People’s Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University M’Hamed BOUGARA – Boumerdes

Institute of Electrical and Electronic Engineering
Department of Power and Control

Final Year Project Report Presented in Partial Fulfilment of the Requirements for the Degree of

MASTER
In Control
Option: Control

Title:

Modeling of Cycling Athlete Performances Using Artificial Intelligence

Presented by:
- MERMOURI Brahim
- TAHIR Khelifa

Supervisor:

Dr. L. SADOUKI

Registration Number:........./2017
DEDICATION

• WE HAVE A GREAT PLEASURE TO DEDICATE THIS WORK TO OUR BELOVED MOTHERS AND DEAR FATHERS.

• TO OUR BROTHERS AND SISTERS AND ALL OUR FAMILIES, COUSINS, UNCLEs AND AUNTs WITHOUT FORGETTING OUR FRIENDS.
Acknowledgements

First of all, all praises and thanks are sent to ALLAH for the guidance, mercy and blessing.

This project is by far the most significant accomplishment in our life and it would be impossible without people who supported us and believed in us.

We would like to extend our deeply gratitude and our sincere thanks to our honorable, esteemed supervisor Dr. L.SADOUKI, great lecturer at the institute. She is not only a great lecturer with deep vision but also most importantly a kind personality. We sincerely thank for her exemplary guidance and encouragement. Her trust and support inspired us in the most important moments of making right decisions and we are glad to work under her supervision.

We would also like to express our special thanks to Mr. SADOUKI Kamel teacher at the National Institute of Higher Education in Sport Science and Technology (INFS / STS Dely-Ibrahim, Ex: ISTS) for providing us with the database of the used samples and the substantial technical explanations about cycling sport.

We want to thank all our teachers of our institute for providing a solid background for our studies. They have been great sources of knowledge to us and we thank them from the bottom of our heart.

We would like to thank all our friends and especially our classmates for all the thoughtful and mind stimulating discussions we had, which prompted us to think beyond the obvious.

Last but not least, we would like to thank our parents, who taught us the value of hard work by their own example. They rendered us enormous support being apart during the whole tenure of our study’s life at the institute.
CHAPTER 1: Generalities about cycling sport and database presentation

1.1) Introduction 2
1.2) Cycle sport 2
1.3) Road bicycle racing 2
1.4) Types of riders 2
1.5) The morphological measurements 3
1.6) Database 4
1.7) Conclusion 5

CHAPTER 2: Machine learning and fuzzy logic systems

2.1) Introduction 6
2.2) Machine learning 6
   2.2.1) Definitions 6
   2.2.2) Categories of machine learning 6
   2.2.3) Applications of machine learning 7
   2.2.4) Machine learning’s flowchart 7
2.3) Fuzzy logic systems 8
### Contents

2.3.1) Definitions 8
2.3.2) Why fuzzy logic? 8
2.3.3) Fuzzy logic systems architecture 8
2.3.4) Membership functions 9
2.3.5) Algorithm 9
2.3.6) Application area of fuzzy logic 10

2.4) Conclusion 10

### CHAPTER 3: AI techniques (SVM, ANN and ANFIS) used for classification

3.1) Introduction 11
3.2) Support Vector Machine (SVM) 11
   3.2.1) SVM history 11
   3.2.2) SVM concept 11
   3.2.3) Mathematical formulation of SVM 12
   3.2.4) Multiclass SVM 15
3.3) Artificial Neural Network (ANN) 15
   3.3.1) ANN concept 15
   3.3.2) Activation function and bias 16
   3.3.3) Multilayer perceptron 17
   3.3.4) Back Propagation algorithm 18
   3.3.5) ANN application 19
3.4) Adaptive Neuro-Fuzzy Inference System 20
   3.4.1) Fuzzy Inference System (FIS) 20
   3.4.2) Adaptive neuro-fuzzy inference system 22
3.5) Conclusion 24
CHAPTER 4: Simulation results and discussions

4.1) Introduction 25

4.2) Technique 1: Classification using SVM 25
   4.2.1) Description of algorithm 25
   4.2.2) Simulation results and discussions 27

4.3) Technique 2: Classification using ANN 33
   4.3.1) definitions and explanations 33
   4.3.2) Simulation results and discussions 33

4.4) Technique 3: Classification using ANFIS 38
   4.4.1) Descriptions 38
   4.4.2) Simulation results and discussions 39
   4.4.2.A) Using a whole database in training 39
     1) ANFIS grid partitioning 39
     2) ANFIS sub-clustering 45
   4.4.2.B) using a part of data base in training 46

4.5) Conclusion 47
General conclusion 48

APPENDIX A: Skinfold measurement A-1
APPENDIX B: Weights formulas A-2
APPENDIX C: Some results of simulation A-3

References
### List of Figures and Tables

- **List of Figures**

**Figure 1.1:** Types of riders (Climber, Sprinter, Rouleur). 3
**Figure 1.2:** Measurement of Skinfold thickness. 4
**Figure 1.3:** The main skinfold sites. 4
**Figure 2.1:** Flowchart of machine learning Algorithm. 7
**Figure 2.2:** Fuzzy logic system architecture. 9
**Figure 3.1:** Hyper planes with different margin. 12
**Figure 3.2:** (a) Dependency of the hyper plane with respect to $\omega$. (b) Support vector, hyper plane and the margin. 12
**Figure 3.3:** A graphical representation of a simple perceptron. 16
**Figure 3.4:** Structure of multilayer perceptron. 18
**Figure 3.5:** Neural network with back propagation algorithm. 19
**Figure 3.6:** Fuzzy inference system. 20
**Figure 3.7:** Commonly used fuzzy if-then rules and fuzzy reasoning mechanisms. 21
**Figure 3.8:** (a) Type-3 fuzzy reasoning; (b) Equivalent ANFIS (type-3 ANFIS) (Basic Structure of ANFIS). 22
**Figure 4.1:** Multiclass SVM classifier. 26
**Figure 4.2:** Different steps of multi-class SVM algorithm. 27
**Figure 4.3:** Simulation results of the first case using quadratic kernel function for testing. 28
**Figure 4.4:** Simulation results of the second case using RBF kernel function for testing (a) for default parameters, (b) for tuned parameters. 30
**Figure 4.5:** Simulation results of the second case using RBF kernel function for testing (a) for default parameters, (b) for tuned parameters. 31
**Figure 4.6:** F-G inputs case. 34
**Figure 4.7:** A-B-C-D inputs case. 36
**Figure 4.8:** All inputs case. 37
**Figure 4.9:** Classification result of the network (all inputs case). 38
**Figure 4.10:** (a) ANFIS structure (two inputs case), (b) ANFIS surface (two inputs case). 40
List of Figures and Tables

Figure 4.11: (a) Target graph (two inputs case), (b) Output class (two inputs case).
Figure 4.12: Target VS outputs (ANFIS & NNT) (two inputs case).
Figure 4.13: (a) Target graph (four inputs case)
(b) Output ANFIS grid (four inputs case)
Figure 4.14: Target VS outputs (ANFIS & NNT).
Figure 4.15: (a) Target graph (all input case), (b) Output graph (all input case).
Figure 4.16: Target VS outputs (NNT & ANFIS) (all input case).
Figure 4.17: Structure model (all inputs case and 34 rules).
Figure 4.18: (a) Target graph (all inputs case), (b) Output graph (all inputs case).
Figure 4.19: Target VS outputs (grid & sub) (all inputs case).

- List of Tables

Table 1.1: Countries and the sample size.
Table 2.1: How fuzzy logic split the input signal.
Table 3.1: Types of activation functions.
Table 4.1: Database parameters definitions.
Table 4.2: Summary or misclassified data for different kernel functions.
Table 4.3: Summary or misclassified data for different kernel functions.
Table 4.4: Summary or misclassified data for different kernel functions.
Table 4.5: Two inputs network parameters results.
Table 4.6: Simulation results of each network (two inputs case).
Table 4.7: Four inputs network parameters results.
Table 4.8: Simulation results of each network (four inputs case).
Table 4.9: All inputs network parameters results.
Table 4.10: Simulation results of each network (two inputs case).
Table 4.11: Simulation results of each network (four inputs case).
Table 4.12: Simulation results of each network (all input case).
Table 4.13: Summary results of obtained networks.
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>ANFIS</td>
<td>Adaptive Neuro-Fuzzy Inference System</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>BP</td>
<td>Back Propagation</td>
</tr>
<tr>
<td>FL</td>
<td>Fuzzy Logic</td>
</tr>
<tr>
<td>FLANN</td>
<td>Functional Link Artificial Neural Network</td>
</tr>
<tr>
<td>FIS</td>
<td>Fuzzy Inference system</td>
</tr>
<tr>
<td>LMS</td>
<td>Least Mean Square</td>
</tr>
<tr>
<td>LSE</td>
<td>Least Square Error</td>
</tr>
<tr>
<td>MF</td>
<td>Membership Function</td>
</tr>
<tr>
<td>MLP</td>
<td>Multilayer Perceptron</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Square Error</td>
</tr>
<tr>
<td>POLY</td>
<td>Polynomial</td>
</tr>
<tr>
<td>RBF</td>
<td>Radial Basis Function</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
</tbody>
</table>


Abstract

To determine the usefulness of artificial intelligence in the identification of a cycling athlete profile using the body build measurement, three non-linear models were used, which are ANN, ANFIS and SVM, in order to investigate the relation between seven morphological parameters of a rider and its profile. A group of 62 professional riders from 10 countries took part in this project. Measurements were carried out by a specialist in cycle sport discipline during the Road bicycle racing of Algeria (Grand Tour d’Algérie de cyclisme 2016).

The aim of this project, is striving to build and develop a network model enables to correctly classify the athletes in the appropriate categories, furthermore different experiences were performed in order to choose the optimal model’s parameters. The reason behind of this work is to correlate a sport field with engineering aspect to help sports professionals make the right decisions using an intelligent system.
Instruments which would allow the determination of personal predispositions for achieving high sport performance have been sought for many years. The used approach was the application of successive phases of training with different selection criteria, to determine the contestant’s chances of achieving high performance [1]. Statistical and mathematical forecasting methods [13] are becoming more and more significant in this area. These methods include multidimensional exploration techniques, which have only been used in the area of sport science.

One of the important current initiative is the collaboration of researchers together to correlate different fields such as our project case, we need to combine engineering with sport field to complete each other and solving appropriate problems, also to cause a consistency and good progress that will positively impact on human being life. In this project, we use artificial intelligence techniques to build different models using Support Vector Machine (SVM), Artificial Neural Network (ANN) and Adaptive Neuro-Fuzzy Inference System (ANFIS) to classify athlete’s profile into the appropriate category based on morphological parameters considered as inputs to our models in order to define the best proper model.

The main advantages of using those models are their ability to learn based on optimization and their excellent performances of nonlinear functions.

Recent efforts to incorporate aspects of artificial intelligence into the design and operation of automatic control systems has focused attention on techniques such as SVM, fuzzy logic, artificial neural networks, and expert systems. Although least mean squares (LMS) algorithm has been considered to be a popular method of system identification but it has been seen in many situations that the accurate system identification is not achieved by employing this technique. On the other hand, artificial neural network (ANN) has been chosen as a suitable alternative approach to nonlinear system identification due to its good function approximation capabilities i.e. ANNs are capable of generating complex mapping between input and output spaces. Thus, ANNs can be employed for modeling of complex dynamical systems with reasonable degree of accuracy. And the idea in SVM is to create a hyper plane between data sets to indicate which class it belongs.

This report is split into four related chapters:

- The first chapter gives an overview of cycling sport, different types of riders we introduced in this project. In addition to that, some references on human body about how morphological measurement were obtained and calculated by an appropriate instruments.
- The second chapter deals with introduction to artificial intelligence field, and talks about machine learning and fuzzy logic systems we used.
- The principle theory of (SVM, ANN, and ANFIS) used techniques are presented in the third chapter.
- The last chapter includes the simulation results obtained and discussions. Finally we end up by a general conclusion.
1.1) Introduction

During years of training, an athlete passes through different stages of development, in which the control of the effects of the applied training methods has the key role. In some endurance sports, such as road bicycle racing, body composition is an important component in the performances of an athlete. The present chapter aimed to define, the morphological indicators of rider's body in cycling sport discipline, and the details of the database used in our study.

1.2) Cycle sport

Cycle sport is competitive physical activity using bicycles. The first bicycle race is popularly held to have been a 1,200 meter race on the 31 May 1868 at the Park de Saint-Cloud, Paris. It was won by expatriate Englishman James Moore who rode a wooden bicycle with iron tires.[2]Bicycle racing is recognized as an Olympic sport. Bicycle races are popular all over the world, especially in Europe.

There are several categories of bicycle racing including road bicycle racing, time trialing, cyclo-cross, mountain bike racing, track cycling, BMX, and cycle speedway. Non-racing cycling sports include artistic cycling, cycle polo, freestyle BMX and mountain bike trials. The Union CyclistInternational (UCI) is the world governing body for cycling and international competitive cycling events.

1.3) Road bicycle racing

Road bicycle racing is the cycle sport discipline of road cycling, held on paved roads. Road racing is the most popular professional form of bicycle racing, in terms of numbers of competitors, events and spectators. Races can typically be split into mass start events where riders start simultaneously (though sometimes with a handicap), racing to set finish point, or individual and team time trials where riders or teams race a course individually against the clock.

1.4) Types of riders

Within the discipline of road racing, from young age different cyclists have different (relative) strengths and weaknesses. Depending on these, riders tend to prefer different events over particular courses, and perform different tactical roles within a team. The main specialties in road bicycle racing are: [3]

a. Climber: A climbing specialist or climber, also known as a grimpeur, is a road bicycle racer who can ride especially well on highly inclined roads, such as those found among hills or mountains. Climbers tend to have a lot of endurance and specifically developed
muscles for long hard climbs. They also tend to have a slim, lightweight physique, but some can become good climbers through concerted training.

b. **Sprinter:** A sprinter is a road bicycle racer or track racer who can finish a race very explosively by accelerating quickly to a high speed[4]. Apart from using sprinting as a racing tactic, sprinters can also compete for intermediate sprints (sometimes called primes), often to provide additional excitement in cities along the route of a race.

Sprinters have a higher ratio of fast-twitch muscle fibers than non-sprinters. Road cycling sprinters sometimes tend to have a larger build than the average road racing cyclist, combining the strength of their legs with their upper body to produce a short burst of speed necessary in a closely contested finish.[5] Some sprinters have a high top speed but may take a longer distance to achieve it, while others can produce short and sharp accelerations.

c. **Rouleur:** A rouleur is a type of racing cyclist considered a good all-rounder. He is particularly efficient on long distances and time trials, who can ride well over most types of course.

   (A time trialist is a road bicycle racer who can maintain high speeds for long periods of time, to maximize performance during individual or team time trials.)

A rouleur will often work as a domestique in support of their team leader, a sprinter or a climber on their team. The best chance for a rouleur to win a stage is by breaking away from the main bunch during the race to win from a small group of riders that does not contain the sprint specialists.

   (A domestique is a rider who works for the benefit of his team and leader, rather than trying to win the race)

![Figure 1.1: Types of riders (Climber, Sprinter, Rouleur)](image)

### 1.5) The morphological measurements

- **Human bodyheight:** Human height or stature is the distance from the bottom of the feet to the top of the head in a human body, standing erect.
✓ **Human body weight or mass:** Body weight is one of the most important indicators of physical development. Body weight is measured in kilograms. The components of the body weight are: the fat mass, bone mass, muscle mass and residual mass.

✓ **Measurement of body circumferences**
  - Calf circumference: Measure the girth around the largest part of the calf.
  - Proximal circumference of the thigh: Measurement of thigh circumference is usually performed 15 cm proximal to the superior pole of the patella.
  - Upper arm circumference: Measure the girth around the largest part of the upper relaxed arm.

✓ **Measurement of body diameters**

✓ **Measurement of skinfold thickness:**

A fold of skin formed by pinching or compressing the skin and subcutaneous layers especially in order to estimate the amount of body fat (see Fig: 1.2).

![Skinfold thickness](image1)

**Figure 1.2:** Measurement of Skinfold thickness.

The common locations used for this measurement are: Abdominal, calf, Thigh, hand, Forearm, Chest, Axilla, tricep, bicep, subscapular and Suprailiac Skinfold. (See Fig: 1.3) (For more details see appendix A)

![Skinfold sites](image2)

**Figure 1.3:** The main skinfold sites.

1.6) **Database**

In order to investigate the relation between the morphological parameters of a rider and its profile such as Climber, Sprinter or Rouleur. A group of 62 professional riders from 10
countries was used in this project. Measurements were performed by a specialist in cycle sport discipline during the Road bicycle racing of Algeria (Grand Tour d’Algérie de cyclisme 2016). The necessary time to take all the measurement is about 15mn/person. Age of these athletes are between 19 and 31 years. (See Table 1.1)

<table>
<thead>
<tr>
<th>Country</th>
<th>Algeria</th>
<th>Eritrea</th>
<th>Spain</th>
<th>France</th>
<th>United Kingdom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample size</td>
<td>22</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Country</th>
<th>Greece</th>
<th>Malta</th>
<th>Turkey</th>
<th>Sweden</th>
<th>Netherlands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample size</td>
<td>3</td>
<td>6</td>
<td>3</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>

Seven variables were computed from the recorded morphological measurement of each athlete. Calculation of components of a body (fat, muscle and bone) was based on Mateika formulas [7].

a. **Muscle mass** : \( MM = \text{Body weight} - (\text{Fat mass} + \text{Bone mass} + \text{Residual mass}) \)

Where:

- **Fat mass** = 1.3*\( D \)*\( \text{Surface} \)
- \( D = \frac{1}{2} \ast (\text{Subscapular} + \text{Chest} + \text{Abdominal} + \text{bicep} + \text{tricep} + \text{Forearm} + \text{handskinfold})/7 \)
- \( \text{Surface} = 100 + \text{Body weight} + \frac{(\text{Body height} - 160)}{100} \)

- **Bone mass** = \( \frac{(\text{Body height} \ast O^2 \ast 1.2)}{1000} \)
- \( O = \frac{(\text{Distal arm diameter} + \text{Forearm diameter} + \text{Thigh diameter} + \text{Calf diameter})}{4} \)
- **Residual mass** = \( \text{Body weight} \ast 0.241 \)

b. **Muscle Weight Ratio** : \( \text{Muscle mass} \ast 100 / \text{Body weight} \)

c. **Sum of 6 Skinfolds** : (Abdominal, subcapular, Suprailiac, tricep, bicep, Chest Skinfold)

d. **Corrected Arm** : Upper arm circumference - (3.14*\( \text{Biceps} \) & \( \text{Triceps} \))

e. **Corrected Thigh** : Thigh circumference - (3.14* (\( \text{Thigh Skinfold} \))/100)

f. **Corrected Calf** : Calf circumference - (3.14* (\( \text{Calf Skinfold} \))/100)

g. **Biceps & Triceps** : \( \frac{(\text{Triceps Skinfold} + \text{Biceps Skinfold})}{2} \)/100

1.7) **Conclusion**

The results of the 62 athletes were used in building the models matrix of 62X 7 and the associate classes (type of each runner is prior known). In our case we have 3 classes: Class 1: Climber (25 samples), Class 2: Rouleur (22 samples) and Class 3: Sprinter (15 samples).
2.1) Introduction

What is artificial intelligence?

Artificial intelligence is the intelligence exhibited by machines and computers; it is increasingly prevalent in our everyday lives. It is used in many aspects like industries, robotics, finance, medical diagnosis, quantum science…etc. All AI techniques used in this project are considered as soft computing (computational learning) since they involve a lot of calculations. Machine learning and fuzzy logic systems are subfields of soft computing where the solution to those problems and uncertain. We have used three techniques, SVM and ANN are types of supervised machine learning in which the machine is learned by a training set of data then it generate a model to predict the classification results of new data and ANFIS technique which is a combination of fuzzy logic system (where it uses degree of truth and not in terms of logic 0 and 1 but the values between them) and ANN.

2.2) Machine learning

2.2.1) Definition

Machine learning is the subfield of computer science that, according to ARTHURSAMUELIN(1959), It gives computer the ability to learn without being explicitly programmed, which is developed from the study of pattern recognition and computational learning theory in artificial intelligence[8], machine learning explores the study and construction of algorithms that can learn from and make predictions on data– such algorithms deals with static program instructions by making data driven predictions or decisions, through building a model from sample inputs. [9]

Machine learning is employed in a range of computing tasks where designing and programming algorithms with good performance is difficult to be achieved; example applications include classifying data of different classes which is our objective, email filtering, detection of network intruders…etc. [11]

Machine learning is closely related to computational statics, which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field [12].

2.2.2) Categories of machine learning

Machine learning tasks are typically classified into three broad categories, depending on the nature of the learning signal
• **Supervised learning:**

The learning algorithm is provided with example inputs and their desired outputs and the goal is to learn a general rule that maps inputs to outputs.

• **Unsupervised learning:**

No labels are given to the learning algorithms, leaving it on its own to structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards on end.

• **Reinforcement learning:**

A computer program interacts with a dynamic environment in which it must perform a certain task (such as driving a vehicle or playing a game against an opponent) [10].

### 2.2.3) Applications of machine learning

There are a lot of applications of machine learning, from them we can name

- ✔ Speech and handwriting recognition.
- ✔ Medical diagnosis.
- ✔ Classifying DNA sequences.
- ✔ Financial market analysis.
- ✔ Classification of data and images.

### 2.2.4) Machine learning’s Flowchart

To apply this type of artificial intelligence we have to pass through different steps as shown on fig 2.1 starting by loading the dataset and try to tune the parameters of the used functions during the procedure, train our network by those data to create a model in order to use it to classify new dataset (test network) and finally we try to evaluate the performances of the model and see if it gives us the desired classification results.

![Flowchart of machine learning Algorithm.](image)

**Figure 2.1:** Flowchart of machine learning Algorithm.
2.3) Fuzzy logic systems

2.3.1) Definition

Fuzzy logic system is a control system based on fuzzy logic which is a mathematical system that analyzes analog input values in terms of logical variables that take on continuous values between 0 and 1, in contrast to classical or digital logic, which operates on discrete values of either 0 or 1 (false or true respectively).[16] Fuzzy logic is widely used in machine control. The term “fuzzy” refers to the fact that the logic involved can deal with concepts that cannot be expressed as the “true” or “false” but rather as “partially true”. Although alternative approaches such as genetic algorithm and neural network can perform just as well as fuzzy logic in many cases, fuzzy logic has the advantage that the solution to the problem can be cast in terms that human operators can understand, so that their experience can be used in the design of the controller.[16]

2.3.2) Why fuzzy logic?

Fuzzy logic is useful for commercial and practical purposes.
✓ It can control machines and consumer product.
✓ It may not give accurate reasoning, but acceptable reasoning.
✓ Fuzzy logic helps to deal with the uncertainty in engineering.

2.3.3) Fuzzy logic systems architecture

It has four main parts as shown

- **Fuzzification Module**: it transforms the system inputs, which are crisp numbers, into fuzzy sets. It splits the input signal into five different types.
- **Knowledge Base**: it stores IF-THEN rules provided by experts.
- **Inference Engine**: it simulates the human reasoning process by making fuzzy inference on the inputs and IF-THEN rules.
- **Defuzzification Module**: it transforms the fuzzy set obtained by the inference engine into a crisp value.

### Table 2.1: How fuzzy logic split the input signal.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LP</td>
<td>Large Positive</td>
</tr>
<tr>
<td>MP</td>
<td>Medium Positive</td>
</tr>
<tr>
<td>S</td>
<td>Small</td>
</tr>
<tr>
<td>MN</td>
<td>Medium Negative</td>
</tr>
<tr>
<td>LN</td>
<td>Large Negative</td>
</tr>
</tbody>
</table>


2.3.4) Membership function

Membership function allows you to quantify linguistic term and represent a fuzzy set graphically. A membership function for a fuzzy set \( A \) on the universe of discourse \( X \) is defined as 
\[
\mu_A : X \rightarrow [0, 1].
\]
Here each element of \( X \) is mapped to a value between 0 and 1. It is called membership value or degree of membership. It quantifies the degree of membership of the element in \( X \) to the fuzzy set \( A \).
- \( X \) axis represents the universe of discourse.
- \( Y \) axis represents the degree of the membership in the \([0, 1] \) interval.

There can be multiple membership functions applicable to fuzzify a numerical value. Simple membership functions are used as use of complex functions does not add more precision in the output.

2.3.5) Algorithm

Fuzzy logic system algorithm has the following characteristics:
- Define linguistic variables and terms.
- Construct membership functions for them.
- Construct knowledge base of rules.
- Convert crisp data into fuzzy data sets using membership functions (fuzzification).
- Evaluate rules in the rule base (inference engine).
- Combine results from each rule (inference engine).
- Convert output data into non-fuzzy value (defuzzification).
2.3.6) Application area of fuzzy logic

The key application areas of fuzzy logic are as given:

- **Automotive systems**: automatic gearboxes, four-wheel steering, vehicle environment control.
- **Consumer electric goods**: photocopiers, still and video camera, television.
- **Domestic goods**: microwave ovens, refrigerators, washing machines.
- **Environment control**: air conditioners, humidifiers.

2.4) Conclusion

Machine learning and fuzzy logic systems are subfields of artificial intelligence, which are widely used in diverse kinds of applications; especially with the great development of computer science. They have a plenty of techniques and different algorithms to solve complex problems and tasks, in the next chapter we are going to describe the three used techniques in this project.
Chapter 3: AI Techniques (SVM, ANN, ANFIS) used for classification

3.1) Introduction

The major interest in this project is in identifying the models of our nonlinear static system and classifying the data using support vector machine, artificial neural network and adaptive neuro-fuzzy inference system. From the automation point of view, the objective of modeling is not to precisely calculate parameters in every point of a system, but to have information that can lead to a successful classification.

What is a classifier?

When working with statistics and other areas where large amounts of data are collected and analyzed, it is often necessary to sort the data points into sub-groups. This can be a very hard task for a human, who often aren’t able to recognize which class a data point belongs to because of the large amounts of data contained in each data point. Instead, a digital classifier is used. There are several different methods of creating a digital classifier, and we introduce here the most common of them. For all classifiers it is known that they work by supervised learning, where the classifier is trained on data with a known output, and then used on data of the same kind, allowing it to use its knowledge from the training data to classify the new once. It is important that the classifier can generalize and can sort data it has never encountered before, based on which sub-group it is most alike.

3.2) Support Vector Machine (SVM)

In machine learning, support vector machines are supervised learning model.

3.2.1) SVM History

The original SVM algorithm was invented by Vladimir N. Vapnik and Alexey Ya. Chervonenkisn in 1963. In 1992, Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik suggested a way to create nonlinear classifiers by applying the kernel trick to maximize margin hyperplanes. The current standard incarnation (soft margin) was proposed by Corinna Cortes and Vapnik in 1993 and published in 1995. [14]

3.2.2) SVM Concept

Support vector machine is a machine learning method that is widely used for data analyzing, pattern recognizing, regression and classification. Classifying data has been one of the major parts in machine learning. The idea is to create a hyperplane in between data sets to indicate which class it belongs. The challenge is to train the machine to understand structure from data and mapping with the right class label; for the best result, the hyperplane has the largest distance to the nearest training data points of any class.
Chapter 3: AI Techniques (SVM, ANN, ANFIS) used for classification

Figure 3.1: Hyperplanes with different margin.

As we can see from figure 3.1, H₃ does not separate the two classes while H₁ separate the two class with a small margin, only H₂ gives a maximum margin between two classes, therefore it’s the right hyperplane used by support vector machine.

However, instead of defining a function for the hyperplane itself; we define the margin in between the two classes. From figure 3.2 (a) we can see that the position of our hyperplane depends on $\omega$ that defines the margin, figure 3.2 (b) shows the margin of SVM. [15]

Figure 3.2: (a) Dependency of the hyperplane with respect to $\omega$, (b) Support vector, hyperplane and the margin.

3.2.3) Mathematical formulation of SVM

A) Case 1: Separable Data

You can use a support vector machine (SVM) when your data has exactly two classes. An SVM classifies data by finding the best hyperplane that separates all data points of one class from those of the other class. The best hyperplane for an SVM means the one with the largest margin between the two classes. Margin means the maximal width of the slab parallel to the hyperplane that has no interior data points. The support vectors are the data points that are closest to the separating hyperplane; these points are on the boundary of the slab. Figure 3.2 (b) illustrates these definitions, with + indicating data points of type 1, and – indicating data points of type −1. [17]
Chapter 3: AI Techniques (SVM, ANN, ANFIS) used for classification

- **Mathematical Formulation: “Primal”**

  The data for training is a set of points (vectors) \( x_i \) along with their categories \( y_i \). For some dimension \( d \), the \( x_i \in \mathbb{R}^d \), and the \( y_i = \pm 1 \). The equation of a hyperplane is:

  \[
  \langle \omega, x \rangle + b = 0 \tag{3.1}
  \]

  Where \( \omega \in \mathbb{R}^d \), \( \langle \omega, x \rangle \) is the inner (dot) product of \( \omega \) and \( x \), and \( b \) is a real number.

  The following problem defines the best separating hyperplane i.e. finding \( \omega \) and \( b \) that minimize \( \| \omega \| \) such that: \( y_i \ast (\langle \omega, x_i \rangle + b) \geq 1 \) \tag{3.2}

  For all data points \((x_i, y_i)\).

  The support vectors are the \( x_i \) on the boundary, those for which:

  \[
  y_i \ast (\langle \omega, x_i \rangle + b) = 1 \tag{3.3}
  \]

  For mathematical convenience, the problem is usually given as the equivalent problem of minimizing \( \frac{1}{2} \langle \omega, \omega \rangle \). This is a quadratic programming problem; the optimal solution \( \omega \), \( b \) enables classification of a vector \( z \) as follows: \[17\]

  \[
  \text{Class (z)} = \text{sign} (\langle \omega, z \rangle + b) \tag{3.4}
  \]

  To obtain the primal, take positive Lagrange multipliers \( \alpha_i \) multiplied by each constraint, and subtract from the objective function:

  \[
  L_P = \frac{1}{2} \langle \omega, \omega \rangle - \sum_i \alpha_i (y_i (\langle \omega, x_i \rangle + b) - 1) \tag{3.5}
  \]

  where we look for a stationary point of \( L_P \) over \( \omega \) and \( b \). Setting the gradient of \( L_P \) to 0, we get:

  \[
  \\begin{align*}
  \omega &= \sum_i \alpha_i y_i x_i \\
  0 &= \sum_i \alpha_i y_i 
  \end{align*} \tag{3.6}
  \]

- **Mathematical Formulation: “Dual”**

  The dual is given by Lagrange multipliers method as:

  \[
  L_D = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \tag{3.7}
  \]

  which you maximize over \( \alpha_i \geq 0 \). In general, many \( \alpha_i \) are 0 at the maximum. The nonzero \( \alpha_i \) in the solution to the dual problem define the hyperplane, as seen in primal case “Equation (3.6)”, which gives \( \omega \) as the sum of \( \alpha_i y_i x_i \). The data points \( x_i \) corresponding to nonzero \( \alpha_i \) are the support vectors.

  The derivative of \( L_D \) with respect to a nonzero \( \alpha_i \) is 0 at an optimum. This gives:

  \[
  y_i (\langle \omega, x_i \rangle + b) - 1 = 0. \tag{3.8}
  \]

  In particular, this gives the value of \( b \) at the solution, by taking any \( i \) with nonzero \( \alpha_i \). \[17\]
B) Case 2: Non-separable Data

Your data might not allow for a separating hyperplane. In that case, SVM can use a soft margin, meaning a hyperplane that separates many, but not all data points.

- **Mathematical Formulation: “Dual”**

There are two standard formulations of soft margins; both involve adding slack variables $s_i$ and a penalty parameter $C$ which is used to control over-fitting.

The $L^1$-norm problem is:

\[
\min_{\omega, b, s} \left( \frac{1}{2} \langle \omega, \omega \rangle + C \sum_i s_i \right)
\]

Such that $y_i (\langle \omega, x_i \rangle + b) \geq 1 - s_i, s_i \geq 0$

The $L^1$-norm refers to using $s_i$ as slack variables instead of their squares.

The $L^2$-norm problem is:

\[
\min_{\omega, b, s} \left( \frac{1}{2} \langle \omega, \omega \rangle + C \sum_i s_i^2 \right)
\]

Subject to the same previous constraints.

In these formulations, you can see that increasing $C$ places more weight on the slack variables $s_i$, meaning the optimization attempts to make a stricter separation between classes. Equivalently, reducing $C$ towards 0 makes misclassification less important. [17]

For easier calculations, consider the $L^1$ dual problem to this soft-margin formulation. Using Lagrange multipliers $\mu_i$, the function to minimize for the $L^1$-norm problem is:

\[
L_P = \frac{1}{2} \langle \omega, \omega \rangle + C \sum_i s_i - \sum_i \alpha_i (y_i (\langle \omega, x_i \rangle + b) - (1 - s_i)) - \sum_i \mu_i s_i
\]

Where you look for a stationary point of $L_P$ over $w, b$, and positive $s_i$. Setting the gradient of $L_P$ to 0, you get:

\[
\left\{
\begin{array}{l}
\omega = \sum_i \alpha_i y_i x_i \\
\sum_i \alpha_i y_i = 0 \\
\alpha_i = C - \mu_i, \alpha_i, \mu_i, s_i \geq 0
\end{array}
\right.
\]

- **Mathematical Formulation: “Dual”**

The previous equations lead directly to the dual formulation:

\[
L_P = \max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle
\]

Subject to the constraints: $\sum_i \alpha_i y_i = 0$ and $0 < \alpha_i \leq C$

The final set of inequalities, $0 \leq \alpha_i \leq C$, shows why $C$ is sometimes called a box constraint. $C$ keeps the allowable values of the Lagrange multipliers $\alpha_i$ in a “box”, a bounded region.

The gradient equation for $b$ gives the solution $b$ in terms of the set of nonzero $\alpha_i$, which correspond to the support vectors. [17]
C) Case 3: Nonlinear Transformation with Kernels

Some binary classification problems do not have a simple hyperplane as a useful separating criterion. For those problems, there is a variant of the mathematical approach that retains nearly all the simplicity of an SVM separating hyperplane.

This approach uses these results from the theory of reproducing kernels:

- There is a class of functions $K(x,y)$ with the following property. There is a linear space $S$ and a function $\varphi$ mapping $x$ to $S$ such that $K(x,y) = \langle \varphi(x), \varphi(y) \rangle$ (3.13)
The dot product takes place in the space $S$.

- This class of functions includes:
  - Linear: where $K(x, y) = (1 + \langle x, y \rangle)$.
  - Quadratic: where $K(x, y) = (1 + \langle x, y \rangle)^2$.
  - Polynomials: For some positive integer $d$, $K(x, y) = (1 + \langle x, y \rangle)^d$ (3.16)
  - Radial basis function (Gaussian): For some positive number $\sigma$
    $$K(x, y) = e^{-\langle(x-y),(x-y)\rangle/(2\sigma^2)}$$ (3.17)
  - Multilayer perceptron (neural network): For a positive number $p_1$ and a negative number $p_2$, $K(x,y) = \tanh(p_1 \langle x, y \rangle + p_2)$ (3.18)

The mathematical approach using kernels relies on the computational method of hyperplanes. All the calculations for hyperplane classification use nothing more than dot products. Therefore, nonlinear kernels can use identical calculations and solution algorithms, and obtain classifiers that are nonlinear. The resulting classifiers are hypersurfaces in some space $S$, but the space $S$ does not have to be identified or examined. [18]

3.2.4) Multiclass SVM

Support vector machines (SVMs) are primarily designed for 2-class classification problems. Although in several research papers it is mentioned that the combination of $K$ SVMs can be used to solve a $K$-class classification problem, such a procedure requires some care. Various normalization methods are proposed to cope with this problem and their efficiencies are measured empirically. [19]

1. One-versus-all

In particular, the most common technique in practice has been to build $K$ one-versus-all classifiers which are typically placed in parallel and each one of them is trained to separate one class from the $K$-1 others. Where we have applied this technique in our simulation. [19]

2. One-versus-one

We build a set of one-versus-one classifiers, and to choose the class that is selected by the most classifiers. While this involves building $K*(K-1)/2$ classifiers. [19]
3.3) Artificial Neural Networks (ANN)

3.3.1) ANN concept

An artificial neural network is a classifier modeled after how the human brain works, which is very different from how one usually writes computer code. A human brain contains an enormous amount of nerve cells “neurons”. Each of these cells are connected to many other similar cells, creating a very complex network of signal transmission. Each cell collects inputs from all other neural cells it is connected to, and if it reaches a certain threshold, it signals to all the cells it is connected to. When writing an ANN using a “perceptron” as the basic unit instead of the neuron, the perceptron can take several weighted inputs and summarize them, and if the combined input exceeds a threshold it will activate and send an output. The output that it sends is determined by the activation function and is often chosen to be between 0 and 1 or between -1 and 1. Since the derivative of the activation function is often used in the training of the network, it is convenient if the derivative can be expressed in terms of the original function value, as few additional computations are needed to calculate the derivative in this case.

The output equation for a perceptron may be represented as:

\[ y(n) = f\left[\sum_{j=1}^{N} \omega_j(n)x_j(n) + b(n)\right] \]  \hspace{1cm} (3.19)

Where, \( b(n) \) = threshold to the neuron is called as bias, \( \omega_j(n) \) = weight associated with the \( j^{th} \) input, \( N \) = number of inputs to the neuron and \( f \) is non-linear activation function.

![Figure 3.3: A graphical representation of a simple perceptron.](image)

3.3.2) Activation Functions and Bias

The perceptron internal sum of the inputs is passed through an activation function, which can be any monotonic function. Linear functions can be used but these will not contribute to a non-linear transformation within a layered structure, which defeats the purpose of using a neural filter implementation. A function that limits the amplitude range and limits the output strength of each perceptron of a layered network to a defined range in a non-linear manner will contribute to a nonlinear transformation. There are many forms of activation functions, which are selected according to the specific problem. All the neural network
architectures employ the activation function which defines as the output of a neuron in terms of the activity level at its input (ranges from -1 to 1 or 0 to 1). Table 3.1 summarizes the basic types of activation functions. The most practical activation functions are the sigmoid and the hyperbolic tangent functions. This is because they are differentiable. [20, 23]

Table 3.1: Types of activation functions.

<table>
<thead>
<tr>
<th>Name</th>
<th>Mathematical Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>$f(x) = k \cdot x$</td>
</tr>
<tr>
<td>Sigmoid</td>
<td>$f(x) = \frac{1}{1 + e^{-\alpha x}}, \alpha &gt; 0$</td>
</tr>
<tr>
<td>Hyperbolic tangent</td>
<td>$f(x) = \frac{1 - e^{-\gamma x}}{1 + e^{-\gamma x}}, \gamma &gt; 0$</td>
</tr>
<tr>
<td>Gaussian</td>
<td>$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(x-\mu)^2}{2\sigma^2}}$</td>
</tr>
</tbody>
</table>

The bias gives the network an extra variable and the networks with bias are more powerful than those of without bias. The neuron without a bias always gives a net input of zero to the activation function when the network inputs are zero. This may not be desirable and can be avoided by the use of a bias.

One problem with the ANN approach is over-fitting of the data, which happens when the classifier becomes too good at recognizing the training examples, at the expense of not being able to recognize a general input. This can be avoided by cross-validation, where the network is trained on one set of data, and then evaluated on a separate one. When the error starts rising in the validation set, the network might be over fitted. If previous networks are saved, the network can then be rolled back to the one which gave the smallest error.

3.3.3) Multilayer perceptron

In the multilayer neural network or multilayer perceptron (MLP), the input signal propagates through the network in a forward direction, on a layer-by-layer basis. This network has been applied successfully to solve some difficult and diverse problems by training in a supervised manner with a highly popular algorithm known as the error back-propagation algorithm. The scheme of MLP using four layers is shown in Figure 3.4. $x(n)$ represents the input to the network, $f_j$ and $f_k$ represent the output of the two hidden layers and $y(n)$ represents the output of the final layer of the neural network. The connecting weights between the input to the first hidden layer, first to second hidden layer and the second hidden layer to the output layers are represented respectively. [20]
If $P_1$ is the number of neurons in the first hidden layer, each element of the output vector of first hidden layer may be calculated as:

$$f_j = \varphi_j \left[ \sum_{i=1}^{N} w_{ij} x_i(n) + b_j \right], i = 1, 2, ... N, j = 1, 2, ... P_1$$

(3.20)

Where $b_j$ is the threshold to the neurons of the first hidden layer, $N$ is the number of inputs and $\varphi$ is the nonlinear activation function in the first hidden layer chosen from the Table 3.1. The time index $n$ has been dropped to make the equations simpler. Let $P_2$ be the number of neurons in the second hidden layer. The output of this layer is represented as, $f_k$ and may be written as:

$$f_k = \varphi_k \left[ \sum_{j=1}^{P_1} w_{jk} f_j + b_k \right], k = 1, 2, ... P_2$$

(3.21)

Where, $b_k$ is the threshold to the neurons of the second hidden layer. The output of the final output layer can be calculated as:

$$y_l = \varphi_l \left[ \sum_{k=1}^{P_2} w_{kl} f_k + b_l \right], l = 1, 2, ... P_3$$

(3.22)

Where, $b_l$ is the neuron’s threshold of the final layer and $P_3$ is the number of neurons in the output layer.

### 3.3.4) Back Propagation Algorithm

An MLP network with 2-3-2-1 neurons (2, 3, 2 and 1 denote the number of neurons in the input layer, the first hidden layer, the second hidden layer and the output layer respectively) with the back-propagation (BP) learning algorithm, is depicted in Fig 3.5. The parameters of the neural network can be updated in both sequential and batch mode of operation. In BP algorithm, initially the weights and the thresholds are initialized as very small random values.

The intermediate and the final outputs of the MLP are calculated by using (3.20), (3.21), and (3.22) respectively. The final output $y(n)$ at the output of neuron $l$, is compared with the desired output $d(n)$ and the resulting error signal $e_l(n)$ is obtained as:

$$e_l(n) = d(n) - y_l(n)$$

(3.23)

The instantaneous value of the total error energy is obtained by summing all error signals over all neurons in the output layer, that is: $\varepsilon(n) = \frac{1}{2} \sum_{l=1}^{P_3} e_l^2(n)$

(3.24)
Where \( P_3 \) is the number of neurons in the output layer.

![Neural network with back propagation algorithm.](image)

**Figure 3.5:** Neural network with back propagation algorithm.

This error signal is used to update the weights and thresholds of the hidden layers as well as the output layer. The reflected error components at each of the hidden layers is computed using the errors of the last layer and the connecting weights between the hidden and the last layer, the error obtained at this stage is used to update the weights between the input and the hidden layer. The thresholds are also updated in a similar manner as that of the corresponding connecting weights in an iterative method until the error signal become minimum. For measuring the degree of matching, squared error cannot be considered when the network has multiple outputs and Root Mean Square Error (RMSE) because over fitting of the model and the weights may not converge. So the Mean Square Error (MSE) is taken as a performance measurement.

The formulas of weights and the thresholds of each layer are written in appendix B.

From the structural point of MLP, if there are more than two hidden layers the structure becomes more complex. As more number of weights are present when implemented in DSP or FPGA memory requirements are considered and during updating of weights in Back Propagation it becomes very complex thereby causing over burden on the processor used. So a very simple and powerful structure is required and thus FLANN is considered.

### 3.3.5) ANN Application

The number of application areas in which artificial neural networks are used is growing daily. Here we simply produce a few representative types of problems on which neural networks have been used.

- **Pattern completion:** ANNs can be trained on sets of visual patterns represented by pixel values. If subsequently, a part of an individual pattern (or a noisy pattern) is presented to the network, we can allow the network’s activation to propagate through the network till it converges to the original (memorized) visual pattern. The network is acting like a content-addressable memory. Typically such networks have a recurrent (feedback as opposed to a feed-forward) aspect to their activation passing. You will sometimes see this described as a network’s topology.
• **Classification**: An early example of this type of network was trained to differentiate between male and female faces. It is actually very difficult to create an algorithm to do so yet an ANN has been shown to have near-human capacity to do so.

• **Optimization**: It is notoriously difficult to find algorithms for solving optimization problems. A famous optimization problem is the Travelling Salesman Problem in which a salesman must travel to each of a number of cities, visiting each one once and only once in an optimal (i.e. least distance or least cost) route. There are several types of neural networks which have been shown to converge to ‘good-enough’ solutions to this problem i.e. solutions which may not be globally optimal but can be shown to be close to the global optimum for any given set of parameters.

• **Feature detection**: An early example of this is the phoneme producing feature map of Kohonen: the network is provided with a set of inputs and must learn to pronounce the words; in doing so, it must identify a set of features which are important in phoneme production.

• **Prediction**: This task may be stated as: given a set of previous examples from a time series, such as a set of closing prices for the FTSE, to predict the next (future) sample.

• **Control**: For example to control the movement of a robot arm (or truck, or any non-linear process) to learn what inputs (actions) will have the correct outputs (results).

### 3.4) Adaptive Neuro-Fuzzy Inference System (ANFIS)

#### 3.4.1) fuzzy inference system

Fuzzy logic (FL) and fuzzy inference system (FIS), first proposed by Zadeh in 1965, they provide a solution for making decisions based on vague, ambiguous, imprecise or missing data. FL represent models or knowledge using IF-THEN rules in the form of "if X and Y then Z". As shown in the Figure 3.6, a fuzzy inference system mainly consists of fuzzy rules and membership function and defuzzification operations.

![Figure 3.6: Fuzzy inference system.](Image)

Basically a fuzzy inference system is composed of five functional blocks:

- A **rule base** containing a number of fuzzy “if-then rules”.
- A **database** that defines the membership function of the fuzzy sets used in the fuzzy rules.
Chapter 3: AI Techniques (SVM, ANN, ANFIS) used for classification

- A decision-making unit which performs the inference operations on the rules.
- A fuzzification interface which transforms the crisp inputs into degrees of match with linguistic values.
- A defuzzification interface which transforms the fuzzy results of the inference into a crisp output.

Usually, the rule base and the database are jointly referred to as the knowledge base.

The steps of fuzzy reasoning (inference operations upon fuzzy if-then rules) performed by fuzzy inference systems are:

1) Compare the input variables with the membership functions on the premise part to obtain the membership values (or compatibility measures) of each linguistic label. (This step is often called fuzzification).

Combine (through a specific T-norm operator, usually multiplication or min) the membership values on the premise part to get firing strength (weight) of each rule.

2) Generate the qualified consequent (either fuzzy or crisp) of each rule depending on the firing strength.

3) Aggregate the qualified consequents to produce a crisp output. (This step is called defuzzification) Several types of fuzzy reasoning have been proposed depending on the types of fuzzy reasoning and fuzzy if-then rules employed, most fuzzy inference systems can be classified into three types: [21, 22]

![Figure 3.7: Commonly used fuzzy if-then rules and fuzzy reasoning mechanisms.](image)

**Type 1:** The overall output is the weighted average of each rule’s crisp output induced by the rule’s firing strength (the product or minimum of the degrees of match with the premise part) and output membership functions. The output membership functions used in this scheme must be monotonically non-decreasing.

**Type 2:** The overall fuzzy output is derived by applying “max” operation to the qualified fuzzy outputs (each of which is equal to the minimum of firing strength and the output membership function of each rule). Various schemes have been proposed to choose the final
crisp output based on the overall fuzzy output; some of them are center of area, bisector of area, mean of maxima, maximum criterion, etc. [23, 24].

**Type 3:** Takagi and Sugeno’s fuzzy if-then rules are used. The output of each rule is a linear combination of input variables plus a constant term, and the final output is the weighted average of each rule’s output.

Fig 3.7 utilizes a two-rule and two-input fuzzy inference system to show different types of fuzzy rules and fuzzy reasoning mentioned above. [25]

### 3.4.2) Adaptive Neuro–Fuzzy Inference System (ANFIS)

If the input data are ambiguous or subject to a relatively high uncertainty, a fuzzy system such as ANFIS may be a better option [Han, 2009]. ANFIS is a multi-layer adaptive network – based fuzzy inference system proposed by functions to learn and tune parameters in a FIS using a hybrid learning mode. In the forward pass, with fixed a premise parameters, the least square error estimate approach is employed to update the consequent parameters and to pass the errors to the backward pass. In the backward pass, the consequent parameters are fixed and the gradient decent method is applied to update the premise parameters.

#### A) ANFIS architecture

For simplicity, we assume the fuzzy inference system under consideration has two inputs \(x\) and \(y\) and one output \(z\). Suppose that the rule base contains two fuzzy if-then rules of Takagi and Sugeno’s type:[25]

**Rule 1:** If \(x\) is \(A_1\) and \(y\) is \(B_1\), then \(f_1=p_1x+q_1y+r_1\).

**Rule 2:** If \(x\) is \(A_2\) and \(y\) is \(B_2\), then \(f_2=p_2x+q_2y+r_2\).

**Figure 3.8:** (a) Type-3 fuzzy reasoning; (b) Equivalent ANFIS (type-3 ANFIS) (Basic Structure of ANFIS).

Then the type-3 fuzzy reasoning is illustrated in Fig 3.8(a), and the corresponding equivalent ANFIS architecture (type-3 ANFIS) is shown in Figure Fig 3.8(b). The node functions in the same layer are of the same function family as described below:
Layer 1: Every node i in this layer is a square node with a node function \( O_{i1} = \mu A_{i}(x) \) Where \( x \) is the input to node \( i \), and \( A_{i} \) is the linguistic label (small, large, etc.) associated with this node function.

In other words, \( O_{i1} \) is the membership function of \( A_{i} \) and it specifies the degree to which the given \( x \) satisfies the quantifier \( A_{i} \). Usually we choose \( \mu A_{i}(x) \) to be bell-shaped with maximum equal to 1 and minimum equal to 0, such as the generalized bell function:

\[
\mu A_{i}(x) = \frac{1}{1 + \left[ \frac{(x-c_{i})}{a_{i}} \right]^{b_{i}}} \quad (3.25)
\]

Or the Gaussian function:

\[
\mu A_{i}(x) = \exp \left[ -\frac{(x-c_{i})^{2}}{a_{i}} \right] \quad (3.26)
\]

Where \{a, b, c\} (or \{a, c\} in the latter case) is the parameter set. As the values of these parameters change, the bell-shaped functions vary accordingly, thus exhibiting various forms of membership functions on linguistic label \( A_{i} \). In fact, any continuous and piecewise differentiable functions, such as commonly used trapezoidal or triangular-shaped membership functions, are also qualified candidates for node functions in this layer. Parameters in this layer are referred to as premise parameters.

Layer 2: Every node in this layer is a circle node labeled \( \pi \) which multiplies the incoming signals and sends the product out. For instance, \( w_{i} = \mu A_{i}(x) * \mu B_{i}(y) \); \( i=1, 2 \).

Each node output represents the firing strength of a rule. (In fact, other T-norm operators that perform generalized AND can be used as the node function in this layer.)

Layer 3: Every node in this layer is a circle node labeled \( N \). The \( i^{th} \) node calculates the ratio of the \( i^{th} \) rule’s firing strength to the sum of all rules’ firing strengths:

\[
\tilde{w} = \frac{w_{i}}{w_{1} + w_{2}}
\]

Layer 4: Every node \( i \) in this layer is a square node with a node function.

\[
O_{i4} = \tilde{w}_{i} f_{i} = \tilde{w}_{i} (p_{i} x + q_{i} y + r_{i}) \quad (3.27)
\]

Where is \( \tilde{w} \) the output of layer 3, and \{p, q, r\} is the parameter set. Parameters in this layer will be referred to as consequent parameters.

Layer 5: The single node in this layer is a circle node that computes the overall output as the summation of all incoming signals, i.e.

\[
O_{i5} = overall \ output = \sum \tilde{w}_{i} f_{i} = \frac{\sum i w_{i} f_{i}}{\sum i w_{i}} \quad (3.28)
\]

Thus we have constructed an adaptive network which is functionally equivalent to a type-3 fuzzy inference system.
B) ANFIS learning algorithm

From the proposed ANFIS architecture above (Fig 3.8), the output $f$ can be defined as

$$f = \frac{w_1}{w_1+w_2} f_1 + \frac{w_2}{w_1+w_2} f_2$$

$$f = \bar{w}_1(P_1x + q_1y + r_1) + \bar{w}_2(P_2x + q_2y + r_2) \quad (3.29)$$

$$f = (\bar{w}_1x)p_1 + (\bar{w}_1y)q_1 + (\bar{w}_1)r_1 + (\bar{w}_2x)p_2 + (\bar{w}_2y)q_2 + (\bar{w}_2)r_2$$

Where $p_1, q_1, r_1, p_2, q_2,$ and $r_2$ are the linear consequent parameters. The methods for updating the parameters are listed as below:

- **Gradient decent only**: All parameters are updated by gradient decent back propagation.
- **Gradient decent and One pass of Least Square Estimates (LSE)**: The LSE is applied only once at the very beginning to get the initial values of the consequent parameters and then the gradient descent takes over to update all parameters.
- **Gradient and LSE**: This is the hybrid learning rule. Since the hybrid learning approach converges much faster by reducing search space dimensions than the original back propagation method, it is more desirable. In the forward pass of the hybrid learning, node outputs go forward until layer 4 and the consequent parameters are identified with the least square method. In the backward pass, the error rates propagate backward and the premise parameters are updated by gradient descent.

3.5) Conclusion

AI has a variety of applications, one of them is classification of data using a classifier that is trained by those data with a known output (i.e. supervised learning), and then we provide this classifier with a new data of the same kind in order to classify and distinguish them using its knowledge from the training data. To do such a task we are going to use SVM, ANN and ANFIS which is a combination of fuzzy logic and ANN described in this chapter in details. In the next chapter we are going to simulate those techniques using MATLAB software.
4.1) Introduction

In chapter 3, we have introduced deeply the theory of all artificial intelligence techniques used in this project. To perform this concept practically, this chapter deals with the simulation of database using MATLAB toolbox to find out the appropriate results.

The used database is an [62x8] matrix; it represents real measured values by sport researchers in the cycle sport discipline, in order to find the non-linear model of cycling athlete performances.

We used in this study, a data base of 7 morphological inputs and one output (three profiles or classes), table 4.1 describes the different parameters.

Table 4.1: Database parameters definitions.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data symbol</td>
<td>Description</td>
</tr>
<tr>
<td>A</td>
<td>muscle mass</td>
</tr>
<tr>
<td>B</td>
<td>muscle weight ratio</td>
</tr>
<tr>
<td>C</td>
<td>sum of 6 Skinfold</td>
</tr>
<tr>
<td>D</td>
<td>Corrected arm</td>
</tr>
<tr>
<td>E</td>
<td>Corrected Thigh</td>
</tr>
<tr>
<td>F</td>
<td>Corrected calf</td>
</tr>
<tr>
<td>G</td>
<td>Triceps &amp; biceps</td>
</tr>
</tbody>
</table>

4.2) Technique 1: Classification using SVM

4.2.1) Description of the algorithm

The database has three different classes and since SVM can just classify two classes, so we need to construct a multiclass SVM. According to chapter 3; there are different methods that we can use to construct a multiclass classifier. In our project we applied one versus all multiclass SVM.

In our case we have three classes, then we need to build up three one-versus-one models (3 binary classifiers) which are placed in parallel; where we tried to train each classifier individually to separate one class from other remaining classes.

- Model 1: is trained to separate class 1 from 2 and 3.
- Model 2: is trained to separate class 2 from 1 and 3.
- Model 3: is trained to separate class 3 from 1 and 2.
The described multiclass classifier is shown as follow:

![Multiclass SVM classifier](image)

**Figure 4.1:** Multiclass SVM classifier.

Our algorithm should perform the following steps:

a) **Load the data:** we load the data to our multi-class classifier which has three different parallel models; where we provide our algorithm by three data files that have the same input data (7 elements and 62 samples) and 3 different classes as output. As we know SVM can’t classify three different classes. For that, we need to apply the technique of one-versus-all. We set the output of class 1 to 1 and classes 2 and 3 to -1 for the first model in order to identify class 1, then we set the output of class 2 to 1 and classes 1 and 3 to -1 for the second model in order to identify class 2 and finally, we set the output of class 3 to 1 and classes 1 and 2 to -1 for the last model to identify class 3. (see Fig. 4.2 stage 1)

b) **Train the data:** we train each of the three different models individually using MATLAB functions where each trained model returns some performances concerning those trained data. An example of the returned performances is shown in Appendix C (Fig C.1).

c) **Test the new loaded data:** after training the data, we need to test the obtained models by loading a set of new data; which are not used in the training process to check the goodness of our classifier with taking in consideration the possible exception of an error (miss classified data). An example is shown in appendix C (Fig C.2).

In this step we plot graphs for each model to compare the original class of the data and the classified ones, then some errors or let’s say a miss classified data are encountered, this operation permits us to compare the results of classification using different kernel functions, namely RBF and MLP, can be performed by the tuning of parameters such as, σ for RBF and “p1,p2” for MLP. (p1>0 and p2>0).

Normally, we expect each model to be classified as in figure 4.2(stage 2) that shows the original trained data loaded to MATLAB.
**d) Classification:** the last step is to use all the classifications results obtained in step 3 by each model and combine them to construct the general SVM multi-class classifier. We are going to extract the classified data of the first class from model 1, second class from model 2 and third data from class 3 (we have programmed our algorithm in this step as follow: class 1 takes value 1, class 2 takes value 2, class 3 takes value 3 and the misclassified data takes the value -1). We’re expecting to get the results shown on figure 4.2(stage 3) as the result of the general multi-class classifier.

All the steps and their functionality that we have described earlier are summarized in the following figure.

![Figure 4.2: Different steps of multi-class SVM algorithm.](image)

### 4.2.2) Simulation results and discussions

In this part we are going to simulate our algorithm through three different cases:

- Case 1: use all the training data as a testing data.
- Case 2: use 53 data for training and 9 new data for testing.
- Case 3: use 44 data for training and 18 new data for testing.

For each case, we are going to simulate it using the following kernel functions (linear, quadratic, RBF, MLP, poly3, poly4, poly5, poly6 and poly7). Results of the misclassified data are shown in a table that compares simulation results of different kernel functions. In the first case, we used quadratic kernel function, RBF for second and MLP in the third case.

**Case 1: Using all the training data in testing**

In this part, we have used entire database in training and testing for each model to see if the classifier gives us a good classification results.
• **Quadratic kernel function:** we are going to explain the simulation results of classification when quadratic kernel function is used as a “testing function”.

![Simulation results of the first model](image1.png)
![Simulation results of the second model](image2.png)
![Simulation results of the third model](image3.png)
![Simulation results of the combined models](image4.png)

**Figure 4.3:** Simulation results of the first case using quadratic kernel function for testing.

**Discussion**

• Figure 4.3(a), shows that the first model succeed to classify all data correctly except one data; where it is expected to be in class -1 but the classifier considers it as class 1; but it doesn’t affect our final combined model since it just takes the class that has value 1 not -1.

• Figure 4.3(b), shows that the second model didn’t succeed to classify three data correctly; where it is expected to classify them in class -1 but the classifier considers them as class 1; but also it doesn’t affect our final combined model for same previous reason.

• Figure 4.3(c), shows that the third model didn’t succeed to classify three data correctly where it is expected to classify them in class -1 but the classifier considers them as class 1 and didn’t succeed to classify another data of class 1 as it considers it to belong to class -1 which lead to a misclassified data in the final combined model.

• Figure 4.3(d), shows that our multiclass classifier succeed to classify all data correctly except one that it is shown in class -1 i.e. a misclassified data, this error is committed by model 3; where our model consider it to belong either to class 1 or 2 which is in reality belongs to class 3.
Our algorithm gives us also the error and its percentage as shown below:

The number of miss classified inputs is: 1
The percentage error is: 1.6129

- **Other kernel functions**

The misclassified data for other kernel functions are shown in the following table.

**Table 4.2**: Summary or misclassified data for different kernel functions.

<table>
<thead>
<tr>
<th>Kernel function</th>
<th>Misclassified data</th>
<th>Error minimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>9/62</td>
<td>-</td>
</tr>
<tr>
<td>Quadratic</td>
<td>1/62</td>
<td>-</td>
</tr>
<tr>
<td>RBF</td>
<td>2/62</td>
<td>2/62</td>
</tr>
<tr>
<td>MLP</td>
<td>22/62</td>
<td>5/62</td>
</tr>
<tr>
<td>Poly3</td>
<td>0/62</td>
<td>-</td>
</tr>
<tr>
<td>Poly4</td>
<td>0/62</td>
<td>-</td>
</tr>
<tr>
<td>Poly5</td>
<td>0/62</td>
<td>-</td>
</tr>
<tr>
<td>Poly6</td>
<td>0/62</td>
<td>-</td>
</tr>
<tr>
<td>Poly7</td>
<td>0/62</td>
<td>-</td>
</tr>
</tbody>
</table>

The second column of RBF and MLP, correspond to the minimization of the error after tuning the parameters, whereas the first column correspond to default case.

In the default case of RBF $\sigma=1$, we have tried different values for $\sigma$ but we remark that the error is not minimized.

In the default case of MLP($[P1 \ P2]=[1 \ -1]$), during the simulation we have used $[P1 \ P2]=[10 \ -100]$ for the first model, $[P1 \ P2]=[10 \ -1000]$ for the second model and $[P1 \ P2]=[2 \ -10]$ for the last model and we minimized the error from 22/62 to 5/62.

From table 4.2 we remark also that the best kernel function for testing is polynomial, quadratic, RBF, linear and MLP consecutively; but this order will not be taken into consideration because same data are used for both training and testing.

**Case 2: Use 53 data for training and 9 new data for testing**

In this part, we have used 53 data to train each model and 9 new data for testing to check the algorithm ability in prediction.

- **RBF kernel function**: we are going to explain the simulation results of classification when we use RBF kernel function as a “testing function”.

Chapter 4: Simulation results and discussions

Figure 4.4: Simulation results of the second case using RBF kernel function for testing, (a) for default parameters, (b) for tuned parameters.

Discussions

a) Default parameters: from figure 4.4(a) we can remark that:

The first model, fails to classify 2 data in class 1 since it considers them as class -1 which will cause a 2 misclassified data in the final combined model, the second model fails to classify 3 data where just one data will cause an error in the combined model because our algorithm concentrates just on data of class 1, the third model fails to classify 4 data where 3 data will cause an error because final combined model concentrates on data of class 1.

From previous discussion and the figure 4.4(a) we remark that our multi-class classifier did 6 misclassified data over 9, which is a big error that is performed by all models, and then we try to minimize this error by tuning the value of $\sigma$ for each model.

b) Tuned parameters: by setting $\sigma=14.2$ and 2 for model 1,2 and 3 respectively, then from figure 4.9 (b) we remark that:

The first model, minimize the error from 2 to 1 misclassified new data, the second model committed 5 errors but no one of them will be present in the final combined model since they belong to class -1 but our model consider them as class 1, the third model committed 4 errors but no one of them will be present in the combined model.

From previous discussion and the figure 4.4(b) we remark that we have minimized the error from 6 misclassified data to just 1 over 9 which is an acceptable result.

- Other kernel functions

The misclassified data for other kernel functions are shown in the next table:
Chapter 4: Simulation results and discussions

Table 4.3: Summary of misclassified data for different kernel functions.

<table>
<thead>
<tr>
<th>Kernel function</th>
<th>Misclassified data</th>
<th>Error minimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>1/9</td>
<td>-</td>
</tr>
<tr>
<td>Quadratic</td>
<td>3/9</td>
<td>-</td>
</tr>
<tr>
<td>RBF</td>
<td>6/9</td>
<td>1/9</td>
</tr>
<tr>
<td>MLP</td>
<td>1/9</td>
<td>1/9</td>
</tr>
<tr>
<td>Poly3</td>
<td>4/9</td>
<td>-</td>
</tr>
<tr>
<td>Poly4</td>
<td>3/9</td>
<td>-</td>
</tr>
<tr>
<td>Poly5</td>
<td>3/9</td>
<td>-</td>
</tr>
<tr>
<td>Poly6</td>
<td>4/9</td>
<td>-</td>
</tr>
<tr>
<td>Poly7</td>
<td>4/9</td>
<td>-</td>
</tr>
</tbody>
</table>

We have listed the parameters of RBF used in this case in previous section. For MLP, this function gave us a good result (1/9 error) by default parameters \([P1, P2] = [1, -1]\), we have tried to eliminate that error but we didn’t succeed, however it stills a good one.

From table 4.3 we remark also that the best kernel function for testing is MLP, linear and RBF then quadratic then polynomial.

**Case 3: Use 44 data for training and 18 new data for testing**

In this part, we have used 44 data in training, and 18 new data in testing to see if the classifier gives a good classification results.

- **MLP kernel function**: we are going to explain the simulation results of classification when we use MLP kernel function as a “testing function”.

![Simulation results of the second case using RBF kernel function for testing, (a) for default parameters, (b) for tuned parameters.](image-url)

**Figure 4.5**: Simulation results of the second case using RBF kernel function for testing, (a) for default parameters, (b) for tuned parameters.
Chapter 4: Simulation results and discussions

Discussions

a) Default parameters: from figure 4.5 (a) we can remark that:

The first model fails to classify 3 new data but just 2 of them will appear in final combined model since it considers just once of class 1 not -1, the second model fails to classify 6 data where all of them belong to class -1 but the classifier classify them in class 1; and no error will appear in the combined model because it concentrate just on data of class 1, the third model fails to classify 7 data where just 1 data will cause an error for same previous reason.

From previous discussion and figure 4.5(a) we remark that our multi-class classifier did 3 misclassified data over 18 which is a considerable error, and then we try to minimize this error by changing the values of P1 and P2 for each model.

b) Tuned parameters: by setting $[P1 \ P2] = [10 \ -1]$ , $[1 \ -1]$ and $[10 \ -1]$  for model 1,2 and 3 respectively, then from figure 4.10 (b) we remark that:

The first model minimized the error from 2 to 1 misclassified data, the second model did no error and the third model eliminated the error that we have in default parameters.

From previous discussion and the figure 4.5 (a) we remark that we have minimized the error from 3 misclassified data to just 1 over 18 which is an acceptable result.

The error and its percentage given by our algorithm is as shown below:

The number of miss classified inputs is:1
The percentage error is:5.5556

- Other kernel functions:

The misclassified data for other kernel functions are shown in the following table:

<table>
<thead>
<tr>
<th>Kernel function</th>
<th>Misclassified data</th>
<th>Error minimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>6/18</td>
<td>-</td>
</tr>
<tr>
<td>Quadratic</td>
<td>8/18</td>
<td>-</td>
</tr>
<tr>
<td>RBF</td>
<td>9/18</td>
<td>1/18</td>
</tr>
<tr>
<td>MLP</td>
<td>3/18</td>
<td>1/18</td>
</tr>
<tr>
<td>Poly3</td>
<td>6/18</td>
<td>-</td>
</tr>
<tr>
<td>Poly4</td>
<td>6/18</td>
<td>-</td>
</tr>
<tr>
<td>Poly5</td>
<td>6/18</td>
<td>-</td>
</tr>
<tr>
<td>Poly6</td>
<td>6/18</td>
<td>-</td>
</tr>
<tr>
<td>Poly7</td>
<td>7/18</td>
<td>-</td>
</tr>
</tbody>
</table>

For RBF we have used the following parameters $\sigma=12$, 4 and 4 respectively for model 1, 2 and 3 respectively; where we have minimized the error from 9 to 1 over 18. We have listed the
parameters of MLP used in this case in previous section, where we have minimized the error from 3 to 1 over 18 as explained before.

From table 4.4 we remark also that the best kernel function for testing is MLP and RBF then linear then quadratic then polynomial.

4.3) Technique 2: Classification using ANN

In fitting problems, we want a neural network to map between a data set of numeric inputs and a set of numeric targets.

The Neural Network Fitting Tool will help us to select data, create and train a network, and evaluate its performance using mean square error and regression analysis.

In our MATLAB version, it is a two-layer feed-forward network with sigmoid hidden neurons and linear output neurons (newfit), can fit multi-dimensional mapping problems arbitrarily well, given consistent data and enough neurons in its hidden layer.

The network will be trained with “Levenberg-Marquardt back propagation” algorithm; unless there is not enough memory “scaled conjugate gradient backpropagation” will be used.

4.3.1) Definitions and explanations

- Neural network tool will divide data base into three kinds of samples as follow:
  - **Training**: These are presented to the network during training, and the network is adjusted according to its error. Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples.
  - **Validation**: These are used to measure network generalization, and to halt training when generalization stops improving.
  - **Testing**: These have no effect on training and so provide an independent measure of network performance during and after training.

- Mean Squared Error (MSE) is the average squared difference between outputs and targets. Lower values are better. Zero means no error.
- Regression R Values measure the correlation between outputs and targets. An R value of 1 means a close relationship, 0 a random relationship.

4.3.2) Simulation results and discussions

Since our data base contains 7 different inputs sets, we have built different network models trained by different number of inputs samples in each case, to compare the performances of each and how a number of inputs can impacts on the result.
Case 1: Two inputs samples

We have chosen three different samples sets (A-B, D-E, and F-G), after several training attempts for each set, we have recorded the following results shown in table 4.5:

**Table 4.5:** Two inputs network parameters results

<table>
<thead>
<tr>
<th>Input set</th>
<th>MSE(training)</th>
<th>R(for all)</th>
<th>Best validation performance (MSE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-B</td>
<td>0.301478</td>
<td>0.6702</td>
<td>0.43492</td>
</tr>
<tr>
<td>D-E</td>
<td>0.252730</td>
<td>0.7543</td>
<td>0.40157</td>
</tr>
<tr>
<td>F-G</td>
<td>0.919189</td>
<td>0.7822</td>
<td>0.4499</td>
</tr>
</tbody>
</table>

The following graph shows the regression plot for (F-G) input set.

![F-G inputs case.](image)

**Figure 4.6:** F-G inputs case.

For (A-B) and (D-E) graphs, see Appendix C, (Fig C.3 and Fig C.4)

The following table shows a result of 12 arbitrary datasets for each network model, taken from the data base, 4sets from each class, and simulated using MATLAB.

Not surprisingly, we see that, each network respond and classify differently, this is due to the relation between each input set and also the size of data base.
Table 4.6: Simulation results of each network (two inputs case).

<table>
<thead>
<tr>
<th>Target</th>
<th>Output (A-B net)</th>
<th>Output (D-E net)</th>
<th>Output (F-G net)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Case 2: Four inputs samples

We have chosen three different samples sets (A-B-C-D, A-B-F-G, and C-E-F-G), after several training attempts for each set we have recorded the following results shown in table:

Table 4.7: Four inputs network parameters results.

<table>
<thead>
<tr>
<th>Input set</th>
<th>MSE(training)</th>
<th>R(for all)</th>
<th>Best validation performance (MSE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-B-C-D</td>
<td>0.083389</td>
<td>0.82463</td>
<td>0.62169</td>
</tr>
<tr>
<td>A-B-F-G</td>
<td>0.509510</td>
<td>0.72905</td>
<td>0.13764</td>
</tr>
<tr>
<td>C-E-F-G</td>
<td>0.123333</td>
<td>0.77792</td>
<td>0.24234</td>
</tr>
</tbody>
</table>

The following table shows a result of 12 arbitrary data sets for each network model, taken from the data base, 4 sets from each class, and simulated using sim command in MATLAB.
Table 4.8: Simulation results of each network (four inputs case).

<table>
<thead>
<tr>
<th>Target</th>
<th>Output (A-B-C-D)</th>
<th>Output (A-B-F-G)</th>
<th>Output (C-E-F-G)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

The classification in each network is quite different, but A-B-C-D network is doing good job in classification according to arbitrary chosen data sets.

The following graph shows the plot regression for A-B-C-D inputs set.

Figure 4.7: A-B-C-D inputs case.

For (A-B-F-G) and (D-E-F-G), see appendix C (Fig C.5 and Fig C.6).
Case 3: All inputs
In this part, we have used entire data base for training, and the following table shows the recorded results of the obtained network after several training attempts.

**Table 4.9:** All inputs network parameters results.

<table>
<thead>
<tr>
<th>Input set</th>
<th>MSE(training)</th>
<th>R(for all)</th>
<th>Best validation performance (MSE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All inputs</td>
<td>0.0902719</td>
<td>0.91396</td>
<td>0.18834</td>
</tr>
</tbody>
</table>

The following graph shows the plot regression for the obtained network.

**Figure 4.8:** All inputs case.

We can see from figure 4.8, that the values of R are close enough to one, fit lines lie on dotted lines with small deviation angle, due to a good mapping between target and output, and an obtained network will have a good performances. Only test fit line is largely deviated from dotted data due to small value of R.

The following figure is an example of 30 arbitrary data sets taken from the data base, 10 inputs from each class, and simulated using `sim` command in MATLAB.
Chapter 4: Simulation results and discussions

Figure 4.9: Classification result of the network (all inputs case).

It is clear from the figure 4.9, that target graph lays on output graph ,except it is different at one point, this interprets that the model is well trained, learned and it is classifying correctly.

Discussions

- Differences in R and MSE values are clearly determined and distinguished between each input case samples, this is due to the number of inputs factor. As we see from previous tables, the more input samples used for training, the better obtained values performance also a big data base size turns to desired results and good learned network.

- One other factor can be taken in consideration, the difference between the values for same used inputs in each case can also be interpreted as the relation and correlation between input set.

- As a result, to build a good learned network we need to provide enough database. The more inputs sets, the better result is achieved.

4.4) Technique 3: Classification using ANFIS

4.4.1) Description

ANFIS in MATLAB implements two methods to identify the inference parameters, namely Grid partitioning and Sub-clustering. Both of them use a given training data set to generate an initial fuzzy inference system that can be fine-tuned via the ANFIS command. However, there are some differences between them, such as:

- Grid partitioning generates rules by enumerating all possible combinations of membership functions of all inputs (and thus is more likely to have the problem of the “curse of dimensionality”); this leads to an exponential explosion even when the number of inputs is moderately large. For instance, for a fuzzy inference system with 10 inputs, each with two membership functions, the grid partitioning leads to 1024 (=2^10) rules, which is practically large for any practical learning methods, while sub-clustering uses SUBCLUST (subtractive clustering) to produces scattering partition.
In case of less than 6 inputs and a large size of training data, we use grid partitioning; otherwise, sub-clustering is used.

So in case of grid partitioning, a large size of memory is needed for a case of many inputs and the fuzzy rules are extremely big.

For both above methods, the most used membership functions are defined as follow:

- **GAUSSMF** (Gaussian curve membership function): GAUSSMF(X, PARAMS) returns a matrix which is the Gaussian membership function evaluated at X. PARAMS is a 2-element vector that determines the shape and position of this membership function. Specifically, the formula for this membership function is:

  \[
  \text{Gaussmf} (X, [\text{SIGMA}, C]) = \frac{\text{EXP}(-\frac{(X-C)^2}{\text{SIGMA}^2})}{\text{SIGMA}^2} \quad (4.1)
  \]

- **TRIMF** (Triangular membership Function): TRIMF(X, PARAMS) returns a matrix which is the triangular membership function evaluated at X. PARAMS = [a b c] is a 3-element vector that determines the break points of this membership function. Usually we require \(a \leq b \leq c\).

  Maximum value for these membership functions is 1.

### 4.4.2) Simulation results and discussions

#### 4.4.2. A) Using a whole data base for training

Here, we use the entire samples of each input in training and the used data for validation belong to the trained data set.

1) **ANFIS grid partitioning**

We start the analysis by first type of ANFIS which is grid partitioning; the following are the different case studies.

**Case 1: Two inputs samples**

We have chosen (F-G) inputs, because according to the neural network results obtained previously, this can be considered as best inputs set to be used for comparison with ANFIS network result.

The following is the summary of the obtained ANFIS network model.

**ANFIS info:**

- Number of nodes: 35
- Number of linear parameters: 9
- Number of nonlinear parameters: 12
- Total number of parameters: 21
- Number of training data pairs: 62
- Number of checking data pairs: 0
- Number of fuzzy rules: 9

Start training ANFIS ...

1 0.6319
2 0.634971
Chapter 4: Simulation results and discussions

The following figures show the structure and surface of the obtained network.

Figure 4.10: (a) ANFIS structure (two inputs case), (b) ANFIS surface (two inputs case).

The number of fuzzy rules is computed exponentially by: $3^2 = 9$.
- Using `evalfis` command, we have chosen data set of $(30 \times 2)$ matrix samples to simulate and test the reliability of the obtained model for classification. Results are resumed in the following graph.

Figure 4.11: (a) Target graph (two inputs case), (b) Output class (two inputs case).

We can see clearly that, the classes are obviously seen in figure 4.11 (a) and the actual output classification is shown in figure 4.11(b), where we can observe that both graphs are not similar. The network has classified only 1/10 outputs correctly from 1st class, 8/10 outputs from 2nd class and 6/10 from 3rd class.

Next, we do comparison for same tested data by using neural network model obtained previously. We can resume the result in the following table.
Table 4.10: Simulation results of each network (two inputs case).

<table>
<thead>
<tr>
<th>Network type</th>
<th>Error</th>
<th>Nº correctness class 1</th>
<th>Nº correctness class 2</th>
<th>Nº correctness class 3</th>
<th>Total of correctness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural network</td>
<td>0.9192</td>
<td>8/10</td>
<td>8/10</td>
<td>8/10</td>
<td>24/30</td>
</tr>
<tr>
<td>ANFIS grid</td>
<td>0.6319</td>
<td>1/10</td>
<td>8/10</td>
<td>6/10</td>
<td>15/30</td>
</tr>
</tbody>
</table>

Figure 4.12: Target VS outputs (ANFIS & NNT)(two inputs case).

Case 2: Four inputs samples

Here, we use one case of four inputs sample formed previously, we choose A-B-C-D inputs sample to obtain ANFIS network, then compare same sample with neural network results.

The following is the summary of ANFIS model for the chosen input

ANFIS info:
  Number of nodes: 551
  Number of linear parameters: 256
  Number of nonlinear parameters: 32
  Total number of parameters: 288
  Number of training data pairs: 62
  Number of checking data pairs: 0
  Number of fuzzy rules: 256

Warning: number of data is smaller than number of modifiable parameters

Start training ANFIS ...

1  0.00250248
2  0.00244804

From above information, if we compare them with case of two inputs, we can remark that the error is greatly reduced. However, the number of rules is increased in addition to other features; we can reduce the number of rules by reducing the number of membership functions, but this one is more accurate than the others.

The obtained graphs of structure and surface models are shown in Appendix C (Fig C.7)

The number of fuzzy rules is found by $4^4 = 256$. 
Chapter 4: Simulation results and discussions

- Using `evalfis` command, we have chosen data set of (30*4) matrix samples to simulate and test the reliability of the obtained model for classification. Results are resumed in the following graphs.

![Figure 4.13: (a) Target graph (four inputs case), (b) Output ANFIS grid (four inputs case).]

We can see that, the classes are obviously seen in figure 4.13 (a) and the actual output classification is shown in figure 4.13b, where we can observe easily that both graphs are absolutely similar. The network has classified all classes correctly.

Next, we do comparison for same tested data using neural network model obtained previously. We can summarize the result in the following table.

**Table 4.11: Simulation results of each network (four inputs case)**

<table>
<thead>
<tr>
<th>Network type</th>
<th>Error</th>
<th>N° correctness Class 1</th>
<th>N° correctness Class 2</th>
<th>N° correctness Class 3</th>
<th>Total of correctness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural network</td>
<td>0.9192</td>
<td>10/10</td>
<td>10/10</td>
<td>7/10</td>
<td>27/30</td>
</tr>
<tr>
<td>ANFIS grid</td>
<td>0.6319</td>
<td>10/10</td>
<td>10/10</td>
<td>10/10</td>
<td>30/30</td>
</tr>
</tbody>
</table>

Both outputs, neural and ANFIS types of tested data are plotted with the target in the following figure:

![Figure 4.14: Target VS outputs (ANFIS & NNT).]
**Discussion**

- The more sufficient multi input sets, the more accurate is the result and good learned network is obtained for all types.
- In case of multi inputs, large enough memory size required for grid partitioning ANFIS type.
- In case of multi inputs, we may not find big difference between artificial intelligence that we used mainly in our project in term of accuracy and resolution of results of obtained learned model.

**Case 3: All inputs**

Entire data base is trained to build ANFIS grid partitioning network, then we compare same data base with neural network model obtained previously.

The following is the summary of ANFIS model for the chosen input

**ANFIS info:**
- Number of nodes: 294
- Number of linear parameters: 1024
- Number of nonlinear parameters: 28
- Total number of parameters: 1052
- Number of training data pairs: 62
- Number of checking data pairs: 0
- Number of fuzzy rules: 128

The minimum fuzzy rules in this case are 128, because we can’t use one membership function for each input. We have taken this because when we have more than two MFs the simulation will not run due to memory problem. The obtained structure and surface models graphs for this case are shown in Appendix C(Fig C.8).

- Using `evalfis` command, we have chosen data set of (30*7) matrix samples to simulate and test the reliability of the obtained model for classification. Results are summarized in the following graph.
We can observe that, the classes are obviously seen in figure 4.15(a) and the actual output classification is shown in figure 4.15(b), where we can remark easily that both graphs are absolutely similar. The network has classified all classes correctly.

Next, we do comparison for same tested data by using neural network model obtained previously. We can summarize the results in the following table.

**Table 4.12: Simulation results of each network (all input case).**

<table>
<thead>
<tr>
<th>Network type</th>
<th>error</th>
<th>N° correctness class 1</th>
<th>N° correctness class 2</th>
<th>N° correctness class 3</th>
<th>Total of correctness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural network</td>
<td>0.0903</td>
<td>10/10</td>
<td>9/10</td>
<td>7/10</td>
<td>29/30</td>
</tr>
<tr>
<td>ANFIS grid</td>
<td>0.00014</td>
<td>10/10</td>
<td>10/10</td>
<td>10/10</td>
<td>30/30</td>
</tr>
</tbody>
</table>

Both outputs, neural and ANFIS types of tested data are plotted with the target in the following figure:

**Figure 4.16: Target VS outputs (NNT&ANFIS) (all input case).**

**Discussions**

- The error is so small in ANFIS grid partitioning technique comparing with neural technique and the ANFIS network model is better learned than neural network model.
We can say that for sufficient trained data base, all artificial intelligence techniques can be used for learning, and mostly turns out to same results.

ANFIS grid partitioning problem is for big inputs sets, a large memory space is needed.

2) ANFIS sub-clustering

It is better for more than 6 inputs to use ANFIS sub-clustering rather than grid partitioning because of (“curse of dimensionality”) problem, to see this difference we establish ANFIS sub-clustering network model of 7 inputs then we facilitate results.

The graph bellow represents the structure model of built network model.

![Figure 4.17: Structure model (all inputs case and 34 rules).](image)

To see the obtained surface model, see Appendix C (Fig C.9).

The obtained sub-clustering network model information is shown below:

**ANFIS info:**
- Number of nodes: 554
- Number of linear parameters: 272
- Number of nonlinear parameters: 476
- Total number of parameters: 746
- Number of training data pairs: 62
- Number of checking data pairs: 0
- Number of fuzzy rules: 34

And the next figures show results of tested data by the obtained network.
Figure 4.18: (a) Target graph (all inputs case), (b) Output graph (all inputs case).

From the above graphs, target and output results are similar, the obtained network is well trained and learned the data base correctly, and is able to test and classify correctly.

So we can say that grid-partitioning and sub-clustering types give same results as we got in our case and we may generalize for other cases. So it is preferred to use sub-clustering instead of grid partitioning in case of many different inputs sets to avoid curse of dimensionality.

Figure 4.19: Target VS outputs (grid & sub) (all inputs case).

4.4.2. B) Using a part of data base for training

In this part, we exclude from our data base a set of data sets from training, and then we use them for testing reason to test the ability of network in identifying, recognizing and predicting new data then classifying it accordingly to its true class.

We have extracted out 12 different data sets samples arbitrary, 4 samples from each class, after that we built new network model using ANFIS sub-clustering.

When the fuzzy inference system is generated, four parameters for “Subtractive Clustering” need to be specified [19, 20]:


**Chapter 4: Simulation results and discussions**

- **Range of influence** $q_1$: (default 0.5), to specify the range of influence of a cluster center. The more neighboring data points a data point can enclose, the higher potential it has as a cluster center.

- **Squash factor** $q_2$ : (default 1.25), multiplying $q_1$ to determine the neighborhood of a cluster center within which the existence of other cluster centers are discouraged.

- **Accept ratio** $q_3$ : (default 0.5), to set the potential above which another data point will be accepted as a cluster center.

- **Reject ratio** $q_4$ : (default 0.15), to set the potential below which a data point will be rejected as a cluster center.

We are striving to obtain good classifier network by tuning ANFIS parameters in each process to obtain the best one because of the complexity of the 7-dimensional data points. After different attempts, we have summed up important results in the following table:

**Table 4.13:** Summary results of obtained networks.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Network 1</th>
<th>Network 2</th>
<th>Network 3</th>
<th>Network 4</th>
<th>Network 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range of influence</td>
<td>0.65</td>
<td>0.50</td>
<td>0.40</td>
<td>0.40</td>
<td>0.60</td>
</tr>
<tr>
<td>Squash factor</td>
<td>1.25</td>
<td>1.30</td>
<td>1.15</td>
<td>2.00</td>
<td>1.20</td>
</tr>
<tr>
<td>Accept ratio</td>
<td>0.55</td>
<td>0.55</td>
<td>0.35</td>
<td>0.90</td>
<td>0.40</td>
</tr>
<tr>
<td>Reject ratio</td>
<td>0.30</td>
<td>0.25</td>
<td>0.20</td>
<td>0.45</td>
<td>0.20</td>
</tr>
<tr>
<td>Number of MF</td>
<td>4</td>
<td>24</td>
<td>45</td>
<td>17</td>
<td>21</td>
</tr>
<tr>
<td>Optim. Method</td>
<td>Hybrid</td>
<td>Hybrid</td>
<td>Hybrid</td>
<td>Hybrid</td>
<td>Hybrid</td>
</tr>
<tr>
<td>error</td>
<td>5.6294E-06</td>
<td>5.6294E-06</td>
<td>7.7399E-06</td>
<td>4.5072E-06</td>
<td>5.4083E-06</td>
</tr>
<tr>
<td>Correct classify</td>
<td>4/12</td>
<td>5/12</td>
<td>9/12</td>
<td>4/12</td>
<td>6/12</td>
</tr>
</tbody>
</table>

From table 4.13, we see that those parameters have significant role in determining the desired network. As a result, we can say that if the data base is so accurate and coherent, the most effective and easiest way to build classifier is to use ANFIS—sub clustering method because, it turns out to preferable results. Among them, the network 3 is found to have the best results.

**4.5) Conclusion**

As a conclusion, we can say that the three techniques that we have used in our simulation permit us to classify correctly our database and also being able to predict the class of the data which were not trained. Those results can be enhanced by providing a higher database. We can summarize also that AI techniques can also be used for prediction but, the significant and necessary condition for that is requiring a good quality and sufficient size of data base to guarantee a better achieved result.
General Conclusion

The aim of this project is to find a proper ANN, SVM and ANFIS models for adaptive nonlinear static system identification based on some athletes morphological inputs to classify them in each specific class.

Away from engineering field, the project topic is naturally sport one, so we are literate in some new things about that field in terms of sport team construction, athletes categories and their some specific morphological features and, some of used instruments to measure those parameters. People are working on that because each athlete has technical descriptions to be determined by continuous training and experience in order to have a qualified team ready for competitions.

Support vector machine classifier, is a good optimization algorithm which is designed for two distinct classes, for that we have used one-versus-others techniques to construct a multiclass classifier where three models are designed. We have trained our classifier by a set of data, then new data are used for testing part to see if the classifier gives the desired classification results; during our simulation we have used different kernel functions (linear, quadratic, polynomial, multilayer perceptron and radial basis function) where the two last types can be controlled by p1 and p2 and α parameters. This algorithm gives us good results and the error is minimized to zero especially using the multilayer perceptron and radial basis kernel functions.

Artificial neural network is also an important technique in learning and inducing good network model that allows us to classify our data based on training step, but we didn’t reach the suitable goal, because we were unable to find the proper network to predict true classification in case of using a part of data base in training and the rest of it in testing and validation.

The last technique used in this project is ANFIS model, were we have used both of its types, all obtained results were satisfactory, however it’s worth noting that the difference between the two used types is the number of inputs used in training. We concluded from our results that for less than 6 inputs, ANFIS grid is recommended otherwise, ANFIS clustering is better. The advantage for this technique is the ability of creating a network model in predicting new data by tuning the ANFIS parameters.

In fact, the work on this project gave us an overview about sport domain and how its practical problems can be solved by the help of engineering field. In our case artificial intelligence models were used in determining the performances of athletes. We can say that we achieved good results even though there are some applications we didn’t do, due to the limited size of our database, and occasionally this is not easy to obtain such kind database. So for further work, we can improve these results by increasing this database by new records, also adding new morphological parameters and giving more importance to the age factor of the athletes, this can affect positively on the obtained results.
1. **Triceps Skinfold (Vertical pinch):** At the level of the mid-point between the acromial (lateral edge of the acromion process, e.g. bony tip of shoulder) and the radial (proximal and lateral border of the radius bone, approximately the elbow joint), on the mid-line of the posterior (back) surface of the arm (over the triceps muscle).

2. **Biceps Skinfold (Vertical pinch):** At the level of the mid-point between the acromial (lateral edge of the acromion process, e.g. bony tip of shoulder) and the radial (proximal and lateral border of the radius bone, approximately the elbow joint), on the mid-line of the anterior (front) surface of the arm (over the biceps muscle).

3. **Subscapular Skinfold (Diagonal pinch):** The lower angle of the scapula (bottom point of shoulder blade); If there is difficulty finding this landmark, get the subject to reach behind their back with their right arm, while feeling for the movement of the scapula.

4. **Thigh Skinfolds (Vertical pinch):** The mid-point of the anterior (front) surface of the thigh, midway between patella (knee cap) and inguinal fold (crease at top of thigh).

5. **Iliac Crest (Iliocristale, Suprailiac) Skinfold (Diagonal pinch):** Immediately above the iliac crest (top of hip bone), on the most lateral aspect (side).

6. **Abdominal Skinfold (Vertical pinch):** a mark is made 5 cm adjacent to the umbilicus (belly-button), to the right side. See notes below about alternate sites.

7. **Calf Skinfold Site (Vertical pinch):** A point on the medial (inside) surface of the calf, at the level of the largest circumference. A tape measure should be used to determine this point.

8. **Chest (Pectoral) Skinfold (Diagonal pinch):** The pinch is taken at a point between the axilla and nipple as high as possible on the anterior axillary fold.

9. **Axilla (Midaxillary) Skinfold (Vertical pinch):** At the point where a vertical line from the mid axilla (middle of armpit) intersects with a horizontal line level with the bottom edge of the xiphoid process (lowest point of the breast bone).

10. **Forearm Skinfold:** The site can beat the maximum circumference of the forearm. The location of the pinch site can be either on the midline of the anterior or posterior aspect.

---

**Fig.A.1: Skinfold caliper.**
Appendix B

Weights formulas

The weights are using the following formulas,

\[ W_{kl}(n + 1) = W_{kl}(n) + \Delta W_{kl}(n) \] \hspace{1cm} B.1

\[ W_{jk}(n + 1) = W_{jk}(n) + \Delta W_{jk}(n) \] \hspace{1cm} B.2

\[ W_{ij}(n + 1) = W_{ij}(n) + \Delta W_{ij}(n) \] \hspace{1cm} B.3

\[ \Delta W_{kl}(n) = -2\mu \frac{\partial e(n)}{\partial W_{kl}(n)} = \mu e(n) \frac{dy_l(n)}{dW_{kl}(n)} = \mu e(n) \phi'_l \left[ \sum_{k=1}^{p_2} W_{kl} f_k + b_l \right] f_k \] \hspace{1cm} B.4

Where, \( \mu \) is the convergence coefficient (0<\( \mu \)<1). Similarly \( \Delta w_{jkn} \) and \( \Delta w_{ijn} \) the can be computed. The thresholds of each layer can be updated in a similar manner, i.e.

\[ b_l(n + 1) = b_l(n) + \Delta b_l(n) \] \hspace{1cm} B.5

\[ b_k(n + 1) = b_k(n) + \Delta b_k(n) \] \hspace{1cm} B.6

\[ b_j(n + 1) = b_j(n) + \Delta b_j(n) \] \hspace{1cm} B.7

Where, \( \Delta b_{ln} \), \( \Delta b_{kn} \) and \( \Delta b_{jn} \) are the change in thresholds of the output, hidden and input layer respectively. The change in threshold is represented as,

\[ \Delta b_l(n) = -2\mu \frac{\partial e(n)}{\partial b_l(n)} = \mu e(n) \frac{dy_l(n)}{db_l(n)} = \mu e(n) \phi'_l \left[ \sum_{k=1}^{p_2} W_{kl} f_k + b_l \right] \] \hspace{1cm} B.8
Appendix C

Some results of simulation

1) Technique 1: SVM

Model1 =

SupportVectors: [19x7 double]
Alpha: [19x1 double]
Bias: 0.5822
KernelFunction: @mlp_kernel
KernelFunctionArgs: {{[10] [-1]}}
GroupNames: [44x1 double]
SupportVectorIndices: [19x1 double]
ScaleData: [1x1 struct]
FigureHandles: []

Model2 =

SupportVectors: [28x7 double]
Alpha: [28x1 double]
Bias: -0.1319
KernelFunction: @mlp_kernel
KernelFunctionArgs: {{[1] [-1]}}
GroupNames: [44x1 double]
SupportVectorIndices: [28x1 double]
ScaleData: [1x1 struct]
FigureHandles: []

Model3 =

SupportVectors: [30x7 double]
Alpha: [30x1 double]
Bias: -2.9641
KernelFunction: @mlp_kernel
KernelFunctionArgs: {{[10] [-1]}}
GroupNames: [44x1 double]
SupportVectorIndices: [30x1 double]
ScaleData: [1x1 struct]
FigureHandles: []

FigC.1: Some performances of data to each model after training.
Appendix C

Some results of simulation

\[
\begin{align*}
test1 &= test2 = test3 = \\
1 & 1 & 1 \\
-1 & 1 & 1 \\
1 & -1 & 1 \\
1 & -1 & -1 \\
1 & -1 & -1 \\
1 & 1 & -1 \\
-1 & 1 & 1 \\
-1 & 1 & 1 \\
-1 & 1 & 1 \\
-1 & 1 & 1 \\
-1 & 1 & 1 \\
-1 & 1 & 1 \\
1 & -1 & 1 \\
-1 & 1 & 1 \\
1 & -1 & 1 \\
1 & 1 & 1 \\
-1 & 1 & 1 \\
-1 & 1 & 1 \\
-1 & 1 & 1 \\
-1 & 1 & 1 \\
\end{align*}
\]

Fig C.2: Classification of data using SVM classify after we load a set of new data for testing.

2) Technique 2: ANN

![Graphs showing data fitting and validation results for different cases.]

Fig C.3: A-B inputs case.
Appendix C

Some results of simulation

Fig C.4: D-E inputs case.

Fig C.5: A-B-F-G inputs case.
Appendix C

Some results of simulation

3) Technique 3: ANFIS

Fig C.6: D-E-F-G inputs case.

Fig C.7: (a) ANFIS structure (four inputs case), (b) ANFIS surface (four inputs case).
Appendix C

Some results of simulation

Fig C.8: (a) Structure model (all inputs), (b) Surface model (all inputs).

Fig C.9: Surface model (all inputs case).
References


