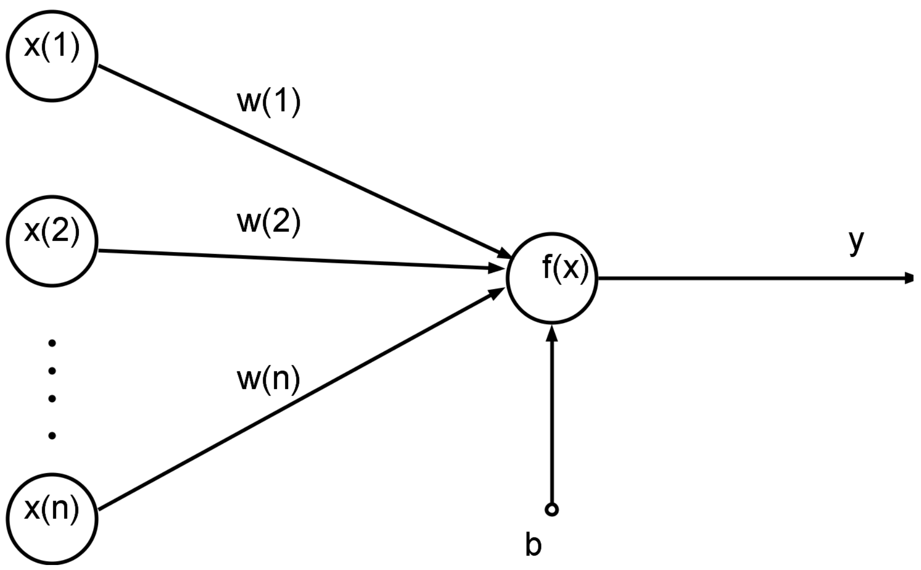

Figure 2.1: Perceptron

inputs must produce a positive value greater than $|b|$ in order to push the classifier neuron over the 0 threshold. Spatially, the bias alters the position (though not the orientation) of the decision boundary. The Perceptron Learning algorithm does not terminate if the learning set is not linearly separable. If the vectors are not linearly separable learning will never reach a point where all vectors are classified properly. The most famous example of the perceptron's inability to solve problems with linearly nonseparable vectors is the Boolean exclusive-or problem. The solution spaces of decision boundaries for all binary functions and learning behaviors are studied in the reference.

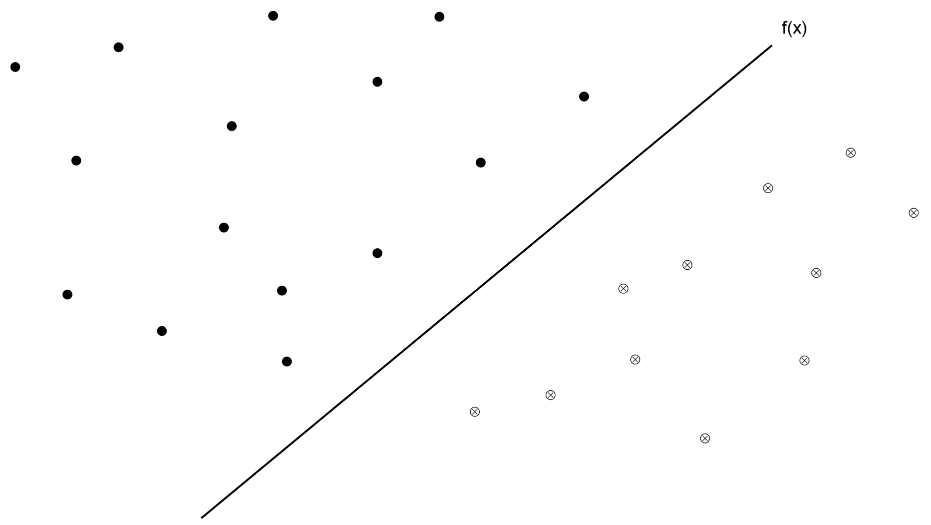

Figure 2.2: Perceptron Classification

Multiclass Perceptron

Like most other techniques for training Linear Classifiers, the Perceptron generalizes naturally to Multiclass Classification. Here, the input $\mathbf{x}$ and the output y are drawn from arbitrary sets. A feature representation function $f(\mathbf{x}, y)$ maps each possible input/output pair to a finite-dimensional real-valued feature vector. As before, the feature vector is multiplied by a weight vector $\mathbf{w}$, but now the resulting score is used to choose among many possible outputs:

$$[\hat{y} = \mathbf{argmax_y} f(\mathbf{x}, y) \cdot \mathbf{w}] \tag{2.2}$$

Learning again iterates over the examples, predicting an output for each, leaving the weights unchanged when the predicted output matches the target, and changing them when it does not. The update becomes:

$$w_{t+1} = w_t + f(x, y) - f(x, \hat{y}) \tag{2.3}$$

This multiclass feedback formulation reduces to the original perceptron when x is a real-valued vector, y is chosen from $\{0, 1\}$, and $f(x, y) = yx$.

2.4 Biological Neural Network

In neuroscience, a Biological Neural Network[3] is a series of interconnected neurons whose activation defines a recognizable linear pathway. The interface through which neurons interact with their neighbors usually consists of several axon terminals connected via synapses to dendrites on other neurons. If the sum of the input signals into one neuron surpasses a certain threshold, the neuron sends an action potential (AP) at the axon hillock and transmits this electrical signal along the axon.

In contrast, a neural circuit is a functional entity of interconnected neurons that is able to regulate its own activity using a feedback loop.

A biological neural network is an essential part of human brain. It is a highly complex system with the ability to process large amounts of information simultaneously. Biological networks are able to recognize and process different visual inputs much faster than any modern high-end computer. For example, human brain is able to recognize familiar face in about 100-200ms, while modern computer requires minutes or even hours for solving the same problem. In general, it is a known fact that in many tasks human brain is much more efficient than computers.

Based on examples and feedback from the ”teacher”, our brain allows us learning how to distinguish an apple from an orange or recognize letters. And even without the ”teacher”, we are

[3]sometimes called a neural pathway

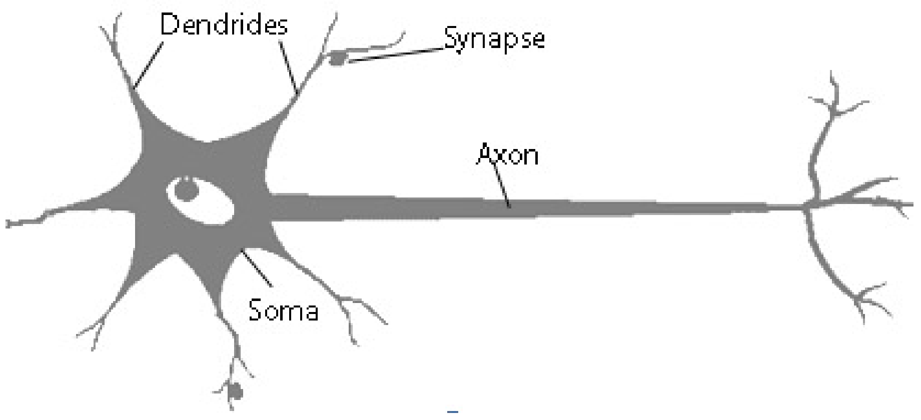

Figure 2.3: Biological Neurons

still able to group similar patterns together. Those and other strengths of human brain challenged scientists to emulate those processes by researching how to use machines for tasks that are common for humans. And one of the concepts that appeared as the result of that research, is the **Artificial Neural Network** (ANN) concept.

2.5 History of Artificial Neural Network

First steps of the ANN theory were made in 1943 by neuro-physiologist Warren McCulloch and mathematician Walter Pitts. They introduced artificial neurons with the threshold[4], that could be arranged into networks. Several years later, in 1949, a psychologist Donald Hebb designed the first learning rule for artificial neural networks - Hebb's rule. Its premise was that if two neurons were active simultaneously, then the strength of the connection between them should be increased.

In subsequent years, the ANN theory made significant progress: in 1960s more neuron architectures - perceptron and adaline - were introduced. Next years, jointly with more powerful computers, brought even more different types of ANN - self-organizing maps, associative networks, RBF-networks and more. Nowadays, artificial neural networks are widely adopted by different scientific and industrial fields - from medicine to speech recognition. More about ANN history can be found in [4].

[4]Later called McCulloch-Pitts neuron. By threshold they meant the following idea: the unit fires only if the input to the neuron is greater than the ”threshold value”.

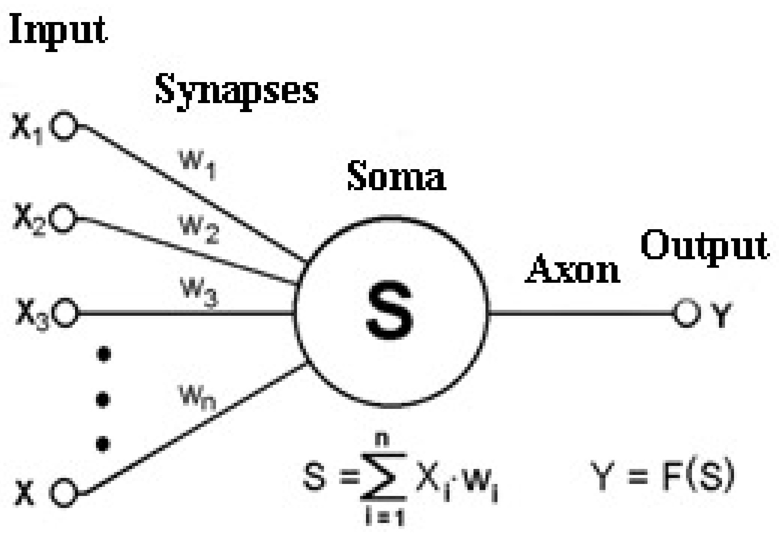

Figure 2.4: Artificial Neuron

2.6 Mathematical Model of the Neuron

Biological Neural Networks are very complex. In contrast, the mathematical model of the network is much more simplified and is based on several assumptions:

- All neurons are synchronized. That means that the signal passing from one neuron to another takes the same time for all connections. Signal processing is also synchronized and is the same for all neurons.
- Every neuron has a so-called transfer function which determines neuron's output signal depending on the input signal strength. That function is time-independent.
- When the signal passes the synapse, it changes linearly, i.e., the signal value is multiplied by some number. That number is called synaptic weight.

The very important property of the synaptic weight is that it changes in time. That feature makes it possible for brain to react differently on the same input in different moments. Or, in other words, to learn.

Of course, those assumptions simplify the initial biological neural network very much. For example, brain signal transmission time naturally depends on the distance between neurons. But despite those simplifications, artificial networks still preserve the most important characteristics of biological networks - adaptability and ability to learn.

The first mathematical model of the neuron was introduced more than a half century ago, but did not change much since then. First of all, the neuron is seen as a simple "automate" that transforms input signals into the output signal (Fig: 2.4).

The model functions as follows: inputs of the neuron's synapses receive N signals $[X_1, ..., X_n]$. Then every synapse makes a linear modification of the signal using its synaptic weight. After that neuron's body (soma) receives signals $[X_1 \cdot w_1, ..., X_n \cdot w_n]$ (where w_i is the corresponding synaptic weight) and sums those signals:

$$S = \sum_{i=1}^{n} X_i \cdot w_i \tag{2.4}$$

Then it applies some given function F (that is also called activation function) and sends the final signal

$$Y = F(S) \tag{2.5}$$

to the output.

Activation Function

There are different functions, that are commonly used as Activation Functions. In a network, it is not necessary to use the same function for all the neurons. However, it's a common practice. In most cases activation functions are non-linear. Otherwise, the whole network will implement some linear transformation and will be equivalent to only one artificial neuron - perceptron (See Section 2.3).

For $\alpha \in \mathcal{R}$, most common activation functions are given in Table 2.1

2.7 Multi-Layer Perceptron

This section describes one of the fundamental neural networks' types - Multi-Layer Perceptron (or MLP, or back propagation network). Lots of other neural networks' types - such as RBF[5] networks or probabilistic networks - are based on this model.

In general, an artificial neural network is a set of artificial neurons. Arrangement and types of those neurons depend on the network type. Multi-Layer Perceptron contains three types of neurons:

1. **Input Neurons** Those neurons are taking input vector that encodes some action or information about the external environment. Input neurons don't perform any type of computation, but only pass the input vector to subsequent neurons.

[5]Radial Basis Function network is an artificial neural network that uses radial basis functions as activation functions. The output of the network is a linear combination of radial basis functions of the inputs and neuron parameters

Function name	Formula	Values range
Linear	$F(S) = kS; k \in \mathcal{R}_+$	$(-\infty, \infty)$
Semi-linear	$F(S) = \begin{cases} kS & \text{if } S > 0, k \in \mathcal{R}_+ \\ 0 & \text{otherwise} \end{cases}$	$[0, \infty)$
Sigmoid	$F(S) = \frac{1}{1+e^{-\alpha S}}$	$(0, 1)$
Bipolar sigmoid	$F(S) = \frac{2}{1+e^{-\alpha S}} - 1$	$(-1, 1)$
Hyperbolic tangent	$F(S) = \frac{e^{\alpha S} - e^{-\alpha S}}{e^{\alpha S} + e^{-\alpha S}}$	$(-1, 1)$
Exponential	$e^{-\alpha S}$	$(0, \infty)$
Sinusoidal	$F(S) = \sin(S)$	$[-1, 1]$
Fractional	$\frac{S}{\alpha + \lvert S \rvert}$	$[-1, 1]$
Step	$F(S) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0 & \text{otherwise} \end{cases}$	$[0, 1]$
Signature	$F(S) = \begin{cases} 1 & \text{if } S \geq 0 \\ -1 & \text{otherwise} \end{cases}$	$[-1, 1]$
Binary step	$F(S) = \begin{cases} -1 & S \leq -1 \\ S & -1 < S < 1 \\ 1 & S \geq 1 \end{cases}$	$[-1, 1]$

Table 2.1: Activation Functions

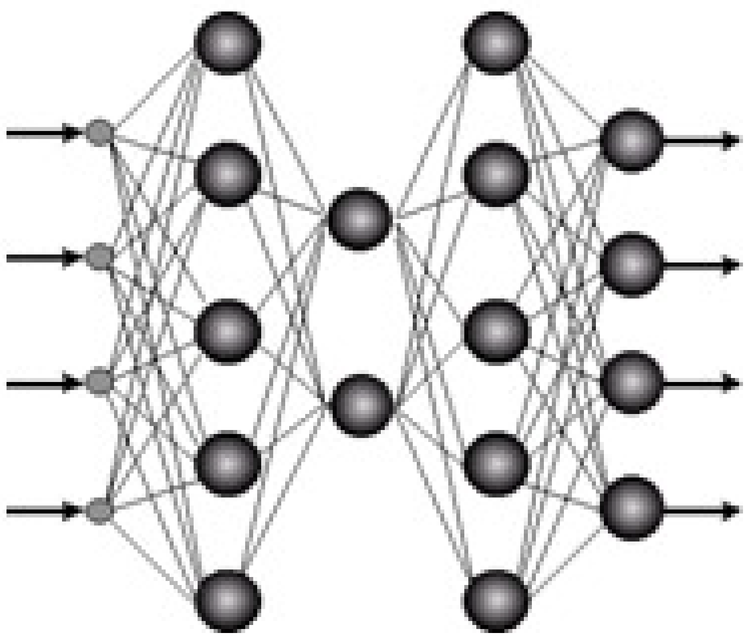
Figure 2.5: Multi-Layer Perceptron

2. **Output Neurons** receive signals from the preceding neurons and transform it using formulas 2.4 and 2.5. Those values represent output of the whole neural network.

3. **Hidden Neurons** are the basis of the neural network. Those neurons receive the sig-

nal from the input neurons or preceding hidden neurons, process it in accordance with formulas 2.4 and 2.5 and then pass result signals to the subsequent (hidden or output) neurons.

In Multi-Layer Perceptron neurons are divided into layers. Input and output neurons form separate layer each - input layer and output layer. Hidden neurons form one or several hidden layers. Every MLP neuron, with the exception of input neurons, is connected via synapses with all neurons of the previous layer. Example of the MLP architecture is shown on Fig. 2.5.

That network receives 4-dimensional input (as it has 4 input neurons). The result is represented by the 4-dimensional output vector. It also contains 3 hidden layers. In publications such network is often referred as 4-5-2-5-4.

2.7.1 Back Propagation Network Training Algorithm

Artificial Neural Networks are the powerful tool for solving different problems - such as classification or prediction. However in order to solve some particular problem, the network should be properly set up and then trained.

The very first step is the network architecture selection - e.g., how many input and output neurons should the network contain, how many hidden neurons and layers. Number of input and output units often follows the context. If the network is expected to produce boolean values, like ”Yes” or ”No”, it is clear that it should have only one output. Similar thoughts can be applied to the input neurons’ number.

In contrast, selecting the number of hidden units and layers is not simple. In most situations, there is no obvious way of how to determine the best number of hidden neurons without training several networks and estimating the generalization error for each of them[6]. Too few hidden units will produce high training error due to underfitting. Too many hidden units lead to low training error but still have high generalization error due to overfitting[7]. Some discussions about number of hidden units and layers can be found, for example, in [5] or [6].

The second step is training itself. Back Propagation algorithm belongs to the group called Supervised Learning (See Section 2.2). It means that to be trained, the network should learn on pairs (input, expected output) which all together form the training set. In contrast, some classes of networks, like Kohonen’s Self-Organizing Maps [7], belong to the **Unsupervised Learning**[8]

[6]The **generalization error** is a function that somehow measures how far the network is from the ”teacher”.

[7]In overfitting, a statistical model describes random error or noise instead of the underlying relationship. Overfitting occurs when a model is excessively complex, such as having too many parameters relative to the number of observations.

[8]Unsupervised Machine Learning is the machine learning task of inferring a function to describe hidden structure from ”unlabeled” data (a classification or categorization is not included in the observations). Approaches to unsupervised learning include: clustering. k-means.

group. It means that the training set does not contain expected outputs for the given inputs. Unsupervised Learning is often used for clustering purposes. At the beginning of the MLP's learning, network's weights are initialized randomly (often by small random values). Then the network

1. processes one by one every item in the training set,
2. compares its output with the desired output,
3. computes error value and
4. After checking the whole training set, the network modifies its synaptic weights in order to minimize the error.

Most of the MLP training methods are based on the **Gradient Descent** approach - iterative change of the network's synaptic weights that gradually decreases the error on the training set. Algorithm that implements that idea is called Error Back Propagation algorithm and can be described as follows:

1. **Initialization.** The first step is selecting the network architecture and setting its initial synaptic weights. The most common method of the synaptic weights initialization is choosing small random numbers with the zero mean value and uniform distribution.
2. **Training Pairs Presentation.** During that phase, elements of the training set are passed one by one to the network. For every training sample, there are two steps - forward computation and backward computation - that are taking place.
3. **Forward Computation.** Assume that the training set consists of N of pairs (x_i, d_i):

$$T = (\vec{x_i}, \vec{d_i}), i = 1, ..., N, \vec{x_i} \in \mathcal{R}^n, \vec{d_i} \in \mathcal{R}^m \tag{2.6}$$

where $\vec{x_i}$ is an input vector and $\vec{d_i}$ is the desired output vector.

The network passes the input vector from layer to layer and sequentially computes the outputs for all the layer's neurons. Those outputs form an input vector for the next layer. For every neuron j in the layer l (with the exception of input neurons) the output y_j^l signals are computed from the previous layer output signals y_i^{l-1} as follows:

$$S_j = \sum_{i=1}^{m_l} w_i^l y_i^{l-1} \tag{2.7}$$

$$y_j^l = F(S_j) \tag{2.8}$$

Here m_l is the number of synaptic weights for all the neurons of the layer l. That number is determined by the number of neurons of the previous layer and is the same for every layer's neuron.

At the first step the neuron calculates the linear combination of outputs of the neurons of the previous layer 2.7. The output value is calculated by applying neuron's activation function to the value computed in the first step 2.8.

For input layer - layer 0 - the formula is simpler:

$$y_j^0 = x_{ij} \tag{2.9}$$

where x_{ij} is the j^{th} element of the input vector $\vec{x_i}$.

After the input vector $\vec{x_i}$ passes all the layers, the network produces the output vector $\vec{o_i} = [y_1^{out}, ..., y_m^{out}]$, that is used for the **error signal** computation:

$$\vec{e_i} = \vec{d_i} - \vec{o_i} \tag{2.10}$$

where e_{ij} is the j^{th} element of the error vector $\vec{e_i}$ and F' is the derevative function for the function F.

4. **Backward Computation.** During that step synaptic weights are adjusted in order to produce smaller error value. Those adjustments are derived from the error signal. For each output unit j calculate δ_j^{out}:

$$\delta_j^{out} = e_{ij} F'(S_j) \tag{2.11}$$

where e_{ij} is the j^{th} element of the error vector $\vec{e_i}$ and F' is the derevative function for the function F.

For hidden neuron j of the layer l calculate δ_j^{hid}:

$$\delta_j^{hid} = F'(S_j) \sum_k \delta_k^{l+1} w_{jk} \tag{2.12}$$

where summarization is done over all neurons k of the following layer, and w_{jk} is the synaptic weight that connects neurons j and k.

When all δ are calculated, synaptic weights are adjusted using:

$$w_{jk}^{t+1} = w_{jk}^t + \Delta w_{jk}^t \tag{2.13}$$

$$\Delta w_{jk}^t = \eta \delta_j y_j^{l-1} + \mu (w_{jk}^t - w_{jk}^{t-1}) \tag{2.14}$$

where η is called **Learning Rate**, y_j^{l-1} is the output of the neuron j of the previous layer and μ is **Momentum**. Learning rate and momentum are numbers between 0 and 1. They will be discussed later.

For algorithm, look at Appendix A.

2.7.2 Training Algorithm - Practical Aspects

In the process of back propagation, two parameters are available to the modeler:

- **Learning Rate.** Learning rate is a number between 0 and 1 that determines how fast the neural network adjusts itself to the patterns in the training data. It does not have to be a constant and can also dynamically decrease or increase with time. That parameter must be chosen carefully - too small one will make the learning process slow, and too large one might lead to the divergence. Jacobs in [8] suggest modifying learning rate dynamically and increasing it as long as gradient keeps pointing the same direction, but lowering it when the gradient changes its sign.

- **Momentum.** Momentum term influences the way of how previous weights affect the current one. It helps the algorithm in preventing being stuck in a local minimum. That value must also be chosen carefully and should be determined experimentally. The use of momentum can be omitted. However it may improve neural network's performance greatly and therefore it is commonly used.

The Back Propagation process can suffer from several problems:

- Reaching of the global error function minimum is not guaranteed by the method. As there can be several local minima, the algorithm might possibly remain at one of them.

- Back propagation method might lead to overfitting: in cases where learning was performed for too long or where the training examples are rare, the network may adjust to very specific random features of the training data, that have no causal relation to the target function. The other possible cause of overfitting is having too many hidden neurons.

- There is no exact way of how to determine that the algorithm has found the best solution (i.e., global minimum). That problem can be generally approached by restarting the algorithm several times with some changes (e.g., shuffling training set) and therefore increase the probability that the reached minimum is the global one.

Despite those problems, Back Propagation is a very popular algorithm that seems to be useful in many areas ([9], [10]). Moreover, other training methods are also available - conjugate Gradient Descent, Quasi-Newton algorithm, Levenberg-Marquardt algorithm. However, those are not widely used.

Genetic Algorithm Neural Network

Another alternative training method is based on the Genetic approach. The main principle is the following: weights of the network are not adjusted by the back propagation algorithm, but by the **Genetic Algorithm**, a heuristic search method used in artificial intelligence and computing. It is used for finding optimized solutions to search problems based on the theory of natural selection and evolutionary biology.. At the beginning, the initial **Population**[9] contains randomly generated sets of weights. An evolution algorithm is then applied to the population - it generates set of new weights and keeps only the most appropriate ones. That solution generally prevents the algorithm from being stuck in a local minimum and also shows a good performance. Networks that employ that approach are generally called **Genetic Algorithm Neural Network** or GANN. Kim in [11] gives an example and evaluates those networks in financial forecasting.

2.8 RBF Networks

Radial Basis Function networks (RBF networks) is the another Artificial Neural Network model that contains several connected layers and shares some other principles with the MLP.

RBF networks are often used for solving approximation and interpolation problems. Commonly, approximation or interpolation solution itself is represented by linear combination of some Basis Functions (e.g., polynomials) that approximates the objective function with some precision. The principle is the same for RBF networks. The difference is that the RBF network uses so-called Radial Basis Functions instead of polynomials.

Formally, interpolation of a set of N data points requires all the input vectors $\mathcal{X} = \vec{x_i}|i = 1, ..., N$ to be mapped onto the corresponding target output vectors $\mathcal{T} = \vec{t_i}|i = 1, ..., N$. So the goal is to find a function $f(x)$ so that

$$f(\vec{x_i}) = \vec{t_i} \quad \forall i = 1, ..., N \tag{2.15}$$

The RBF approach employs functions, each of those functions takes the form $\phi(|\vec{x} - \vec{x_i}|)$ where $\phi(.)$ is some radial basis function and $(|\vec{x} - \vec{x_i}|)$ is some metrics, normally **Euclidean distance**.

The output of the mapping is then takes the form:

$$f(\vec{x}) = \sum_{i=1}^{N} \vec{w_i}\phi(|\vec{x} - \vec{x_i}|) \tag{2.16}$$

The last step is finding the weights $\vec{w_i}$ such that function goes through all the data points.

[9]Population is a subset of solutions in the current generation. It can also be defined as a set of chromosomes.

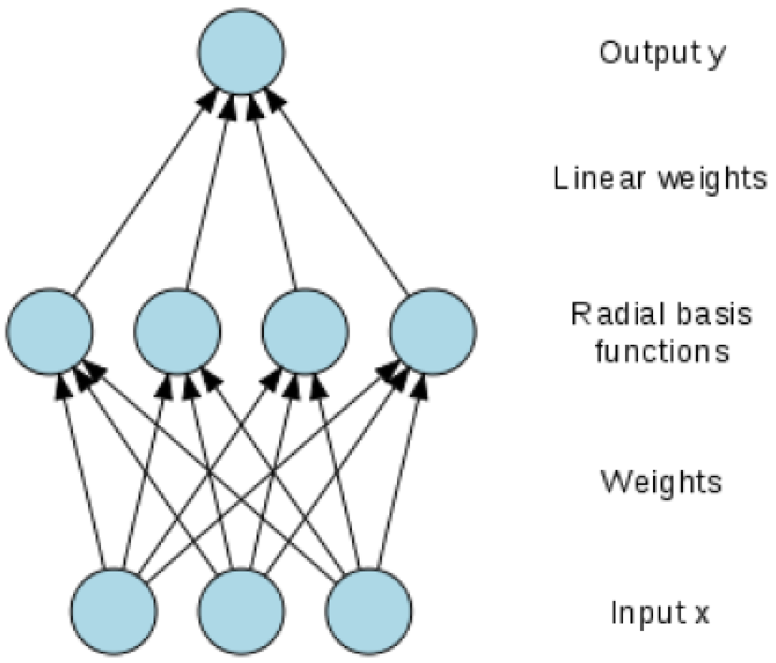

Figure 2.6: RBF Network

Radial basis function networks implement the idea described above. In order to do so, they employ a particular Artificial Neural Network Architecture that is described below.

2.8.1 RBF Network Architecture

Radial Basis Function Networks are built using so-called RBF units, and each of them implements RBF function. Every RBF unit has n inputs and one output. It also has two additional parameters that are used for the corresponding RBF function computation - **function center** $\vec{c}$ and **function width** b.

Then the output of the unit for the given input $\vec{x} = (x_1, x_2, ..., x_n)$ can be expressed as

$$\phi(\xi) = \phi\frac{|\vec{x} - \vec{c}|}{b} \tag{2.17}$$

where $\phi(.)$ is a radial basis function. The most common radial basis functions are listed below:

RBF network itself is has three layers - input layer, RBF layer and output layer - and is shown on Figure 2.6. The input layer passes the input vector $\vec{x} = (x_1, x_2, ..., x_n)$ to the RBF layer. That layer - which is a hidden layer of H RBF units - using Equation 2.17 transforms the input into new vector $\vec{y} = (y_1, y_2, ..., y_h)$ that is subsequently passed to the output layer using the linear transformation. Then the final result $\vec{f} = (f_1, f_2, ..., f_n)$ can be expressed as

Function name	Formula
Gaussian	$\phi(\xi) = \exp\frac{-\xi^2}{2\sigma^2}, \quad \sigma > 0$
Multi-quadric	$\phi(\xi) = (\xi^2 + \sigma^2)^{1/2}, \quad \sigma > 0$
Inverse multi-quadric	$\phi(\xi) = (\xi^2 + \sigma^2)^{-1/2}, \quad \sigma > 0$
Generalized multi-quadric	$\phi(\xi) = (\xi^2 + \sigma^2)^{\beta}, \quad \sigma > 0, \beta \in (0,1)$
Thin plate spine	$\phi(\xi) = \xi^2 \ln(\xi)$
Cubic	$\phi(\xi) = \xi^3$
Liner	$\phi(\xi) = \xi$

Table 2.2: Different $\phi(.)$ Functions

$$f_k(\vec{x}) = \sum_{j=1}^{h} w_{jk} y_j = \sum_{j=1}^{h} w_{jk} \phi(\frac{|\vec{x} - \vec{c_j}|}{b_j}) \tag{2.18}$$

where f_k is the output of the k^{th} unit of the output layer.

2.8.2 Learning of the RBF Networks

Similarly to MLP, RBF networks use Supervised Learning for the training. So the main idea remains the same: based on the training set, at every step the error value is calculated and subsequently used for the network parameters adjustment.

RBF networks employ two different approaches of how to train them:

1. Gradient Training
2. Hybrid Training

In that section I will describe the main idea of both of those methods. More information can be found in [12].

Gradient Training

Because RBF networks are very similar to MLP, the same training idea can also be applied to the RBF learning. The only difference is that RBF networks parametrize not only their weights, but also RBF units (by setting custom centers and widths). The main goal of the training process can be stated as setting all required parameters in order to minimize overall training error. Then the common rule for that step can be expressed as:

$$p(n+1) = p(n) - \alpha \frac{\delta E}{\delta p} \tag{2.19}$$

where p is the value of some parameter during the iteration n, α is the learning rate and n is an iteration. That approach reminds the MLP Gradient Descent learning algorithm (See Section 2.7).

See Appendix A for the algorithm.

RBF Hybrid Training

Hybrid training is based on the fact, that RBF network parameters can be divided into several groups - parameters that determine location (centers), parameters that determine size (widths) and weights. Hybrid training is using that classification in order to set those parameters separately in tree steps.

Hybrid Training Algorithm

1. **Set RBF Units' Centers.** The idea behind that process is very simple: at some parts of the input space, there is a lot of items and therefore that part should be ”covered” by the large number of the RBF units. In contrast, few items may be ”covered” by a significantly smaller number of the RBF units. For finding those confluence regions, the training algorithm employs one of the clustering methods - such as Kohonen's self-organizing maps or K-means. Then it uses found clusters for setting the appropriate centers' values.

2. **Select b_i.** If centers of RBF units define their ”location”, then the width can be seen as the ”size” of the unit. It is important to set those parameters not too small as that poor selection may create areas into the input space that are not covered by RBF units at all. At the same time, widths should not be too large, because in that case RBF units will collide and that is also not desirable. The one of the possibilities of how to set width is based on a simple idea - the width should be in a direct proportion to the distance between the unit and its closest neighbor. The closer is the closest RBF unit, the smaller width value will suffice.

3. **Weight Vectors.** While steps 1 and 2 are setting parameters of the hidden layer, the last step sets up weights that are connecting hidden and output layers. That can be done, for example, by solving the system of linear equations. Because for every x_i the corresponding output should be:

$$f_k(\vec{x_i}) = \sum_{j=1}^{h} w_{jk} y_j = \vec{t_i^k} \tag{2.20}$$

That condition can also be expressed in matrix notation:

$$\begin{bmatrix} y_1(\vec{x_1}) & \dots & y_h(\vec{x_1}) \\ \vdots & \ddots & \vdots \\ y_1(\vec{x_k}) & \dots & y_h(\vec{x_k}) \end{bmatrix} \begin{bmatrix} w_{11} & \dots & w_{1m} \\ \vdots & \ddots & \vdots \\ w_{h1} & \dots & w_{hm} \end{bmatrix} = \begin{bmatrix} t_1^1 & \dots & t_1^m \\ \vdots & \ddots & \vdots \\ t_h^1 1 & \dots & t_h^m \end{bmatrix} \tag{2.21}$$

Or,

$$\mathbf{Y} \cdot \mathbf{W} = \mathbf{T} \tag{2.22}$$

Because the number of the training pairs is normally greater than the number of the RBF units, equations above will have more than one solution. Then the solution with the smallest error should be chosen. The other approach of how to determine weight values is based on the **Least Square Method** and is described in details in [12].

2.9 Conclusion

In that chapter I have described two models of Artificial Neural Networks: Multi-Layer Perceptrons and RBF networks. Those networks both share a lot of similarities: both of them contain several layers of neurons and they can employ the same principle of the error back propagation learning. They also both play role of the non-linear mappers between multidimensional spaces. In both cases mappings are expressed in terms of composition of functions of one variable. At the same time, there are substantial differences between those two types:

1. Network architectures may be different: while RBF networks have only one hidden layer, MLP networks may have one or more hidden layers.

2. MLP nodes usually use the same activation function, while RBF nodes use different functions that are parametrized by centers' and widths' values.

3. MLP learning process determines all the parameters at the same time, while RBF learning requires two or more stages of learning to set up the network.

The other substantial difference is that - MLP constructs global approximations, while RBF construct local approximations. That means that for a given input vector, in MLP many units contribute to determine the output value. By contrast, RBF networks form a representation in the space of hidden units which is local. So for a given input vector, only few of the hidden RBF-units will have significant activations.

Choosing between RBF and MLP networks should be motivated by the nature of the input space. For more ”clustered” input spaces where some inputs may be considered less relevant then the others, RBF networks may perform better. While in more uniformly distributed spaces, MLP may be the better choice. However, there is no exact rule or algorithm for choosing between those two alternatives and some researchers suggest trying both. The other alternative is the combined RBF-MLP network that tries to take advantage of both MLP and RBF networks [13].

For the purpose of my research, I have chosen MLP networks. The first reason for that choice was the greater architecture provided by the MLP networks. The second and more important reason is the nature of the problem I will try to solve. Using clustering methods, in Chapter 3 I will show that the nature of the input space I will use can not be naturally divided into a set of localized clusters.

Chapter 3

Kohonen's Self-Organizing Maps

Self-Organizing Maps (SOM) were invented by Tuevo Kohonen in 1984. That neural networks type proved to be a very powerful clusterization and visualization tool and is known as one of the most popular models in the unsupervised learning category.

The main idea of the self-organized neural network was inspired by nature: SOM is based on the same principle of self organization as the human brain. Human visual perception and brain together make a very complex cognition system. More than 90% of the information is entering brain through the visual inputs. Those inputs are then mapped onto corresponding areas of the cerebral cortex, so that neurons dealing with the closely related information are close to each other and can interact via short synaptic connections.

Self-Organizing Model employs the same principle: it perceives some complex input and maps it onto lower dimensional surface of artificial neurons by activating some of them. Also, same as it happens in brain, similar signals activate nearby neurons. As a result, self-organizing maps can, in some sense, simplify multi-dimensional input space and visualize it in human-friendly two or three dimensions.

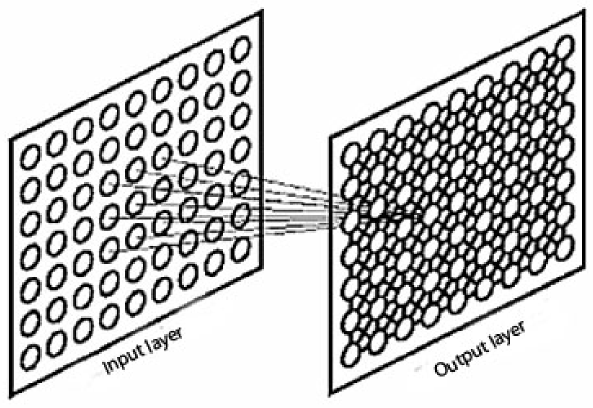

Figure 3.1: Willshaw-von der Malsburg Model

It is worth mentioning that Kohonen's model was not the first one inspired by the brain self-organization features. Its predecessor - **Willshaw-von der Malsburg Model** - was introduced in 1976 [14]. It contained two two-dimensional lattices (also called layers) of the exactly same size - one for the input and one for the output (in contrast, the SOM's input is one-dimensional, as we will see later in that chapter). Those lattices were interconnected using synaptic weights and nearby neurons on the input layer activated nearby neurons on the output layer (Fig. 3.1).

Kohonen's model appears to be more general then the Willshaw-von der Malsburg model (that is specialized to mapping inputs and outputs of the same dimensions) [15]. What Kohonen proposed was a much simplified learning mechanism that is capable of performing data compression and don't require the same dimensions of the input and the output. That is the reason why that model became much more popular then the previous one.

3.1 Self-Organizing Maps General Overview

SOM's architecture differs from the Multi-Layer Perceptron. Same as Willshaw-von der Malsburg model, it contains only two layers. The first layer is the input layer that only passes input vector to the next layer. The second layer is represented by the two-dimensional[1] lattice of interconnected neurons. Every neuron of the input layer is connected to the every neuron of the output layer. Typical SOM architecture is shown on Fig. 3.2.

Output lattice is often two-dimensional and is organized as a rectangular grid. Every neuron of that layer represents some class of possible inputs. That is done by assigning it a numerical n-dimensional **weight vector** that is also sometimes called **synaptic weight**. Also, every output neuron is connected with its closest neighbors.

Like most artificial networks, SOM operates in two modes: training mode and mapping mode.

1. **Training Mode.** At the beginning, newly created SOM does not now anything about the input space it should represent and therefore can not work correctly. So the first step is to teach the network of how to classify possible inputs. That process is called self-organization or SOM training and is very important. Training phase will be described in details later in this chapter.

2. **Mapping Mode.** Mapping is performed by the trained network. It automatically classifies a new input vectors.

[1]Higher or lower dimensions are also possible, but not common.

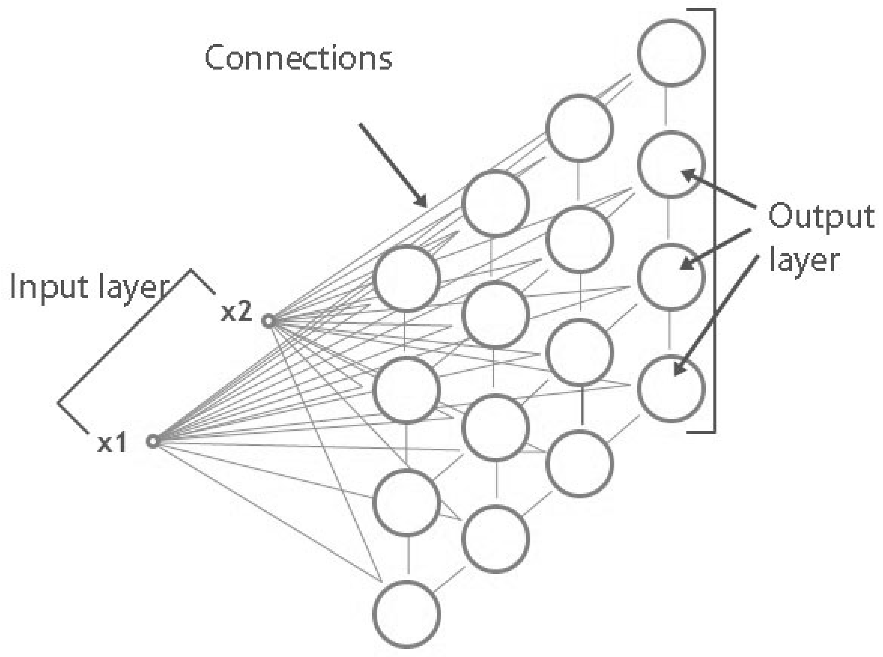

Figure 3.2: SOM Architecture

3.2 Self-Organizing Maps Training Algorithm

As mentioned before, SOM training is the most important phase because it de facto determines quality of the whole network and its ability to process input data correctly. In that section that process is explained in details.

SOM training algorithm can be divided into four main steps that can be summarized as:

1. **Initialization.** During that phase synaptic weights are being initialized. That can be done by assigning them some small random values. Also, it's important to select the right size of the output layer that will represent the input space properly.
2. **Competition.** For each input pattern from the **training set**, neurons are competing with each other by computing their output (i.e., as scalar product of the input vector and output neuron weight vector) and on the ground of this computation the particular neuron is declared as a winner of the competition.
3. **Cooperation.** During that step topological neighborhood of the winning neuron is being determined. Thereby providing the basis for cooperation.
4. **Synaptic Adaptation.** Weight vectors of the winning neuron and neurons from its topological neighborhood are being adjusted under the impact of the input vector.

3.2.1 Initialization Process

During the initialization process network parameters (e.g., number of output layer neurons and synaptic weights) are set. It's important to select good initial values to make training more effective.

The very first step is the output map size selection. On the one hand it's important to select number of neurons that will represent the input space, on the other hand if the map is too large it may not be efficient. The other important step is to select neurons' initial weights. There are three main possibilities how to do so:

1. **Random Initialization.** Vectors are being initialized with the small random values.
2. **Sample Initialization.** Initial neurons' weights values are selected among sample values.
3. **Linear Initialization.** In that case initial weights are being initialized on the basis of the training set principal **eigenvectors**[2] (that could be found by using Gram-Schmidt procedure). The main goal of that method is to distribute initial synaptic weights uniformly.

Other important initialization parameters - initial constants, neighborhood and initial radius - will be discussed later.

3.2.2 Competition Process

Let

$$\mathbf{X} = \mathbf{x_i} | \mathbf{x_i} = \{x_{i1}, x_{i2}, \ldots, x_{im}\}^T, i = 1, 2, \ldots, N \tag{3.1}$$

be the training set which contains N of n-dimensional vectors.

Then let

$$S = j | \mathbf{w_j} = \{w_{j1}, w_{j2}, \ldots, w_{im}\}^T, j = 1, 2, \ldots, M \tag{3.2}$$

be the set of the output lattice neurons. There are M neurons on the output layer (their order is not important at that moment), and every neuron j is assigned a weight vector $\mathbf{w_j}$ of the same dimensionality as every of the input vectors.

During the competition process vectors from the training set are being submit, one by one, to the network's input layer that passes it to the output layer. The next step is finding a **winning neuron** (also called a **best-matching neuron** or simply a **winner**) - the neuron of the output layer, that, in some sense, corresponds the given input vector best of all. The most established

[2] A vector that when operated on by a given operator gives a scalar multiple of itself.

approach for the winner selection is based on the inner product criterion: neuron j with the maximum value of inner product $\mathbf{w_j^T x}$ is selected as a winning neuron.

As known from the geometry, inner product is equivalent to the distance between two vectors in some Euclidean space, so the inner product condition can be rewritten as:

$$i(\mathbf{x}) = argmin|\mathbf{x} - \mathbf{w_j}|, j = 1, 2, \dots, M \tag{3.3}$$

where i is a winning neuron and —.— stands for Euclidean distance.

3.2.3 Cooperation Process

One of the basic SOM properties is that in trained network, neurons that are spatially located close to each other also represent similar classes of input space patterns. That property can be achieved during the learning process when the winning neuron is taken as the center of some topological neighborhood where all neurons are cooperating. When the winner is selected and its weight vector changes, all neighborhood neurons' weights are adjusted as well. Those adjustments depend on how far the adjusted neuron is located from the winner.

That section explains the process of cooperation in details and also answers the question: what is the topological neighborhood and how it changes in time.

Definition. Let i and j be two neurons on the output layer. Then the lateral distance between those two neurons $d_{i,j}$ can be defined as $d_{i,j}^2 = |\mathbf{r_i} - r_j|^2$ where r_i defines the position of the neuron i, and r_j defines the position of the neuron j. Both are in the discrete n-dimensional output space.

For example, for one-dimensional space $d_{i,j}$ is an integer equal to $|i - j|$ where i, j are neurons' coordinates on the one-dimensional line. For two-dimensional case, $d_{i,j}$ turns into Euclidean distance between neurons i and j, where neuron's position is defined by its coordinates on the lattice.

Definition. Neighborhood (or **topological neighborhood**) of the neuron i with the neighborhood radius σ (sometimes also called **effective width**) is the set of neurons with the lateral distance $d_{i,j}$ smaller then σ:

$$N(i, r) = \{j | d_{i,j} \leq \sigma\}, j = 1, 2, \dots, M \tag{3.4}$$

For the topological neighborhood we can also define a **neighborhood function**.

Definition. Let $N(i, \sigma)$ is the topological neighborhood, centered in the neuron i. Then neighborhood function $h_{i,j}$ is a a function of the lateral distance $d_{i,j}$ defined on the $N(i, \sigma)$, such that satisfies:

1. The topological neighborhood $h_{i,j}$ is symmetric to the central neuron for which $d_{i,j} = 0$ and $h_{i,j}$ has its maximum at that neuron.
2. Topological function $h_{i,j}$ decreases monotonically with the lateral distance $d_{i,j}$ increasing and $\lim_{d_{i,j} \to \infty} h_{i,j} = 0$. This condition is necessary for convergence.

A typical example of the topological function is the Gaussian function:

$$h_{i,j} = \mathrm{e}^{-\frac{d_{i,j}^2}{2\sigma^2}} \tag{3.5}$$

One of the features of the SOM training algorithm is that topological neighborhood shrinks with time. It means that parameter σ is a function that is decreasing in time. A popular choice for the $\sigma(n)$ is the exponential decay:

$$\sigma(n) = \sigma_0 \mathrm{e}^{-\frac{n}{\tau_1}} \tag{3.6}$$

where σ_0 indicates initial neighborhood radius and τ_1 is a time parameter. Therefore, topological neighborhood function can now be written as:

$$h_{i,j}(n) = \mathrm{e}^{-\frac{d_{i,j}}{2\sigma^2}} \tag{3.7}$$

3.2.4 Adaption Process

The last part of the learning process is the **Synaptic Adaptation**. That means that for every neuron within winner's neighborhood $N(i, \sigma)$, its synaptic weights need to be adjusted with a glance of the input vector $\mathbf{x}$. That adjustment could be made using generalized Hebb's rule:

$$\Delta \mathbf{w_j} = \eta y_j \mathbf{x} \tag{3.8}$$

where η is the learning rate parameter and y_j is the response of the neuron j to the input x.

However, the Hebb's rule in its initial form is not good for the Unsupervised Learning and should be modified by including the **forgetting term** $g(y_j)\mathbf{w_j}$, where $\mathbf{w_j}$ is a synaptic weight of the neuron j and $g(y_j)$ is some positive scalar function of the response y_j . The only requirement on the function $g(y_j)$ is

$$y_j = 0 \implies g(y_j) = 0 \tag{3.9}$$

Given such function, the synaptic change of the output neuron j can be expressed as follows:

$$\Delta \mathbf{w_j} = \eta y_j \mathbf{x} - g(y_j)\mathbf{w_j} \tag{3.10}$$

The simplest and the most obvious choice for the forgetting term is the linear function:

$$g(y_j) = \eta y_j \tag{3.11}$$

It's obvious that is satisfies the condition 3.9. 3.10 can be also simplified by setting

$$y_j = h_{i,j} \tag{3.12}$$

where i is the winning neuron. Then, using 3.9, 3.10 and 3.11 we get:

$$\Delta \mathbf{w_j} = \eta h_{i,j}(\mathbf{x} - \mathbf{w_j}) \tag{3.13}$$

And finally, adding discrete time:

$$\mathbf{w_i}(n+1) = \mathbf{w_i}(n) + \eta(n)h_{i,j}(n)(\mathbf{x} - \mathbf{w_i}(n)) \tag{3.14}$$

that is applied to all the neurons that are inside of the topological neighborhood of the winning neuron i.

In 3.14 the Learning-Rate parameter $\eta(n)$ should be time varying. In particular, it should start at initial value η_0 and then decrease gradually with increasing time n. That requirement can be satisfied by choosing

$$\eta(n) = \eta_0 \mathrm{e}^{-\frac{n}{\tau_2}} \tag{3.15}$$

Here τ_2 is another SOM algorithm constant.

Equation 3.14 visually means moving of the winner's synaptic weights vector $\mathbf{w_i}$ towards the input vector $\mathbf{x}$. Neurons in the winner neighborhood are impacted as well, but in proportion to the neighborhood function value. Repeating presentation of the training data means that synaptic neuron weights tend to follow input vector distribution. The algorithm therefore leads to a SOM layer topological ordering in sense that neurons that are adjacent in the lattice will tend to have similar synaptic weights vectors.

3.2.5 Parameters Selection

It is important to select initial training parameters well. In [16] Haykin argues for the following selection:

1. The Learning-Rate parameter $\eta(n)$ should begin with value close to 0.1 and then decrease gradually, but remain above 0.01 and not allowed to be decreased to zero. The values of

the initial parameters for the formula 3.15 that satisfy that demand may be set as:

$$\eta_0 = 0.1$$

$$\tau_2 = 1000$$

2. The Neighborhood Function $h_{i,j}(n)$ should initially include all or almost all the neurons of the lattice, centered in the winning neuron i and then diminish slowly with time. To satisfy that, in 3.5 we may choose:

$$\tau_1 = \frac{1000}{\log \sigma_0}$$

At the end of the training, the neighborhood function should contain only the closest neighbors of the winning neuron.

3.2.6 Training Algorithm

Now we can describe the SOM algorithm. The first step is initialization of the key parameters that were discussed in the previous section. In short, we need to initialize:

- **Training Patterns.** Those vectors should be selected carefully and reflect the input
- **Network Topology** i.e., number of the output neurons, their locations and initial weights.
- **Time-Varying Neighborhood Function** $h_{i,j}(n)$ that decreases in time and was described in details above.
- A Learning-Rate parameter $\eta(n)$ that starts at initial value η_0 and then decreases.

The algorithm is then can be summarized as Algorithm 3. In Algorithm 3 while cycle tests stopping condition. It can be, for example, whether maximum number of iterations is reached or there are no significant changes in the map.

See Appenxid for the algorithm.

3.3 Other Clustering Methods

Kohonen's SOM is not the only one clustering method and there are different alternatives to it. Two of them - **K-means clustering** and **Fuzzy c-means clustering** (FCM) - will be in short described in that section.

3.3.1 K-means Clustering

K-means Clustering is one of the simplest and the oldest clustering methods. It is called K-means, because the number of clusters K should be fixed "a priori".

In general, the K-means clustering idea can be described as following: given a set of N data points (observations) $\mathbf{X} = \{\mathbf{x_1}, \dots, \mathbf{x_N}\}, \mathbf{x_i} \in \mathcal{R}^\backslash$, the goal is to partition those observations into K sets $S = \{S_1, \dots, S_K\}$ so that the value of the function 3.16 is minimal.

$$F(\mathbf{X}) = \sum_{i=1}^{k} \sum_{x_j \in S_i} |x_j - c_i|^2 \tag{3.16}$$

Here c_i is the mean of all points in S_i. The algorithm that fulfills that idea is shown in the procedures below.

1. Select initial K points from the set $\mathbf{X}$ as representatives for each cluster (centroids) $\mathbf{C} = \{\mathbf{c_1}, \dots, \mathbf{c_K}\}$. These points can be selected randomly, or using some heuristics.
2. Assign each point from $\mathbf{X}$ to the cluster, that has the closest centroid.
3. When all points from $\mathbf{X}$ are assigned, recalculate centroids using the following formula:

$$\mathbf{c_i^n ew} = \frac{1}{|S_i|} \sum_{\mathbf{x_j} \in S_i} \mathbf{x_j} \tag{3.17}$$

4. Repeat Steps 2 and 3 until centroids no longer move.

The K-means clustering algorithm is simple and is relatively undemanding on the resources (comparing to Kohonen's SOM, for example). However, it has some weaknesses. Such as:

- The ways of how to initialize centroids is not specified. However, the result may depend on that selection.
- Number of clusters K should be fixed in advance and the result depends on that value. On the other hand, determining that number for a complex data sets may become a very non-trivial task.
- The result also depends on the metric, that is used to measure $|x_j - c_i|$.

  See Appendix A for algorithm.

3.3.2 Fuzzy C-means Clustering

Fuzzy C-means Clustering employs a very similar idea to the K-means algorithm - it divides a set of observations into several clusters. However, in contrast to the previous method it also maintains one additional function - a degree of belonging to clusters. Therefore, each data item may belong to more than one cluster and there is a function $u_i(\mathbf{x_j}) = u_{ij}$ that measures the degree of being in the cluster S_i for the item $\mathbf{x_j}$. Also, the sum of all the coefficients must be designed to be equal 1:

$$\sum_{i=1}^{K} u_{ij} = \sum_{i=1}^{K} u_i(x_j) = 1, \quad \forall x_j \in X \tag{3.18}$$

Then the main goal of the FCM algorithm is to minimize the the following objective function:

$$F(\mathbf{X}) = \sum_{i=1}^{K} \sum_{x_j \in S_i} u_{ij}^m |x_j - c_i|^2, \quad 1 \leq m \tag{3.19}$$

Here c_i is the mean of all points in S_i. The algorithm of the fuzzy partitioning is iteratively optimize the objective function 3.19 by updating parameters u_{ij} and c_i using the following rules:

$$c_i = \frac{\sum_{j=1}^{|X|} u_{ij}^m x_j}{\sum_{j=1}^{|X|} u_{ij}^m} \tag{3.20}$$

$$u_{ij} = \frac{1}{\sum_{k=1}^{|S|} \left(\frac{|x_i - c_j|}{x_i - c_k}\right)^{\frac{2}{m-1}}} \tag{3.21}$$

See Appendix A for algorithm.

3.4 Conclusions

In that chapter I've described tree methods: Kohonen's self-organizing maps, K-means and Fuzzy c-means clustering. All those algorithms are used for similar tasks - grouping similar input space items together.

K-means and FCM clustering methods are very similar. They both are starting with the pre-defined number of random clusters and then gradually recalculate those clusters. The only difference between them is one additional parameter - degree of belonging - for FCM. Both those algorithms are rather simple and effective when the number of result clusters in known in advance. Those algorithms also have their drawbacks - for example, clusters depend on the initial state and the order of the input. But the main obstacle for their practical usage is the need to know number of clusters in advance.

In contrast, Kohonen's SOM employs different principle - it maps highly dimensional input space into lower dimensional output space. Basically Kohonen's maps rearrange the data in order to preserve some topological characteristics of the input space. That method does not operate with clusters directly and therefore knowing number of clusters in advance is not necessary.

Because my research is based on highly dimensional and complex data, I have decided to use Kohonen's maps mainly because they can operate on data without additional input information (such as number of clusters).

Chapter 4

Stylometry

In this chapter, we are going to discuss Stylometry, a quantitative analysis of some written text that yields information about the style it is composed with and through that about the author of this text. The chapter begins with the definition of stylometry. Its history and different methods are the discussed at length. Finally the chapter ends with the reasoning of how our work will be used with stylometry.

4.1 Stylometry in Writting Language

Stylometry is the application of the study of linguistic style. The application domain is usually confined to written language, but it has successfully been applied to music and to fine-art paintings as well. (Source: Wikipedia)

Main stylometric tasks, belonging within information retrieval domain [17], they are considered author characterisation, similarity detection, and finally, considered as the most important, author identification. The later one has legal as well as academic and literary applications, ranging from the question of the authorship of Shakespeare's works to forensic linguistics.

Author Characterisation brings conclusions about the author, such as gender, education, social background etc. Similarity Detection involves comparing texts of several authors in order to find, if they exist, some properties in common. Author Identification (or Attribution) means attributing an unknown text to a writer basing on some feature characteristic or measure. It can be used when several people claim to have written some text or when no one is able or willing to identify the real author of this text.

Stylometry is used for detect the plagiarism, which is a very serious issue in education system. Stefan Gruber and Stuart Noven proposed a software tool that support detection of plagiarism in 2005 [18]. For Communication purpose now a day, the messages are passed through e-mail.

The misuse of e-mail is increasing day by day. The people are doing crime by emails and they send spam messages, hoaxes and threats. Therefore, it is important to properly identify the author of the e-mail. There has been growing interest in applying stylometry to the content generation where the content is checked whether it is original or copied from others style. An example of an e-mail hoax is sending a false computer virus warning with the request to send the warning on to all the recipients, thus the mail server time and bandwidth would be wasted. Computer viruses or worms are now commonly distributed by e-mail, by making use of loose security features in some e-mail programs. These worms copy themselves to all of the addresses in the recipients address book. Author identification have grown in several different areas in past years in practical manner such as, civil law in which identification of copyright and estate disputes, criminal law in which identification of writers of ransom notes and harassing letters, and computer security in which mining email content are identified [19]. The analysis of the texts for evidence of authenticity, authorial identity has also increased the stylometry techniques. The English professor John Burrows concluded that the intellectual propensities of the authors display inherently and written texts have a particular style. If we dont know the authorship of the word-use patterns in a text and then comparing and contrasting those patterns to the patterns in texts of known authorship, the similarity and dissimilarities of the textual patterns can provide supporting evidence for or contradicting evidence against an assertion of authorship [20].

4.2 History of Stylometry

Stylometry grew out of earlier techniques of analyzing texts for evidence of authenticity, authorial identity, and other questions. An early example is Lorenzo Valla's 1439 proof that the Donation of Constantine was a forgery, an argument based partly on a comparison of the Latin with that used in authentic 4^{th} century documents.

The modern practice of the discipline received major impetus from the study of authorship problems in English Renaissance drama. Researchers and readers observed that some playwrights of the era had distinctive patterns of language preferences, and attempted to use those patterns to identify authors in uncertain or collaborative works. Early efforts were not always successful: in 1901, one researcher attempted to use John Fletcher's preference for ”'em,” the contractional form of ”them,” as a marker to distinguish between Fletcher and Philip Massinger in their collaborationsbut he mistakenly employed an edition of Massinger's works in which the editor had expanded all instances of ”'em” to ”them” [21].

The basics of stylometry were set out by Polish philosopher Wincenty Lutosawski in Principes de stylomtrie (1890). Lutosawski used this method to build a chronology of Plato's Dialogues.

The development of computers and their capacities for analyzing large quantities of data enhanced this type of effort by orders of magnitude. The great capacity of computers for data analysis, however, did not guarantee quality output. In the early 1960s, Rev. A. Q. Morton produced a computer analysis of the fourteen Epistles of the New Testament attributed to St. Paul, which showed that six different authors had written that body of work. A check of his method, applied to the works of James Joyce, gave the result that Ulysses was written by five separate individuals, none of whom had any part in *A Portrait of the Artist* as a Young Man. [21] (Source: Wikipedia)

In time, however, and with practice, researchers and scholars have refined their approaches and methods, to yield better results. One notable early success was the resolution of disputed authorship in twelve of **The Federalist Papers**[1](See [22]). While questions of initial assumptions and methodology still arise (and, perhaps, always will), few now dispute the basic premise that linguistic analysis of written texts can produce valuable information and insight. (Indeed, this was apparent even before the advent of computers: the successful application of a textual/linguistic approach to the Fletcher canon by Cyrus Hoy and others yielded clear results in the late 1950s and early '60s.)

In April 2015, researchers using stylometry techniques identified a play, *Double Falsehood*, as being the work of William Shakespeare. Researchers analyzed 54 plays by Shakespeare and John Fletcher and compared average sentence length, studied the use of unusual words and quantified the complexity and psychological valence of its language.

4.3 Objectives of Stylometry

The primary aim of stylometry is to remove uncertainty about the author of some text, which can be used in literary tasks of textual analysis for works edited, translated, with disputed authorship or anonymous, but also with forensic aspect in view to detect plagiarism, forgery of the whole document or its constituent parts, verify ransom notes, etc.

Stylometric analysts claim that each writer possesses some unique characteristic, called the authorial or writer invariant, that keeps constant for all texts written by this author and different for texts of other authors. To find writer invariants there are used style markers which are based on textual properties belonging to either of four categories: **lexical, syntactic, structural, and content-specific**.

Lexical Descriptors provide statistics of total number of words or characters, average number of words per sentence, characters per sentence or characters per word, frequency of usage for

[1]**The Federalist** (later known as The Federalist Papers) is a collection of 85 articles and essays written (under the pseudonym Publius) by Alexander Hamilton, James Madison, and John Jay promoting the ratification of the United States Constitution. Seventy-seven were published serially in The Independent Journal and The New York Packet between October 1787 and August 1788.

individual letters or distribution of word length.

Syntactic Features reflect the structure of sentences, which can be simple or complex, or conditional, built with punctuation marks. **Structural attributes** express the organisation of text into paragraphs, headings, signatures, embedded drawings or pictures, and also special font types or its formatting that go with layout. **Content-Specific** properties recognise some keywords: words of special meaning or significant importance for the given context.

4.4 Methodologies in Practice

Stylometry evolved mainly from historical textual analysis methods dedicated to proving or disproving authenticity of documents or settling questions of authorial identity for anonymous or disputed texts.

As early as in 1439 Lorenzo Valla proved the forgery of the Donation of Constantine by comparing the Latin used in other documents dated to 4th Century that were unquestionably original. Yet these early attempts could hardly rely on anything else but striking elements of texts such as distinct vocabulary or specific language structures.

The new era for Stylometry downed in 1887 when Mendenhall proposed to use not qualitative but quantitative measures such as word length, its average and distribution or word frequency.

This was followed by Yule and Morton, in 1938 and 1965, who selected sentence length as descriptive feature for authorship identification.

Numerical measurements of texts were not fully exploited at first but the development of computers with their high and permanently increasing computational powers made possible the application of **Statistical-Oriented Analysis** to constantly growing corpus of texts in the cyberspace of Internet, enabling also to employ algorithms from Machine Learning (See Chapter 2) domain to Stylometric tasks.

Contemporary Stylometric procedures are typically representatives of either computer-aided **Statistic Based Analysis**, or **Artificial Intelligence Techniques**.

4.4.1 Statistic Based Analysis

In **Statistical Analysis** there are used computations of probabilities and distributions of occurrences for single letters or other characters such as punctuation marks, words, patterns of words or sentences [23].

One such method, called QSUM or CUSUM, was developed by Jill M. Farringdon [24]. The name of this method comes from the step of calculating the cumulative sum for two textual features. The first of these is the sentence length whose deviations from the average are plotted as the graph for the whole text sample of some known author. As the second descriptor typically there is chosen either the usage of the 2 and 3 letter words, using words starting with a vowel, or the combination of these two together. The two descriptors reflect the writing habits and are the key to detecting the author. If the two graphs match, the writer is identified.

Markovian Models consider a text as a sequence of characters (letter, punctuation marks, spaces, etc.) that corresponds to a Markov chain [25]. In probabilistic model of natural language letters appear with some probability, depending on which characters precede them. In the simplest model there is considered only the immediate predecessor which gives rise to the **1st order Markov chain**. Thus for all pairs of letters in the alphabet there are obtained matrices of transition frequencies of one letter into another. These statistics are calculated for all texts by known authors and for some unattributed text as the true author there is selected the one with the highest probability [26].

4.4.2 Artificial Intelligence Techniques

In Artificial Intelligence, a wide verity of approaches are used. Some are fairly simple while others are tricky. A few of them are described in short.

Writer Invariant

The primary Stylometric method is the writer invariant: a property held in common by all texts, or at least all texts long enough to admit of analysis yielding statistically significant results, written by a given author. An example of a writer invariant is frequency of function words used by the writer.

In one such method, the text is analyzed to find the 50 most common words. The text is then broken into 5,000 word chunks and each of the chunks is analyzed to find the frequency of those 50 words in that chunk. This generates a unique 50-number identifier for each chunk. These numbers place each chunk of text into a point in a 50-dimensional space. This 50-dimensional space is flattened into a plane using methods such as **Linear Discriminant Analysis**, **Principal Component Analysis** (PAC) or **Cluster Analysis** (See Chapter 3). This results in a display of points that correspond to an author's style. If two literary works are placed on the same plane, the resulting pattern may show if both works were by the same author or different authors.

Artificial Neural Network

Artificial Neural Networks have been used to analyze authorship of texts. Text of undisputed authorship are used to train the neural network through processes such as **Back Propagation**(See Chapter 2), where training error is calculated and used to update the process to increase accuracy. Through a process akin to non-linear regression, the network gains the ability to generalize its recognition ability to new texts to which it has not yet been exposed, classifying them to a stated degree of confidence. Such techniques were applied to the long-standing claims of collaboration of Shakespeare with his contemporaries Fletcher and Christopher Marlowe, and confirmed the view, based on more conventional scholarship, that such collaboration had indeed taken place.

A 1999 study showed that a Artificial Neural Network program reached 70% accuracy in determining authorship of poems it had not yet analyzed. This study from Vrije Universiteit examined identification of poems by three Dutch authors using only letter sequences such as ”den” [27]. (Source: Wikipedia)

One problem with this method of analysis is that the network can become biased based on its training set, possibly selecting authors the network has more often analyzed.

Genetic Algorithms

Genetic Algorithms provide an example of **Artificial Intelligence Technique** [28] applied in Stylometric analysis. The whole procedure starts with definition of a set of rules describing textual properties. Next these rules are checked against the text of known authorship and each rule if evaluated for fitness, basing on which score some rules (with the lowest score) are discarded leaving only these with fitness satisfying some criterion (selection process). The selected rules are slightly modified (mutation) and some new added, after which they are tested again. The process continues till there is obtained some number of rules that best describe features of the known text. At this point the evolved rules can be tested on a text of unknown author and if their fitness remains the same, the author is found.

4.5 Authorship Attribution in Instant Messaging

The diffusion of Internet has shifted the authorship attribution attention towards online texts (web pages, blogs, etc.) electronic messages (e-mails, tweets, posts, etc.), and other types of written information that are far shorter than an average book, way more informal and much richer in terms of expressive elements like colors, layout structures, fonts, graphics, emoticons, etc. Efforts to take into account such aspects at the level of both structure and syntax were reported in [29]. In addition, content-specific and idiosyncratic cues (e.g., topic models and grammar checking tools) were introduced to unveil deliberate stylistic choices (See [30]).

Standard Stylometric features have been employed to categorize the content of a chat over instant messaging or the behavior of the participants, but attempts of identifying chat participants are still few and early. Furthermore, the similarity between spoken conversations and chat interactions has been neglected while being a key difference between chat data and any other type of written information.

4.6 Conclusions

Although stylometry was in use for thousand years in human history, the true power of stylometry is revealed fairly recently by the introduction of computers. Statistic approach is naive for small scale input feature vectors while on the other hand, artificial neural network offers more dimensionality. Instant messaging produces dynamic stream of data and every individual leaves a trace of linguistic pattern that can be approximated using stylometry. In this work, stylometry is used to measure feature vectors to create neural network to predict digital users.

Chapter 5

Real-Time User Identification

In this chapter we are going to put everything together to come up with a real-time solution that can predict a digital users' authenticity. We will use concept of Stylometry (see chapter 4) with the help of Artificial Neural Network as a tool (see chapter 2) to produce such a system. Kohonen's Self-Organizing Maps (see chapter 3) is also used to cluster our input. The chapter will begin by introducing the feature extraction process followed by feature vectors and the implementation. Finally, different performance matrices will be used to show the different outcome of our work.

For predicting the authentication of a user, first we need to define the input feature vector set $\mathbf{x_i} = (x_{i1}, x_{i2}, \dots, x_{im})$ where $\mathbf{x_i}$ is the i^{th} input vector that is $m - dimentional$. To get our input feature vectors, we first need to analize our data set and extract feature vectors properly.

5.1 Data set

In our case, the data stream is mainly written text, weither it is instant messaging (IM) or e-mail or blog. This texts are fairly non-formal and uses native symbols and words most often. Due to the randomness of natural languages, the data set shows a wide range of variation.

We need to find the features that remains constant for a particular user for a fair amount of time. This can be called **Author Behavior Categorization**. Author behavior categorization uses a set of characteristics that remain relatively constant for a large number of digital text messages written by an author. These characteristics, known as Stylometric features, include syntactic and structural layout traits, patterns of vocabulary usage, unusual language usage, and stylistic features. Each author has various Stylometric features that are sufficient to uniquely identify him or her. Stylometric features are often word-based, including word and character frequency distributions, word length, and sentence length. Literary analysts and computational linguists often use frequency lists. Various syntactic features are also included, such as the use of function words (short all-purpose words such as the and to), punctuation, greetings and farewells, and emoticons. Users also use abbreviations for common phrases such as LOL (laughing out loud) and ROTFL (rolling on the floor laughing), as well as shortened spellings of words such as ru (are you) and 4 (for). Table 5.1 shows the Stylometric features that may be collected and analyzed for author classification.

Category	**Attribute**
Special characters	. , ! ? @ # $ % & * - + = { } ( ) [ ] ; ’ ” /
Short Forms	R U K 2 4 BRB LOL BTW NP IDK OMG WTF FYI PLS PLZ THX ...
Emoticons	:-) :) :-(:(;-) ;) :-P :P ;-P ;P :-D :D :-(:O :-O :o :-o XD X-D :(
Stickers	STICKER1 STICKER2 STICKER3 ...
Function Words	TO FROM HE THEY THE ON IN AT INTO UPON ...
Sentence Structure	Average words per sentence
Word Structure	Average characters per word
Common spelling mistakes	AMAR JANI TUI HOBE ACCHA ...

Table 5.1: Data set

Special Characters

In text data, it is often the case that the user will try to express a feeling or emotion through special characters. It is a good feature to use as a user is often biased to use particular types of special characters. For example, UserX may have a tendency to end a sentence with multiple exclamation marks (!!!) or multiple question marks (??). Thus it can be an indicator that can be used in combination with other features.

Short Forms

Using short form to express a word or feelings became a norm in digital conversations. User often use one or two characters that is universally recognized as a word or collection of words. Such as BRB is a short form for Be Right Back and AFK is a short form for Away From Keyboard. In everyday conversation in digital world, these words are used frequently, but some user prefer to use them more often than others and again have their own preferences.

Emoticons and Stickers

With the emerge of digital conversation, emoticons and Stickers plays a significant role. According to research, a digital user has a set of emoticons that he or she prefers over other. Using the list of emoticons as our data set to produce feature vector is obvious.

Function Words

Function Words include determiners (for example, the, that), conjunctions (and, but), prepositions (in, of), pronouns (she, they), auxiliary verbs (be, have), modals (may, could), and quantifiers (some, both). According to Argamon [31], function words offers a unique identifier of an author if significant amount of data is available. There are 255 function words in english language. As our work is limited to real-time data, we can not relay only on function words. Thus our data set offers other variety of features. Here is a list of common function words.

Type	Examples
Prepositions	of, at, in, without, between
Determiners	the, a, that, my, more, much, either, neither
Conjunctions	and, that, when, while, although, or
Auxiliary	is, am, are, have, got, do
Particles	no, not, nor, as

Table 5.2: Function Words

Word and Sentence Structure

Some people use small words and sentences instead of complex and long words or sentences. While others prefer long sentences and complex words. But these measurement deviates largely for a particular user. So a mean is always a good measure for such parameters.

Common Spelling Mistakes

A user tends to make same spelling mistakes in a written conversation. Apart from that, digital user often use local language spelled in english that is taken as a spelling error by the spell-checker. These local words as well as common mistakes are very unique to a particular person. For example, in bangla, the word ’AMAR’ means ’MINE’ and it is often the case that a bangali will use ’AMAR’ often. Again, ’AMR’ and ’AMAR’ are both the same thing, but user prefers to use one over the other.

5.2 Feature Extraction

As digital conversation is an interactive process, the data are in combination of different users. We first filter out the streams of our observed user (UserX) according to the following formate:

$$[\textbf{timestamp}][\textbf{username} :][\textbf{message}]$$

Then we group C characters into a segment. We take N such segments as train data. From every segment we extract our different matrices. Assuming that segment $i \in N$ has I number of interactions (timestamps), W number of words, C number of characters and S number of sentences, our extracted feature vectors **for every segment** are shown in below:

1. Average number of each function words normalized by the total amount total used of function words, FN0, FN1, FN2,FN255

$$FN_i = \frac{NumberoftimesFN_iused}{\sum_{i=1}^{2} 55NumberoftimesFN_iused} \forall i = 1, \ldots, 255 \tag{5.1}$$

2. Average delay between two consecutive interactions

$$Delay = \frac{\sum_{i=1} Timestamp_i - Timestamp_{i-1}}{I, Totalnumberofinteractions} \tag{5.2}$$

3. Average length of words

$$WL = \frac{\sum_{i=1}^{W} Lengthofwordw_i}{W, Totalnumberofwords} \tag{5.3}$$

4. Average length of sentences

$$SL = \frac{\sum_{i=1}^{S} Lengthofsentences_i}{S, Totalnumberofwords} \tag{5.4}$$

5. Average number of emoticons

$$EMO = \frac{Totalnumberofemoticonsused}{S, Totalnumberofwords} \tag{5.5}$$

6. Average number of stickers

$$ST = \frac{Totalnumberofstickers}{S, Totalnumberofwords} \tag{5.6}$$

7. Average number of special characters

$$SC = \frac{Totalnumberofspecialcharacters}{S, Totalnumberofwords} \tag{5.7}$$

8. To calculate short forms of $SF_i \forall SF_i \in R, U, K, 2, 4, BRB, \ldots$ let us assume that for

SF_i the standard deviation is σ_{SF_i} and the mean is μ_{SF_i}. So,

$$SF_i = \begin{cases} 1 & \text{if } \eta_{SF}|\mu_{SF_i} - \text{number of} SF_i| < \sigma_{SF_i} \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

Where η_{SF} is a constant that changes accodring to the user. Now, to calculate μ_{SF_i}

$$\mu_{SF_i} = \frac{\sum_{j=1}^{N} SF_{i,j}}{N} \tag{5.9}$$

And

$$\sigma_{SF_i} = \frac{\sqrt{\sum_{j=1}^{N} (\text{Total number of} SF_i \text{in} j - \mu_{SF_i})2}}{N} \tag{5.10}$$

9. To calculate common spelling mistakes CS_i for all K most frequent mistakes by the user, let us assume that the standard deviation is σ_{CS_i} and the mean is μ_{CS_i}. So,

$$CS_i = \begin{cases} 1 & \text{if } \eta_{CS}|\mu_{CS_i} - \text{number of} CS_i| < \sigma_{CS_i} \\ 0 & \text{otherwise} \end{cases} \tag{5.11}$$

Where η_{CS} is a constant that changes accodring to the user. Now, to calculate μ_{CS_i}

$$\mu_{CS_i} = \frac{\sum_{j=1}^{N} SF_{i,j}}{N} \tag{5.12}$$

And

$$\sigma_{CS_i} = \frac{\sqrt{\sum_{j=1}^{N} (\text{Total number of} CS_i \text{in} j - \mu_{CS_i})2}}{N} \tag{5.13}$$

10. Average number of dots (.) used

$$Dots = \frac{Totalnumberofdots}{S, Totalnumberofwords} \tag{5.14}$$

11. Average number of exclamation (!) used

$$Exclamations = \frac{Totalnumberofexclamations}{S, Totalnumberofwords} \tag{5.15}$$

12. Average number of question (?) used

$$Questions = \frac{Totalnumberofquestions}{S, Totalnumberofwords} \tag{5.16}$$

Input Features Vectors

Although there are about 256 Functions words and the number of common spelling mistakes can extend to thousands, for computational reasons, we limit this numbers according to our preferences. Here is a table that shows a typical Feature Vectors. These values are gathered empirically and is subject to vary depending on the desired accuracy and data set.

Dimention Number	Variable Name	Description
$1-83$	$FN_i \quad \forall i = 1,\ldots,83$	Average function words
84	$Delay$	Average delay between two interactions
85	WL	Average word length
86	SL	Average sentence length
87	EMO	Average number of emoticons
88	$STICKERS$	Average number of stickers
89	SC	Average number of stickers
$90-110$	$SF_i \forall i = 1,\ldots,20$	Short form bias
$110-125$	$CS_i \forall i = 1,\ldots,15$	Common spelling mistake bias
126	$Dots$	Average number of dots
127	$Exclamation$	Average number of exclamation
128	$Questions$	Average number of questions

Table 5.3: Feature Vectors

5.3 Experiment and Results

The experiments used text logs to collect features. For dynamic purposes, the data were collected from conversations in IM. The IM conversations were logged to ASCII text files in the following format:

$$[\textbf{timestamp}][\textbf{username}:][\textbf{message}]$$

For example:

$$(14:19:29) User1 : hey, whattimeisthemeetingtoday?$$

$$(14:19:35) User2 : Itisat11AMareyougoing?$$

$$(14:19:39) User1 : yeah, Iwillbethere, itsoundsveryinteresting! :) :)$$

The data required a series of preprocessing steps. First, the raw IM logs were parsed to extract data for each user. The data were prepared for analysis by removing all entries that did not belong to the user under analysis **(UserX)**, as well as removing both the timestamp and username.

Thus, an example of a formatted log for User 1 looks like the following:

$$hey, whattimeisthemeetingtoday?$$

$$yeah, Iwillbethere, itsoundsveryinteresting! :) :)$$

Next, the logs were split into C character segments (in our experiment C = 2500) to create instances. Sometimes this was a complete single conversation log, and sometimes it involved combining smaller conversation logs to meet the required length. Finally, the instances were processed to generate frequency totals for each attribute of a behavior category for the author. The frequency total data was outputted in **CSV format** and formatted as a **Weka** data file.

We divided the total data as $8 : 1 : 1$ as **Training samples** and **Validation samples** and **Testing samples**.

The data used in this research consisted of IM conversation logs for four users categorized in the following classes: User1, User2, User3, and User4. The data was parsed to calculate the feature vectors according to the Table 5.3.

Then using Kohonen's Seft Organizing Map (See Chapter 3) each feature was centered in its cluster by finding a centroid[1]. Classifying was done by SOM algorithm that continues to iterate till error is less than the epsilon value that was chosen as 10^6, by using the centers.

USER	Training Segments	True Positive	False Positive	Accuracy
User1	1000	88.32	11.68	88.32%
User2	1000	87.19	12.81	87.19%
User3	1000	89.02	10.98	89.02%
User4	1000	91.22	8.78	91.22%

Table 5.4: Training Data Results

USER	Validation Segments	True Positive	False Positive	Accuracy
User1	100	77.21	22.79	77.21%
User2	100	81.91	18.09	81.91%
User3	100	80.16	19.84	80.16%
User4	100	80.76	19.24	80.76%

Table 5.5: Validation Data Results

The Artificial Neural Network, more specifically MLP (See Chapter 2) is trained with the aim of classifying segments of four users. The value of Learning-Rate parameter (η), as well as number

[1]In centroid-based clustering, clusters are represented by a central vector, which may not necessarily be a member of the data set. When the number of clusters is fixed to k, k-means clustering gives a formal definition as an optimization problem: find the k cluster centers and assign the objects to the nearest cluster center, such that the squared distances from the cluster are minimized

USER	Test Segments	True Positive	False Positive	Accuracy
User1	100	81.38	18.62	81.38%
User2	100	81.01	18.99	81.01%
User3	100	76.51	23.49	7.51%
User4	100	79.25	20.75	79.25%

Table 5.6: Test Data Results

of layers were set using the validation samples (See Chapter 2) empirically. In our case, a two level layer of artificial neural network with input vector of 128 dimension was appropriate.

After training, the synaptic weights are recorded. During testing and evaluation, that information is used. Tables 5.4-5.6 show the average of 10 runs as the outcome on Training data, Validation Data and Testing Data respectively.

5.4 Conclusions

The result shows approximately 80% accuracy in unknown data set. As the data set is inherently unpredictable and natural language has its own form of randomness, the accuracy of above 80% is hard to achieve. Using more input feature vectors and multiple layer of perceptrons might increase the accuracy theoretically but it has a trade-off with speed and performance. As the system is designed for real-time use, a significant amount of computational budget is not desirable but one can surely parsue to improve given certain constraints.

So this technique can not be used on its own but can surely be used to trigger a verification check if certain threshold is crossed for a particular user. Thus it can be used as a dynamic authentication tool for a digital user profile as real-time data is being generated and checked.

Chapter 6

Concluding Remarks

Digital identity is becoming a norm of this century. Although current authentication schemes are protecting our identity in the virtual world, but static authentication system such as password protection is still not enough to protect privacy completely.

Our work proposed a scheme that can overcome this barrier and identify a user in real-time. Every user has unique language preference, wither its sentence structure or choice of words and phrases. Stylometry offers a unique solution to identify this pattern in a quantitative way. Thus using enough information from existing research on Stylometry, our work proposed a system that classifies a user based on their previous behavior. The system learns on every interaction and thus over enough data, predicts better. Using MLP as an Artificial Neural Network as a self-intelligent tool, the system can predict a user almost 70% of the time with significant performance. As the system is still not totally reliable on its own, it can be used in conjunction with the current authentication schemes (i.e. Two Step Verification[1]) and as our proposed scheme is dynamic in nature, it can trigger an immediate verification on real-time if the behavior of a particular individual differs significantly.

The Stylometric approach is still at its rudimentary form, so is Artificial Neural Network and thus significant performance increase is still possible. Artificial Neural Network with double layer and 128-dimentional input feature vector is computationally expensive. As the system is dynamic and data being generated is on real-time, there is a significant amount of computational cost is involved and any system adopting this scheme must pay the price. Overcoming the barrier of computational time can be the key focus in this area as the whole scheme is based on real-time. The faster the performance, the better and more generally it can be utilized. So further research is needed to ensure better performance so that this system can be applied with almost no significant computational cost.

[1]Two Factor Authentication, also known as 2FA, two step verification or TFA (as an acronym), is an extra layer of security that is known as ”multi factor authentication” that requires not only a password and username but also something that only, and only, that user has on them, i.e. a piece of information only they should know or have immediately to hand - such as a physical token.

Also, as our systems accuracy is about 70% and thus relays much on existing authentication schemes. Having a lower accuracy will certainly annoy user as the system will often classify wrongly and ask for a static verification. If better accuracy is achieved, this unnecessary verifications can be reduced. A 100% accuracy can, in theoretical sense, replace the current static schemes like passwords or two-step-verifications as a whole. So future work can definitely focus on improving the accuracy keeping in mind the trade-off with computational cost.

Artificial Neural Network offers a great classification tool. But there are other methods that can classify a given pattern, for example Decision Tree[2]. In future research can also focus on different classification tool to get better accuracy or better performance or both.

Stylometry is a concept of linguistic pattern and there are still many other attributes that are not yet discovered. A better attribute will certainly help better prediction. So this area can also be the focus on research in further study.

[2]A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm

References

[1] P. C. Van Oorschot and T. Wan, "Twostep: An authentication method combining text and graphical passwords," in *International Conference on E-Technologies*, pp. 233–239, Springer, 2009.

[2] M. Can, "Authorship attribution using principal component analysis and nearest neighbor rule for neural networks," *Southeast Europe Journal of Soft Computing*, vol. 1, no. 2, 2012.

[3] C. H. Ramyaa and K. Rasheed, "Using machine learning techniques for stylometry," in *Proceedings of International Conference on Machine Learning*, 2004.

[4] L. Fausett, "Fundamentals of neural networks: Architectures, algorithms and applications," *Computers & operations research*, vol. 1, no. 5, pp. 1259–1273, 1993.

[5] D. L. Chester, "Why two hidden layers are better than one," in *Proceedings of the international joint conference on neural networks*, vol. 1, pp. 265–268, 1990.

[6] S. Lawrence, C. L. Giles, and A. C. Tsoi, "Lessons in neural network training: Overfitting may be harder than expected," in *AAAI/IAAI*, pp. 540–545, Citeseer, 1997.

[7] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, no. 1, pp. 1–6, 1998.

[8] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural networks*, vol. 1, no. 4, pp. 295–307, 1988.

[9] C. M. Bishop, *Neural networks for pattern recognition.* Oxford university press, 1995.

[10] M. Adya and F. Collopy, "How e! ective are neural networks at forecasting and prediction? a review and evaluation," *J. Forecasting*, vol. 17, pp. 481–495, 1998.

[11] K.-j. Kim, "Artificial neural networks with evolutionary instance selection for financial forecasting," *Expert Systems with Applications*, vol. 30, no. 3, pp. 519–526, 2006.

[12] P. Kudová, "Learning methods for rbf networks," *Master's Thesis, Charles University, Faculty of Mathematics and Physics, Prague*, 2001.

[13] G. W. Flake, “Square unit augmented radially extended multilayer perceptrons,” in *Neural networks: tricks of the trade*, pp. 145–163, Springer, 1998.

[14] D. J. Willshaw and C. Von Der Malsburg, “How patterned neural connections can be set up by self-organization,” *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 194, no. 1117, pp. 431–445, 1976.

[15] H. Yin, “Learning nonlinear principal manifolds by self-organising maps,” in *Principal manifolds for data visualization and dimension reduction*, pp. 68–95, Springer, 2008.

[16] S. Haykin and R. Lippmann, “Neural networks, a comprehensive foundation,” *International journal of neural systems*, vol. 5, no. 4, pp. 363–364, 1994.

[17] G. Bonanno, F. Moschella, S. Rinaudo, P. Pantano, and V. Talarico, “Manual and evolutionary equalization in text mining,” in *Proc. of the 7th WSEAS International Conference on Simulation, Modelling and Optimization*, pp. 262–267, 2007.

[18] S. Gruner and S. Naven, “Tool support for plagiarism detection in text documents,” in *Proceedings of the 2005 ACM symposium on Applied computing*, pp. 776–781, ACM, 2005.

[19] D. Pavelec, L. S. Oliveira, E. Justino, F. N. Neto, and L. V. Batista, “Compression and stylometry for author identification,” in *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, pp. 2445–2450, IEEE, 2009.

[20] C. S. Butler, *Computers and written texts*. Blackwell, 1992.

[21] A. Sherbo and S. Schoenbaum, “Internal evidence and elizabethan dramatic authorship: An essay in literary history and method,” 1967.

[22] F. Mosteller and D. Wallace, “Inference and disputed authorship: The federalist,” 1964.

[23] R. Peng, R. P. Cc, *et al.*, “Statistical aspects of literary style,” 1999.

[24] W. Buckland, “Forensic semiotics,” *The Semiotic Review of Books*, vol. 10, no. 3, 1999.

[25] D. V. Khmelev and F. J. Tweedie, “Using markov chains for identification of writer,” *Literary and linguistic computing*, vol. 16, no. 3, pp. 299–307, 2001.

[26] D. T. Tran and T. D. Pham, “Markov and fuzzy models for written language verification.,” *WSEAS Transactions on Systems*, vol. 4, no. 4, pp. 268–272, 2005.

[27] J. F. Hoorn, S. L. Frank, W. Kowalczyk, and F. van Der Ham, “Neural network identification of poets using letter sequences,” *Literary and Linguistic Computing*, vol. 14, no. 3, pp. 311–338, 1999.

[28] U. Stańczyk and K. A. Cyran, "Machine learning approach to authorship attribution of literary texts," *International Journal of Applied Mathematics and Informatics*, vol. 1, no. 4, pp. 151–158, 2007.

[29] O. De Vel, A. Anderson, M. Corney, and G. Mohay, "Mining e-mail content for author identification forensics," *ACM Sigmod Record*, vol. 30, no. 4, pp. 55–64, 2001.

[30] S. Argamon, M. Koppel, J. W. Pennebaker, and J. Schler, "Automatically profiling the author of an anonymous text," *Communications of the ACM*, vol. 52, no. 2, pp. 119–123, 2009.

[31] S. Argamon and S. Levitan, "Measuring the usefulness of function words for authorship attribution," in *Proceedings of the 2005 ACH/ALLC Conference*, 2005.

Index

Activation Functions, 10
Artificial Intelligence, 4, 37
Artificial Neural Network, 3, 8, 38, 46, 48
Authentication, 49
Author Attribution, 2, 33
Author Behavior Categorization, 40
Author Characterisation, 33

Back Propagation, 12, 13, 15, 38
    Backward Computation, 14
    Forward Computation, 13
    Initialization, 13
    Training, 13
Binary Classifiers, 5
Biological Neural Network, 7

Chat, 39
Computational Cost, 48
CUSUM, 36

Data Analysis, 35
Data Stream, 40
Decision Tree, 49
Double Falsehood, 35

Eigenvectors, 25
Email, 33
Emoticons, 41

Feature Vectors, 5
Federalist Papers, 2, 35
Forgetting Term, 27
Function Words, 42, 45
Fuzzy C-means Clustering, 31

Gaussian Function, 27
Genetic Algorithm, 16, 38
Global Minima, 15
Gradient Descent, 13, 19

Hypothesis Space, 5

Identity Thief, 1
Instant Messaging, 38
Internet, 1
    Blog, 2
    Instant Messaging, 2
    Mail, 2

K-means Clustering, 30
Kohonen's Seft Organizing Maps, 46
Kohonen's Self-Organizing Maps, 3, 12

Learning Rate, 14, 15, 28, 29, 46
Least Square Method, 20
Lexical Descriptors, 35
Linear Classifiers, 7
Linguistic Analysis, 35
Local Minima, 15
Login Session, 3

Machine Learning, 4, 36
Markovian Models, 37
    1st Order, 37
MLP, 23
Momentum, 15

Multi-Layer Perceptron, 10
Hidden Neurons, 11
Input Neurons, 10
Output Neurons, 11
Multiclass Perceptron, 7

Neighborhood, 26
Neighborhood Function, 26, 27, 29
Network Architecture, 12
Network Topology, 29

Overfitting, 12, 15

Password, 1
Authentication, 2
PIN, 1
Two-Step Verification, 1
Perceptron, 5
Plagiarism, 33
Poem, 38

Quantitative Measures, 36

Radia Basis Function
Input Layer, 17
Output Layer, 17
RBF Layer, 17
Radial Basis Function, 16
Gradient Training, 18
Hybrid Training, 19
Basis Function, 16
RBF Units, 18, 19
Weight Vectors, 19
Renaissance Drama, 34

Segment, 43
Short Forms, 41
Similarity Detection, 33
SOM, 22
Competition, 24
Cooperation, 24
Initialization, 24
Linear Initialization, 25
Random Initialization, 25
Sample Initialization, 25
Mapping Mode, 23
Synaptic Adaption, 24, 27
Training Mode, 23
Winner, 25, 27
Special Characters, 41
Static Authentication, 48
Statistical Analysis, 36
Stickers, 41
Stylometric Features, 40
Stylometry, 2, 33, 35, 38, 48
Linguistic Style, 2
Stylistics, 2
Supervised Learning, 5, 12, 18
Synaptic Weights, 9, 14, 23, 47
Syntactic Features, 36, 40

Threshold, 6–8
Training Data, 5
Two-Step Verification, 48

Unsupervised Learning, 12, 27

Weight Vector, 23, 28
Willshaw-von der Malsburg Model, 23
World Wide Web, 1
Writer Invariant, 35, 37

Appendix A

Algorithms

A.1 Multi-Layer Perceptron

Algorithm 1 Multi-Layer Perceptron Algorithm

```
Require: All network input output
  for all weight $w_i$ in weight vector $\mathbf{W}$ do
    $w_i \leftarrow$ random value
  end for
  while Less than Maximum number of Iterations or Error Function is less than a Threshold
  do
    for all pattern in training set do
      Present pattern to the network
      for all Layers in the network do
        for all Nodes in the network do
          Calculate the weight sum of the inputs to the node
          Add the threshhold to the sum
          Calculate the activation for the node
        end for
      end for
      for all Nodes in the output layer do
        Calculate error signal
      end for
      for all Hidden Layers do
        for all Nodes in the Layer do
          Calculate the Node's signal error
          Update each Node's weight in the network
        end for
      end for
      Calculate the Error Function
    end for
  end while
```

A.2 RBF Gradient Training Algorithm

Algorithm 2 RBF Gradient Training Algorithm

1. **Initialization.** Set $n = 0$. Set random values to $w_{ij}(0), c_i(0), b_i(0)$ for all $i = 1, ..., h, j = 1, ..., m$, where m is the number of outputs.

2. **Next Iteration.** Set $n = n + 1$.

3. **Error Calculation.** Calculate error E.

4. **Parameter Update.** Update network's parameters:

$$w_{ij}(n+1) = w_{ij}(n) - \alpha \frac{\delta E}{\delta w_{ij}} \tag{A.1}$$

$$c_i(n+1) = c_i(n) - \alpha \frac{\delta E}{\delta c_i} \tag{A.2}$$

$$b_i(n+1) = b_i(n) - \alpha \frac{\delta E}{\delta b_i} \tag{A.3}$$

5. **Testing Stop Condition.** If stop condition is met (e.g., the overall error is small enough or maximum number of iterations is reached), then stop. Otherwise, continue at step 2.

A.3 SOM Training Algorithm

Algorithm 3 SOM Training Algorithm

```
Require: Training set $X$
  Initialization:
  Initialize synaptic weights $w_{ij}$
  Set topological neighborhood parameters
  Set learning rate parameters
  Preprocessing:
  while stopping condition false do
    for all input vectpr $\mathbf{x}$ in $X$ do
      for all output neuron $j$ do
        compute $j(\mathbf{x}) \leftarrow |\mathbf{x} - \mathbf{w_j}|$
      end for
      winning neuron $i(\mathbf{x}) \leftarrow$ neuron with the smallest value of $j(\mathbf{x})$
      for all neuron $j$ within a winner's neighborhood do
        update synaptic weights $w_j$:
            $\mathbf{w_j}(\mathbf{new}) \leftarrow \mathbf{w_j}(\mathbf{old}) + \eta(old)h_{i,j}(\mathbf{x} - \mathbf{w_j}(\mathbf{old}))$
      end for
    end for
    update learning rate $\eta(old) \leftarrow \eta(new)$
    reduce topological neighborhood radius $\sigma$
  end while
```

A.4 Fuzzy C-means Clustering Algorithm

Algorithm 4 FCM Algorithm

1. Select number of clusters K and initialize u_{ij} randomly or using some heuristics.
2. Calculate clusters' centers using 3.20
3. Update u_{ij} using 3.21
4. If $max_{ij}|u_{ij}^{new} - u^(old)_{ij}| < \epsilon$, where ϵ is termination criterion, then stop. Otherwise return to the step 2.

Fuzzy C-means Clustering Algorithm is based on the fuzzy logic where the values are not descrete but real numbers.

A.5 K-means Clustering Algorithm

Algorithm 5 K-means Clustering Algorithm

```
Require: Data set, K clusters
  for all clusters do
    Select a data from the data set as Centroid
  end for
  while error is greater than $\eta$ do
    for all $i$ in data set do
      Calculate the minimum distance from $i$ to any cluster
      Assign $i$ to cluster with minimum distance
      Calculate new centroid
    end for
  end while
```

Appendix B

Codes

B.1 Real-Time User Identification

We use this code to find out the predicted class for a particular user so that the system can ensure authenticity. Here we have used MLP as Artificial Neural Network to classify our user into ”Authentic” and ”Non-authentic” binary class. The attributes were selected based on Stylometry.

```
import java.util*
inport weka.*
public void IdentifyUser(String trainingSet, String validSet, String test
        try{
        //Reading training arff or csv file
        FileReader trainreader = new FileReader(trainingSet);
        Instances train = new Instances(trainreader);
        train.setClassIndex(train.numAttributes()     1);
        FileReader validreader = new FileReader(validSet);
        Instances valid = new Instances(validreader);
        valid.setClassIndex(train.numAttributes()     1);
        FileReader testreader = new FileReader(testSet);
        Instances test = new Instances(testreader);
        test.setClassIndex(train.numAttributes()     1);
        //Instance of NN
        MultilayerPerceptron mlp = new MultilayerPerceptron();
        //Setting Parameters
        mlp.setLearningRate(0.1);
        mlp.setMomentum(0.2);
        mlp.setTrainingTime(2000);
        mlp.setHiddenLayers("2");
        mlp.buildClassifier(train);
        }catch(Exception e){
        e.printStackTrace();
        }
        Evaluation eval = new Evaluation(train);
        eval.evaluateModel(mlp, train);
        System.out.println(eval.errorRate());
        System.out.println(eval.toSummaryString());
        eval.crossValidateModel(mlp, valid, 10, new Random(1));

        for (int i = 0; i < test.numInstances(); i++) {
                double clsLabel = mlp.classifyInstance(test.instance(i));
                test.instance(i).setClassValue(clsLabel);
        }
        //Storing again in arff
        BufferedWriter writer = new BufferedWriter(
                                        new FileWriter(output));
        writer.write(test.toString());
        writer.newLine();
        writer.flush();
        writer.close();
}
```

Generated using Undegraduate Thesis LaTeX Template, Version 1.1. Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh.

This thesis was generated on Monday 26th December, 2016 at 10:48pm.