Adversarial Learning through Red Teaming: From Data to Behaviour

Shir Li Wang

A thesis submitted in fulfilment

of the requirements of the degree of

Doctor of Philosophy

School of Engineering and Information Technology

University of New South Wales

Canberra Campus

September, 2012

Declaration

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgment is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the projects design and conception or in style, presentation and linguistic expression is acknowledged.

Signed: .....

Shir Li Wang

Date: .....

Acknowledgements

I would like to take the opportunity to express my deepest gratitude to many people for making the journey of my study possible and exciting.

First of all , I would like to express my gratefulness to my supervisor, Prof. Hussein A. Abbass. He has been a driving force behind my research, by inspiring me to view problems from different angles, providing constructive advice, asking critical questions that trigger/ guide my thinking. I would like to express my gratitude to my co-supervisors, Dr. Chris Lokan and Dr. Kamran Shafi for not only providing guidance and encouragement, but also sharing their knowledge and ideas.

Next, my special grateful acknowledgement goes to the University of Education Sultan Idris (Universiti Pendidikan Sultan Idris, UPSI) and Ministry of Higher Education, Malaysia for sponsoring my study in UNSW. I would like to thank the staff of the School of Engineering and Information Technology as well as other departments which have been providing support and assistance in easing my study. I also owe thanks to Prof. Lim Chee Peng and Assoc. Prof. Zarina bt. Abdul Aziz who have been so wonderful to me.

A special thanks to all of my fellow friends at UNSW, Amr Ghoneim, Dominik Beck, Eman Samir Hasan Sayed, George Leu, Irman Hermadi, Miriam Brandl, Miriam Glennie, Mohammad Esmaeil Zadeh, Nizami Jafarov, Theam Foo Ng, Van Viet Pham, Wenjing Zhao for sharing ups and downs, joys and worries, excitement and frustration of academic research, and anyone who shares a friendship with me. Also, not to forget my dearest friends in Malaysia, Dr. Siew Chin Neoh, Wei Lee Lim, Wei Wei Tan and Yean Ching Tan for organising a unforgettable trip to New Zealand together. Let's do again in the future.

Most of all, I am most grateful to my father and siblings for their unconditional love and believing in me. I wish to thank my dad in particular for helping me to realising my dream.

Abstract

The concept of an adversary applies to all facets of the human life as they continue to dwell in a hostile world. The knowledge of an adversary's behaviour is therefore of paramount importance to us to make sound decisions and succeed in our lives. Red teaming is an ancient technique where an adversary's role is emulated by playing a devil's advocate role to improve own defences and decisions. The approach involves dealing with two competing entities, namely blue and red agents, and has been widely used in military planning to role-play the enemy; test and evaluate its course of actions or judgement; assess the vulnerabilities of the red team; and learn to understand the dynamics between the red and blue entities. However, the red teaming concept is easily mapped to domains that share similar characteristics to military planning, such as adversarial learning, risk assessment, and behavioural decision making in a competitive environment. Computational red teaming is a recent approach that extends red teaming concept in the cyber space and benefits from replacing the physical red and blue with simulated entities.

The focus of this thesis is to study the effect of information on adversarial behaviour. A Computational Red Teaming based framework is developed to analyse four forms of an adversary or a red agent operating in a fixed self or blue agent's environment: a static red having direct access to manipulate randomly the information received by the blue agent; a dynamic red having the ability to learn and evolve to counteract blue's actions; a real human playing a red agent's role; and a red approximating human behaviours.

To understand the impact of information, a statistical framework for simulating adversarial attacks is proposed to model and explore the effect of red. The heart of the simulation lies in attacks against representative samples of the training data available to blue, and the generation of attack is based on statistical sampling methods. The underlying assumption for the simulation is, red has the capability to identify representative samples and attacks them directly. Under the influence of red, the performance of blue, represented by a single neural network and neural ensemble, is evaluated in static and non-stationary environment.

A synthetic red teaming game environment is then created to study the second, third and fourth forms of red. Here, CRT assists in the process of understanding the differences

and similarities in a behavioural context between a computational red (machine learning agent) and a natural red (human). Neuroevolution is selected as the computational model, owing to its abilities to evolve and learn which are very important to mimic human behaviours. Besides that, neural networks are used to approximate human behaviours from the data collected in human red teaming.

The literature lacks metrics and methodologies to analyse the behaviour of both machine and human red, and to compare these behaviours. The metrics and methodologies need to be able to represent, process, analyse, and compare the behaviours in an objective manner. Several metrics and methodologies are proposed in this thesis for the purpose of analysing and comparing the possible red's behaviours.

The thesis demonstrates that:

1. Blind purposeful manipulation of data can be counteracted with an ensemble of learning machines.
2. In a demanding task, where the time to make a decision is very short, humans tend to ignore the information available to them and instead focus on using their skills to achieve the task.
3. A deceptive behaviour is beneficial in an environment where the frequency of receiving information is low and the noise in received information is high.
4. A deceptive behaviour is not beneficial in an environment where information is frequent and noise free.
5. Machine behaviour encompasses human behaviours but extend it with more creative behaviours.

List of publications

Journal Articles

1. **Shir Li Wang**, Kamran Shafi, Chris Lokan and Hussein A. Abbass. Adversarial learning: the impact of statistical sample selection techniques on neural ensembles. *Evolving System*, 1(3):181–197, 2010.

2. **Shir Li Wang**, Kamran Shafi, Chris Lokan and Hussein A. Abbass. An agent based model to simulate and analyse behaviour under noisy and deceptive information. *Adaptive Behavior*, 2012, accepted.

Conference Papers

3. **Shir Li Wang**, Kamran Shafi, Chris Lokan and Hussein A. Abbass. Robustness of neural ensembles against targeted and random adversarial learning. In *Proceedings of the 2010 IEEE International Conference on Fuzzy Systems (FUZZ)*, pages 1–8, Barcelona, Spain, July 2010.

4. **Shir Li Wang**, Kamran Shafi, Chris Lokan and Hussein A. Abbass. Neuro-evolution of escape behaviour under high level of deception and noise. *The First International Conference on Robot Intelligence Technology and Applications 2012 (RiTA)*, Gwangju, South Korea, December 2012, accepted.

Contents

List of Figures

List of Tables

List of Abbreviations

ACRE	Adversarial Classifier Reverse Engineering
ARTL	Army Red Team Leader
CM	Covariance-Mahalanobis
CRT	Computational Red Teaming
ER	Evolutionary Robotic
FCM	Fuzzy C Mean
GA	Genetic Algorithm
HRT	Human Red Teaming
ISO	International Standards Organisation
ICM	Inversion based on Covariance-Mahalanobis
IRND	Inversion based on Random
MEBRA	Multiobjective Evolutionary Based Risk Assessment
MGT	Magic Gamma Telescope
MSE	Mean Squared Error
NN	Neural network
PB	Page Block
POC	Pareto Operating Curve
PDF	Probability Density Function
RND	Random
RONI	Reject On Negative Impact Defense
RT	Reaction time
STDEV	Standard Deviation
UFMCS	University of Foreign Military and Cultural Studies

Chapter 1

Introduction

1.1 Introduction

Red teaming is an approach to study a task by anticipating the action of an adversary. An adversary refers to any entity affecting the objective of completing the task optimally and rationally. On the one hand, optimality refers to the decision maker's ability to maximise or minimise some explicit and measurable criterion with respect to the conditional nature of the environment [26]. On the other hand, rationality refers to a decision making process whereby an agent takes the action associated with the highest attainable value based on a measurable function from the available alternative actions. The available actions are associated with different values whereby the values are computed based on the measurable function given a set of inputs and constraints [26, 4].

Red teaming normally involves two entities, namely *blue* agents and *red* agents. A *blue* agent refers to an entity whose goal is to achieve a specific task, while a *red* agent refers to a circumstance and/or entity which acts to prevent *blue* from achieving its task. Originally, red teaming is an approach widely used in military operations to role-play the enemy; test

and evaluate its courses of actions or judgement; assess the vulnerabilities of *blue*; and learn to understand the dynamics that exist between *red* and *blue*.

In recent literature, *red* does not necessarily refer to an enemy. It is generalised to any entity that has conflicting objectives with *blue* or causes *blue* a level of uncertainty in achieving its objectives [4]. Based on this definition, we are actually facing a virtual *red* for most of our lives because as we know, life is about making decisions and it usually involves conflicting and uncertain objectives.

Red teaming as a concept can be traced back thousands of years. Its importance is reflected by several famous quotes in The Art of War by Sun Tzu, in about 500 B. C.[30]: to know your enemy, you must become your enemy; if you know your enemies and know yourself, you will not be imperilled in a hundred battles; if you do not know your enemies but do know yourself, you will win one and lose one; if you do not know your enemies nor yourself, you will be imperilled in every single battle. In short, the quote emphasises the importance of anticipating adversary's behaviours during decision making in a conflict situation.

Despite the long history, red teaming still remains a developing and evolving concept which has great potential to offer. Only a few years ago, the importance of the red teaming concept was formally recognized in the military community with the introduction of the Army Red Team Leader (ARTL) course at the University of Foreign Military and Cultural Studies (UFMCS), Fort Leavenworth, in 2006. The Canadian military has also shown intention to establish the concept and its capability [41]. The concept was limited to the physical realm originally, but it has been extended to computer simulation owing to advancements in computer technology as shown in [89, 41]. Computer-based red teaming, which is commonly known as Computational Red Teaming ("CRT") is no longer limited to military applications. Its implementation has been and can be further extended to

security systems, commercial organisations, governments, and even countries to study the effectiveness of its strategies in anticipating adversarial behaviours.

From the above examples, red teaming can basically be divided into two types, physical realm-based red teaming and computer-based red teaming. Physical realm-based red teaming usually involves human beings. On the other hand, CRT involves the use of computational methods and models. The involvement of humans in red teaming is not only time and cost consuming, but also limits the exploration of possible aspects of the relevant task. In contrast, CRT allows the exploration of abstract higher level of scenarios of different vulnerabilities in the problem space [89, 4].

Owing to the definition of *red* in [4], CRT can be mapped into domains which share similar characteristics such as adversarial learning, risk assessment, and behavioural decision making as long as the domains contain entities that have the potential to influence the decision making process. The discussion on CRT may depend on the purpose to either discover vulnerabilities or to learn about the opponents. Besides that, the implementation of CRT also depends on the interest of decision makers on either the problem space or solution space. With explicit representation of *red*, CRT can be used to explore the weakness of *blue* which impacts the plans or solutions at hand. In this case, the implementation of CRT is to explore the problem space. On the other hand, the implementation of CRT focuses on the solution space if it is used to explore possible solutions against *red*.

The representation of *red* depends on types of domains and applications. For example, adversarial learning refers to a branch of machine learning, where the objective is to understand the impact of an adversary whose intention is to counteract what the machine learning algorithm is learning. In this domain, an adversarial attack is referred to as *red* while the machine learning algorithm/system is known as *blue*. *Red* in adversarial learning is usually information driven. To design secure and robust machine learning environments,

the performance of *blue* under the influence of *red* is usually evaluated by simulating the *red* effect through data manipulation.

By using the work in [89] as an example, the question that can be asked is: can autonomous machine red teaming produce results similar to human-based red teaming? The core of the question means that can computational red teaming actually model and explore the effect of a natural *red* (human), who is behaviour driven [4]?

To make comparisons by using a military application would very costly and time consuming. Instead, we propose a synthetic simulation environment which allows both machine red teaming and human red teaming to be carried out in the same environment so that a fair comparison can be made between them. It is important to implement the same red teaming task in both environments because red teaming is context dependent and user specific [41]. To be specific regarding the context of our research, we are interested to know whether computational red teaming can produce or mimic the behaviour of a natural *red* agent, specifically humans, on the same given task and environment?

A synthetic simulation environment allows us to have control over the environment so that a specific change in the dependent variable can reliably be produced by specific manipulations of the independent variables, and the change is unlikely to be the result of confounding variables. Besides that, the focus of the study is on the feasibility of CRT, which relates to understanding the similarities and differences between CRT and Human Red Teaming (HRT), but not on the complexity of the environment. If the feasibility of CRT is proven in our work, the study can be further extended into a more complex and complicated environment. In short, the focus of our work is to understand the impact of different forms of *red*.

1.2 Motivation

Red teaming usually involves two sides: the *red* and the *blue* agents. The *blue* can refer to any entity which can vary from a single individual to an organisation that has intentions, objectives or goals. In contrast, the *red* agent refers to any entity which has the potential to influence the achievements of the *blue* agent. The purpose of red teaming is to represent and understand the *red* agent which has the potential to influence a system of interest, so that the system is more prepared for possible adversarial influences and less exposed to risk. Owing to the advancements of information technology, red teaming has been expanded to CRT with the involvement of computational methods and models. With an explicit representation of the *red* agent in CRT, it is designed to anticipate and simulate adversarial behaviours so that we have better understanding on its effects on the system that we would like to protect. The use of CRT to mitigate risk is not only time and cost effective, but also has great potential to (see [4] for a good review):

- Explore a space of possibilities;
- Discover vulnerabilities;
- Learn about competitors;
- Reveal biases;
- Create a database of cases for future events;
- Unlearn to learn.

However, the simulation of adversarial behaviours will merely be hypothetical if without studies to model and explore of how a *red* agent thinks, which can be either a machine or a human. The impact of these two forms of agents could be different. Therefore, we

would like to understand the dynamics of how the *red* and *blue* agents interact in machine red teaming and human red teaming, as shown in Figure 1.1. The study of understanding the dynamics of interaction between the agents is not only important in establishing the feasibility of CRT, but also in increasing the confidence of decision makers to use it as a decision making tool. Besides that, the study will also provide some kinds of empirical knowledge of the adversary's decisional process that can be beneficial in red teaming.

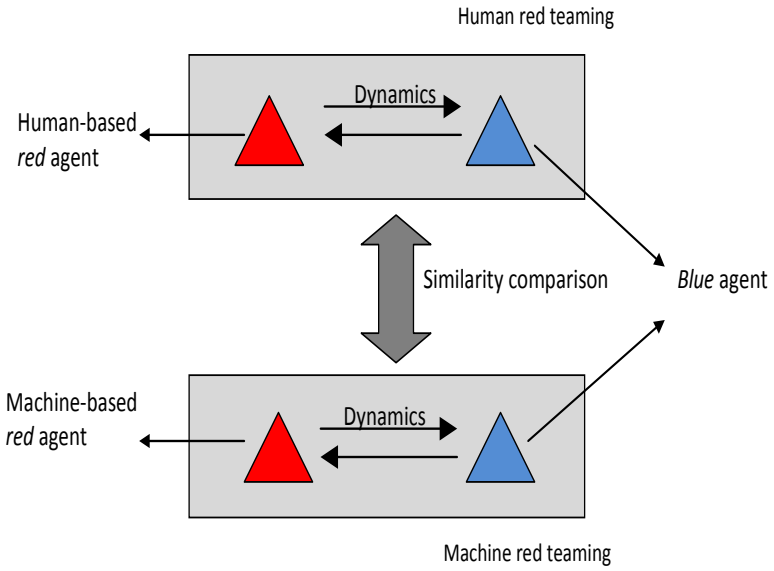


Figure 1.1: Comparison of the dynamics of interaction between the *red* and *blue* agent in machine red teaming and human red teaming.

1.3 Key Questions/Hypothesis

In our daily lives, we always need to face risk which may come from an explicit or implicit *red* agent. The existence of the *red* agent often makes us choose from the available options which affect our abilities to achieve our objective(s). The introduction of CRT as a decision making tool improves our preparedness to encounter the emerging threat from the *red* agent, specifically in adversarial learning.

The primary focus of CRT in an adversarial learning environment lies in how to produce the effect of *red*. Given that *red* can be either a machine or a human, its impact on adversarial learning may be different. In other words, different forms of *red* could have different impacts in adversarial learning. This also means the impact of *red* lies in its manipulation approach, which can be either information driven or behaviour driven. Therefore, it is important to understand the impact of information and behaviour in adversarial learning if we would like to design secure and robust machine learning environments.

In this thesis, we study four forms of *red*: (i) one with direct access to stochastically manipulate the information received by *blue*; (ii) one based on machine learning with the ability to learn and evolve to counteract *blue*'s behaviour; (iii) one that is a real human playing *red*; and (iv) one based on machine learning to approximate human behaviours. These four forms motivated the primary research question in this thesis:

"**What is the role of information and behaviour in adversarial learning**?"

The four forms of *red* are proposed based on a selective rather than exhaustive review on several domains shown in the chapter which are interrelated such as adversarial learning, prey-predator simulation, risk assessment, behavioural decision making and red teaming, where the research studies consist of at least two entities with conflicting interest.

In most research related to adversarial learning, one of the common practices found in the research is an explicit assumption being made about the ability of the adversary. On the other hand, in the behavioural research focusing on human behaviour in decision making, human subjects are usually evaluated in the simulated situations or scenarios.

By comparing the research between these two domains, we can see that their main difference depends on the perception of the source of risk produced by a *red*. In the former domain, the *red* is treated as a fixed source of risk rather than a participant in an interaction. Besides that, the degree of responsiveness of the *red* is varied from simple fixed

rules to complex human beings. Given that the nature of adversarial learning, involving new attacks, and repetitions or evolutions from old attacks, it is likely that the *red* exhibited human behaviour, if not complete, e.g., adaptivity.

One important point mentioned by Lima [46] in prey-predator simulation is that simple modifications of the assumptions made about the interactions between agents might reveal some insights on new classes of behavioural phenomena. Therefore, four forms of *red* agents are proposed in our work with their differences being determined by the degree to which individual red's behaviour approximates human behaviour. At one side, the *red* acts based on explicit functions. On the opposite side, the *red* actually refers to a real human being. The *red* agents mentioned in (ii) and (iv) lie in between these two extreme categories, where they mimic and reproduce human behaviour based on computational models and methods. We say these *red* agents are driven by implicit behaviour.

Adversarial learning is a common term in cyber security such as spam filtering, fraud detection and intrusion detection, where the domain usually involves machine learning algorithms from both sides, i.e., defence system and adversary. Owing to this common practice of using learning machines in the domain, the term usually refers to the learning process of a machine learning algorithm in the presence of an adversary whose main goal is to cause dysfunction of the learning machine. However, we believe that the term "adversarial learning" is not necessarily limited in cyber security but can be expanded to other domains as long as the learning process is viewed from the perspective or interest of understanding the adversary. Since the term is commonly used in cyber security but not in the other domains, we limited the use of the term to Chapter 3 to eliminate unnecessary confusion to the readers.

To attempt answering the main question, there are several related sub-questions that arise from the attempt and they need to be investigated as well:

1. What is the impact of intentional manipulation of information/data on a learning machine?

   The impact of information (data) in adversarial learning lies on the way data is being manipulated [84, 85]. A statistical framework for simulating adversarial attacks is proposed to simulate the *red*'s effect which is information driven. The heart of simulation lies in attacks against representative samples of the training data available to *blue*, and the generation of attacks is based on statistical sampling methods. The underlying assumption is *red* is able to identify the representative samples and attack them directly. We are interested to understand the impact of *red*'s information manipulation on *blue*'s performance. Two types of *blue* are evaluated under the influence of *red*, in static and non-static environments, using a single neural network and a neural ensemble.

2. How to characterise and understand behaviour for a machine or a human?

   It is important to compare human and machine within the same environment to eliminate the possible differences which may be due to the specifics of the task and the behaviour adequacy. For the latter difference, it means we hardly know in advance which behaviour (human or machine) is more adequate for the task. This leads to the importance of comparing the differences on the same environment – the development of red teaming environment. In the proposed environment, we have three types of agents as follows:

   - $B(A_b)$ - the role is as a *blue* agent, with a scripted strategy.
   - $B(A_{mr})$ - the role is as a *red* agent and the actions are determined by a machine learning algorithm.
   - $B(A_{hr})$ - the role is as a *red* agent and the actions are determined by a human.

To characterise and understand behaviour for machines and humans, both *red* agents are exposed to the *blue* agent associated with different scripted strategies. Then, we would like to understand how *blue*'s strategies affect *red*'s behaviours which can be either machine *red* or human *red*. For machine *red*, neuroevolution is used owing to its ability to evolve and learn, which are very important to mimic human behaviours.

3. How can we compare the differences and similarities between a machine and a human behaviour?

   The differences and similarities between machine and human are compared in behavioural context between a computational *red* (machine learning agent) and a natural *red* (human). In our work, behaviour represents a sequence of actions in the proposed environment over time t. The behaviour of an agent i in the environment can be denoted as $B(A_i) = \{a_i^1, a_i^2, a_i^3, \ldots, a_i^t\}$, with a_i^t refers to the same attributes which describe the actions taken by an agent i over time t.

   However, the literature lacks metrics and methodologies to analyse the behaviour of both machine and human *red*, and to compare these behaviours. The metrics and methodologies need to be able to represent, process, analyse, and compare the behaviours in an objective manner. Several metrics and methodologies are proposed in this thesis for analysing and comparing the possible *red*'s behaviours based on their actions distribution and actions similarity. The details on the metrics and methodologies are explained in Chapter 4, Section 4.2.3.

1.4 Organisation of the Thesis

The remainder of the thesis is organized in 7 chapters as follows:

- Chapter 2 - Literature Review

- Chapter 3 - Adversarial Learning
- Chapter 4 - Synthetic Simulation Environment
- Chapter 5 - Human Red Teaming Experimental Design
- Chapter 6 - Machine Red Teaming Experimental Design
- Chapter 7 - Machine Approximating Human Behaviour Experimental Design
- Chapter 8 - Conclusion and Future Work

In Chapter 2, the context of the problem which focuses on behavioural modelling is provided. We also include a summary of selected literature on the domains in which the concept of computational red teaming has a great deal to offer. The first part of the chapter covers background materials on adversarial learning, risk assessment and behavioural decision making. The later part of the chapter covers background materials on the methodology that is used in our research, which consists of evolutionary robotics, genetic algorithms and artificial neural networks.

In Chapter 3, a statistical framework simulating adversarial attacks is proposed to model the effect of *red* which is information driven. Two types of adversarial attacks are simulated to investigate their effects on machine learning, namely *Causative Availability Targeted* and *Causative Availability Indiscriminate* attacks. We investigate the effects of the adversarial attacks on a single neural network and a neural ensemble.

The simulated adversarial attacks represent *red* while the single neural network and neural ensemble represent *blue*. In this chapter, we are interested to know the effects of *red* on *blue*. Therefore, we replicate part of the existing adversarial taxonomy using a data mining task to simulate a red teaming environment. The simulation is based on the assumption that *red* has the capability to identify representative samples and attacks them.

The problem we encountered is that we hardly know whether the assumption is valid in a natural *red* or not. This leads to the idea of developing a synthetic red teaming game environment which allows a natural *red* and computational *red* to perform the same task.

Chapter 4 provides the information about a synthetic simulation adversarial environment that accommodates a *red* agent and *blue* agent. The chapter provides in detail the constraints of the environment, the roles of the agents in the environment and the strategies adopted by both agents. Furthermore, the experimental methodology, analysis methodology and evaluation metrics involved in the thesis are explained in this chapter.

In Chapter 5, the experimental designs for the red teaming exercise involving humans are explained in detail. The chapter also consists of the results and analysis of the experiments.

In Chapter 6, we describe the approach that is used to simulate machine red teaming. The chapter also provides in detail the experimental designs for machine red teaming. At the end of the chapter, analysis on the result of the machine red teaming is carried out and findings are concluded.

Chapter 7 provides the experimental designs on the machine learning approach to approximate human behaviours based on the data collected from human red teaming. The chapter also consists of the results and analysis of the experiments.

In Chapter 8, we summarise the contributions of our research, point out the limitations of our work and discuss the future directions that stem from this work.

1.5 Thesis Contribution

The main contribution of this thesis focuses on the study of feasibility of CRT through behavioural comparison between humans and computational models. Further contributions

of thesis are listed as following:

- **The impact of information driven** *red*. A framework simulating adversarial attacks is developed to study the effect of *red* on *blue* in an adversarial environment, where *red* refers to adversarial attacks while *blue* represents the deployed machine learning system against the attacks. Here, the simulation of *red*'s effect is through information manipulation (information driven). In the framework, two types of adversarial attacks, namely *causative availability targeted* and *causative availability indiscriminate* attacks, are generated based on sampling. The heart of the simulation lies on the attack against the outputs instead of the inputs. Instead of adding extra corrupted instances into data, e.g., white noise, the outputs are inverted based on sampling methods. Under the influence of adversarial attacks, the performances of *blues*, which are represented by a single neural network and neural ensemble, are evaluated in both static and non static environment.

- **Metrics/methodologies for behaviour production, analysis and comparison**. Basically, our study involves two types of red teaming, i.e., CRT and HRT. Behavioural production in HRT refers to the involvement of human beings in decision making for the environment. When examined from CRT, the selection of models for behavioural production lies in the ability to mimic or produce optimality and rationality of human behaviours. Owing to this reason, neuroevolution is chosen and we attempt to study its usefulness in this area. Behaviour is defined as a sequence of actions in our study and it can be viewed as a representation of internal preferences and judgement. As far as we know, the literature lacks metrics and methodologies to analyse the behaviours of both machine and human *red*, and to compare these behaviours. The metrics and methodologies need to be able to represent, process, analyse, and compare the behaviours in an objective manner. Therefore, we propose

several metrics and methodologies for the purpose of analysing and comparing the possible *red*'s behaviours between the human and computational models.

- **Metrics/methodologies to characterise and understand the impact of behaviour driven** *red*. The focus in the framework of adversarial attacks is the CRT is information driven. The inversion of *red* depends on the fixed assumption, whereby the *red* is assumed to have the ability to search for representative samples of the training data available to *blue*, and invert the outputs of representative samples. If *red* is replaced by a machine approximating human action or even a real human, it would be interesting to know whether CRT can actually model and explore the effects of a behaviour driven *red* or not. Besides that, we bring to light the feasibility of CRT to model the effect of behaviour driven *red*. To answer the question, we need a red teaming teaming environment which allows the natural *red* and computational *red* to perform on the same task. Here, we need to have a good simulation environment which allows the simulation of red teaming's dynamics which flow between *red* and *blue*. The dynamics are captured by the behaviours between *red* and *blue*. Therefore, a synthetic game environment based on red teaming is developed to allow behavioural comparison being made between machine and human. In the proposed environment, the strategies of *blue* are scripted to understand how these strategies influence the behaviours of machine *red* and human *red*.

Chapter 2

Literature Review

2.1 Introduction

Originally, red teaming is an approach widely used in military operations to role-play the enemy; test and evaluate its courses of actions or judgements; assess the vulnerabilities of the red team; and learn to understand the dynamics existing between the red and blue entities. However, the red entity is not necessarily referring to an enemy per se but to the concept of entities with conflicting objectives or uncertainty in objectives [4]. Owing to this definition, the red teaming concept can be mapped into domains that share similar characteristics, such as adversarial learning, risk assessment, and behavioural decision making. As long as a domain contains entities with conflicting objectives, red teaming will be in play. The concept can be further expanded with the the use of computational models or methods in a red teaming environment.

The following sections present some background on the domains that share similar characteristics with red teaming, and the potential that CRT has in these domains.

2.2 Risk Assessment

By taking the adversarial learning problem as an example, the improvement of the state of preparedness of an organisation/system for countering the emerging threat of adversarial attacks is important and it is directed to the need of risk assessment in adversarial domains. Unfortunately, the main problem in adversarial learning is that the types of adversarial attacks can be many. New types of attacks and old ones may keep on arising and evolving. Therefore, whatever preparedness plans/solutions that an organisation/system has on their minds need to be robust and adaptive. To protect an organisation/system against adversarial attacks, which plans/solutions to be used depend greatly on the decision makers, and risk assessment tools becomes prominent in assisting them to make decisions.

According to the definition from the International Standards Organisation (ISO) [90], risk is "the effect of uncertainties on objectives". The definition consists of two important elements: uncertainty and objective. It means the effect of uncertainty needs to be measured relative to the objective, when risk assessment is performed. There are two main categories of risk assessment methods: qualitative risk assessment and quantitative risk assessment. They complement each other. With the involvement of computational models and methods, CRT can be viewed as a generic encompassing framework combining both quantitative and qualitative risk assessment. In risk assessment, the uncertainties in objective(s) are viewed as *red* while the actions taken to achieve the objective(s) are referred to as *blue*.

A broad risk assessment approach characterising uncertainties beyond probabilities and expected values is needed in the decision making environment. As pointed out by Aven and Renn [12], the main component in risk is uncertainty, not probability. Probability is just one of way of expressing uncertainties. The main problem in risk assessment is not about the uncertainty of the estimates of the probabilities or expected values, but lies in

the uncertainty about an activity, as a result of limited knowledge about the multitude of contextual conditions and motivational factors which could lead to consequences. The need of seeing beyond probabilities or expected values is a new direction in risk assessment. The use of CRT to explore a space of possibilities offers a great potential in this direction, as is demonstrated in [2, 8, 7, 9]. In [2], a framework known as Multiobjective Evolutionary Based Risk Assessment (MEBRA) is introduced to explore and evaluate the algorithms under risk. The focus of CRT demonstrated in MEBRA is to explore the scenario spaces in which the algorithms may perform poorly, known as the failure regime of the algorithms. Through evaluating the performances of the algorithms under the scenarios that could lead to failure, decision makers can know the severity of the consequences (or outcomes) of these scenarios with respect to their objectives. Implementation of CRT in this way narrows down the area of concern, whereby the purpose of the exploration is to construct scenarios outlining concerns regarding the threats towards achieving objectives.

Conflicting objectives arise when decision making involves more than a single objective, and no option best meets all of the relevant objectives simultaneously. When conflicting objectives exist, how do decision makers choose an option among the multi-attribute alternatives when no alternative dominates?

The most common approach for making decisions in problems involving conflicting objectives is to consider the extent to which one is willing to trade-off the relevant objectives. The presence of conflicting objectives and the difficulty to compare the trade-off among the alternatives cause tentativeness in decision making. Therefore, a tool that is able to represent the trade-off among the alternatives will be beneficial in decision making. With a conflict being represented by *red*, CRT can be used to explore the objective spaces; the concept is shown in [3].

In [3], the non-dominance in a multi-objective evolutionary search generates a set of

non-dominated solutions. The topology of the generated set in the objective space forms a topological structure known as a Pareto curve, as illustrated in Figure 2.1. Assuming a problem involves two objectives, each solution on the Pareto curve describes a particular trade-off between the objectives. Thus, they are considered to be independent; additional information, such as user preference, is required to determine the solution to be selected.

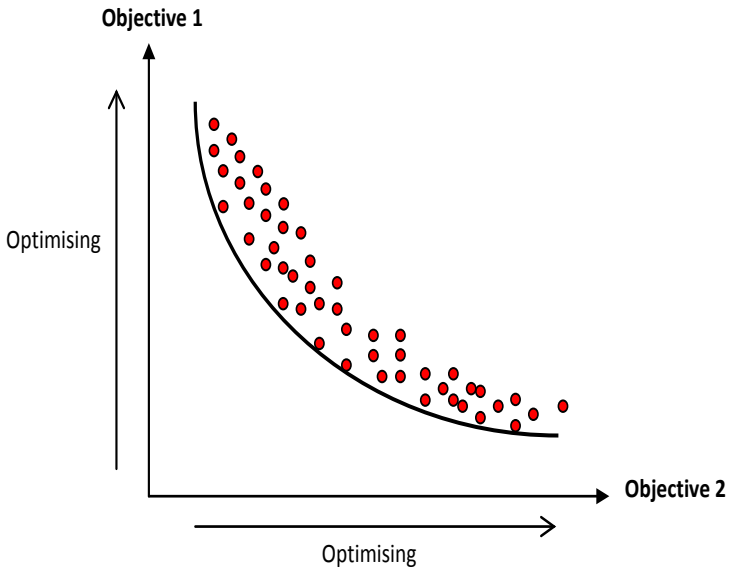


Figure 2.1: Pareto CURVE.

Another benefit that the Pareto curve can offer is to be used as an operating curve, known as a Pareto operating curve (POC). It is named as such because the relevant objective function needs to reflect the performance of a solution in an operating environment. If there are changes in the level of trade-off, the selection of solution changes from one point to another point on the Pareto curve. The advantage of POC is it offers adaption and/or robustness, given that systems operating along a POC are exposed to risk, complexity and trade-offs.

The development of software-based red teaming, which explores the weakness of the defence plan or operation, is another good example that shows the potential of CRT. CRT is used to discover vulnerabilities in a military plan or operation to mitigate risk before the

real implementation. Several examples of simulation engines that can be used for CRT in a warfare environment are WISDOM, EINSTein and MANA [89, 88, 31].

In these simulations, a military operation or plan that needs to be studied is viewed as the *blue* team, while the adversary is viewed as the *red* team. Since the risk analysis is conducted before the implementation of human-based red teaming, the exercise of the operation or plan involving real soldiers can be carried more effectively and efficiently based on the findings from the analysis. This approach is not only cost effective, but may also reduce physical and financial risks faced by the planners.

With an explicit representation of *red* and *blue* in CRT to simulate war scenarios, organisations are interested to know the possible strategies that can be used to outperform an adversary and what the possible outcomes are. The findings obtained from CRT can be used to educate planners on the selection of strategies to maximise the chance of *blue*'s survival, and success in real wars, thus, reduce physical and financial risks.

2.3 Adversarial Learning

.

Adversarial learning is a recently introduced term which refers to the learning process of a machine learning algorithm, in the presence of an adversary whose main goal is to cause dysfunction of the learning machine. An adversary, in this context, refers to an agent who explores possible ways to cause machine learning to fail.

A preliminary taxonomy categorizing adversarial attacks was suggested by Barreno et al. [15]. The taxonomy describes the capabilities of the adversary, which consist of *influence*, *specificity* and *security violation*. Later, Barreno et al. [13] suggested that the threat model in adversarial learning can be described as the adversary's *goal* and *capabilities*; where

capabilities can be referred to as capabilities of *information* and capabilities of *control*.

Adversarial information refers to the adversary's knowledge of the learning system and environment, such as the learner's features, the learning algorithm, the current decision function, the policy for training and retraining and other types of information. On the other hand, *adversarial control* refers to the extent of the adversary's ability to influence the learner's training and/or test data.

The most recent proposed taxonomy describes adversarial learning from two axes: *security goals* and *threat model*[14]. *Security goals* describe the goals from the learner's perspective, which relates to protecting the learning system from adversarial attacks. The security goal can be categorised into two goals: *integrity goal* and *availability goal*. According to Barreno et al.[14], *integrity goal* is to prevent adversaries from reaching a learning system, while *availability goal* is to prevent adversaries from interfering with normal operation. On the other hand, *a threat model* describes the profile of an adversary, which includes its motivation and capabilities. The threat model can be further categorised along three axes as suggested by Barreno et al [15]: *influence*, *specificity* and *security violation*. Based on different types of categorisations, we can observe that the earlier categorisations focus on the adversary's aspect only, while the more recent one views adversarial learning from two aspects: the learner and the adversary.

In adversarial learning problems, machine learning techniques are usually used to discriminate the malicious and benign instances as they provide an automated and adaptive approach. The reason that machine learning techniques are widely used is owing to their ability to adapt to the constantly changing nature of adversarial domains by extracting knowledge from the supplied data, and using the obtained information in the classification of unseen data. Therefore, machine learning algorithms are used in our research as well. There are several comprehensive reviews on the applications of machine learning al-

gorithms in adversarial domains, such as spam filtering [18, 32], fraud detection [40, 71], and intrusion detection [20, 40].

Most of the research done on adversarial learning focuses on the possible adversarial attacks faced by a machine learning algorithm, and then an attempt is made to develop some appropriate defense strategies against the attack. For the former focus, investigation on the possible ways of training data being subverted by adversaries is a popular research area. In [55], an adversary is assumed to attack an outlier detector by progressively inserting data until a malicious threat is no longer detectable. The attack involves an adversary determined to alter the detector to include a specific point by constructing data to shift the hypersphere towards the target as the hypersphere is retrained. The work in [56] shows that an adversary can control a large training data set used to build a classifier for worms and spam. They showed that the presence of a *delusive* adversary can obstruct the learning process of a classifier. A *delusive* adversary is an adversary that provides the classifier with correctly labeled data but manipulates the features in the target-class samples to mislead the classifier.

In [83], an approach was proposed to discover hidden patterns in a message delivered by a spam, and a classifier was built based on exploration of the discovered patterns. Instead of building a single classification model which performs averagely well on all messages, they proposed to use the lazy approach which only takes into account the useful messages to induce the rules for classification. The experimental result has empirically shown that the approach is effective in terms of classification accuracy and computational complexity.

The *good word* attack is another common adversarial attack in spam filtering. According to [36], *good words* are words that are commonly found in legitimate emails but not in spam. However, such words have been strategically injected into spam messages to avoid spam filters. To defend against this type of attack, [36] transformed each email into a

bag of multiple segments, and then applied multiple instance logistic regression on the bags. In the work, a spam filtering strategy was proposed based on the classical multiple instances assumption, whereby an email is classified as a spam if at least one instance in the corresponding bag is spam. On the other hand, an email is only classified as a legitimate email if all the instances in the corresponding bag are legitimate. The research shows that a classifier which adopts such counterattack strategy is more robust to good words attack than the single instance counterpart.

The research in [54, 14] shows how an adversary manipulates the SpamBayes spam filter specifically to cause failure by controlling only 1% of the training data. They also present three new attacks which cause the filter dysfunction, i.e., *dictionary*, *focused* and *pseudospam* attacks. They also present a potential defense known as Reject On Negative Impact Defense (RONI) against *dictionary* and *focused* attacks. RONI measures the incremental effect of a query email, by testing the performance difference with and without the email. This method managed to filter out the attacked messages successfully based on the impact of a message on the classifier performance.

Some researchers prefer to formulate adversarial learning as a game between learners and adversaries [22, 48, 14]. In [22] and [48], they formulated the classification problem as a game between classifiers and adversaries. The main difference between those two studies [22, 48] is the former assumed that the adversary has perfect knowledge of the classifier while the latter does not. As proposed in [22], an optimal classifier was produced to defend itself against adversarial attack by assuming the adversary launches its optimal attack.

On the contrary, [48] developed the adversarial classifier reverse engineering (ACRE) framework, by assuming the adversary learns a sufficient amount of information only about a classifier instead of perfect knowledge. According to them, adversaries must learn about the classifiers using some combination of prior knowledge, observations and experimenta-

tions. This was done with the adversary issuing queries to identify the negative instances with the minimum adversarial cost.

According to [14], an adversarial attack on a learning system can be modelled as games between them, with moves representing strategic choices made by either the learner or the adversary. The choices in a move depend on information derived from previous moves. In the game model, the learner's move is to choose a learning algorithm H for selecting hypotheses from the provided data sets, while the adversary chooses procedures for generating attacks against the learner according to its derived information from the learning environment. Therefore, the game between the learner and adversary is a repeated process.

2.4 Computational Red Teaming

Most research studies on adversarial learning focus on the possible adversarial attacks faced by machine learning algorithms, development of defence strategies against adversarial attacks, or both. However, the types of adversarial attacks, as well as the possible defence strategies, can come in many forms. It is just a matter of time that new attack techniques arise and existing ones evolve over time. Therefore, it will be beneficial to the research of adversarial learning if a general platform can be developed to cover both adversarial attacks and defense strategies simultaneously. This research direction was mentioned by Barreno *et al.* [14]. The CRT concept does exactly this. It studies the reciprocal interaction between defence and attack strategies.

The studies conducted in adversarial learning can be mapped into a CRT problem, in which the adversarial attacks are referred to as *red* while the underlying machine learning algorithms defending against the attacks are referred to as *blue* [1, 5, 6]. If the interest of decision makers is to know the limitations/weakness of the underlying machine learning

algorithms that are used to protect against adversarial attacks, CRT will be used to discover vulnerabilities. Whenever machine learning is used to provide protection against some illegal activities, adversaries will attempt to circumvent these approaches and thus form an arms race between legal and illegal activities. With an explicit representation of *red* in CRT to simulate adversaries, we are interested to know the possible effects that *red* can cause on *blue*, representing the machine learning algorithms. Since the focus lies on the simulation of *red*, *blue* will trigger response from *red* to circumvent the strategy from *blue*, thus, this will lead to the changes in the actions taken by *red*. Then, the decision makers would want to know how the changes made by *red* affect *blue*. For example, can the already implemented strategy of *blue* withstand the further changes from *red*?

CRT can be used to learn about the opponents if the interest is to understand an adversary's impact in a problem. Given that decision makers have sufficient information of the relevant aspects of the adversaries, if not entirely complete, CRT can be used to study the possible actions taken by the opponents in the environment. The exploration of the action space allows decision makers to anticipate the actions of the opponents, and organise their strategies in order to compete with opponents. As such, CRT can be used as a potential tool to teach decision makers in organising their strategies so that they are more prepared in real world situations.

By having appropriate representations of *red* and *blue*, CRT can be used to explore the emergence of behaviour of the decision makers and opponents/competitors. For instance, CRT is used to explore the space of possible actions taken by the opponents/ competitors, and the decision makers are trained in such scenarios. Since the decision makers are placed in a partially unknown and unpredictable environment, their actions may be based on trial and error through repeated cycles of interaction with the environment until their objectives are achieved. Through CRT, the emergence of behaviour is described, starting with a recurrent behaviour, leading to the construction of preferences for a behaviour, and

to the acceptance of a behaviour by most people [67]. Based on this case, CRT can be used to extract the emerged behaviour of the decision makers in an environment of interest. The emerged behaviour can be compared with human personalities of the decision makers. It is believed that personalities and other traits comparable to personalities influence the responsiveness of individuals to environmental stimuli in human and animal species [86]. In short, CRT can become a great potential tool to study the emergence of behaviour.

Another reason to have CRT is due to difficulties in acquiring and sharing data sets, especially those involving intelligent and adaptive adversaries. Without such data sets, it is difficult to compare the performance of the deployed learning algorithms in an adversarial environment, thus, limiting progress in adversarial learning. CRT not only can be used as a general platform for constructing and evaluating adversarial attacks as well as defence strategies, but also overcomes the difficulties of acquiring adversarial data sets. On top of that, the possible adversarial attacks from the implementation of CRT can be kept in databases to create future events, which can be used as precaution references. Whenever there is a new type of adversarial attack, the patterns of the attack can be mapped and compared with those in database so that appropriate solutions can be taken to minimise the threats.

Research on behavioural decision making, especially on the construction of preferences and/or beliefs, plays an important role in decision making. This is the case because a better understanding of the construction of decision behaviour could help us to manage our preferences more effectively, leading to the changes of the outcome of a decision. The selection of information and strategies to construct preferences and/or belief depend upon the task, context, and factors affecting individuals' differences [70]. In other words, the construction of preferences and/or beliefs can be viewed as a function of task, context and individual differences.

Task factor refers to general characteristics of a decision problem; it does not depend on particular values of alternatives in decision making. On the other hand, context factors refer to the characteristics that are associated with particular values of alternatives such as similarity of alternatives. Lastly, individual difference factors refer to the characteristics that describe the differences between one decision-maker from others, such as personality characteristics.

The study of the the interactions among these three factors can be combined, as shown in [89]. Based on the implementation of CRT, which normally involves the use of evolutionary computation, a wargame between the *blue* and *red* teams is simulated. The focus of the study is to investigate the effect of personality characteristics of the *red* team on the *blue* team, by evolving the best personality characteristics of the *blue* team. From the study, planners are able to know the possible appropriate combinations of personalities for the *blue* team that should be deployed for a given combination of personalities for the *red* team.

2.5 Evolutionary Robotics

Evolutionary Robotics (ER) refers to the use of evolutionary computation in the autonomous production of behavioural robot controllers. Usually, the control system used to determine the robot's behaviours are neural networks; this approach is adopted here. In ER, evolutionary algorithms are used to evolve neural networks (individuals) that control the robot's behaviour and the selected evolutionary algorithm is a genetic algorithm ("GA"). The research works related to ER can be found in [52, 82, 62, 65, 79, 78, 77, 76, 80].

The popularity of ER for synthesising and studying adaptive behaviour of natural and artificial agents stems from the possibility to rely on self-organizing processes [61, 57, 58]. Instead of explicitly dividing the desired behaviour into simple basic behaviours that are

handled separately by different layers or modules of the robot control system, neural networks (individuals) are selected based on their abilities to perform the desired behaviour as a whole. The fitness of neural networks is evaluated based on behavioural outcomes of the robot in performing a given task. Through evolutionary computation, users are released from the burden to decide how the whole desirable behaviour should be broken down. Instead, user's responsibility is just limited to determining the appropriate fitness function that should be used to evaluate the controllers.

Several reasons contribute to the selection of ER as a suitable approach to synthesize desired behaviour in machine red teaming. One of the main reasons is the ability to approximate the behaviour of a natural agent in red teaming.

In our research, a natural agent in red teaming refers to a human that will be placed in the same environment. According to Simon [75], rationality that can be observed in human beings in decision making is actually "bounded rationality" instead of global rationality. The concept of "bounded rationality" refers to decisions that are made, based on access to information and limited computational capabilities that are actually possessed by human beings, to produce functional behaviour in the environment in which human beings are placed. The concept actually lies in the adaptive ability of humans, which consists of the abilities to learn and evolve [64, 28, 60, 61, 63].

In ER, the abilities to learn and evolve are handled by neural networks and GA respectively. The main differences that distinguish them is the difference in spaces and time scale that the form of adaptation operates on. Evolution refers to the process of selective reproduction and recombination of individuals, based on the existence of a population of diverse individuals; while, learning refers to the modification process that happens within each neural network during its own life span. In terms of capturing changes based on time scales, evolution is able to adapt to the relatively slow changes that might spread over

several generations, but learning is not able to do so. Instead, learning can only adapt to the changes in environment that happens during the life span of an individual. The use of GA and neural networks in ER form the adaptive ability which is analogical to human's biological adaptation.

Another reason which encourages the use of ER is its ability to produce a population of diverse solutions, approximating the variability of human choices. Owing to the variability of human choices, it is undeniable that we will observe inconsistency of humans' strategies in the same task, where the use of different strategies may lead to similarity in the outcomes.

In the next section, we will introduce some fundamentals of GA and neural networks that form the basis for ER.

2.5.1 Genetic Algorithm

GA [35] is an evolutionary computation method motivated by an analogy to biological evolution. The principle of the method is based on the generation of successor individuals by repeatedly selecting and recombining parts of the best currently known individuals [51]. This procedure is repeated for several cycles known as generations. An individual, sometimes known as a genotype or chromosome, refers to a string that is made of units ("genes") that are arranged in linear succession. Each gene has control over the inheritance of one or several characteristics of an individual. Therefore, each individual (chromosome or genotype) represents a potential solution for a problem. The meaning of an individual, known as "phenotype", is defined by users. To evaluate the performance of the phenotype of each individual, a fitness function determined by users is used. The fitness function is used to determine the ranking potential and probability selection of the individuals to be included into the next generation. The higher the fitness value, the better the individual is.

GA starts with the random generation of a population of individuals, each yielding a potential solution for the problem at hand. The population of individuals is evaluated based on the designed fitness function, and each individual is associated with a fitness value. Then the population of individuals goes through a selective reproduction, whereby the best individuals are replicated in the next generation's population. This means the individuals associated with higher fitness values tend to have a higher number of copies in the next generation. Once a population has been created based on selective reproduction, the individuals in the new population are paired for crossover followed by mutation, to form the population for new generation. Crossover is applied with a given probability p_c to a pair of selected individuals, by swapping the genes between the pair at a random point along the individuals (Figure 2.2). After that, mutation is applied by arbitrarily modifying the genes of a selected individual based on a given probability p_m (Figure 2.3). A complete generation consists of fitness evaluation of all individuals, selective reproduction, crossover and mutation. This procedure is repeated for a number of generations until the stopping criterion is met.

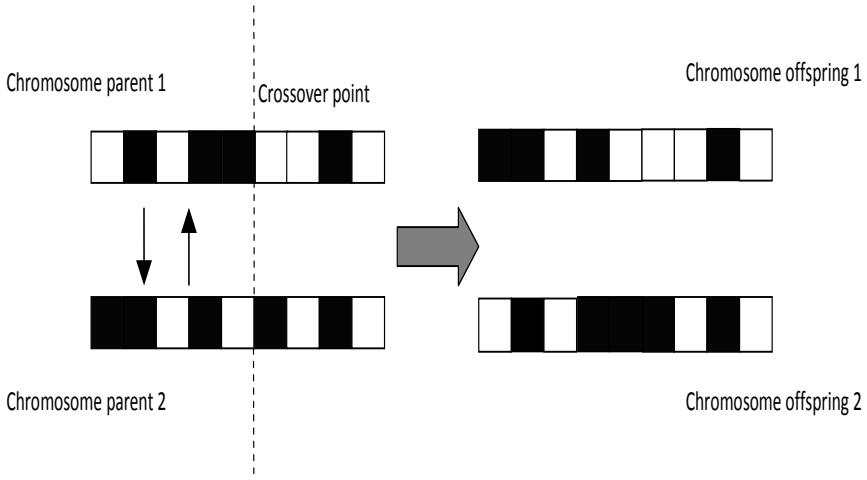


Figure 2.2: A conceptual diagram depicting examples of the crossover operator.

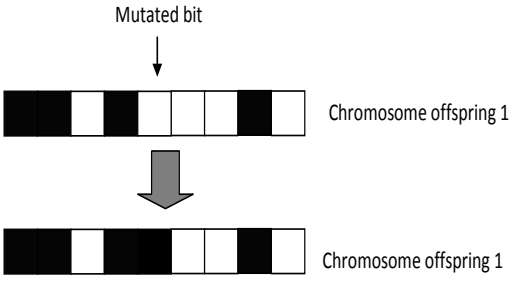


Figure 2.3: A conceptual diagram depicting examples of the mutation operator

2.5.2 Neural Network

A neural network consists of a collection of units known as neurons, connected by weighted links (also known as connection weights) used to transmit signals as shown in Figure 2.4. Units that receive signals from the external environment are known as input neurons while units which produce signals to the external environment are known as output neurons. Units that are located in between the input neurons and output neurons are known as hidden neurons.

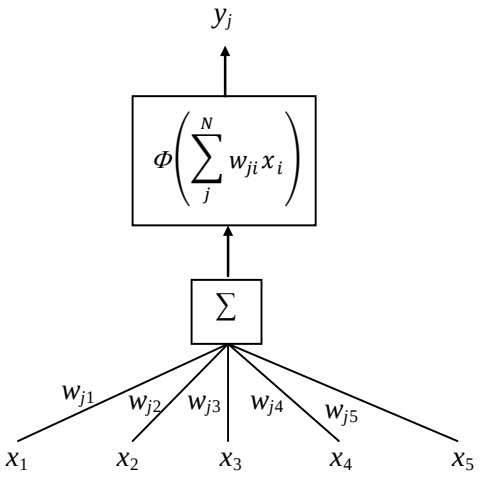


Figure 2.4: A simple neuron model

Based on the simple neuron model [50], the output unit y_j is a function of the sum of incoming inputs x_i weighted by the connection strengths w_{ji} between them as shown in Equation (2.1).

$$y_j = \Phi\left(\Sigma_i^N w_{ji} x_i\right) \tag{2.1}$$

The function Φ can be a step function, sigmoid function or linear function which are shown separately in Equations (2.2) to (2.4).

$$\Phi(y) = \begin{cases} 1 & \text{if } y > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

$$\Phi(y) = \frac{1}{1 + e^{-y}} \tag{2.3}$$

$$\Phi(y) = \lambda y \tag{2.4}$$

where λ is a constant that controls the inclination.

The architecture of a neural network is defined by the number of and pattern of connectivity. The type of pattern of connectivity that we would like to focus on in this section is a feedforward neural network trained using the back-propagation algorithm, because it has been proven to be successful in various domains [51]. Usually, a feedforward neural network consists of three layers of neurons. Similar to the simple neuron model, the layer that receives signals from the external environment is known as the input layer while the layer that produces a signal to the external environment is known as the output layer. The hidden layer is the layer located in between the input and output layers.

The behaviour of a neural network largely depends on the values of connection weights. Learning occurs in a neural network through the repeated adjustment of connection weights when a sample set of input/output is presented to the neural network. The adjustment of connection weights Δw_{ji} can be carried out either after the presentation of each pair input/output (online learning), or after the presentation of the entire set of input/output (offline learning). Based on Equation (2.5), the previous weights are updated by adding the newly computed modification Δw_{ji}. A parameter η known as learning rate is used to control oscillations that may happen during the weight update as shown in Equation (2.6).

$$w_{ji} = w_{ji} + \Delta w_{ji} \tag{2.5}$$

$$w_{ji} = w_{ji} + \eta \Delta w_{ji} \qquad 0 < \eta \leq 1 \tag{2.6}$$

2.5.2.1 Back-propagation algorithm

The back-propagation algorithm adjusts the weights in a multilayer neural network which has a fixed number of neurons and connection weights. The algorithm attempts to minimise the squared error E between the network outputs and the target values for these outputs, as shown in Equation (2.7).

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2 \tag{2.7}$$

where outputs in the equation refers to the set of output neurons in the network, while t_{kd} and o_{kd} refers to the target and output values associated with the kth output neuron and training example d. The purpose of gradient descent is to search for a hypothesis that minimises E. Assume each training example has a pair of vectors $\langle \overrightarrow{x}, \overrightarrow{t} \rangle$, where $\overrightarrow{x}$ refers to the input vector and $\overrightarrow{t}$ refers to the target vector. A feedforward network with 3 layers (input layer, hidden layer, and output layers) is created. The numbers of input neurons, hidden neurons and output neurons are denoted as n_{in}, n_{hidden} and n_{out} respectively. The input from neuron i into neuron j is denoted as x_{ji} and the weight connected from neuron i into neuron j is denoted as w_{ji}. With the mentioned denotations, the back-propagation algorithm is summarised as following:

- Create a feedforward network which consists of n_{in} input neurons, n_{hidden} hidden neurons and n_{out} output neurons.
- Initialise randomly all the connection weights, e.g., $[-1, +1]$.
- Until the stopping criterion is met, Do
  - For each $\langle \overrightarrow{x}, \overrightarrow{t} \rangle$ in training examples, Do
    1. Input the instance $\overrightarrow{x}$ to the network, compute the output o_u of every unit u in the network.
    2. For each network output unit k, calculate its error term δ_k based on Equation (2.8).
    3. For each hidden unit h, calculate its error term δ_h based on on Equation (2.9).
    4. Update each network weight w_{ji} based on Equation (2.6).

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k) \tag{2.8}$$

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{kh}\delta_k \tag{2.9}$$

Procedure 1 refers to the forward propagation of the input through the network, while procedures 2 to 4 refer to the backward propagation of the error through the network.

2.6 Gaps and Challenges in CRT

Based on the above literature, CRT has a great deal to offer in these domains. The question that we are attempting to answer rests on the feasibility of CRT. In other words, can an autonomous machine red teaming produce results that are similar to the results in real life? To answer this question, we need to model or explore the effect of *red* in the real world.

The context of our feasibility study focuses on behaviour modeling in an adversarial environment. Specifically, we are interested to know whether computational red teaming can produce similar behaviour results as HRT. To achieve the objective of our research, there are several challenges that require our attention:

Environment. In our study, comparisons need to be made between the behaviours of computational models and human responses in an adversarial environment. The comparisons may become invalid if both computational models and human beings are evaluated based on a task that is inadequately represented. Therefore, before a behaviour comparison

is made between the computational models and human beings, it is important to compare each of them with respect to the same environment. The decisions resulting from computational models and human beings are produced based on optimality and rationality with respect to the environment. If the environment is not the same, comparisons between both models could be inappropriate if they interpret the task in different ways, and thus influence the optimality and rationality of decision making in both models.

As pointed out in [26], optimality of decision making depends greatly on the nature and complexity of the environment. It can be evaluated based on two alternatives, i.e., build optimal models based on simplifying assumptions, or build heuristic models that maintain greater environmental realism. The importance of this statement rests in the need to have an environment which allows the decision making process during a task to be evaluated. By having the same environment, behavioural comparisons between computational models and human beings can be performed adequately.

Agent. In our work, we use the definition in [4] to define an agent, whereby it is equipped with three main capabilities, i.e., information receiving, decision making, and action production. In a human red teaming environment, the decision making of an agent is determined by human beings. With the received information, a human being will be responsible to process the information and justify the alternatives of actions that should be taken.

The next question will be how can we produce a machine red teaming? The question includes the selection of appropriate computational methods or models to produce machine red teaming, and the rationale behind it. One of the main factors that determines the selection is to mimic human beings in making optimal and rational decisions in order to produce functional behaviours.

Behaviours are functional if they contribute to the achievement of a certain intention,

objective or goal [26]. The central concepts of rationality and optimality lie in the abilities to learn and evolve. Here, optimality is defined as decisions that maximise or minimise some measurable criterion with respect to a given task in a given environment. On the other hand, rationality refers to a decision making process which allows decision makers to choose from alternative courses of action to maximise their payoffs. Computation of the payoffs for the actions are based on a "utility" or "value" function of the given sets of inputs and constraints [4]. To produce machine red teaming, we need computational methods or models that are able to produce optimal and rational decisions with the given inputs and constraints. Besides that, they need to have the ability to produce diverse solutions which are similar to diversity observed in human choices.

Task representation. Differences between the actions of computational models and human responses will be affected by task representation. To capture only the behavioural differences between them, task representation needs to be fixed so that both models could represent the task adequately in a similar way. Besides that, the chosen task needs to be able to be performed by both a natural agent and an artificial agent: human beings and computational models respectively. For example, the comparison of HRT and CRT in a warfare environment will be difficult if not impossible because the simulation of red teaming involving people is costly and risky.

The above mentioned challenges have led to the synthesis of a well-controlled red teaming game environment. With an appropriate representation of *blue* in a red teaming game environment, decisions made by the *red* agent to achieve certain goal(s) can be either determined by human beings or machines. The decisions made by human beings or machines are reflected in their actions. Patterns may be exhibited in the actions of the computational and human models and could be extracted for comparison. A better understanding of the differences/similarities of behaviours between the models helps to improve the implementation of CRT because the knowledge will prevent us from taking lightly the importance of

the representation of human cognition in CRT.

Chapter 3

Synthesis of Adversarial Attacks

3.1 Introduction

Adversarial learning refers to the learning process of a machine learning algorithm in the presence of an adversary whose main goal is to cause dysfunction of the learning machine. An adversary, in this context, refers to an agent who explores possible ways to cause failure to machine learning. The situation becomes worse if an adversary fulfills the contamination assumption [54], where it can control some of the user's training data. Instead of adding corrupted instances to confuse the machine learning system, the adversary intercepts the training process of the machine learning system by modifying the training data, thus leading to modifications of the learning model itself. The threats from adversaries become severe especially in domains that are easily exposed to malicious activities and electronic crimes. Instead of being useful, the results provided by a machine learning system in an adversarial environment lead to poor decisions. Measuring or evaluating the effect of an adversarial attack on machine learning helps system designers to reason and equip the system with some appropriate defense strategies. Therefore, assessing the vulnerability of a machine

learning algorithm to adversary attack is the focus of current chapter.

In this chapter, we investigate the effect of *Causative Availability* attacks on machine learning, with a specific focus on artificial neural networks. A *Causative* attack is one in which an adversary interferes with the learning process by modifying some of the training data. An *Availability* attack results in a broader type of classification errors, both false negatives and false positives. Therefore, a *Causative Availability* attack launched by an adversary disrupts the learning process and causes broader types of errors in a machine learning system. A *Causative Availability* attack can be further divided into either *Targeted* and *Indiscriminate*, whereby both attacks refer to the types of data being attacked. These two types of attacks are related closely with the samples selection methods which are explained in the later part in this section. From an adversary perspective, it can use the sample selection methods to carry out its attack.

The simulations of adversarial attack are carried out in both static and non-stationary environments. The purpose of carrying out the experiment in a non-stationary environment is to investigate the performance of the proposed ensemble and single neural networks under the condition that an adversary can dynamically launch its attack. However, the ability to detect changes is not our main investigation. The assumption here is a re-training mechanism is triggered to re-train the learning model whenever a change is detected. Our main concern is whether a re-trained ensemble performs better than the single neural networks or not in such an environment.

The main contributions of this chapter are as follows:

Sample selection based on the Covariance-Mahalanobis (CM) method. We propose a sample selection method known as the Covariance-Mahalanobis (CM) method. In the method, representative samples are selected from a large data set, a.k.a the frame, using measures from statistics. The basis of our measure of representativeness is the statistical

significance test between the frame and the chosen sample, as proposed in [37]. The details of representativeness test is explained further in Section 3.3.1.1.

Evaluation based on bias-variance analysis. Instead of using only the common performance measures, e.g., mean squared error (*MSE*), to measure the effectiveness of the sample selection methods, bias-variance is used to evaluate the network performance which is associated with the proposed sample selection methods. This is because bias-variance provides a better insight on the stability and reliability of neural network performance associated with the sample selection methods. The details of bias-variance is described in Section 3.3.1.2.

Simulation of *Causative Availability* attacks. *Causative Availability* attacks are carried out based on sample selection methods: a *Targeted* attack is simulated based on our proposed sample selection method, CM, and a *Indiscriminate* attack is simulated using a random sampling method, RND. The inversions based on CM and RND are referred as ICM and IRND in short. Both attacks are explained further in Section 3.3.2.

In detecting malicious activity, e.g., computer intrusion detection, the *positive* class (label 1) indicates malicious *intrusion* instances while the *negative* class (label 0) indicates benign *normal* instances. The adversary inverts some labels that resemble the positive and negative classes when the data are collected to train the learner. When the learner is trained on the attacked data, the learner may classify *intrusion* instances as *normal* and vice versa. If the labels of positive and negative classes are inverted arbitrarily, the inversion is known as IRND. On the other hand, the inversion is known as ICM if the attacked data are representative.

The details of the sample selection methods and simulation of the inversions are described in Section 3.3. The next concern in this chapter is whether an ensemble of artificial neural networks would perform better under both types of attacks than a single neural

network. In our experiments we use standard artificial feedforward neural networks as the learning algorithm.

Neural ensemble in adversarial learning. A neural ensemble is proposed to defend a system against the proposed *Causative Availability* attacks. Our hypothesis is the performance of an ensemble may not degrade as rapidly as a single classifier when they are exposed to adversarial attacks. We test this hypothesis as the neural networks in the ensemble are varied by their hidden neurons to decrease common vulnerabilities so that it is difficult to attack them as a group. Even though both the ICM and IRND methods are basically based on random sampling method, the main difference is ICM focuses on the representative samples. We would like to know whether a neural ensemble can still defend itself or not even the attack is focused on its representative samples. Furthermore, it is difficult to know in advance whether the attack is specific one or not. Accuracy and *MSE* are used to evaluate the performance of the ensemble and single neural networks on corrupted data. Here, bias-variance analysis is not used to evaluate the performance of ensemble and single neural networks because they are trained with the whole data set instead of samples.

The rest of this chapter is organised as follows. Section 3.2 presents related work on sample selection and adversarial attack simulation. In Section 3.3 we present our proposed sample selection method and two inversion methods, along with some background material on the statistical techniques used for measuring sample representativeness and bias-variance analysis. Section 3.4 describes the experimental setup. Section 3.5 presents and discusses the experimental results. Finally, Section 3.6 presents conclusions and future work.

3.2 Related Work

3.2.1 Sampling Techniques

Several researchers have proposed categorisations of sampling methods. Daszykowski et. al. [23] categorise sample selection methods into two groups, i.e., cluster-based designs and uniform designs. A more general categorisation approach is based on the evaluation scheme of the selected samples. Evaluation based methods can be divided into two groups, i.e., *direct measure* and *indirect measure* [47] a.k.a *filter* and *wrapper* approaches respectively [72]. For the *filter* approach (or *direct measure*), the evaluation of selected samples deals with the data only and does not involve any machine learning algorithm. As opposed to the *filter* approach, the *wrapper* approach (or *indirect measure*) involves data mining algorithms to guide sample selection. An initial sample is selected from the original data set and a machine learning algorithm is applied to build a model from the selected sample. The outcome of the algorithm is then used to determine whether the selected sample should be kept or eliminated.

In summary, the main distinguishing feature between *filter* and *wrapper* approaches is the order in which a learning algorithm is used to build the model from selected samples. For the *filter* approach, instance selection is carried out prior to any model building exercise and vice versa. In this paper, we are concerned with the *filter* approach which is more general than the *wrapper* approach as it works irrespective of the data mining algorithms. The categorisation of *filter* approaches can be further refined into two types; one that considers the frame properties for sample selection and the other that does not. The methods such as sampling and D-optimal design [87] ignore frame properties while our approach, *Kennard-Stone* [87, 39] design and *OptiSim* [21, 23] use frame properties for sample selection. An extended categorisation of sample selection methods is shown in Figure 3.1.

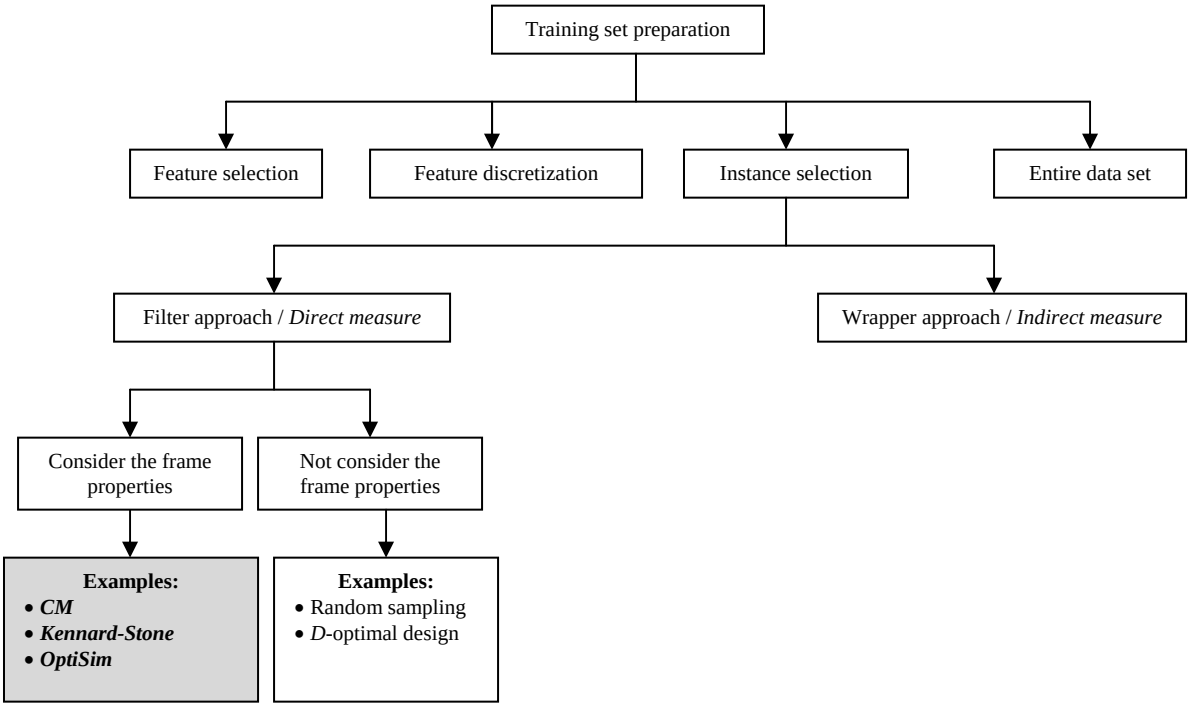


Figure 3.1: The refined categorisation of instance selection methods

The main difference between our proposed method and the *Kennard-Stone* and *OptiSim* methods is the types of frame properties that are considered. Both the *Kennard-Stone* and *OptiSim* methods take into account the frame's mean only. On the contrary, our method considers not only the mean but also covariance. Covariance reflects the amount of variation in the samples and also the extent to which variables are correlated.

3.2.2 Adversarial Learning

A taxonomy of adversarial learning is presented in [15], which describes adversarial attacks form three main perspectives: *influence*, *specificity* and *security violation*. *Influence* refers to the capability of an attack. It can be further divided into *causative* and *exploratory* attacks. A *causative* attack means an adversary has the capability to control training data while an *exploratory* attack does not. An *exploratory* attack explores the possible

information based on other techniques, such as off-line analysis. *Specificity* focuses on the types of data being attacked. *Targeted* and *indiscriminate* attacks fall under this category. A *targeted* attack happens when a particular data point or a small portion of data points becomes the aim of an attack. On the other hand, any arbitrary data is the aim of an *indiscriminate* attack. Lastly, *security violation* refers to the effects caused by an attack. There are two types of attack under this category, i.e., *integrity* and *availability*. An *integrity* attack causes false negatives while an *availability* attack causes both false negatives and false positives. The research studies in adversarial learning are shown in Section 2.3.

Although most research studies on adversarial attacks have focused on the spam detection domain, there is nothing special in this regard about spam detection: any machine learning in any domain can easily become the target of adversarial attack.

3.3 Methodology

3.3.1 Sample selection

In this subsection, we present steps involved in CM, which is also summarised in Figure 3.2. Assume that we have a large size of frame which consists of N data points. Instead of using the whole data set, n data points are randomly selected from the frame to form a sample, with $n \ll N$. Then we would like to know whether the selected sample is representative of the frame before it is included into the pool of training samples. Representativeness is evaluated in two stages, i.e., comparison of covariance and Mahalanobis distance.

In the representativeness test, we need to determine the critical values for the comparison of covariance and Mahalanobis distance, which are denoted as C_{crit} and D_{crit} re-

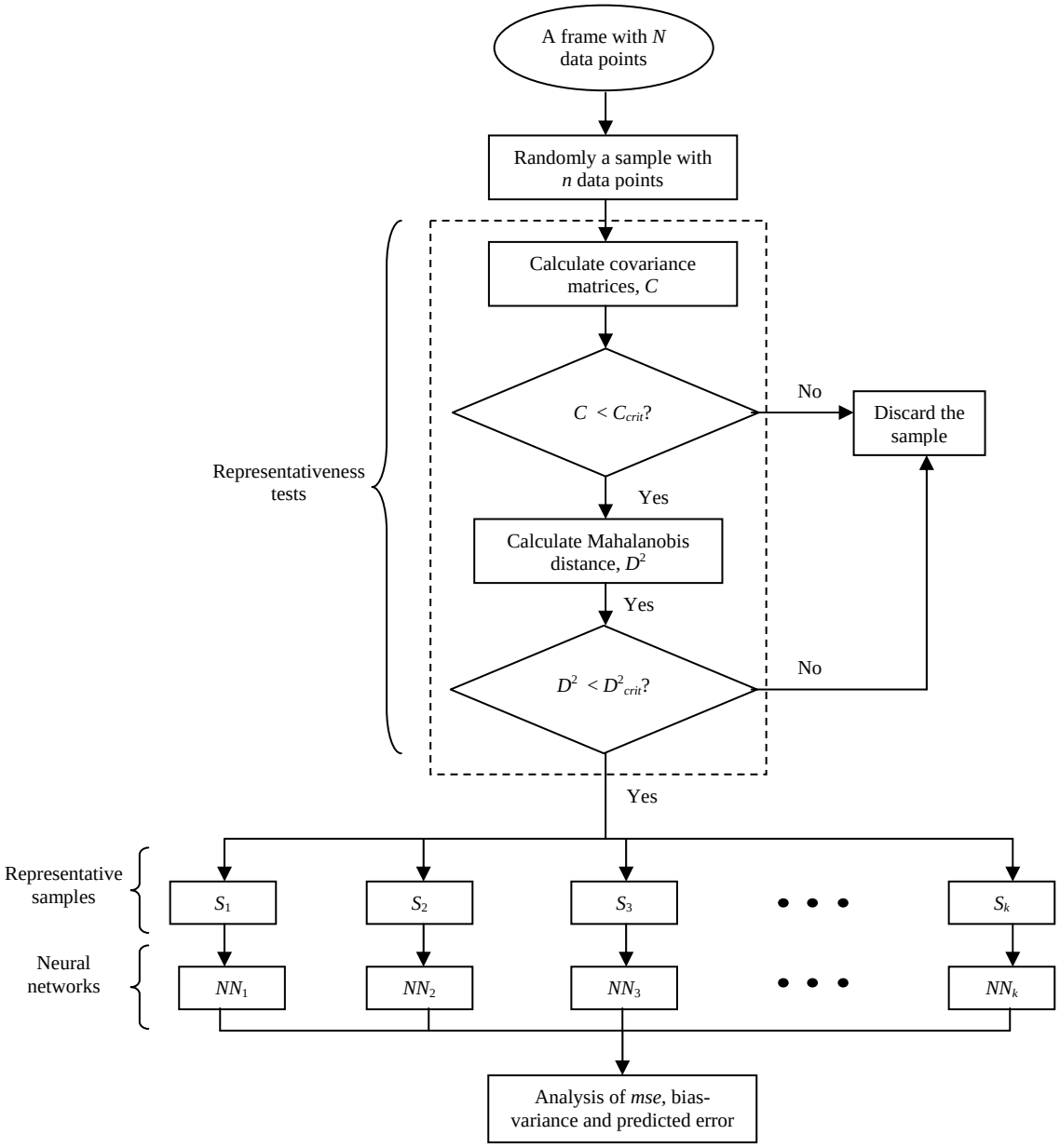


Figure 3.2: The flow chart of representative instances selection

spectively. Here, we would prefer C_{crit} and D_{crit} to be as minimum as possible because they reflect the measures of how far the sample is from its frame by taking into account all the variables being considered. If the calculated C of the sample is smaller than C_{crit}, the sample will go through the second stage of evaluation – the Mahalanobis distance. Otherwise, the sample will be eliminated from becoming a training sample. Again in the

second stage, if the calculated D of the sample is smaller than D_{crit}, this means the sample is a representative sample and thus becomes a training sample. If the samples is unable to meet the condition, it will be eliminated. To be included into the training samples, the sample need to meet two properties of representativeness – the covariance matrices and Mahalanobis distance.

3.3.1.1 Mahalanobis Distance and Covariance Matrices

In our proposed method, representativeness of a sample is measured by the different properties between the sample and the frame, based on significance tests. Hotelling's T^2 test is used to evaluate the significant difference of Mahalanobis distance, D between two data sets. Assume $\bar{\mathbf{x}}_1$ and $\bar{\mathbf{x}}_2$ are the mean column vector of the q variables in data sets $\mathbf{X}_1$ and $\mathbf{X}_2$, both having n_1 and n_2 instances respectively. The squared Mahalanobis distance, D, between the mean of each data set is defined in Equation (3.1):

$$D = (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)'\mathbf{S}^{-1}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2) \tag{3.1}$$

Then, Hotelling's T^2 statistic is defined as:

$$\begin{aligned} T^2 &= \frac{n_1 n_2[(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)'\mathbf{S}^{-1}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)]}{(n_1 + n_2)} \\ &= \frac{n_1 n_2 D}{(n_1 + n_2)} \end{aligned} \tag{3.2}$$

The significance or lack of significance T^2 is determined by using the fact that in the null hypothesis case of equal means, the transformed statistic as shown in Equation (3.3)

$$F = \frac{(n_1 + n_2 - q - 1)T^2}{(n_1 + n_2 - 2)q}$$

$$= \frac{(n_1 + n_2 - q - 1)}{(n_1 + n_2 - 2)q}[\frac{n_1 n_2 D}{(n_1 + n_2)}]$$
$$= \frac{n_1 n_2 (n_1 + n_2 - q - 1)}{q(n_1 + n - 2)(n_1 + n_2 - 2)} D \tag{3.3}$$

follows an F distribution with q and $(n_1 + n_2 - q - 1)$ degrees of freedom. Hotelling's T^2 statistic is based on the assumption that the two data sets are assumed to come from multivariate normal distributions with equal covariance matrices [49].

The comparison of covariance between two data sets is evaluated based on Bartlett's test, with the assumption that the samples are from multivariate normal distributions [49]. Assume $\mathbf{S}_1$ and $\mathbf{S}_2$ refer to the respective estimates of the covariance matrices of data sets $\mathbf{X}_1$ and $\mathbf{X}_2$, while $\mathbf{S}$ is the pooled covariance of these data sets. The Bartlett test checks that the computed value

$$C = \frac{1}{v\{(n_1 + n_2 - 2)ln(|\mathbf{S}|) - (n_1 - 1)ln(|\mathbf{S}_1|) - (n_2 - 1)ln(|\mathbf{S}_2|)\}} \tag{3.4}$$

with $v = 1 + \frac{2q^2+3q-1}{6(q+1)}(\frac{1}{n_1-1} + \frac{1}{n_2-1} - \frac{1}{n_1+n_2-2})$ follows a χ^2 distribution, with $q(q+1)/2$ degrees of freedom. To reject a null hypothesis of equal covariance, the computed value of $C > \chi^2_{q(q+1)/2,\alpha}$, where $\chi^2_{q(q+1)/2,\alpha}$ is the upper tail critical value for the distribution and α refers to the significance level. The calculated C needs to be as small as possible, compared to the critical value, to claim the covariance matrices between the two data sets are statistically equal.

The representativeness of a sample can be measured based on C and D values. The smaller are these values, the more representative is the sample with reference to its frame. However, we hardly know in advance the C-D limit (the lowest or highest) of a data set in real-world application. Therefore, we also propose a procedure to estimate the limit of C-D and its details are explained in 3.4.1.

3.3.1.2 Bias-Variance

In this section, we presents the details of bias-variance which is used to investigate the generalisation performance achieved by a neural network. Since the neural network is trained with samples selected by different sample selection methods, this means the bias-variance is used to evaluate the effectiveness of a sample selection method. For the explanation on bias-variance, assume that a single estimator after being trained by some training data, Υ produces estimates $y(\mathbf{x})$, so the squared error between the estimates $y(\mathbf{x})$ and target function $\langle t|\mathbf{x}\rangle$.

$$\{y(\mathbf{x}) - \langle\ t|\mathbf{x}\rangle\}^2 \tag{3.5}$$

The squared error depends on the training data that are used to train model that produces estimates $y(\mathbf{x})$. Therefore, the expected error produced by the model can be written as:

$$E_\Upsilon\{y(\mathbf{x}) - \langle\ t|\mathbf{x}\rangle\}^2 \tag{3.6}$$

Using some algebraic manipulation [38], equation (3.6) can be rewritten as:

$$E_\Upsilon\{y(\mathbf{x}) - \langle t|\mathbf{x}\rangle\}^2 = \{E_\Upsilon[y(\mathbf{x})] - \langle t|\mathbf{x}\rangle\}^2 + E_\Upsilon\{y(\mathbf{x}) - E_\Upsilon[y(\mathbf{x})]\}^2 \tag{3.7}$$

The first term refers to the squared bias, and the second term refers to the variance. Bias measures the difference between the average estimate over all possible samples of fixed size and the true function. On the other hand, the variance measures the difference between an estimate obtained for a single training sample and the average estimate obtained over all possible samples.

3.3.2 Adversarial Attacks Simulation

In this section, we present the simulations of *Causative Availability* attacks based on the samples selection methods, namely the CM and RND methods. Using the RND method, a sample is selected randomly from the training set. Inverting these instances is effectively launching a *Causative Availability Indiscriminate* attack. Using the CM method as mentioned in 3.3.1, a representative sample is selected from the training set. Inverting these instances is effectively launching a *Causative Availability Targeted* attack.

The following summarises the steps involved in the proposed *Causative Availability Targeted* attack. Assuming that we have a large frame that consists of N data points. Instead of attacking any arbitrary data, a percentage $m\%$ of training data is randomly selected from the frame to form a sample, where $m\%$ of data $< N$. We would like to know whether the selected sample is more "representative" of its frame than other equally sized random samples. The representativeness evaluation is based on C and D as mentioned in 3.3.1.1. If the sample meets the representativeness test, it is inverted. We call this inversion ICM. The alternative is to invert a sample chosen randomly; we call this inversion adversarial approach IRND.

Based on the statistics shown above, it is noted that representativeness increases with the decrease of C and D values. Therefore we propose the following procedures to select a representative sample, based on the C–D values, and then invert its outputs.

1. Randomly select 1000 samples, with the desired sample size, from the training set.
2. Calculate the C and D values for each sample.
3. Sort the samples based on the C–D values.
4. Randomly select a sample from a desired percentage $p\%$ of the total number of samples that are associated with the lowest C–D values. For example, if we set $p\%$ as 5%

(5% of 1000 is 50), a sample is randomly selected from 50 samples that are associated with the lowest *C–D* values.

5. Remove the sample from the training set and invert its outputs.

6. Return the modified sample to the training set.

7. Train the ensemble and single neural networks with the training set.

3.4 Experimental Design

3.4.1 Sample selection

An experimental design is presented to investigate the effects of representative samples selected based on the CM method on machine learning. As explained in previous section, the representativeness of a selected sample can be evaluated based on *C* and *D*. The main purpose of these experiments is to investigate the effect of these two properties on the *MSE* and bias-variance.

In most real world applications, we hardly know the true function of the collected data. Therefore, we investigate our proposed methodology on an artificial data set that is generated based on Equation (3.8). The true function of the underlying data need to be known for the analysis of bias-variance.

$$y_i \sim N(\mu = 10sin(\Pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5, \sigma_\varepsilon = 1) \tag{3.8}$$

In this data set, 100 combinations of x_1, x_2, x_3, x_4, x_5 is randomly generated in the domain [0,1], $T_1, T_2, T_i, \ldots, T_{100}$. For each combination, T_i, 100 y_i is generated based on Equation (3.8). Therefore, the frame of this artificial data consists of 10000 points. The frame is

divided into three sets, i.e., training set, validation set and test set. The sample size for the training set, validation and test set is 5000, 2500 and 2500 respectively.

Since the data set is generated artificially, we know the real values of mean and covariance of the frame of the artificial data. A sample with an arbitrary number of instances (sample size, n_s) is selected from the training set. A comparison of covariance and mean between the selected sample, S and the frame is carried out. Setting the significance level, α as 0.05, a selected S is known as a representative pre-sample, S_{pre} if the comparison of covariance, C and Mahalanobis distance, D are lower than C_{crit} and D_{crit} respectively. In the CM method, the selection S_{pre} with $n_s = 100$ is carried out until a large arbitrary number of S_{pre} are accumulated, e.g., 1500. From these representative S_{pre}, 30 S_{pre} associated with the lowest and highest *C-D* values respectively are used for neural network training. These 30 samples are known as representative post-sample, S_{post}. The above mentioned procedures are repeated to select 30 representative S_{post} with $n_s = 150$.

The performances of neural network are then compared with those of the RND method. In our experiment, we use different neural network sizes to investigate model independence in place of different classifiers. For the RND method, a sample, S with $n_s = 100$ is randomly selected from the training set but no comparison of *C-D* values is carried out between S and its frame. Unlike the CM method, there is no involvement of S_{pre} and S_{post} in the RND method. The sample selection is repeated until we have 30 S and they are used for neural network training. The same procedures are repeated for $n_s = 150$ in the RND method.

Since we don't know the true *C-D* limit (the lowest or highest of *C-D*), the following investigation is suggested to estimate it. This information is useful in guiding the selection of representatives samples from a pool of samples which associated with different *C-D* values. To indicate from which proportion pool of samples associated with the lowest *C-D* values that we should draw representative sample from, a flexibility parameter denoted as

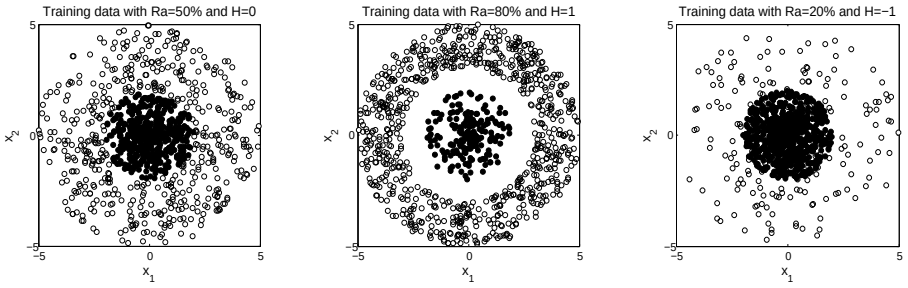
$p\%$ is introduced in this investigation. The purpose of this experiment is to investigate the effect of $p\%$ on the representativeness of a sample. To perform this investigation, a pool of samples with an arbitrary sample size needs to be firstly generated. A sample of 100 instances is randomly selected and included into the initially empty pool. The procedure is repeated until the pool sizes reaches 1000 samples. These samples are then sorted based on their *C-D* values, in ascending and descending orders.

In the case of ascending order, assume that we need 30 representative samples, the parameter $p\%$ will determine from which proportion pool of sorted samples the selected samples are drawn. Since we have 1000 samples and the $p\% = 5\%$, we select 30 samples from 50 sample associated with the lowest *C-D* values (5% of 1000 is 50). *MSEs* of the neural networks trained on these samples are measured and plotted. The range of $p\%$ to be investigated are 5%, 10%, 15%, 20%, 25%, 30%, 40%, ..., 100%. The above procedures are then repeated for the samples, whereby their *C-D* values are sorted in a descending order. Two graphs (*MSE* versus $p\%$) are produced, where one of them contains the results of the *C-D* values that are sorted in an ascending order while the other is in a descending order. The *C-D* limit can be estimated from the intersection of these graphs and it is denoted as $p\%_{threshold}$. In other words, it means that if we want to select representative samples from a pool of samples which are sorted in an ascending order for the *C-D* values, the suggested value of $p\%$ should be lower than $p\%_{threshold}$.

3.4.1.1 Artificial Data

The purpose of this initial experiment is to show how each single neural network which forms part of the ensemble performs differently when they are under adversarial attacks.

In the artificial data, let us assume there are two classes of points, x_1 and x_2, that can be separated or classified using a separating hyperplane as shown in Figure 3.3(a). The classes



(a) Data generated with Ra=50% and H=0 (b) Data generated with Ra=80% and H=1 (c) Data generated with Ra=20% and H=-1

Figure 3.3: Artificial data, generated by varying Ra and H

(positive and negative cases) are equally balanced in number and uniformly distributed in the space. Positive and negative cases are represented by black and white dots respectively. We can further generate different characteristics of data by varying two variables, Ra (the ratio between the positive and negative cases) and H (the distance between the separating hyperplane which separates the two classes). In effect, this mechanism varies the degrees of freedom around the support vectors in different data sets. Figure 3.3(b) and 3.3(c) show examples of data generated with different values of H; (the former is generated with positive H and the latter with negative H).

We created data sets with eleven different values of H $(-1.0, -0.8, -0.6, \ldots, 1.0)$ and nine different Ra values $(10\%, 20\%, 30\%, \ldots, 90\%)$. Therefore we have 99 sets of data with different characteristics. Each data set contained 1000 instances.

For each data set, samples of 100 instances ($m\% = 10\%$) are inverted based on each of ICM and IRND. Therefore, a series of training sets with 10% of inverted data is produced. The series of training sets is used to train the ensemble and single neural networks. Besides that, we also generate validation set and test set, which are used to monitor the generalization and to verify the network performance respectively. The ratio of training set, validation set and test set is 2:1:1.

3.4.1.2 Static Simulation

Four data sets were obtained from the UCI repository[29], to compare the performances of the ensemble and single neural networks against the proposed attacks under static environment. Three are classification problems while the fourth is a regression problem, i.e., Wave, Page Block (PB), Magic Gamma Telescope (MGT) and Wine Quality (Wine). The details of each data set are shown in Table 3.1.

Here, we refer to the original data set as the frame. We divided the frame into three sets (training, validation, and test set) in the ratio 2:1:1.

For each data set, samples with different $m\%$, i.e., 0.5%, 1%, 5%, 10%, 15%, ..., 30% are inverted based on ICM and IRND. Therefore, a series of training sets with different percentages of inverted data are produced for each UCI data. The series of training sets is used to train the ensemble and single neural networks.

Table 3.1: The details of data sets obtained from UCI repository.

Data set	Types	Attributes	Class	Frame size	Training size	Validation size	Test size
Wave	Classification	22	3	5000	2500	1250	1250
PB	Classification	11	5	5473	2736	1368	1369
MGT	Classification	11	2	19020	9510	4755	4755
Wine	Regression	12	-	4898	3248	1624	1625

3.4.1.3 Non-Stationary Simulation

The simulation of non-stationary environment is carried out by using the same UCI data sets and a spam data set [33]. Firstly, each of the original data sets need to be divided into 10 subsets and 1 test set. The sizes of test set for the wave, PB, MGT, wine and spam are 125, 130, 456, 179 and 116 respectively. The remaining data are then divided into 10 subsets, with each subset is then further divided into training set and validation set according to the ratio of 2:1. The inversions based on ICM and IRND are then carried out

on these 10 training sets, with different $m\%$, i.e., 0.5%, 1%, 5%, 10%, 15%, ..., 30%. The neural ensemble and single neural networks with the same parameters as the ones in static environment are performed by using these training sets. Their generalization performances are monitored by using the validation set. The trained neural networks are then validated on the test set. The inversions of several training sets is to simulate the dynamics of an adversary. The performance of the neural ensemble and single neural networks in such environment are evaluated based *MSE* of the test set.

3.4.2 Neural Ensemble and Single Neural Network

In this chapter, we use artificial neural networks as the machine learning algorithm owing, to their ability to learn complex and nonlinear functions even with little prior knowledge about the underlying true function. However, conceptually any machine learning algorithm can be substituted in place of neural networks.

There are two ways of using neural networks to evaluate the effect of training data on their performance[87]. One is to use a fixed structure, while the other is to use the optimal network structure obtained with each training sample. We use the former method, to eliminate the confounding effects of other factors so that the changes in performance depend on the characteristics of the data rather than on the structure of the network.

The learning rate r for the network is set as 0.1 and the numbers of hidden neurons are deployed for the network training according to the types of data: (i) artificial generated data - in both artificial data sets in sample selection and adversarial attack simulation, different number of hidden neurons. i.e., 5, 10, 15 are deployed for network training; (ii) UCI data - the number of hidden neurons used in network training are 5, 10, 15, 20, and 25.

MSE of the validation set is used as the stopping criterion in the network training. For

each 100 iterations, *MSE* of the validation set is monitored until it reaches the maximum iterations, 1,500,000. Based on the monitoring, we would like to identify the iteration where the validation set produces the minimum *MSE* and thus, it is used to finally train the neural network. The network performance is then evaluated with the test set. The effectiveness measurement used in the experiments of sample selection and adversarial attack simulation are different. In the experiment of sample selection, the effectiveness of the CM and RND methods are measured in terms of *MSE* and bias-variance. On the other hand, the effectiveness of the ICM and IRND are measured in terms of accuracy and *MSE* in the classification and regression problems respectively in the experiment of adversarial attack simulation. 30 runs, using different seeds, were made with each combination of parameters.

In the adversarial attack simulation, a neural ensemble can be formed by combining the outputs from each single neural network based on a simple combination rule (average, majority-voting, or winner-take-all) to produce a consensus output. We investigated each of these combination rules with every data set, except the Wine data. The ensemble deployed with the Wine data set uses only the average and winner-take-all methods to produce the final output.

3.5 Main Results

3.5.1 Sample selection

This section presents the findings that we obtain from the experiment which investigates the effectiveness of sample selection. It contains two major parts:

- Analysis on the effect of *C-D*

- Estimation of the *C-D* limit

For the box plots shown in this section, the notation a, b, c in the x-axis , refers to 5, 10 and 15 hidden neurons, e.g., $a(1), a(2), a(3), \ldots, a(6)$ refers to the combinations of 5 hidden neurons with different sample sizes and *C-D* values . In the legend, the notation of x in CM(x,y) refers to sample size and while y refers to high or low values of *C-D*. The notation of x in RND(x) refers to sample size only.

Figure 3.4 presents the results obtained in the *MSE* analysis for the CM and RND methods. By referring to the figure, the *MSE* produced by the random sampling method reduces significantly as the sample size increases, i.e., from sample size 100 to 150 when the hidden neuron is 5. We observe similar effect for the random sampling method when the hidden neurons are 10 and 15 respectively. This means the random sampling method shows similar effect irrespective of hidden neuron. Therefore, the random sampling is said to be model independent. Based on the observations, our proposed methods (i.e., the CM method) can be viewed as model independent as well because the increase of performance for different sample sizes is similar as the hidden neuron increases. To show there is a significance decrease of *MSE* as the sample size increases, a t-test is carried out on the CM and RND method which associated with different samples sizes. Setting the significance level, α as 0.05, the following hypotheses are set out:

- *Null hypothesis*: The sample mean of measurement, i.e., *MSE* for the samples associated with low sample size is less than or equal to those of high sample size;
- *Alternative hypothesis*: The sample mean of measurement, i.e., *MSE* for the samples associated with low sample size is greater than to those of high sample size.

Table 3.2 show the results of t-test for the above hypotheses and we can observe that all of the null hypotheses are rejected for both the CM and RND methods irrespective of

the network size. This means that *MSE* produced by both methods are significantly lower as the sample size increases. Beside that, Figure 3.4 shows the samples associated with high *C-D* produce larger *MSE* than those with low *C-D*. This similar effect can be observed irrespective of the sample size and hidden neuron. Since the performance of different neural networks associated with the CM is similar, we can empirically show that our proposed methods are indeed model independent.

Table 3.2: The t-test of *MSE* test sets between samples which have different sample sizes, i.e., 100 and 150 for the CM and RND methods. For the CM method, the t-test is carried out on the samples which share the same *C-D* values.

No	HN	Method	MSE_{test}		
			p-value	t-value	Reject H_0
1	5	CM(high *C-D*)	1.5933E-16	11.368	Yes
2		CM(low *C-D*)	3.7440E-11	7.9718	Yes
3		RND	1.2026E-14	10.069	Yes
4	10	CM(high *C-D*)	1.0210E-30	22.812	Yes
5		CM(low *C-D*)	4.8677E-24	17.211	Yes
6		RND	1.0836E-24	20.379	Yes
7	15	CM(high *C-D*)	9.0977E-34	27.714	Yes
8		CM(low *C-D*)	1.2544E-30	23.579	Yes
9		RND	2.5404E-30	25.108	Yes

We also hypothesise that the *C-D* value may significantly influent the network performance. The hypotheses of our study is set out below:

- *Null hypothesis*: The sample mean of measurement, i.e., *MSE*, bias, variance and predicted error for the samples associated with high *C-D* is less than or equal to those of low *C-D*;
- *Alternative hypothesis*: The sample mean of measurement, i.e., *MSE*, bias, variance and predicted error for the samples associated with high *C-D* is greater than those of low *C-D*.

To confirm the effect of *C-D* in the CM method, an analysis is carried out on *MSE* and bias-variance. Table 3.3 shows a two-sample t-test on *MSE* for high and low *C-D*. By

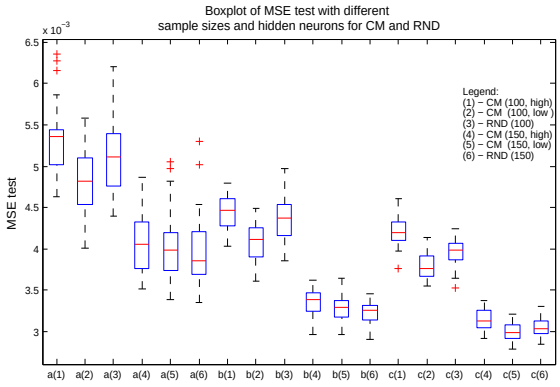


Figure 3.4: *MSE* of test sets for CM and RND with various network sizes.

Table 3.3: The *t*-test of *MSE* test sets between samples which have high and low *C-D* values but same sample sizes for the CM method.

No	HN	SS	MSE_{test}		
			p-value	*t*-value	Reject H_0
1	5	100	0.0002	3.7023	Yes
2		150	0.2934	0.5461	No
3	10	100	3.40E-08	6.1852	Yes
4		150	0.1119	1.2297	No
5	15	100	5.33E-14	9.7208	Yes
6		150	3.65E-05	4.2747	Yes

observing the *t*-test of *MSE* when sample size is 100, all of the null hypotheses are rejected irrespective of the network size. For sample size 100, there is enough evidence to claim that the samples associated with low *C-D* produces smaller *MSE* than those of high *C-D*. On the other hand, for sample size 150, some of the null hypotheses are not rejected and this happens when the network size is small. However as the network becomes larger, the effect of *C-D* becomes significant since the null hypothesis is rejected. Besides that, the *p*-value reduces as the hidden neuron increases for the same sample size. Therefore, we can conclude that the *C-D* value has significant effect on *MSE* especially with larger network size.

Another performance measurement that is used to evaluate the effect of *C-D* is bias-variance. Figure 3.5 shows the boxplots of bias, variance and predicted error of the test set for CM and RND methods. The figure shows that the RND method produced the

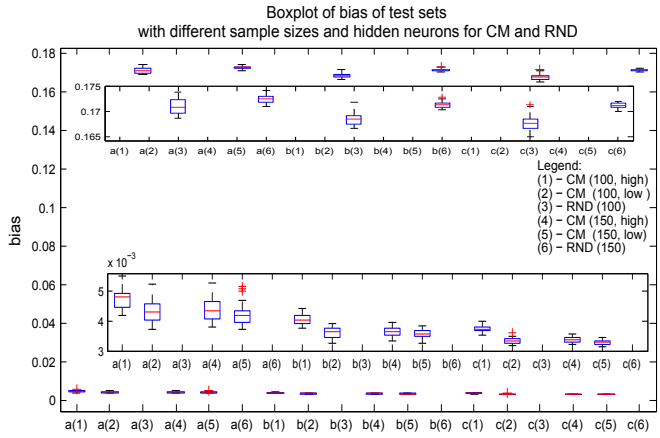
Table 3.4: The t-test of bias, variance and predicted error applied on the test set between samples which have high and low *C-D* values but the same sample sizes for the CM method.

No	HN	SS	$Bias_{test}$			$Variance_{test}$			$Predicted\ error_{test}$		
			p-value	t-value	Reject H_0	p-value	t-value	Reject H_0	p-value	t-value	Reject H_0
1	5	100	2.53E-05	4.3813	Yes	6.62E-06	4.7682	Yes	2.92E-09	6.8244	Yes
2		150	0.0413	1.7668	Yes	0.1228	1.1732	No	0.0001	3.8591	Yes
3	10	100	1.23E-13	9.4468	Yes	5.18E-13	9.0769	Yes	1.07E-22	15.62	Yes
4		150	0.0442	1.7338	Yes	0.0427	1.7501	Yes	3.54E-10	7.4243	Yes
5	15	100	9.03E-23	15.874	Yes	2.08E-19	13.406	Yes	1.14E-28	21.448	Yes
6		150	0.0006	3.4004	Yes	6.54E-05	4.1001	Yes	1.17E-16	11.406	Yes

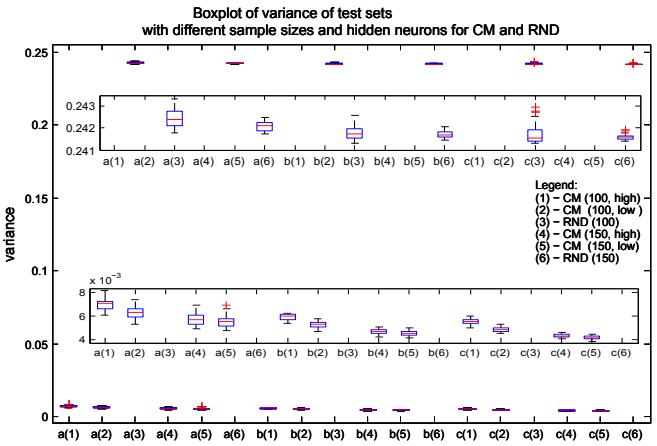
highest bias-variance for the test set. For both the CM method, the samples associated with low *C-D* produce smaller bias-variance, which lead to smaller predicted error as well because it is the sum of bias-variance. Based on the t-test of the bias and variance as shown in Table 3.4, they shows similar findings as those of *MSE*; the null hypotheses are significantly rejected as the network size grows except for the CM method with sample size 150. Despite the null hypothesis is not rejected, it's p-value only exceeds the significance level marginally. According to [19], the alternative to rejecting the null hypothesis is not necessarily to accept it. Otherwise the null hypothesis would be accepted in all instances once the p-value is in excess of the significance level. The rough guide is the null hypothesis might be accepted if $|t| \leq 1$. The smaller is the t-value below 1, the greater the firmness in accepting the null hypothesis. A smaller difference is much more likely to be owing to sampling error than a larger one. Since the t-value for the CM method with sample size 150 is 1.1732, which is greater than 1, it is much more unlikely in accepting the null hypothesis. Therefore, we can conclude that the value of *C-D* has significant effect on the bias-variance, especially when the network size grows.

In spite of showing compatible results as the random sampling method in term of *MSE*, the CM method produce much smaller bias and variance than the RND method, which leads to smaller predicted errors as well.

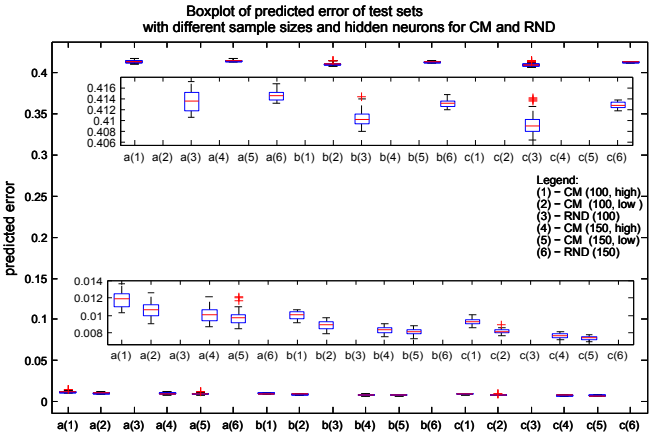
In reality, it is important to estimate the limit of *C-D* so that we can know where the representative sample should be drawn from. Figures 3.6 shows the average *MSE* test



(a) Bias



(b) Variance



(c) Predicted error

Figure 3.5: Bias, variance and predicted error of test sets for CM and RND with various network sizes.

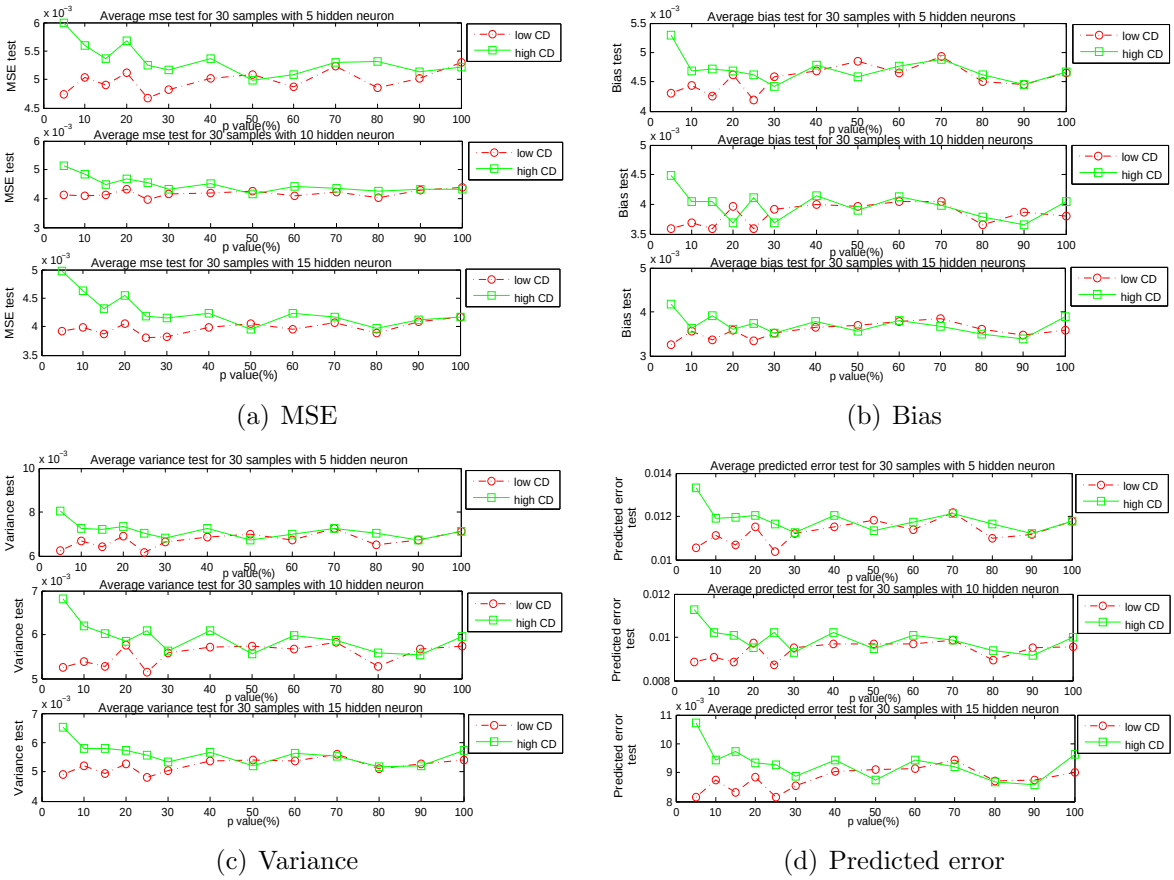
of 30 samples against $p\%$ for low and high *C-D* with sample sizes 100. For sample size 100, as $p\%$ is small, there is a significant difference of *MSE*, bias-variance and predicted error between low and high *C-D*, where low *C-D* produces smaller measurement values compared to high *C-D*. However, the difference becomes less apparent as the value of $p\%$ increases. In general, low *C-D* produces smaller *MSE*, bias, variance and predicted errors than those of high *C-D* for small $p\%$. Furthermore, the plots provide a threshold value, $p\%_{threshold}$ of where the representative sample should be selected from. The threshold value refers to the point where the lines of low and high *C-D* intersect. The samples drawn from the pool of samples associated with the lowest *C-D*, where its $p\% \leq p\%_{threshold}$, produce small *MSE*, bias, variance and predicted error in general. In other words, these samples are more representative than those of $> p\%_{threshold}$. This finding is used as a guideline to find and invert the representative samples in the following simulation of *Causative Availability Targeted* attack.

3.5.2 Adversarial attack simulation

3.5.2.1 Artificial Data

In the artificial data set, we carry out two-sample t-test on *MSE* produced by each single neural network, after being trained by the ICM data and the IRND data. Here, we introduce an index R which indicates the outcome of the t-test. The following shows the possible outcomes:

- If the network has equal sample mean of *MSE* after being trained with the ICM data and IRND data, $(MSE_{ICM} = MSE_{IRND})$, $R = 0$.
- If $MSE_{ICM} > MSE_{IRND}$, $R = 1$.
- If $MSE_{ICM} < MSE_{IRND}$, $R = -1$.

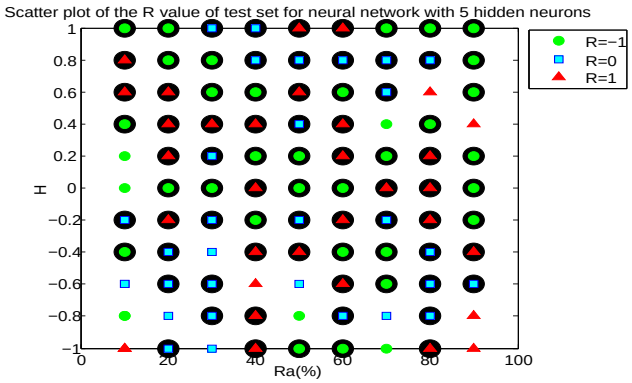


(a) MSE (b) Bias

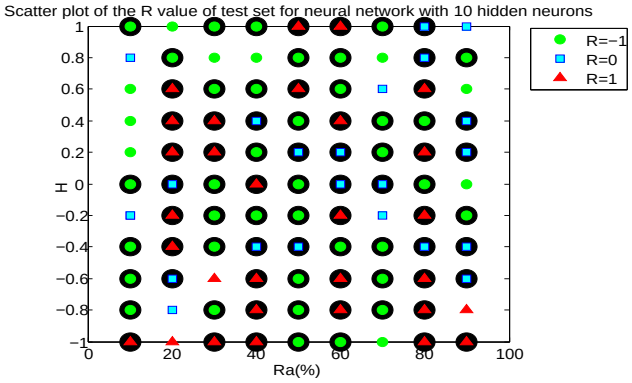
(c) Variance (d) Predicted error

Figure 3.6: *MSE* and generalization error of test sets for various $p\%$ and network sizes when the sample size is 100

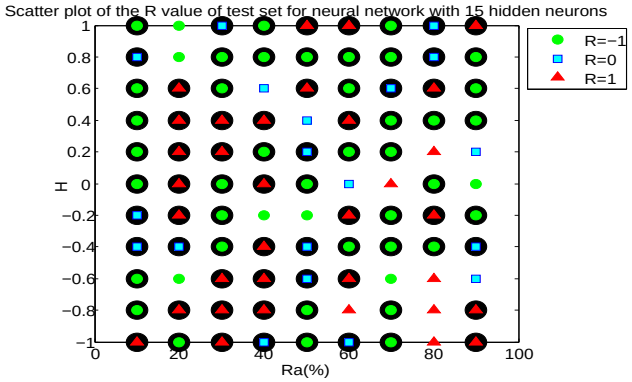
Then, we use the outcome of the t-tests to produce scatter plots (see Figures 3.7(a) to 3.7(c)). The x-axis refers to Ra while the y-axis refers to H. Different colours of markers in the scatter plot indicate the performance difference of each single neural network, when trained with the ICM-data and IRND-data. Therefore, the green, blue and red markers in the scatter plot indicate the R values -1, 0 and +1 respectively. When the ICM-data and IRND-data are used to train the single neural networks, their performances against these attacks are different. One of the neural networks may defend itself better than others with the ICM-data while the another defends itself better with the IRND-data. In other words, the vulnerability of each neural network against adversarial attacks is different.



(a) The R value for neural networks with 5 hidden neurons



(b) The R value for neural networks with 10 hidden neurons



(c) The R value for neural networks with 15 hidden neurons

Figure 3.7: Scatter plots of R for neural networks with different hidden neurons

With the same characteristic of data, the R values for the single neural networks with 5, 10 and 15 hidden neurons are different. This means their defences against these two attacks are different. Therefore, the combination of these neural networks could help to

Table 3.5: Percentage of instances where the ensemble performs at least as well as single neural networks, with artificial data

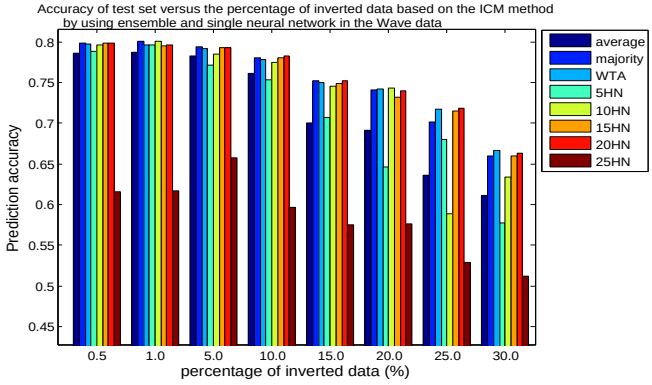
Hidden neuron	ICM			IRND		
	Average	Majority	Winner-take-all	Average	Majority	Winner-take-all
5	99.71%	99.70%	99.57%	99.69%	99.68%	99.53%
10	99.75%	99.81%	99.72%	99.73%	99.79%	99.67%
15	99.74%	99.81%	99.82%	99.73%	99.80%	99.79%

reduce common vulnerabilities against adversarial attacks, thus produce better performance than a single neural network.

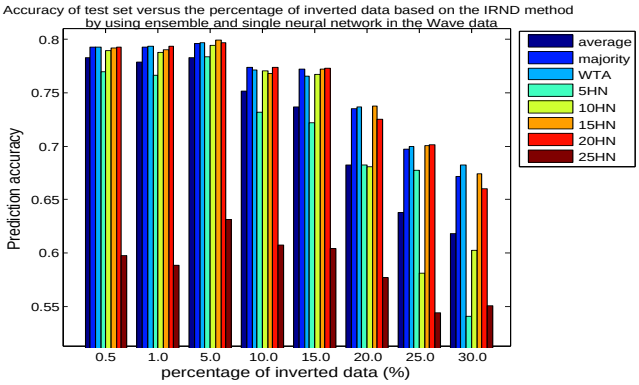
To create a neural ensemble, we combine the outputs from each single neural network based a combination rule: average, majority-voting, or winner-take-all. The predictions of the ensembles are compared with each of the single neural network. If the performance from at least one of the ensembles is better than the single neural network, the marker in the scatter plots in Figure 3.7 is circled in black. From Figures 3.7(a) to 3.7(c), we can observe that most of the markers are circled in black, showing the ensemble performs better than the individual neural networks in most of the artificial data sets. This also suggests that the use of ensemble to carry out classification is a better option, because we hardly know what is the true characteristic of the inverted data.

The accuracy of the ensemble in this classification problem is then compared with each of the individual neural networks. Table 3.5 shows how often the ensemble performs at least as well as the individual neural networks when they are applied to data with different characteristics. The results show that the ensemble is better than a single neural network more than 99% of the time, regardless of the types of combination rules in the test set.

Having demonstrated the feasibility of using neural ensembles with artificial data sets, we turn to real data sets.



(a) The performances of the ensemble and single neural network in the Wave data after being inverted based on ICM



(b) The performances of the ensemble and single neural network in the Wave data after being inverted based on IRND

Figure 3.8: Performances of the ensemble and single neural networks after adversarial attacks, with the Wave data set

3.5.2.2 UCI and Spam Data

Figure 3.8 presents the accuracy of the ensemble and single neural networks with the Wave data, when trained with the ICM-data and IRND-data. We can observe that the accuracy of both decreases as the percentage of inverted data increases. Despite this, the performance of the ensemble, using average or majority-voting, performs as well as or better than most of the single neural networks.

From Figure 3.8 we can also see that the performance of the single neural networks with 5, 10 and 25 hidden neurons deteriorates as the percentage of inverted data increases, irrespective of the inversion type. This means if we rely on a single neural network in a classification problem, the prediction most probably is less accurate when it is exposed to adversarial attack. This finding may also indicate that the performance of an under-fitted or over-fitted neural network degrades more than a generalised network as the percentage of inverted data increases. Unfortunately, it is difficult to determine in advance which single neural network can generalise better even if it is attacked by an adversary.

Figure 3.8 also shows that the performance of an ensemble which uses the average is not as competitive as the majority-voting and winner-take combination rules, because its final output is affected more by members that perform poorly. This implies that averaging the networks, which is biased towards their mean performance, is not reliable in these test cases. Nevertheless, it still performs better than the worst single neural network.

For the PB and MGT data sets, the trends are similar to the Wave data set (graphs are not presented here due to space limitations). Table 3.6 presents the key results for all of the UCI data sets.

Table 3.6 shows how often at least one of the ensembles performs at least as well as the individual neural networks, with UCI data. The ensembles perform better than the single

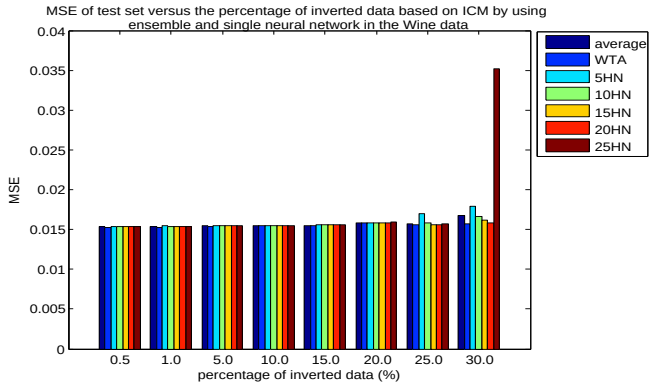
Table 3.6: Percentage of instances where the ensemble performs at least as well as single neural networks, with UCI data under a static environement. HN refers to hidden neurons.

Data	Method	ICM					IRND				
		5HN	10HN	15HN	20HN	25HN	5HN	10HN	15HN	20HN	25HN
Wave	average	94.03%	94.43%	93.00%	93.40%	95.90%	93.68%	93.84%	93.80%	93.72%	95.19%
	majority	97.71%	98.59%	98.01%	98.56%	94.94%	97.90%	98.00%	98.41%	98.85%	94.23%
	WTA	96.87%	98.03%	98.06%	98.96%	94.40%	97.24%	97.47%	98.26%	98.70%	94.07%
PB	average	90.95%	90.81%	90.76%	90.73%	90.75%	90.17%	90.03%	89.95%	89.91%	89.92%
	majority	91.08%	90.98%	90.94%	90.90%	90.92%	90.37%	90.27%	90.19%	90.15%	90.16%
	WTA	91.71%	91.61%	91.57%	91.53%	91.55%	92.18%	92.09%	92.01%	91.97%	91.98%
MGT	average	98.92%	99.02%	98.96%	98.90%	99.07%	98.77%	99.04%	99.04%	98.97%	99.02%
	majority	99.11%	99.28%	99.22%	99.15%	99.00%	98.91%	99.31%	99.31%	99.21%	98.98%
	WTA	98.42%	98.69%	98.70%	98.70%	98.40%	98.46%	99.06%	99.17%	99.29%	98.70%
Wine	average	51.83%	51.83%	50.17%	50.21%	50.51%	51.54%	51.54%	50.34%	49.87%	51.42%
	WTA	59.86%	59.86%	56.45%	64.80%	63.36%	58.58%	58.58%	60.33%	61.18%	65.56%

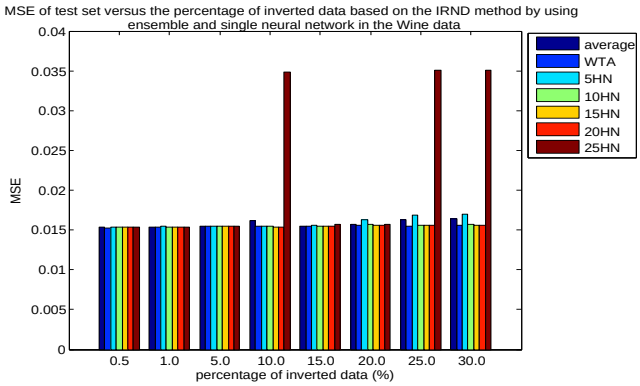
neural networks more than 90% and 50% of the time in the classification and regression problems respectively. Generally, the above findings suggest that an individual neural network is more vulnerable to adversarial attacks than an ensemble.

Since the Wine data is a regression problem, *MSE* is used to measure network performance instead of accuracy (see Figure 3.9). Figures 3.9(a) and 3.9(b) show that *MSE* of the single neural network with 25 hidden neurons increases drastically when the percentage of inverted data is large. In contrast, the ensemble manages to maintain its performance. For ease of comparison, Figure 3.10 shows *MSE* of the ensemble and single neural networks from 0.5% to 20.0% of inverted data. Figure 3.10 clearly shows the ensemble — especially the one based on winner-take-all — always produces smaller *MSE* than the others even with a high percentage of inverted data.

The performances of the proposed neural ensemble and single neural networks under the non-stationary environment are evaluated as well. Owing to space limitations, we only present the graphs for the spam data. However, the key results of all data sets are summarised in Table 3.7. Figure 3.11 shows the performance of the neural ensemble and single neural networks in the Spam data under the non-stationary environment for both the ICM and IRND inversions. Based on the graphs in Figure 3.11, we can observe the performance of a neural ensemble is either comparable to or better than the single neural



(a) The performances of the ensemble and single neural network in the Wine data after being inverted based on ICM



(b) The performances of the ensemble and single neural network in the Wine data after being inverted based on IRND

Figure 3.9: Performance of the ensemble and single neural networks after adversarial attacks, with the Wine data set

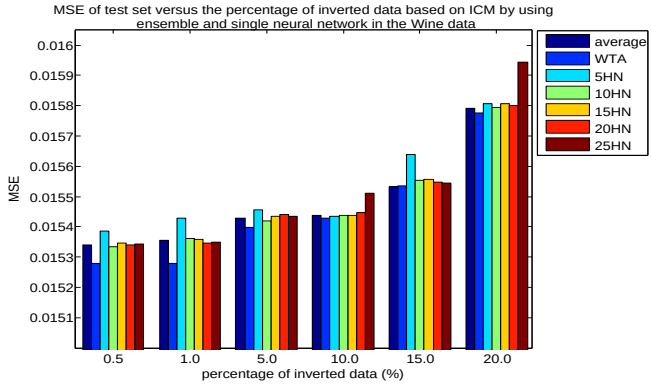
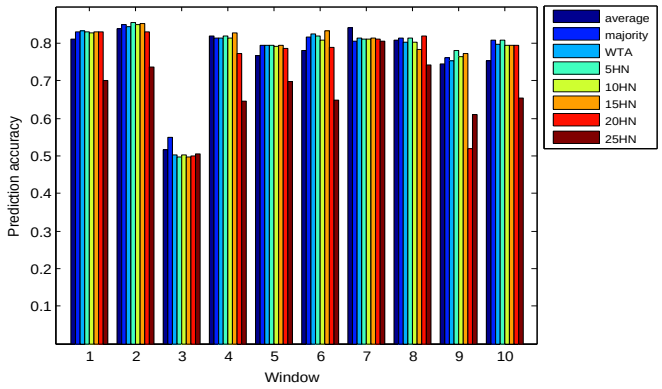


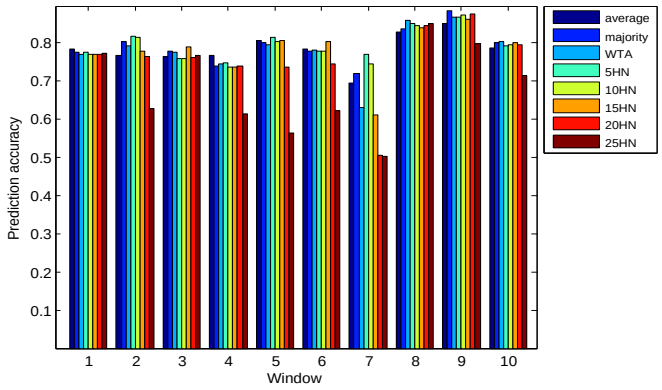
Figure 3.10: The zoom in of Figure 3.9(a)

networks under the non-stationary environment. This means the proposed ensemble shows promising results in spam detection.

Table 3.7 summarises the frequency of at least one of the ensembles perform at least as well as the single neural networks under the non-stationary environment. Except for the Wine data which is based on the average method, the ensembles perform better than the single neural networks in the remaining cases, with the frequencies distributed between 52% - 99%. This suggests that an ensemble generally performs better than single neural networks even under the non-stationary environment.



(a) The performances of the ensemble and single neural network in the Spam data after being inverted based on ICM



(b) The performances of the ensemble and single neural network in the Spam data after being inverted based on IRND

Figure 3.11: Performance of the ensemble and single neural networks after adversarial attacks, with the Spam data set

Table 3.7: Percentage of instances where the ensemble performs at least as well as single neural networks, with UCI data and spam data sets under a non-stationary environment. HN refers to hidden neurons.

Data	Method	ICM					IRND				
		5HN	10HN	15HN	20HN	25HN	5HN	10HN	15HN	20HN	25HN
Wave	average	92.33%	89.56%	91.40%	84.38%	87.33%	91.76%	89.54%	90.23%	82.67%	86.47%
	majority	97.22%	96.97%	96.60%	89.41%	84.09%	96.60%	97.00%	96.49%	86.35%	84.80%
	WTA	95.62%	95.90%	95.40%	88.03%	83.22%	95.01%	94.56%	95.46%	82.63%	81.28%
PB	average	97.02%	95.46%	96.10%	95.01%	95.35%	90.76%	93.43%	87.50%	89.27%	90.77%
	majority	98.93%	98.12%	98.90%	98.11%	97.89%	97.30%	99.02%	96.54%	98.73%	97.48%
	WTA	98.41%	97.16%	97.80%	96.72%	96.70%	96.92%	98.34%	96.60%	98.23%	97.27%
MGT	average	90.06%	90.07%	90.20%	90.62%	95.04%	91.79%	92.12%	92.11%	91.92%	94.70%
	majority	98.68%	99.00%	99.10%	98.65%	97.14%	98.41%	98.82%	98.72%	98.75%	98.03%
	WTA	98.57%	98.61%	98.80%	98.34%	96.64%	98.10%	98.48%	98.41%	98.42%	97.51%
Wine	average	39.78%	39.13%	39.80%	37.95%	51.96%	40.45%	41.57%	42.24%	41.57%	54.73%
	WTA	62.42%	62.54%	66.10%	54.44%	60.42%	63.89%	63.25%	62.41%	52.61%	61.51%
Spam	average	89.62%	89.78%	90.70%	93.33%	95.79%	93.19%	93.46%	92.23%	94.10%	95.99%
	majority	97.64%	98.65%	97.90%	94.94%	92.63%	98.09%	98.59%	97.79%	98.10%	93.90%
	WTA	97.52%	98.30%	97.10%	93.71%	91.34%	96.72%	97.14%	97.52%	97.24%	92.41%

3.6 Conclusions

This chapter proposes a framework which simulates *causative availability targeted* and *causative availability indiscriminate* attacks, and then assesses neural networks' performance against these threats. *Causative availability targeted* attacks are simulated based on CM, while *causative availability indiscriminate* attacks are simulated based on RND, whereby both CM and RND are the sample selection methods. Before these two sample selection are methods used to simulate the attacks, their effects on network performance are found. Despite being based on random sampling, CM causes higher bias-variance as compared to RND if it is associated with low *C-D*. In other words, the *causative availability targeted* which simulated based CM may cause higher deterioration in network performance.

The heart of our inversion lies in the attacks against the outputs instead of inputs. Instead of adding extra corrupted data into the data set, the outputs of the selected sample are inverted to confuse the neural network.

The performances of the proposed ensemble and single neural networks are evaluated in static and and non-stationary environments. Based on the experiment results, they show an ensemble performs comparable to or better than single neural networks in both

simulated environments. The experiment results also show that the performances of the ensemble and individual neural networks deteriorate with the increase of inverted data in both simulated environments. The deterioration is more significant in larger single neural networks. This may be due to over-fitted networks. Despite this, the study shows that the ensemble performs better than most of the single neural networks when they are exposed to both adversarial attacks. This is important because it shows that the ensemble can still defend itself even the attack is focused on a specific group of instances. Also, an ensemble has better potential to minimise rapid deterioration than a single neural network after being exposed to adversarial attacks. This suggests the vulnerability of an ensemble against adversarial attacks is lower than a single neural network, which means the prediction from an ensemble is more reliable than a single neural network. If we rely heavily on a single neural network, it is difficult to know whether the network can still defend itself if it is incrementally attacked by an adversary.

We have observed that the performance of an ensemble may be affected by some of its members that perform poorly. Therefore, in the future we would like to focus on developing a mechanism which can automatically increase or decrease the influence of some of the members in producing the consensus output. Besides that, a mechanism to detect and react to changes in the data should be investigated. The work done in [11, 10] has introduced a powerful concept which can address the problem. Therefore, the concept can be borrowed to further improved our proposed ensemble in the future work.

Chapter 4

Synthetic Simulation Environment

4.1 Synthetic Simulation Environment

The objective of our research is to investigate the similarities/differences of behaviours between human red teaming and machine red teaming. To perform the investigation, it is very important to carry out the comparison in the same environment, so that the possible differences which may arise due to task representation and behaviour adequacy can be reduced or eliminated. We hardly know in advance which of the two behaviours (human or machine) presents a more adequate representation of the needs of a task.

This leads to the development of the red teaming environment. It needs to be a well-controlled environment so that a specific change in the dependent variables can reliably be produced by specific manipulations of the independent variables, and the change is unlikely to be the result of confounding variables. To create a well-controlled environment, a synthetic red teaming game environment is created and an agent simulation is used.

The game environment involves two agents known as *blue* and *red*, whereby the *blue* agent is represented by a predator and the *red* agent is represented by a prey. Both agents

have opposite goals: the *blue* agent tries to catch the *red* agent while the *red* agent tries to escape the *blue* agent.

For an entity to be declared as an agent, it needs to be equipped with the abilities to sense/observe inputs, to make decisions based on the observed inputs and to act according to these decisions [4]. This means the *blue* and *red* agents need to have these three abilities in order to achieve their goals.

The ability to obtain information is abstracted by the availability of inputs to the agents, since the experiment is carried out entirely in a simulation environment. On the other hand, the ability to make decisions refers to the selection of which location to move to by the agents based on the received information. Lastly, the ability to act refers to the movements of the agents in the environment.

The simulation is based a low-resolution strategic behavioural model because it ignores detailed physics of agents and only captures behaviours in the model. Interesting behaviours emerge as a result of maneuvering of the agents in space and times of the simulation. Several examples of low-resolution strategic behavioural simulation models are WISDOM, EINSTein and MANA [89, 88, 31] which are widely used to study a military plan or operation. For these models, the attraction-repulsion weighting approach is used to determine the movements of the agents in the simulation.

Unlike these models, the approach to determine the movements of agents in the environment is influenced by information and deception. The details about the implementation will be explained in Section 4.1.2.

Both the *blue* and *red* agents are equipped with a so called "sensorimotor system" which enables them to move in the environment based on their decisions. The sensorimotor system produces the planned actions based on received input information about the environment and their opponents. For the *blue* agent, its sensorimotor system produces a direct action

according to scripted strategies after receiving information through the agent's sensor. For the *red* agent, its sensorimotor system can be either a machine or a human. For both machine and human sensorimotor systems, the production of planned actions involves more complicated computational processes of input information. Since the game is carried out in simulation, the relevant inputs needed by the sensorimotor systems based on scripted strategies and machine are made directly available to them.

Figure 4.1 shows the schematic representation of the environment while Figure 4.2 shows the interface of the game environment. The *blue* and *red* agents are represented by the triangles in blue and red respectively. The perimeters of the circles surrounding the *blue* and *red* agents (shown in dotted lines) represent the possible locations that the agents can move to in one time of step due to the constraints that have been imposed on them (as explained in detail in Section 4.1.1).

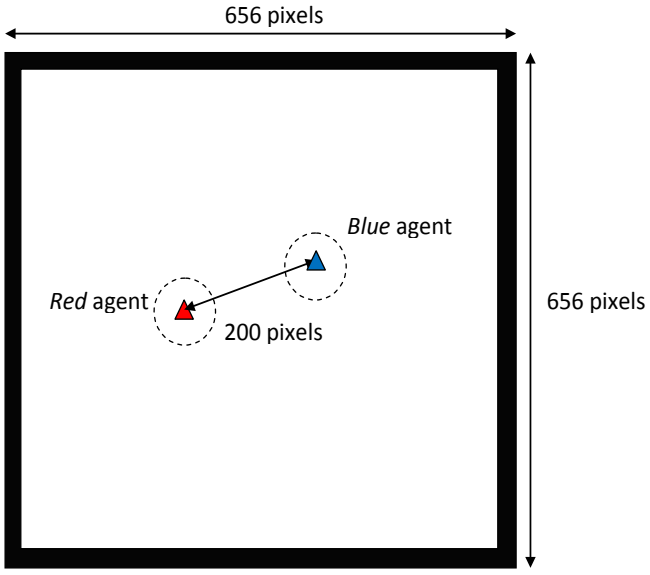


Figure 4.1: Schematic representation of the initial setup of the *blue* and *red* agents in the game environment.

The simulation is limited to one-to-one agent only. Instead of focusing on the complexity of the interaction between the agents, we focus on understanding the behaviour between a

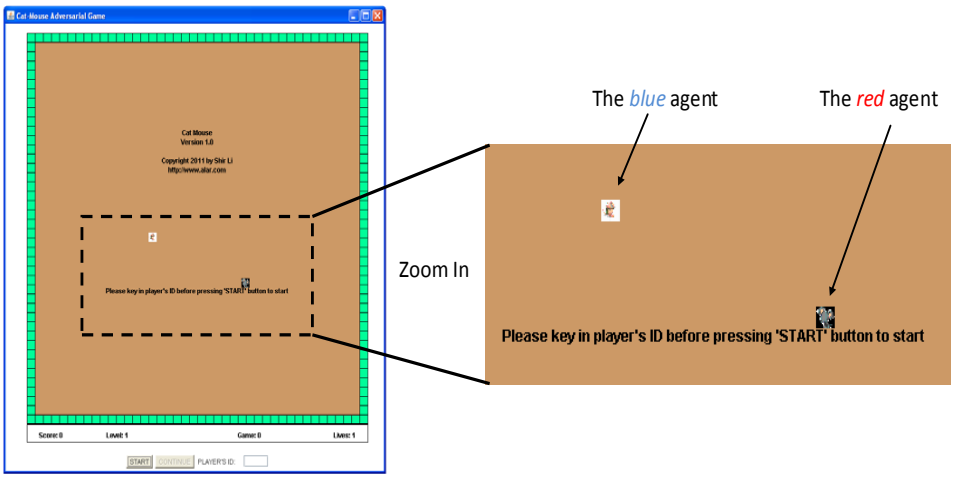


Figure 4.2: Interface of the game environment.

single pair of agents. Increasing the number of agents in the simulation, e.g., many-to-one and many-to-many relations may become overwhelming and complicated if the study lacks appropriate metrics and methodologies to analyse the behaviour of the agents. Instead, we would like to understand the behaviour from this basic relation based on our proposed behavioural analysis framework. Once the metrics and methodologies have been proven to work adequately, the work can be expanded to more complex relations.

The ultimate goal of the proposed framework is to understand human behaviour and machine behaviour in the same environment, through analysing how these two forms of *red* behave in response to specific formations of the *blue* agent. This could be done by having a human or a machine as the *red* agent. The formations of *blue* are characterised by information and deception. The question can be answered appropriately if the *blue* agent's strategy is fixed but the *red* agent's strategy is evolved. Having also the *blue* agent to operate based on learning machines, e.g., evolutionary computation, will introduce more complexity and stochasticty into the investigation. Consequently, it will be difficult to obtain an insight on the effect of the formations on the behaviour of the *red* agent. Having full automation of red teaming by involving learning machines for both agents does not necessarily provide answers or insights on the complexities of red teaming, which is cen-

tralised on human activities [4]. Some key questions related to this issue are: how different or similar is a computational *red* agent's behaviour compared to a human's behaviour? how to represent or simulate a *red*? how to extract the information? etc. All these questions can be summarised as the core of red teaming, which consists of three main components, i.e., context, computation and analysis [4].

4.1.1 Constraints

The environment is bounded by several constraints.

- In the game environment, 1 cm = 1 pixel. Both agents are placed in a $2D$ continuous environment of 16cm $\times$ 640cm, where x and y refer to coordinates in environment. The coordinates (x, y) start with the value 16 instead of 0 because the $2D$ grid environment is surrounded by rectangular walls.

- The speeds for the *blue* and *red* agents (represented by ξ_b and ξ_r respectively) are both fixed at 10cm per time step ($\xi_b = \xi_r = 10$). The reason that the speeds for both agents are set to be equal is to emulate the scenario that both agents have equal capabilities, so that capturing can only occur through good strategy.

- The movements of the agents are determined by their travel angle; the travel angles for the *blue* and *red* agents are denoted as θ_b and θ_r respectively. The values of θ_b and θ_r are within the range of [0, 2π]. The generation of travel angles depends on the strategies used by the agents.

The game starts and ends as follows:

- The initial locations for both agents are randomly generated, with distance between them, $d_{opp} = 200$cm.

- The game is terminated if one of the following constraints is met:
  - the distance between the *red* agent and *blue* agent, $d_{opp} < D_{min}$, with $D_{min} =$ 20cm. D_{min} refers to the distance at which *red* is considered to be caught by *blue*.
  - maximum number of time steps is reached, $S = 100$ steps.

4.1.2 Strategies

The movements for both agents are determined by their strategies to generate travel angles. In other words, the generation of travel angles depends on their decision models. The *blue* agent's strategies are scripted. The *red* agent's strategies are determined by either a machine or a human. For the *red* agent whose movements are determined by a machine, neuroevolution is selected as the approach.

The *blue* agent arranges its strategies with the knowledge of the existence of an intelligent *red* agent. The intelligence of the *red* agent is reflected in its actions resulting from its own situation awareness. For example, a thief who has information about the police's patrol route would not break into the houses which are located within the route. Due to the thief's situation awareness, it is unlikely that the police will be able to catch the thief.

Therefore, the access that the *blue* agent has to the *red* agent's intelligence can be obtained through observing the *red* agent's actions. However, this may be affected by the frequency of observing the *red* agent's actions and the noise which may exist in the observations. Given that the *red* agent's actions are perceived by the *blue* agent as intelligence about the *red* agent, we call the frequency of receiving information about the *red* agent the "frequency of intel" N_I, while the noise contained in the information is the "noise in intel" $\hat{\alpha}^{(t)}$. Here, $\hat{\alpha}^{(t)}$ actually reflects the control that the *red* agent has over how it is perceived

by the *blue* agent. This means the *red* agent intentionally moves away from its targeted location so that its movements become unpredictable by the *blue* agent.

Based on the received information, the *blue* agent arranges its actions to deceive the *red* agent. The deception is produced by the *blue* agent intentionally to confuse the *red* agent and disrupt the *red* agent in achieving its objectives. This can be demonstrated using the previous example of a thief and the police. In this case, the police will take the same patrol route for a few days in order to make the thief believe that the police will take the same route for the following day. Instead, the police changes their routes. Consequently, with a higher probability the thief will be caught by the police if the police patrols at the areas which are not in their previous patrol route. To create a deception produced by the *blue* agent in the environment, we introduced two parameters: namely deception cycle length, N_D; and deception range, $\zeta^{(t)}$.

The pseudo code shown in Algorithm 1 describes the strategy used by the *blue* agent. The parameters used in the strategy are as follows:

- $\hat{P}_b^{(t)}$ - Actual position of the *blue* agent at time t.
- $\hat{P}_r^{(t)}$ - Actual position of the *red* agent at time t.
- $\hat{P}_{ir}^{(t)}$ - Intel reporting on the position of the *red* agent perceived by the *blue* agent at time t.
- $\hat{\alpha}^{(t)}$ - the relative change in x and y coordinates due to the level of noise added to $\hat{P}_r^{(t)}$, expressed as $\hat{\alpha}^{(t)} = (\Delta x, \Delta y)$.
- Counter - Game clock.

The demonstration of the above proposed parameters means the *red* agent has some control over how it is perceived by the *blue* agent. The *blue* agent also demonstrates its

Algorithm 1 Update intel and position

while *Counter* is not stopped **do**
  Update intel:
  if (*Counter* modulo N_I) $= 0$ **then**
    $\hat{P}_{ir}^{(t)} \leftarrow \hat{P}_{r}^{(t)} + \hat{\alpha}^{(t)}$
  else $\{$(*Counter* modulo N_I) $\neq 0\}$
    $\hat{P}_{ir}^{(t)} \leftarrow \hat{P}_{ir}^{(t-1)}$
  end if
  Update position:
  $\theta_c = \hat{P}_{ir}^{(t)} - \hat{P}_{b}^{(t)}$
  if (*Counter* modulo N_D) $= 0$ **then**
    $\theta_b^{(t)} = \theta_c^{(t)}$
  else $\{$(*Counter* modulo N_D) $\neq 0\}$
    $\theta_b^{(t)} = \theta_c^{(t)} + \zeta^{(t)}$
  end if
end while

control over actions in the environment based on the frequency of its observation on the *red* agent. The *blue* agent uses the received information about the *red* agent's intelligence, to generate deceptive movements through the adjustment of N_D and $\zeta^{(t)}$.

For the *red* agent, neuroevolution is used as the decision-making model to construct its decision behaviour. Neuroevolution refers to the use of evolutionary computation and neural networks in the autonomous production of behavioural robot controllers. In this case, the *red* agent is viewed as a robot, whereby neuroevolution is responsible to control its movements. The use of neuroevolution is justified according to Simon's concept of "bounded rationality" [75], which lies in the ability to learn and evolve to produce functional behaviours [75]. It approximates the possible limited computational capabilities of different humans interacting with the environment. According to this concept, behaviours are functional if they contribute to certain objectives of an individual. Evolution is about the persistence and survival of functional behaviours, while learning refers to the ability of an individual to change its decision making processes resultant from its actions and interactions with the environment. Neuroevolution is represented by a population of diversified

solutions which are associated with their own strengths and limitations.

Evolution and learning are complementary forms of an individual is ability to adapt. This is supported by Baldwin's argument that learning accelerates evolution because suboptimality can reproduce by acquiring during life necessary features for survival. Within an evolutionary perspective, learning allows individuals to adapt to changes in the environment that occur in the lifespan of an individual or across several generations. Then, the information extracted from the environment through learning is used by evolution to produce potential suboptimal individuals so that they can survive in the environment. Since neuroevolution is associated with both abilities to learn and evolve [69, 68, 58], it emerges as a suitable approach to determine the *red* agent's actions in the environment.

Even though evolutionary computation is usually applied to solve optimisation problems, we would like to distinguish our main idea from the evolutionary optimisation problem. The ultimate goal of our research is not to use evolutionary computation to optimise the games, but to emulate human behaviour as closely as possible.

4.1.3 Agent-based Model

Based on the strategies by either the *red* or *blue* agent, the agent-based model in our research can be represented by the following schematic diagram in Figure 4.3. For the *red* and *blue* agents, their decision models represent intelligence, processing the received information to produce and justify outputs. The inputs and outputs involved for both agents are listed in the Figure 4.3.

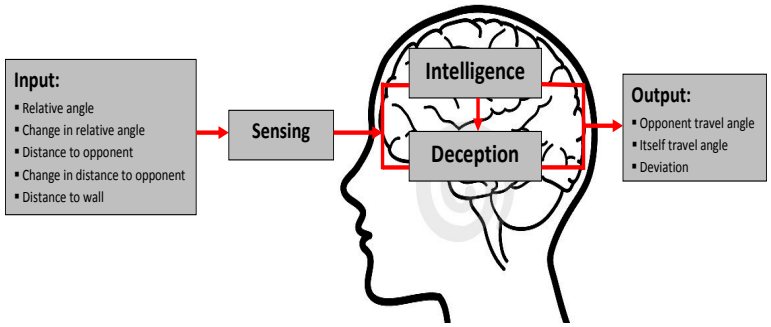


Figure 4.3: Schematic representation of the agent-based model.

4.1.4 Parameter Selection

N_I and $\hat{\alpha}^{(t)}$ represent the information that the *blue* agent has of the *red* agent, while N_D and $\zeta(t)$ represent the deception generated by the *blue* agent. The *blue* agent's strategies are influenced by the combinations of these 4 parameters.

Table 4.1 shows the possible combinations of N_I and $\alpha^{(t)}$, given that the deception effort from the *blue* agent is fixed. On the other hand, Table 4.2 shows the possible combinations of N_D and $\zeta(t)$, given that the received information about the *red* agent is fixed.

Table 4.1: The combinations of N_I and $\hat{\alpha}^{(t)}$ given that the deception effort from the *blue* agent is fixed.

Combination	N_I	$\hat{\alpha}^{(t)}$
1	1	0
2	1	$U(0, 20)$
3	10	0
4	10	$U(0, 20)$

Table 4.2: The combinations of N_D and $\zeta^{(t)}$ given that the information received about the *red* agent's intelligence is fixed.

Combination	N_D	$\zeta^{(t)}$
1	1	0
2	5	$U(-15°, 15°)$
3	5	$U(-30°, 30°)$
4	10	$U(-15°, 15°)$
5	10	$U(-30°, 30°)$

The value $N_I = 1$ means that the *blue* agent receives the information about the *red*

agent at each step. In contrast, $N_I = 10$ indicates that the *blue* agent receives information at every 10^{th} step.

A uniform distribution, $U(0, 20)$ is used to generate $\zeta^{(t)}$. The reason that 20 is used as the maximum values of the uniform distribution is based on the scenario that the *red* agent is almost caught by the *blue* agent , $d_{opp} \approx D_{min}$ ($D_{min} = 20$cm).

The first combination in Table 4.2 represents the scenario when the *blue* agent moves in the direction where it expects the *red* agent to be, $\hat{P}_{ir}(t)$. The other combinations represent scenarios in which the *blue* agent deviates from its expected trajectory (which leads to the capturing of the *red* agent) once after a number of steps. The deviation from the trajectory depends on the values of $\zeta^{(t)}$, which is based on a uniform distribution. The maximum and minimum deviations are selected as 15° and 30° respectively, so that the *blue* agent does not deviate too far away from the original trajectory which may end up with a failure to capture.

The four combinations in Table 4.1 and the five combinations in Table 4.2 produce a total of 20 combinations with varying perception and deception.

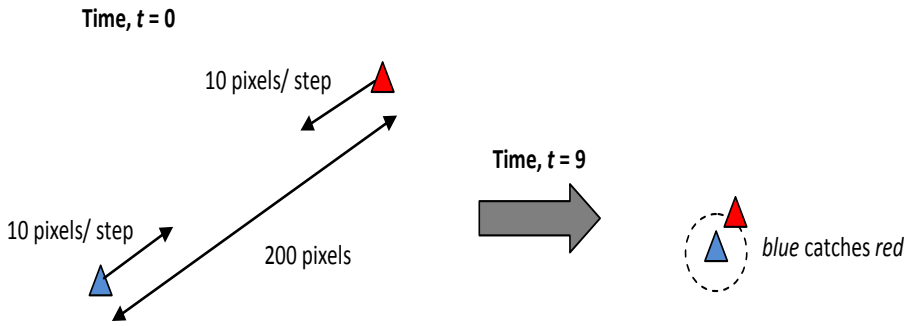


Figure 4.4: Selection of maximum values for N_I.

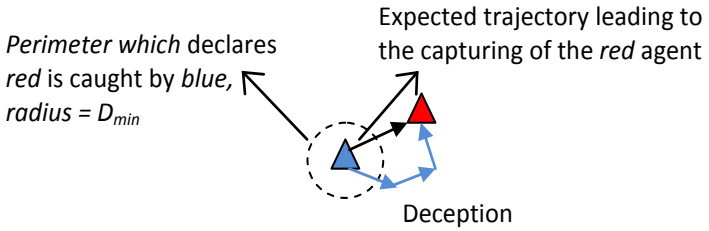


Figure 4.5: Selection of maximum values of the uniform distribution for the generation of $\zeta^{(t)}$.

4.2 Analysis Methodology

As we mentioned before, the main objective of our research is to compare similarities between machine and human red teaming. For a fair and valid comparison, a synthetic simulation environment is created to carry out red teaming, in which the *red* agent which can be either a human or a machine is required to maximise the optimality of decision-making. Here, optimality refers to decisions that maximise or minimise some explicit and measurable criterion conditional on certain environmental assumptions and a specified time horizon [26]. From this definition, we can see that Einhorn and Hogarth [26] emphasise the conditional nature of optimality. Therefore, the comparison of optimality between a human *red* and a computational *red* is compared with respect to the environmental constraints mentioned in Section 4.1.1. Based on this definition, we can observe also that optimality is influenced by the changes in decision-making and the changes are reflected in the actions taken. In other words, the changes in behaviours influence the outcomes. Owing to this, we are interested to compare the similarities between a human and computational red teaming in terms of behaviours and outcomes of behaviours. A behaviour refers to a

sequence of actions of the *red* agent in the environment. On the other hand, outcome of the red agent's behaviour can be either represented by the fitness values of a population of strategies in machine red teaming or the scores obtained by the human players in the proposed environment.

4.2.1 Analysis of Scores

4.2.1.1 Significance test

Behaviours of the *red* agent are determined by a population of strategies in an evolutionary process and a number of human players. In the demonstration of machine red teaming, the outcomes of the behaviours are represented by fitness values of the population of strategies at the end of the evolutionary process. On the other hand, the outcomes are represented by the scores obtained by the human players. Since the changes in behaviour will affect the outcomes, we are interested to know whether there is any significant difference of outcomes between the machine behaviour and human behaviour. Both machine and human *red* agents are exposed to two scenarios, as shown in Table 4.3, and their differences lie in the availability of perceived information about the *blue* agent. If the *red* agent operates with the absence of perceived information about the *blue* agent, such scenario is called known-unknown. On the other hand, another scenario is called known-known scenario given that the *red* agent acts with the presence of perceived information about the *blue* agent.

Table 4.3: The denotation of treatments.

Red agent	Scenario	Definition
Machine / Human	Known-unknown	A *red* agent with the absence of perceived information about the *blue* agent.
	Known-known	A *red* agent with the presence of perceived information about the *blue* agent.

The outcomes of both scenarios refer to the mean of the outcomes, $\overline{o}$, which are represented by the mean-fitness for the population and mean-scores for the human players. To test whether there is a significant difference or not for $\overline{o}$, a t-test is conducted between the

paired scenarios for the same configuration of N_I, $\alpha^{(t)}$, N_D and $\zeta^{(t)}$. Setting the significance test level of t-test as 0.05, the hypotheses of our study are set out as following:

- *Null hypothesis*(H_0): The sample mean of the difference in $\overline{o}$ between the paired treatments is equal;
- *Alternative hypothesis*(H_1):The sample mean of the difference in $\overline{o}$ between the paired treatments is unequal.

4.2.2 Analysis of Action Sequences

Given that both the agents have the same speed, the *red* agent is not able to escape by running faster. Therefore, the *red* agent needs to change its position in such a way to ensure it is not caught by the *blue* agent. As a result, there will be changes in the *red* agent's velocity as well as acceleration. With the measurements, the relative change in the movement is captured in terms of amplitude and time. For the calculation of velocity and acceleration as shown in Equation (4.1) and Equation (4.2), the locations visited by the *red* agent in each game are recorded.

$$v_r^{t_2} = \frac{\parallel \hat{P}_r^{t_2} - \hat{P}_r^{t_1} \parallel^2}{t_2 - t_1} \tag{4.1}$$

$$\dot{v}_r^{t_2} = \frac{v_r^{t_2} - v_r^{t_1}}{t_2 - t_1} \tag{4.2}$$

Each action sequence of the *red* agent is associated with different locations at different

time periods. To extract the information from a set of action sequences of different lengths, we need to have a set of action sequences of the same dimensions. An extracted information at time t can be seen as a single dimension. Owing to this, we need to determine the dimension for the action sequences. In other words, we need to determine the number of sub-sequences for each action sequence, N_S. To ensure there is no loss of generality, the number of steps for the shortest action sequence is selected as the desirable dimension N_S. By using a sliding window size, $w_i = len(A_i)/N_s$, the relevant attributes within the window are extracted as shown in Figure 4.6.

For ease of understanding, an action sequence of an agent i, A_i is represented by a single attribute over time t and the number of sub-sequences in the shortest sequence is denoted as N_s. Divide the total time T of a complete sequence of actions into N_s equal steps of times with window size, $w = T/N_s$. Therefore, $t = \{t_1, t_2, t_3, \ldots, t_{N_s}\}$ and the relevant attribute associated with the action is extracted at each point of time. Therefore, $A_i = \{a_i^1, a_i^2, a_i^3, \ldots, a_i^{N_s}\}$ represents a sequence of single attribute for agent i. Here, a_i^t is a sub-sequence of sequence A_i and can be considered as a single feature of a multi-dimensional data set. The proposed method can be easily extended to action sequences associated with multi-attributes with the relevant attributes being extracted at each point of time.

With the proposed method, a sequence of length $len(A_i)$ is mapped to a trail in feature space, consisting of N_s points, with a point for each possible offset of the sliding window. By doing so, we can measure the velocities at these points for different lengths of sequences and there will be N_S velocities. These are not velocities for the actual moves of an agent, but for the relative position of an agent from one window to another.

Having measured a velocity, we can also obtain the changes of velocities for these points. They are represented by $N_S - 1$ accelerations. As a result, a sequence of changes in locations over time is transformed into a sequence of velocities and accelerations instead for the *red*

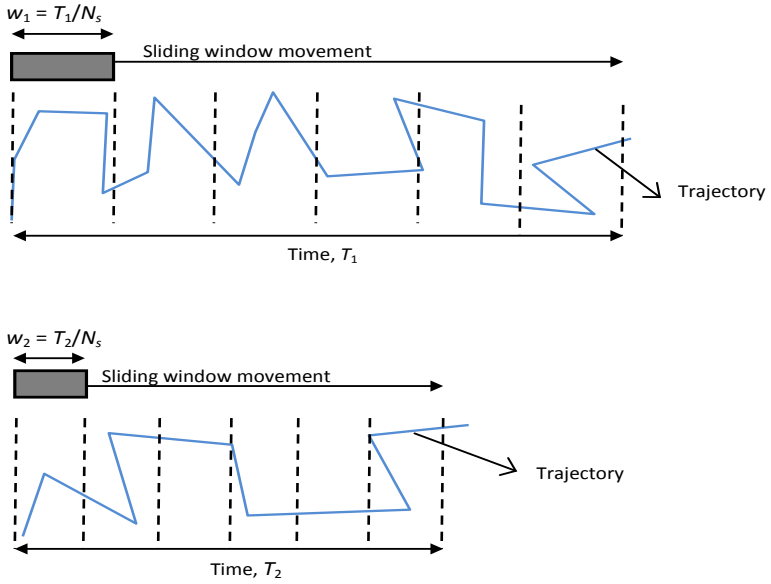


Figure 4.6: Attributes extraction based on sliding window.

agent, with new dimension, $\ell = 2N_S - 1$. The behaviour of agent i can be viewed as $B(A_i) = \{V_i, \dot{V}_i\} = \{v_i^1, \ldots v_i^{N_s}, \dot{v}_i^1, \ldots, \dot{v}_i^{N_s-1}\}$, where V has N_s sub-sequences, and $\dot{V}$ has $N_s - 1$ sub-sequences respectively.

The collection of action sequences is represented by a matrix, $\mathbf{M}$ of dimension $k \times \ell$, where k refers to the total number of action sequences and ℓ refers to the dimension of each action sequence. Clustering, specifically fuzzy c-mean ("FCM"), is used as the data mining technique to extract the underlying patterns that may exist in the action sequences. Clustering is a data mining technique that categorises a set of instances into several groups that share higher similarities within the clusters but lower similarities between the clusters, without the use of actual labels for the instances. The details of the clustering is shown in Section 4.2.4. With two sets of action sequences, are obtained from the experiments of machine and human red teaming respectively, clusterings are performed on the sets of action sequences. Given that both experiments use the same computer based decision environment and the same evaluation system, we can compare the similarity of action sequences between the human and machine based on their cluster centroid. For the clusters between the human

and machine that share high similarity, we are able to identify the possible characteristics that are associated with them. This is because in the human red teaming, the relevant characteristics of the human players have been collected through the questionnaire and the significance of differences in characteristics for the clusters can been carried out.

4.2.3 Behavioural Analysis between *Red* and *Blue*

4.2.3.1 Histogram Plot

To describe the relationship between the action sequences of the *red* and *blue* agents at time t, the relative angle between both agents denoted as ϑ^t, and its change, $\Delta\vartheta^t$ are measured.

For each time step, ϑ^t is calculated after the movements of both agents are updated simultaneously. Given that each game may be terminated at different time T, we will have a vector, $\hat{\vartheta}$ representing a sequence of ϑ^t, with $\hat{\vartheta} = \{\vartheta^1, \vartheta^2, \ldots, \vartheta^t, \ldots, \vartheta^T\}$. At the same time, we can obtain a sequence of changes in $\hat{\vartheta}$; this vector is denoted as $\Delta\hat{\vartheta} = \{\Delta\vartheta^2, \Delta\vartheta^3, \ldots, \Delta\vartheta^t, \ldots, \Delta\vartheta^T\}$. Here $\Delta\vartheta^t$ refer to $\vartheta^t - \vartheta^{t-1}$ as illustrated in Figure 4.7. To perform statistical analysis, each single sequence of ϑ and $\Delta\vartheta$ can be viewed as a sample. Each sample can be represented by a general measure that summarises the frequency distribution of observations in the sample. An approach that reflects tendency of where the observations are located is used: it is known as central tendency.

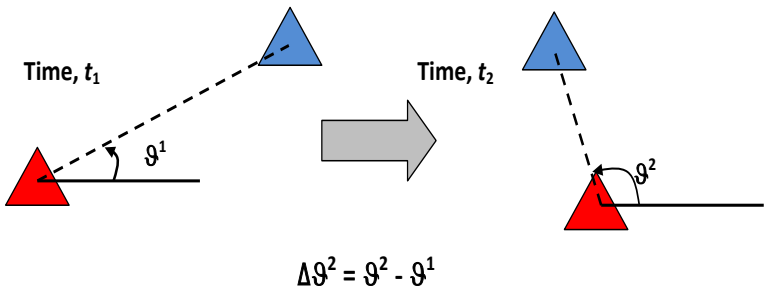


Figure 4.7: Calculation for the travel angle change, $\Delta\vartheta^t$.

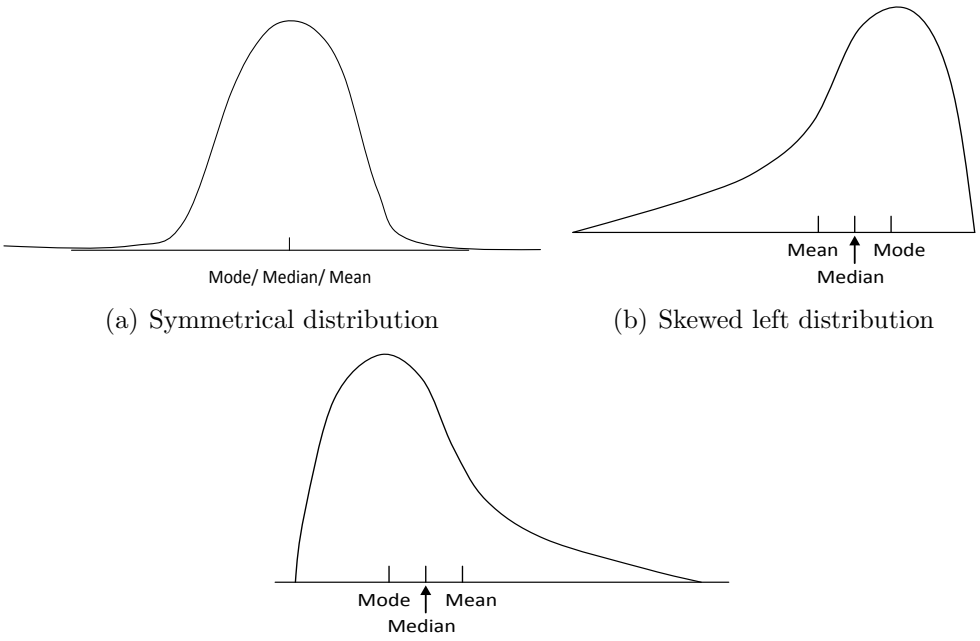
Central tendency for both ϑ and $\Delta\vartheta$ can be measured based on mean, mode and median. When a data distribution is symmetrical, the values for mean, mode and median will be similar, as shown in Figure 4.8(a). In contrast, the values for these measurements will be different in skewed distributions. Instead, the mean will tend to get dragged towards the tail of the distribution depending on whether it is skewed left or skewed right, as shown in Figure 4.8(b) to 4.8(c).

The collection of action sequences between the *red* and *blue* agents can be represented by the following vectors, where k refers to a individual sample for the measurements in human or machine red teaming, and $1 \leq k \leq K$. Here, K can either refer to the total samples in the human or machine red teaming based on the measurements of ϑ and $\Delta\vartheta$. Frequency histograms are used to display the distributions for the following measurements.

- Angle between the heading of both agents
  - $\hat{\vartheta} = \{\vartheta^1, \vartheta^2, \ldots, \vartheta^k, \ldots, \vartheta^K\}$
- Change of angle between the heading of both agents
  - $\Delta\hat{\vartheta} = \{\Delta\vartheta^2, \Delta\vartheta^3, \ldots, \Delta\vartheta^k, \ldots, \Delta\vartheta^K\}$

4.2.4 Clustering Analysis

Clustering analysis is a method used to categorise instances in a data set into several clusters in which the instances within the same cluster (group, subset, or category) share high similarity and the instances between different clusters share low similarity. In the clustering methods, the k-means (or also known as hard c-means) is a popular clustering method which restricts the constituting instances to have only two possible values of membership, they either belong to the cluster or not. Unlike the k-means method, FCM allows the instances



(a) Symmetrical distribution

(b) Skewed left distribution

(c) Skewed right distribution

Figure 4.8: A comparison of the mean, mode and median for distributions differing in shape.

to belong to several clusters simultaneously with different degrees of membership to each cluster [16]. Due to the complexity of actions produced by humans and machines in red teaming, we are hardly able to categorise an action into a single category. Therefore, we decided to use FCM in our research, because the method can capture the ambiguity in the data. This is a more realistic approach to address the concept of behaviour similarity. The FCM method is able to tolerate uncertain situations that often happen in human behaviours in the real world.

Let an action sequence be represented by $\hat{A}_i = \{a_{i1}, a_{i2}, \ldots, a_{i\ell}, \ldots, a_{iL}\}$, where the length of each sequence can be viewed as L-dimensional vector for an entity i, and $a_{i\ell}$ refers to a feature describing the action sequence, e.g., velocity. A collection of action sequences is represented by a $\mathbf{M}$ of $i \times \ell$ dimension, with $1 \leq i \leq I$ and $1 \leq \ell \leq L$. Here, i refers to the number of vectors $\hat{A}$ and ℓ refers to the number of dimensions in $\hat{A}$.

Clustering is performed by mapping the set of vectors $\mathbf{M} = \{\hat{A}_1, \hat{A}_2, \ldots, A_I\}$ with L-dimensional vectors in the vector space $\Re^I$ into a finite set of vectors $\mathbf{O} = \{\hat{B}_1, \hat{B}_2, \ldots, \hat{B}_J\}$, where $I \gg J$. Each L-dimensional vector $\hat{B}_j = \{b_{j1}, b_{j2}, \ldots, b_{j\ell}, \ldots, b_{jL}\}$, where $j = 1, 2, \ldots, J$, is called as a cluster centroid.

Fuzzy c-partition of $\mathbf{M}$ into centroid $\hat{B}_j$ where $j = 1, 2, \ldots, J$ refers to the number of clusters, $\mathbf{U} = [u_{\hat{A}_i}(\hat{B}_j)] = [u_{ij}]$ is called the degree of membership (membership grade), and $\tau \in [1, \infty)$ (fuzzy exponent) be a parameter weight for u_{ij} denoting the degree of fuzziness. $\mathbf{U}$ are subject to $\sum_{j=1}^{J} u_{ij} = 1$ respectively, where $0 \leq u_{ij} \leq 1$ for $\forall i = 1, 2, \ldots,$ and $\forall j = 1, 2, \ldots, J$, where $j \geq 2$ is the number of split regions (clusters) in FCM clustering. Membership grade is generated with random values between 0 and 1. Fuzzy partitioning is carried out through an iterative optimisation that minimises the fuzzy objective function $\Im$:

$$\Im_\tau(\mathbf{U}, \mathbf{O}) = \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{\ell=1}^{L} u_{ij}^{\tau} d_{ij\ell}^2 \tag{4.3}$$

subject to Equations (4.4) and (4.5) as shown below:

$$b_{j\ell} = \left[\sum_{i=1}^{I} u_{ij}^{\tau} \cdot a_{i\ell} \right] \left[\sum_{i=1}^{I} u_{ij}^{\tau} \right]^{-1} \tag{4.4}$$

where $i = 1, 2, \ldots, I$ and $j = 1, 2, \ldots, J$.

$$u_{ij} = \left(\sum_{\ell=1}^{L} d_{ij\ell}^2 \right)^{\frac{1}{1-\tau}} \left[\sum_{j=1}^{J} (\sum_{\ell=1}^{L} d_{ij\ell}^2)^{\frac{1}{1-\tau}} \right]^{-1} \tag{4.5}$$

where $d_{ij\ell}^2 = d^2(a_{i\ell}, b_{j\ell}) = \| a_{i\ell} - b_{j\ell} \|^2$.

4.2.4.1 Cluster Validity Analysis

Two fundamental issues that need to be addressed in clustering analysis are to determine the appropriate cluster size (the number of clusters) and also the goodness of the formed clusters. As an emerged solution, cluster validity analysis is used to determine an appropriate cluster size that produces meaningful output. Cluster validity analysis is the assessment of a clustering process's output. It usually involves a specific criterion of optimality. There are three main categories of validity indices, i.e., external assessment, internal assessment and relative test. An external assessment compares the discovered structure with respect to an external criterion that supervises the quantification of the degree of compatibility between the discovered clusters and the actual ones. For example, the use of data labels to evaluate clustering performance is an example of an external assessment. On the other hand, an internal assessment determines intrinsically whether the discovered structure is appropriate for the data without any prior information. Lastly, a relative test compares two structures and measures the relative indices between them. In our work, cluster validity based on internal assessment is used because there is no label involved in our data. To perform the analysis, 4 well-known validity indices, namely Silhoutte index [73], Davies-Bouldin index [24], Calinski-Harabasz index [17] and Dunn index [25], are selected for the assessment because they evaluate the discovered structure according to their appropriateness for the data such as cluster separation and compactness. Since there is no so called "gold standard" for selecting cluster validity indices, the four indices form an ensemble to determine an appropriate cluster size.

Chapter 5

Human Red Teaming

5.1 Introduction

The advancement in computer technology has allowed computational models to be used to mimic or reproduce human behaviours in decision making regardless of whether they are rational or irrational [4]. We can observe that the focus of many models is on reproducing human behaviour in decision making. However, a human's actual decision behaviour can be hard to approximate by the global rational behaviours exhibited in computational predictive models of decision tasks [75].

But why is the reproduction of human behaviours in decision-making important? It might be true that human behaviours do not approximate the global rationality owing to their psychological limits, but it is undeniable that human's computational capacities are able to handle various decision tasks. Besides that, there is a lot of issues that still can be learnt about the possible mechanisms from an examination of the schemes of approximation that are actually employed by humans in specific situations.

Owing to the above, computational red teaming is incomplete without the involvement

of humans in the process. For example, human reflection is a mechanism that offers a great deal to learn about the schemes of approximation that are actually employed by humans [4].

There are three main components that are impacted by human reflection, i.e., measures of performance, intelligence, and decision-making and planning. In a red teaming environment, intelligence refers to the perception that the agents have about each other, themselves, and their environment. These information are collected through the agents' sensors. The decision-making and planning component refers to computational capabilities to produce a decision on how should an agent act in the environment. In other words, the component is reflected in the agents' behaviours. Lastly, measures of performance refer to the metrics used to measure achievements of agents' objectives.

Based on these three components, we can observe that an agent's perception of its environment including its surrounding and the other agent's influence on its behaviour affect the outcome of an agent's objective. Therefore, perception can be seen seen as the cause for changes in behaviours and affects the achievement of agent's goal. We hypothesise that the differences in human behaviours are influenced by the effect of perception. To investigate this, information display about the opponent is used as a way to vary the level's of human's perception.

5.2 Experimental Design for Human *Red* Agent

In this experiment, the role of the *red* agent is taken by a human player. We are interested to know how human players acting as the *red* agents behave facing a deceptive *blue* agent. Unlike a machine which acts like an "economic man", normal humans may be influenced by small changes in the same task.

In this experiment, we are interested to know whether human behaviour would be influenced by the access that humans have on the *blue* agent's information or not in the same environment. We hypothesise that the differences in human behaviours are influenced by the perceived information about *blue*, leading to differences in the outcomes as well. Therefore, two scenarios are simulated to allows us to investigate the hypothesis, which can be called as the known-unknown and known-known scenarios.

The game environments for both scenarios can be shown in Figures 5.1 and 5.2. In the known-unknown scenario, as shown in Figure 5.1, the human player has no access on the *blue* agent's perception. In contrast, an extra yellow box can be seen in the Figure 5.2; it represents what the *blue* agent thought was the position of the *red* agent. This means the human player will know the level of noise, $\hat{\alpha}^{(t)}$ in information received by the *blue* agent through the information display on the computer screen. In contrast, the level of noise, $\hat{\alpha}^{(t)}$ in information is unknown for the *red* agent in the known-unknown scenario, in which the access on the *blue* agent's perception is absent. For each scenario, 20 different configurations as mentioned in Section 4.1.4 are used as *blue*'s strategies, and the game is repeated twice continuously for the same configuration.

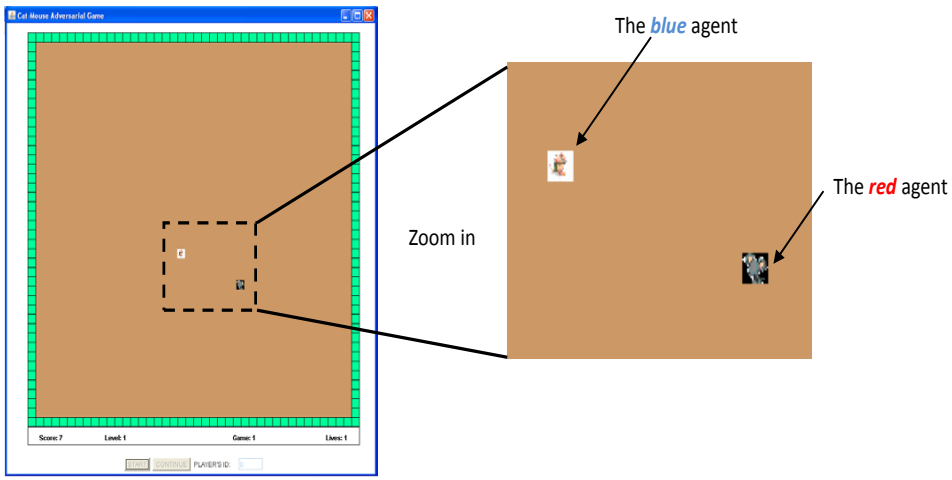


Figure 5.1: Illustration of the known-unknown scenario.

In this experiment, the human player is placed in a computer based decision environ-

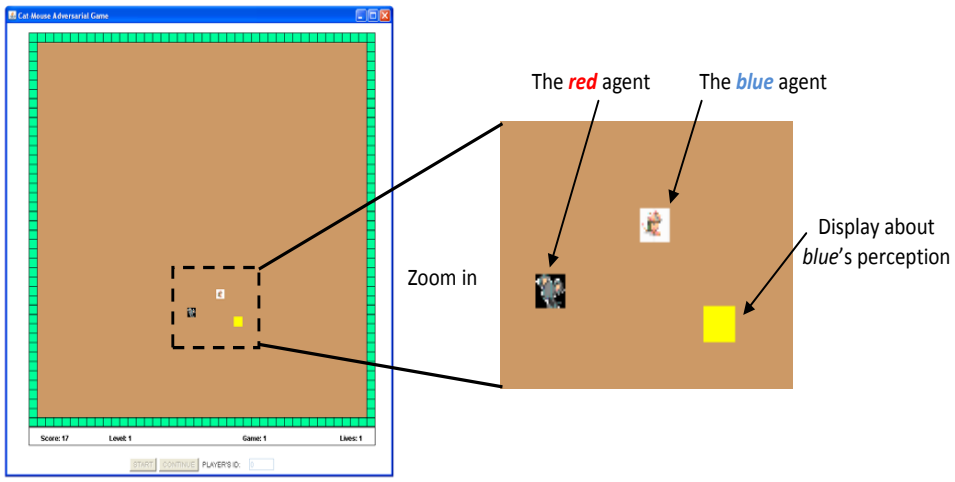


Figure 5.2: Illustration of the known-known scenario.

ment, in which the human player needs to determine the travel angle of the *red* agent, θ_{hr}^{t}. This type of approach is appropriate because it ensures that all individuals will receive the same information from computer software, and thus, eliminate the descriptive and subjective effects. For example, in an interview, the same description of a decision problem by a researcher may be framed and therefore interpreted differently by the subjects. In contrast, the use of computer based decision environment allows behavioural analysis to be performed based on the actions taken by the subjects only; as variations in problem framing are eliminated.

The use of computer software also removes the artefactual problems of an interaction between the researcher and the subject, and reduces the likelihood of descriptive effects from peer group pressure, public performance and perception of others relative to self. At the same time, the human players are asked to fill in a questionnaire so that we are able to collect some information about their personality characteristics.

None of us is completely innocent of acquaintance with the gross characteristics of human choice in affecting human behaviours. An important factor that influences human behaviour in a given task is individual-difference which is related to perception capacities,

processing capacities, prior knowledge, expertise and also personality characteristics of the humans. Besides that, the order in which configurations of the game are played can actually influence the human players' behaviour or cause false response, e.g. fatigue factor or other factors that may change the behaviour of the subject. These types of factors are not of real interest in the experiment and are known as nuisance factors. To reduce the impact of the order of configuration and nuisance factors, a counterbalanced design is used in our experiments. Details of the counterbalanced design are explained in the following section.

For each game, the initial locations for both agents are randomly generated, where $d_{opp} = 200$cm. The goal of the *red* agent in each game is the same: to extend its survival period as long as possible. The longer the *red* agent can survive, the more likely it will win the game.

The scoring system used in this experiment is simple: the *red* agent will score a point for each step it takes as long as the game is not terminated. However, each configuration is repeated twice in HRT. The longer that the *red* agent can survive from being caught by the *blue* agent, the more scores will be collected until the game is terminated. For each time step, the human player controlling the movement of the *red* agent needs to determine the travel angle of the *red* agent, $\theta_{hr}^{(t)}$, through mouse clicks. Once this is done, the locations for both the *red* and *blue* agents are updated simultaneously. The same scoring system will be used in the following chapter.

The same procedures are repeated for each scenario (known-unknown and known-known) perceived information (known-known scenario).

To investigate our hypothesis, there are two important elements to be focused on, i.e., the outcomes of the games and human behaviour. In this experiment, the outcome refers to the collected scores while human behaviours are described by the action sequences in the environment. To perform analysis on the action sequences, the locations visited by the

red agents in each game are recorded.

5.2.1 Counterbalanced Measures Design

The basic concept of the counterbalanced measures design is to reduce the impact of confounding due to the order of treatments or nuisance factors. For example, suppose the human players are asked to play continuously a series of games associated with different conditions, always in the same order. Due to fatigue factor, we may observe that most of the players achieve high scores for the earlier games but lower scores for the later games. This means that the scores may actually be affected by the order of play rather than the conditions of the games, and thus, we may make the wrong conclusion. One way to address this problem is to counterbalance the order of presentations. In our case, the human players would be playing the games in different orders, in such a way that each game is played in each sequential position an equal number of times.

To illustrate the counterbalanced measures design is explained based on two possible conditions of games, A, and B. Assume that we have 2 players; one of the players is treated with condition A followed by condition B, and the other player is treated with condition B followed by condition A. The illustration of counterbalanced measures design based on two conditions are shown in Figure 5.3. When there are three conditions involved, i.e., A, B and C, the same procedures are repeated by having at least 6 players, with the orders carried out as shown in Figure 5.4.

The main drawback with the complete counterbalanced measures design is the permutations for an experiment with multiple conditions multiply quickly, and the size of the experiments become too large to handle. For example, four possible conditions will require 4! ($4\times3\times2\times1$) orders of treatments. In short, the number of treatments required for n conditions is $n!$. Since our experiments involves 20 configurations, the number of treatments

required to perform the complete counterbalanced measures design will be 20! at least even each configuration of game is played by a subject. Therefore, a Latin-squared design [74] is used to keep the experiment to a reasonable size because the design will require $2n$ of treatments only.

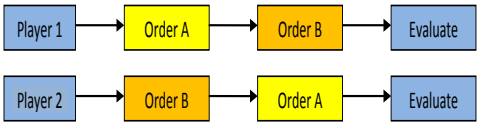


Figure 5.3: Counterbalanced measures design for 2 conditions.

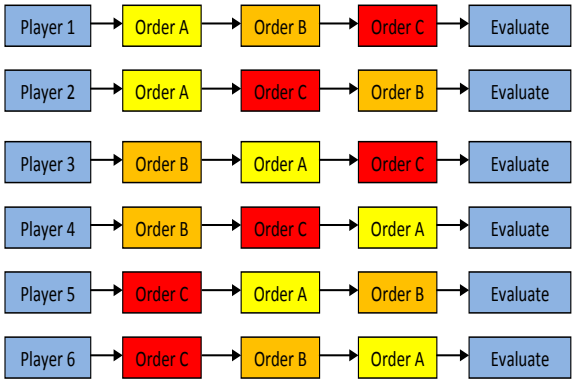


Figure 5.4: Counterbalanced measures design for 3 conditions.

Table 5.1 illustrates the method for four conditions given that each condition is treated with a subject. The elements in the first rows of the two squares from left to right are:

- 1st Square - $\{0, n-1, 1, n-2, 2, n-3, \ldots, n/2+1, n/2-1, n/2\}$
- 2nd Square - $\{n-1, 0, n-2, 1, n-3, 2, \ldots, n/2-2, n/2, n/2-1\}$

For the elements in the subsequent rows, the value of one is added to the elements in previous row. If the value of element exceeds $n-1$, the values of zero will be returned.

Table 5.1: Latin squares for 4 conditions.

Individual	1st Square Order				Individual	2nd Square Order			
	1	2	3	4		1	2	3	4
Player 1	0	3	1	2	Player 5	3	0	2	1
Player 2	1	0	2	3	Player 6	0	1	3	2
Player 3	2	1	3	0	Player 7	1	2	0	3
Player 4	3	2	0	1	Player 8	2	3	1	0

5.3 Result and Analysis

This section presents the results and analysis on human behaviours. It is divided into two subsections, i.e., action distribution (Section 5.3.1) and action similarity (Section 5.3.2). For the figures related to trajectory, the distributions of ϑ and $\Delta\vartheta$, they are plotted in such a way that:

- The sequence of information in the same row from left to right: frequent-accurate, frequent-noisy, infrequent-accurate, infrequent-noisy;
- The sequence of deception in the same column from top to bottom: none, infrequent-low, infrequent-high, frequent-low, frequent-high.

5.3.1 Action Distribution

There were 34 human players participating in the experiment. Each player needed to face *blue* associated with 20 different combinations of perception and deception, and each combination of a game was repeated twice. There were 1360 (34×20×2) different sets of action sequences generated in the experiment. Due to limited space, we only show the trajectories between *blue* and *red* in a game for a single player in the known-unknown scenario (Figure 5.5) and the known-known scenario (Figure 5.6). The markers of • and ×

in the trajectory plots represent the initial and end positions for both agents.

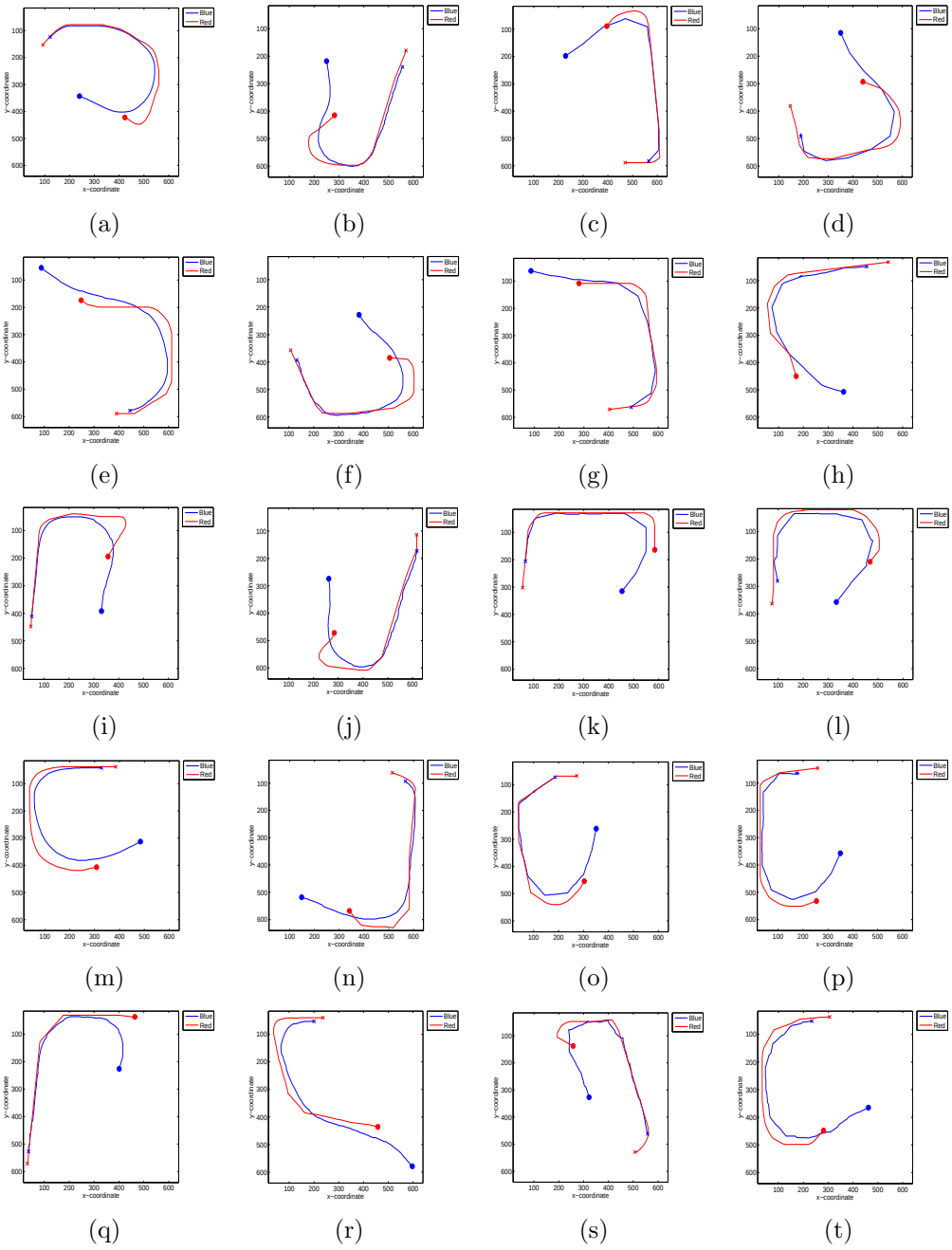


Figure 5.5: The trajectories between *blue* and *red* in the known-unknown scenario.

The measurements based on ϑ and $\Delta\vartheta$ are extracted from the trajectories between the *red* and *blue* agents for each configuration. Then, probability density functions (pdfs) for these measurements in the known-unknown and known-known scenarios are approximated

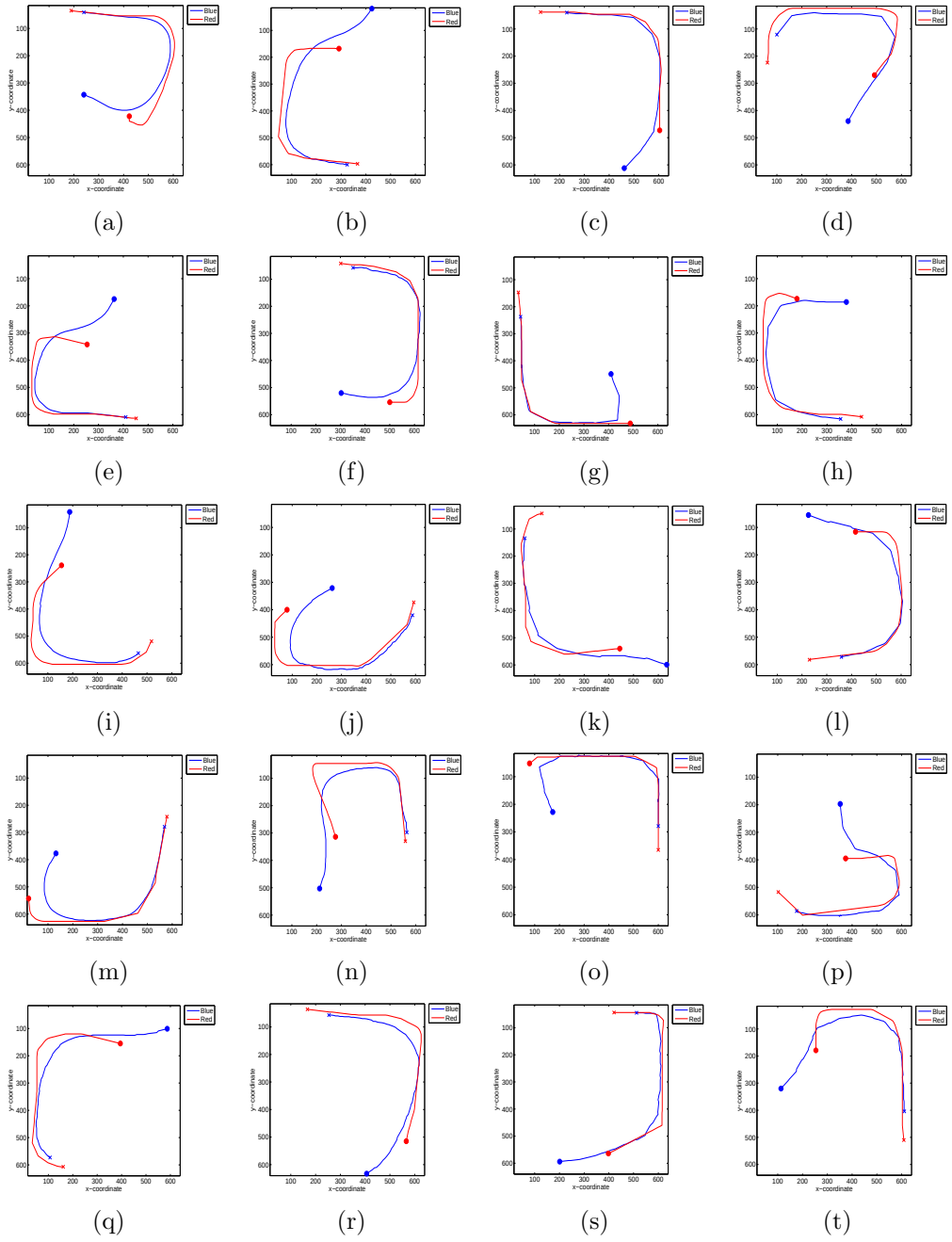


Figure 5.6: The trajectories between *blue* and *red* in the known-known scenario.

and shown in Figures 5.7 to 5.10.

The pdf plots of ϑ for both scenarios, shown in Figures 5.7 and 5.8, show that the distributions are multi-modal. Unlike the machine behaviours in the following chapter,

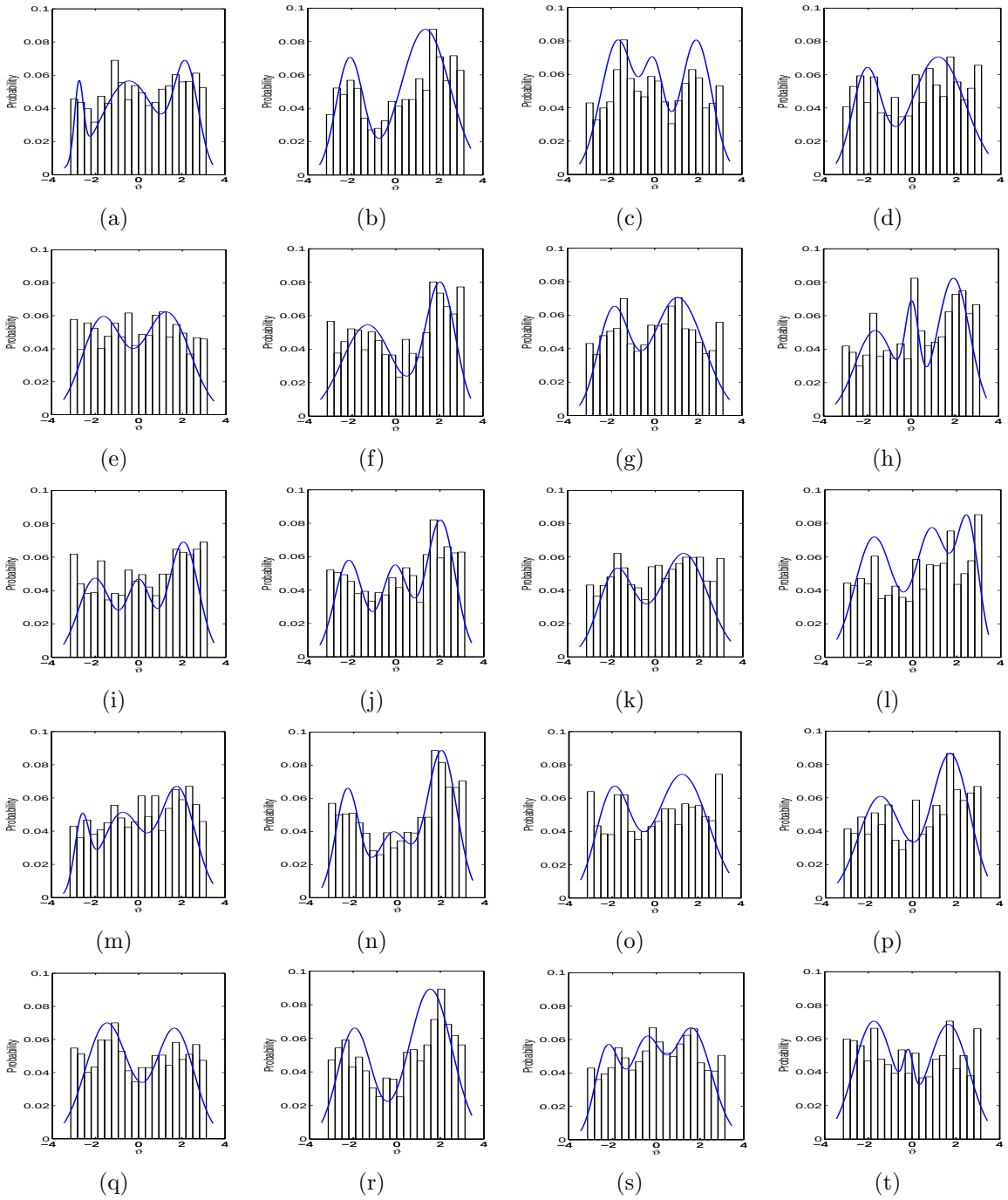


Figure 5.7: The plots of ϑ for various combinations of information and deception in known-unknown scenario for HRT.

there is lack of pattern consistency in the pdf plots as the information changes from being frequent and accurate to infrequent and noisy. Again, there is no clear pattern found as

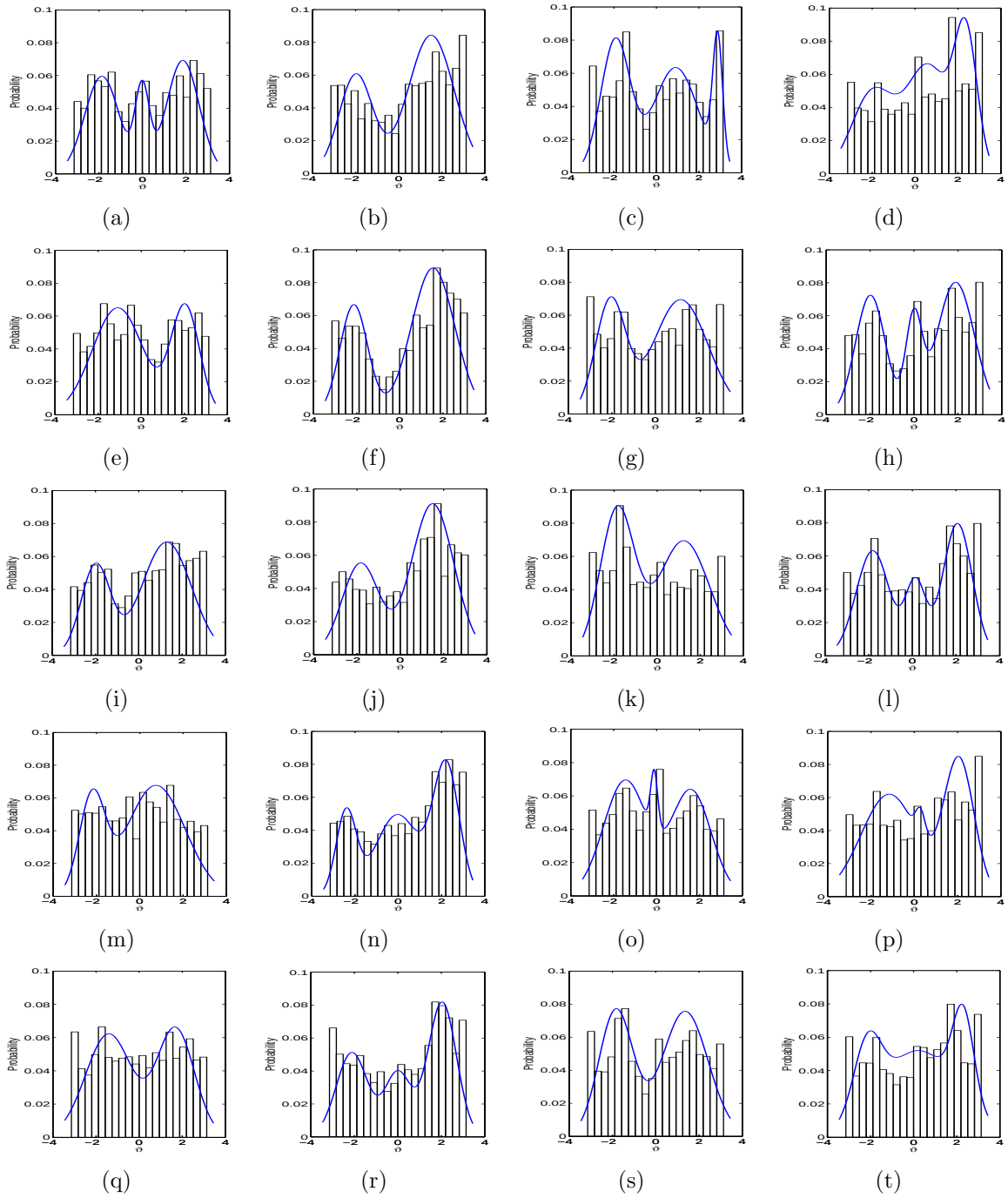


Figure 5.8: The plots of ϑ for various combinations of information and deception in known-known scenario for HRT.

the level of deception changes from being not deceptive to highly deceptive. Besides that, most of the pdf plots in both scenarios do not share similarity for the same configuration.

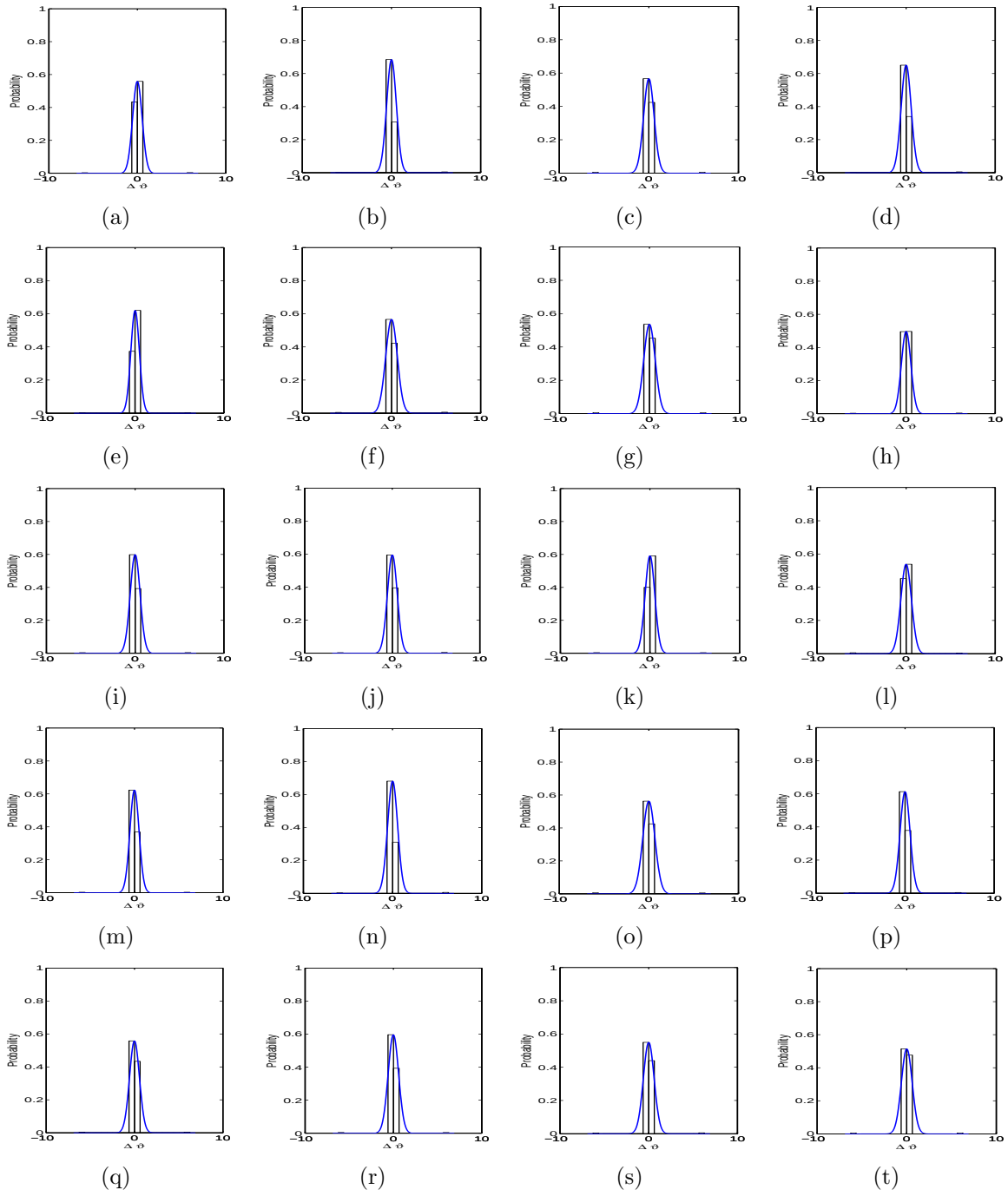


Figure 5.9: The plots of $\Delta\vartheta$ for various combinations of information and deception in known-unknown scenario for HRT.

On the other hand, Figures 5.9 and 5.10 show that the distributions of $\Delta\vartheta$ are unimodal in both scenarios, where the values of $\Delta\vartheta$ are highly concentrated around zero. This means

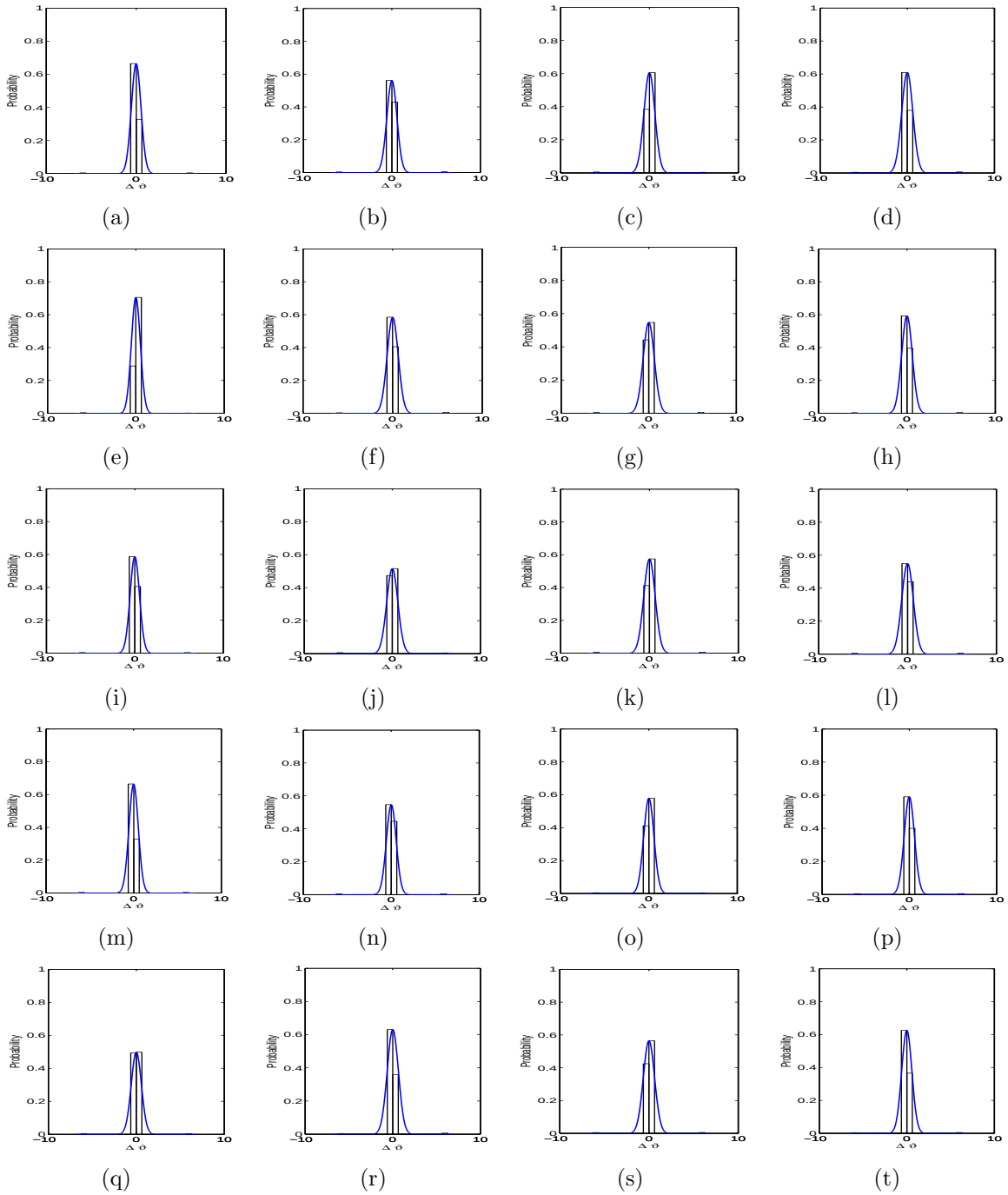


Figure 5.10: The plots of $\Delta\vartheta$ for various combinations of information and deception in known-known scenario for HRT.

the human players hardly or slightly make changes in their movements regardless of having access to *blue*'s perception or not.

The findings in both scenarios suggest that there is a lack of obvious patterns in the human behaviours across different combinations of information and deception. In the known-unknown scenario, human players are unlikely to be affected by information and deception, and thus, their actions are independent of the information and strategy of *blue*. In other words, the distraction in *blue*'s actions are not clearly perceived by the human players. As a result, the factors of information and deception are not taken into account in determining their movements. In contrast, access to the *blue*'s perception is provided for in the known-known scenario and the access can be viewed as a task-relevant stimulus. However, the findings obtained in the known-known scenario also indicate that there are no obvious patterns observed in the involved comparisons. This leads us to believe that human behaviours in the scenarios were not affected by information and deception even though a task-relevant stimulus is provided. Given that the scores of the human players in both scenarios are high, as shown in Table 5.2, possible reasons for this are the engagement of human players in the games and that they exhibit goal-directed behaviour.

Table 5.2: Scores and RTs (in ms) for the human players in known-unknown and known-known scenarios.

N_I	$\hat{\alpha}^{(t)}$	N_D	$\zeta^{(t)}$	Score($Mean \pm stdev$)		RT($Mean \pm stdev$)	
				Known-unknown	Known-known	Known-unknown	Known-known
1	0	1	0	82.53±27.75	89.91±18.65	406.5±5.5	407.6±7.4
		5	$U(-15°, 15°)$	83.04±25.82	88.88±23.36	407.2±5.6	407.0±6.0
		5	$U(-30°, 30°)$	88.81±20.92	93.99±17.71	407.3±6.4	407.9±6.3
		10	$U(-15°, 15°)$	89.31±21.15	92.31±18.41	408.1±5.9	407.1±6.3
		10	$U(-30°, 30°)$	90.02±19.33	91.79±17.69	407.6±6.0	408.4±6.0
1	$U(0, 20)$	1	0	90.91±18.26	92.32±16.86	409.5±9.5	406.7±5.9
		5	$U(-15°, 15°)$	89.74±21.64	93.04±18.49	407.6±6.0	406.7±6.1
		5	$U(-30°, 30°)$	88.37±21.55	90.22±19.83	407.5±5.6	406.9±6.1
		10	$U(-15°, 15°)$	87.15±24.91	93.84±14.63	407.2±6.4	407.2±5.4
		10	$U(-30°, 30°)$	86.60±26.22	90.19±19.51	406.8±7.1	406.5±6.3
10	0	1	0	95.60±17.69	99.62±3.13	408.5±6.0	408.2±6.3
		5	$U(-15°, 15°)$	97.59±11.72	95.85± 17.00	408.1±6.1	406.9±7.2
		5	$U(-30°, 30°)$	97.63±8.70	97.93±11.91	408.2±6.2	407.1±5.8
		10	$U(-15°, 15°)$	95.87±17.12	97.74±11.78	408.1±6.3	407.9±7.3
		10	$U(-30°, 30°)$	95.60±15.71	95.65±15.76	408.0±6.1	407.5±5.9
10	$U(0, 20)$	1	0	95.84±16.01	96.74±13.52	408.8±6.6	407.6±5.0
		5	$U(-15°, 15°)$	97.77±11.11	95.85±16.39	408.7±6.2	407.7±5.4
		5	$U(-30°, 30°)$	99.07±5.52	97.49±11.81	408.7±5.8	408.2±5.9
		10	$U(-15°, 15°)$	97.16±13.89	96.03±15.19	408.8±6.5	407.5±6.0
		10	$U(-30°, 30°)$	95.68±16.82	98.40±9.58	408.4±6.7	408.2±5.9

Goal-directed behaviours require vital attention on goal-relevant stimuli while ignoring the potential interferences from distractions that are irrelevant to the task (task-irrelevant stimuli) [45]. Selective attention is the mechanism that affects the goal-directed behaviour. Most research on load theory of selective attention suggests that perceptual load plays an important role in determining whether task-irrelevant stimuli are perceived or not [42, 44, 45, 43]. Instead of focusing on task-irrelevant stimuli, the findings in our work suggest that perceptual load also affects the perception of goal-relevant stimuli. The perception load in the experiment is represented by the visualisation of the movements for the *red* and *blue* agents. Besides that, the perceptual load is considered high because the reaction between the *blue* and *red* agents controlled by machine and human players respectively happens very fast.

To demonstrate that the perceptual load in both known-unknown and known-known scenarios are equally high, a t-test is carried out with 0.05 significance level to evaluate the null hypothesis that the reaction times ("RTs") come from both treatments with equal means, against the alternative that the means are unequal for the same combination as shown in Table 5.2. The significance test in Table 5.3 shows that all of the null hypotheses are not rejected. Therefore, we do not have enough evidence to claim that the RTs for both scenarios are different.

Besides RTs, we are interested to know whether there is any significant differences of scores between the presence and absence of perceived information about the *blue* agent for different combinations of information and configurations. To investigate the effect of the presence and absence of the perceived information, a t-test is carried out with 0.05 significance level to test the null hypothesis that the samples come from both scenarios with equal means, against the alternative that the means are unequal. Table 5.3 shows the result of t-test on the scores between the known-unknown and known-known scenarios based on the combinations of information and perception. Based on the significance test,

all of the null hypotheses are not rejected. This means there is not enough evidence to claim that the scores for both scenarios are significantly different.

Table 5.3: t-test for RTs and scores between the known-unknown and known-known scenarios.

N_I	$\hat{\alpha}^{(t)}$	N_D	$\zeta^{(t)}$	*RT*		Score	
				Reject H_0	p-value	Reject H_0	p-value
1	0	1	0	No	1.0	No	0.0732
		5	$U(-15°, 15°)$	No	1.0	No	0.1723
		5	$U(-30°, 30°)$	No	1.0	No	0.1246
		10	$U(-15°, 15°)$	No	1.0	No	0.3828
		10	$U(-30°, 30°)$	No	1.0	No	0.5792
1	$U(0, 20)$	1	0	No	1.0	No	0.6427
		5	$U(-15°, 15°)$	No	1.0	No	0.3430
		5	$U(-30°, 30°)$	No	1.0	No	0.6054
		10	$U(-15°, 15°)$	No	1.0	No	0.0606
		10	$U(-30°, 30°)$	No	1.0	No	0.3706
10	0	1	0	No	1.0	No	0.0716
		5	$U(-15°, 15°)$	No	1.0	No	0.4928
		5	$U(-30°, 30°)$	No	1.0	No	0.8706
		10	$U(-15°, 15°)$	No	1.0	No	0.4633
		10	$U(-30°, 30°)$	No	1.0	No	0.9871
10	$U(0, 20)$	1	0	No	1.0	No	0.7266
		5	$U(-15°, 15°)$	No	1.0	No	0.4310
		5	$U(-30°, 30°)$	No	1.0	No	0.3212
		10	$U(-15°, 15°)$	No	1.0	No	0.6533
		10	$U(-30°, 30°)$	No	1.0	No	0.2526

As shown in Table 5.2, the scores are high and the RTs are very fast in both scenarios. Given that there is not enough evidence to support any significant differences in terms of RTs and scores in both scenarios, we can assume that both scenarios require equal focusing attention from humans. In other words, both scenarios can be considered to have equally high perceptual loads.

The high perceptual load that engages full capacity in relevant processing leaves the human players no spare capacity for the perception of task-relevant stimuli. Therefore, simply informing the human players about the task-relevant stimuli is not sufficient for encouraging their use. As a result, there are no significant patterns that can be observed in the comparison of human behaviours in the known-unknown and known-known scenarios.

In order to support that high perceptual load does affect the attention on task-relevant stimuli, analysis is conducted on human behaviours in terms of $V_{Rrel\,B}$ and $\Delta V_{Rrel\,B}$, ex-

plained in detail in section 5.3.2.

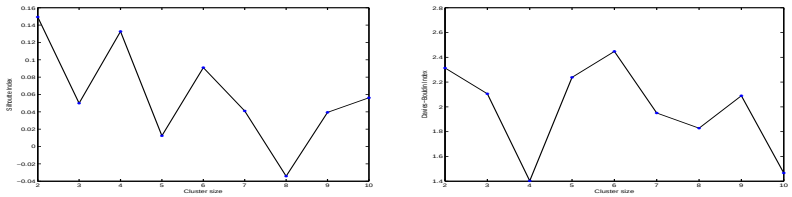
5.3.2 Action Similarity

The action sequences of the human players are described by $V_{Rrel\,B}$ and $\Delta V_{Rrel\,B}$. Since human behaviours are analysed based on different measurements in this section, it would be interesting to know whether the analysis leads to similar findings to those in section 5.3.1 or not.

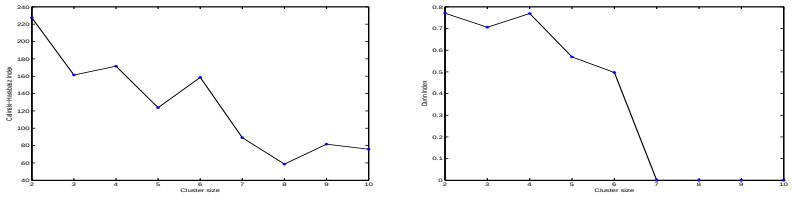
From the collections of action sequences in the known-unknown and known-known scenarios, we are interested to know if there are similarities among the action sequences. To achieve this purpose, clustering is used on the collections of action sequences. Similar to previous chapter, an ensemble of cluster validity, consisting of Silhoutte, Davies-Bouldin, Calinski-Harabasz and Dunn indices, is used to determine the appropriate cluster size based on the range [2 10].

The results of cluster validity for the known-unknown and known-known scenarios are shown in Figure 5.11 and 5.12 respectively. Based on Figure 5.11, the results show a slight inconsistency in determining the cluster size. To solve this problem, we rank the involved cluster size for each index, with a lower rank being given to the cluster size associated with better indices and vice versa. After that, the appropriate cluster size is determined based on the average rank, as shown in Table 5.4. The results in Table 5.4 suggest that the appropriate cluster size to be used in the known-unknown scenario is 4 with its average ranking value of 1.50. On the other hand, Figure 5.12 shows consistent result on the cluster size to be used in the known-known scenario of 2. As a result, cluster sizes of 4 and 2 are used to perform clustering in both scenarios respectively.

Based on the suggested cluster sizes, clustering is carried out. The cluster centroid is shown in Table 5.5 for both treatments. As we mentioned before, the cluster centroid

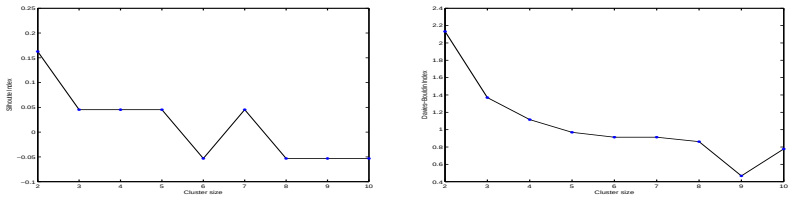


(a) Silhoutte Index (b) Davies-Bouldin Index

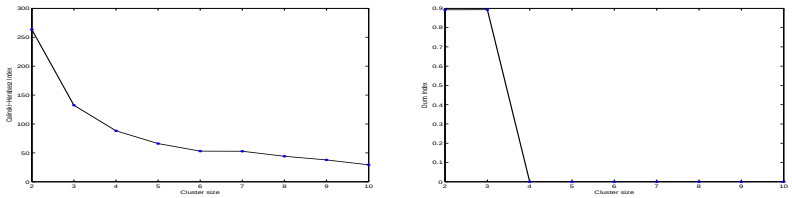


(c) Calinski-Harabasz Index (d) Dunn Index

Figure 5.11: Cluster validity indices for the action sequences associated with the known-unknown scenario for HRT.



(a) Silhoutte Index (b) Davies-Bouldin Index



(c) Calinski-Harabasz Index (d) Dunn Index

Figure 5.12: Cluster validity indices for the action sequences associated with the known-known scenario for HRT.

Table 5.4: Ranking of cluster size based on cluster validity indices in the known-unknown scenario.

Index	Cluster size								
	2	3	4	5	6	7	8	9	10
Silhoutte	1	7	2	8	3	5	9	6	4
Davis-Bouldin	8	6	1	7	9	4	3	5	2
Calinski-Harabasz	1	4	2	5	3	6	9	7	8
Dunn	2	3	1	4	5	6	7	8	9
Average	3.00	5.00	**1.50**	6.00	5.00	5.25	7.00	6.50	5.75

describes the strategies based on $V_{R\,rel\,B}$ and $\Delta V_{R\,rel\,B}$ and the centroid can be viewed as a general representation of a group of strategies. For ease of interpretation, the $V_{R\,rel\,B}$ and $\Delta V_{R\,rel\,B}$ against different windows are plotted for these strategies as shown in Figure 5.13.

Figure 5.13 shows that there is only a single type of strategy in the known-unknown scenario because all of the strategies overlap with each other. A similar observation is found in the known-known scenario even though there are 2 types of strategies in it. Besides that, the strategies used in the known-unknown and the known-known scenarios are very similar to each other in $V_{R\,rel\,B}$. In fact, the only difference between them is on $\Delta V_{R\,rel\,B}$ which is very marginal.

As we mentioned before, we believe that high perceptual load will influence the perception of task-relevant stimuli. The findings in the analysis of human behaviours $V_{R\,rel\,B}$ and $\Delta V_{R\,rel\,B}$ show that there are high similarities in the strategies regardless of the absence or presence of the task-relevant stimuli. High perceptual load requiring fast reaction times to achieve high scores in the game prevent the human players to pay attention to the task-relevant stimuli, exhaust their capacity in processing the available task-relevant stimuli, and thus, lead them to use similar strategies in both situations. Despite this, human players are still able to select the appropriate correct actions. The ability to take correct actions, in conditions whereby the human players pay more attention on scenarios of avoiding *blue* rather than on the extra task-relevant stimuli (represented by the yellow square box), suggests that the goal-directed behaviour is less influenced by the provided task-relevant stimuli. As a result, the strategies used by the human players, regardless of having the extra task-relevant stimuli or not, are quite similar as shown in Figure 5.13.

Table 5.5: Cluster centroid that represents the actions of the *red* relative to *blue* in the known-unknown and known-known scenarios.

Feature	Window	Centroid					
		Known-unknown				Known-known	
		1	2	3	4	1	2
$V_{R\,rel\,B}$	ω_1	7.6416	7.6410	7.6410	7.6414	7.5535	7.5765
	ω_2	6.6697	6.6690	6.6691	6.6695	6.4407	6.4587
	ω_3	5.8484	5.8479	5.8481	5.8482	5.6480	5.6496
	ω_4	5.9684	5.9682	5.9683	5.9683	5.6573	5.6462
	ω_5	5.6058	5.6057	5.6058	5.6057	5.4829	5.4686
	ω_6	5.1830	5.1827	5.1828	5.1829	5.0092	4.9999
	ω_7	4.6648	4.6644	4.6645	4.6646	4.3811	4.3739
	ω_8	4.2199	4.2192	4.2194	4.2196	3.8222	3.8182
	ω_9	3.7684	3.7677	3.7679	3.7681	3.4488	3.4450
	ω_{10}	3.3445	3.3439	3.3441	3.3443	3.2066	3.2031
$\Delta V_{R\,rel\,B}$	ω_2	-0.0668	-0.0668	-0.0668	-0.0668	-0.1056	-0.1061
	ω_3	-0.0871	-0.0871	-0.0871	-0.0871	-0.0795	-0.0813
	ω_4	-0.0060	-0.0060	-0.0060	-0.0060	0.0001	-0.0013
	ω_5	-0.0515	-0.0515	-0.0515	-0.0515	-0.0353	-0.0357
	ω_6	-0.0423	-0.0423	-0.0423	-0.0423	-0.0521	-0.0516
	ω_7	-0.0482	-0.0482	-0.0482	-0.0482	-0.0634	-0.0632
	ω_8	-0.0542	-0.0542	-0.0542	-0.0542	-0.0595	-0.0591
	ω_9	-0.0441	-0.0441	-0.0441	-0.0441	-0.0378	-0.0378
	ω_{10}	-0.0452	-0.0452	-0.0452	-0.0452	-0.0286	-0.0286

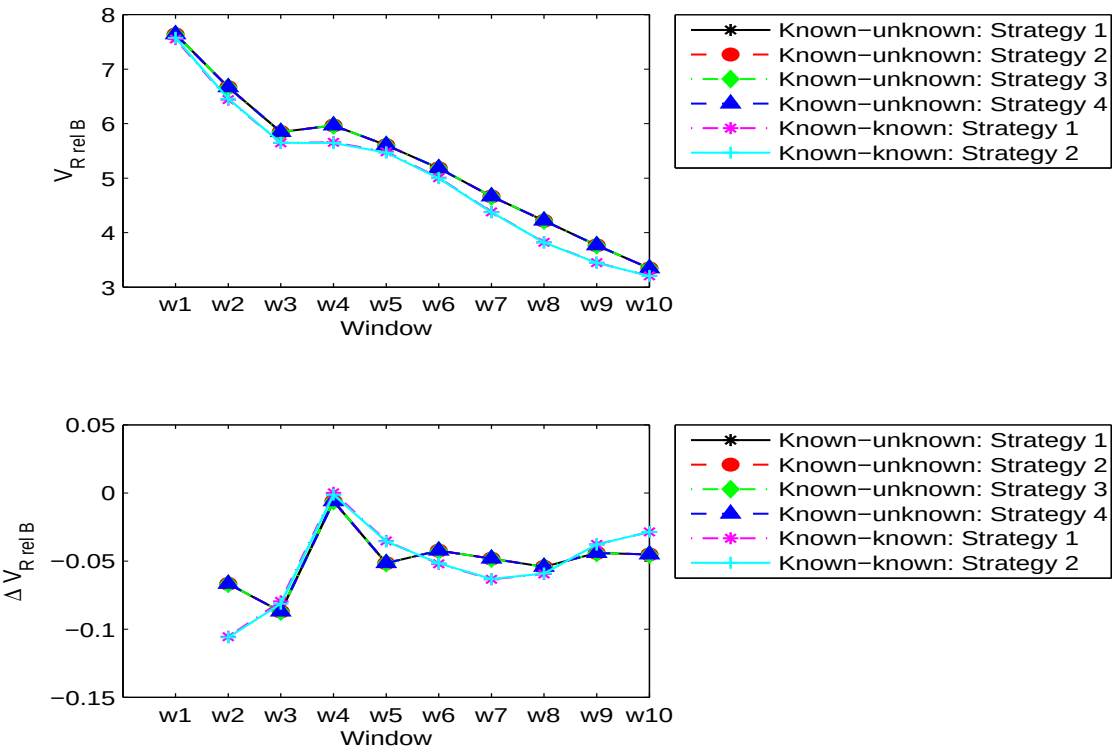


Figure 5.13: $V_{R\,rel\,B}$ and $\Delta V_{R\,rel\,B}$ in the known-unknown and known-known scenarios for HRT.

5.4 Conclusion

The study illustrates that human behaviours are less affected by the information and deception of *blue* when humans engage in high perceptual load. This is supported by the lack of dependencies between the human actions and information as well as deception in both scenarios. In both scenarios, the human players show goal-directed behaviour and it is reflected by the high scores achieved by them. When humans exhibit goal-directed behaviour in an environment that demands attention, high perceptual load prevents humans from processing task-relevant stimuli even if the stimuli are clearly perceived. Consequently, high perceptual load engaging to full capacity in relevant processing leaves humans no extra capacity for perception of task-relevant stimuli. Furthermore, explicitly informing humans about the availability of task-relevant stimuli is also not sufficient to encourage them to process the stimuli. Therefore, this suggests that humans have limited perception capability which prevents them from processing task-relevant stimuli when they are engaged in a task with high perceptual load. Even though how the solution to a decision problem is selected will be a function of individual-differences factors such as processing capacities, the nature of the task with high perceptual load appears to be the factor preventing humans from utilising the task-relevant stimuli. Thus, the limited processing capacities lead to the selection of similar strategies in both treatments regardless of the absence or presence of perception access. In other words, there is a lack of invariance in human decision behaviours owing to high perceptual load.

Chapter 6

Machine Red Teaming

6.1 Introduction

This chapter focuses on the construction of behaviour of a *red* agent based on a machine in the game environment. However, the purpose of using a machine is not to win or lose in the game environment, but rather to use the advances in computational modelling to mimic or reproduce the limited computational capabilities of humans interacting with the task environment to produce bounded rationality.

None of us is completely innocent of the acquaintance with the diversity of human choices. Therefore, one of the important factors in the selection of computational models is the ability to produce diversified solutions. Another important factor is the ability to mimic biological adaptation of the human. Evolution and learning are two forms of adaptation differing in space and time scales, which are important elements to form the adaptive ability [64, 28, 60, 61, 63] of an artificial agent. Therefore, neuroevolution emerges as a potential solution to control the *red* agent in the game environment.

The main contribution of this chapter does not lie in the use of the neuroevolution to

optimise the games. Instead, our contribution focuses on how to represent a *red* and how to produce its behaviour. Reliance on learning machines to automate red teaming does not necessarily account for the complexities of red teaming; the core of red teaming depends on its context, computation and analysis [4].

Since CRT is to mimic human behaviours, the possible factors that could influence human behaviours need to be taken into account in CRT as well. The ways that a human acts in an environment could be affected by his/her observations about its surroundings and the other agent, and the actions taken will affect the achievement of objectives. This means that sensory capabilities creating human perception can be seen as a source that contributes to changes in human behaviours, and we would like to investigate its effect in CRT. To investigate this, information display about the opponent can be used as a way to vary the level of a human's perception in HRT.

The next question will be on how to simulate similar scenarios in CRT? In our work, two scenarios, called known-unknown and known-known, are simulated to investigate the effect of perception in CRT. We hypothesise that the difference in perception will affect machine behaviour.

6.2 Methodology

To synthesise and analyse behaviours in our synthetic simulation environment, neuroevolution is selected as a suitable approach for the task. This involves the evolution of neural networks. Neuroevolution is mostly used in the development of photoaxis behaviours, object avoidance and navigation in a single robot [61]. Neuroevolution has also been used to simulate complex behaviours in a pair of robots [59] as well as a colony of mobile robots [53, 81]. Furthermore, the research also focuses on the effectiveness of either evolution-

ary algorithms or neural networks to simulate a desirable complex behaviour in a given environment.

Neuroevolution can be categorised based on the types of coordination of basic behaviours, i.e., monolithic coordination and layered coordination. Monolithic coordination involves the implementation of a single neurocontroller to support the emergence of a complex behaviour. In contrast, layered coordination involves several neurocontrollers, each controlling a basic behaviour. There are many examples for monolithic neuroevolution [61] while layered neuroevolution can be found in [27].

The focus of the above studies was on the effectiveness of producing desirable behaviour to complete a task. In contrast, we are interested in the possible patterns hidden in the behaviours in a red teaming environment.

Neuroevolution is preferable as a potential solution in the emergence of complex behaviours. The main reasons are its autonomous ability and ability to operate without prior information [53]. Instead of handcrafting the desired behaviour into simple basic behaviours which are handled separately by different modules, neuroevolution can be used to automate the production of the desirable behavioural controller. Beside that, it is able to produce un-characterised behavioural domains. Owing to this strength, neuroevolution is a suitable approach to develop control systems in environments which lack sufficient information. Furthermore, neuroevolution tends to produce a variety of solutions resulting from the interaction between a robot (agent) and its environment [61].

The abilities to evolve and learn play a very important role in behavioural decision making, and they complement each other to produce functional behaviours. In the absence of learning, dysfunctional behaviour will persist. On the other hand, the absence of evolution will make it difficult for individuals to adapt to changes in an environment and will lead to the extinction of individuals. Therefore, neuroevolution is selected to automatically

produce the desirable behaviour for the *red* agent in the environment; that is, for the *red* agent to escape the *blue* agent.

6.3 Neural Networks and Genetic Algorithm

In the proposed environment, the *red* agent is viewed as the robot. Neuroevolution is used to develop the robotic system which is responsible for its movement in the $2D$ environment. In our case, neuroevolution is used to develop the controller of the *red* agent, where GA is responsible for evolving a population of neural networks.

The ultimate goal of our research is not to study evolutionary computation or system optimisation, but to reproduce and augment human behaviours. Therefore, a fixed architecture of neural network is used, and GA is used to evolve the connection weights of the neural network. The neural network for the *red* agent's control system is a multi-layer perceptron of sigmoid units. The network has 5 input neurons, 7 hidden neurons, and 2 output neurons. The architecture of the neural network is shown in Figure 6.1. **IW** and **LW** refer to the matrices of connection weight between the input-hidden layers and hidden-output layers. Besides that, $\hat{\rho}^1$ and $\hat{\rho}^2$ refer to the the vectors of bias units between the input-hidden layers and hidden-output layers. **IW** is a matrix of $HN \times IN$ dimension while **LW** is a matrix of $ON \times HN$. The values of IN, HN and ON refer to total number of input neurons, hidden neurons, and output neurons. $\hat{\rho}^1$ is a vector of $IN \times 1$ dimension while $\hat{\rho}^2$ is a vector of $ON \times 1$ dimension.

Given that the architecture of a neural network is fixed, **IW**, **LW**, $\hat{\rho}^1$ and $\hat{\rho}^2$ are mapped into a vector of weights, which is represented by a chromosome. Therefore, each chromosome in the evolutionary process represents a vector of weights for the input-hidden layers ($\hat{W}_{IW}$) and the hidden-output layers ($\hat{W}_{LW}$), as shown in Figure 6.2. Based on

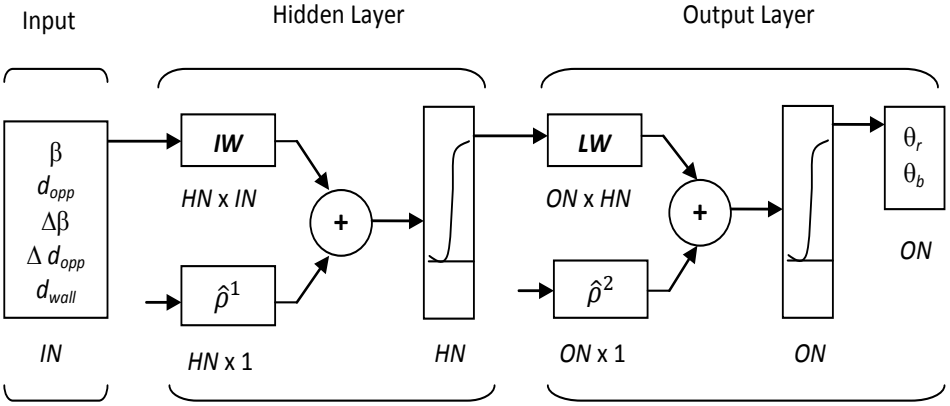


Figure 6.1: Selection of maximum values of the uniform distribution for the generation of $\zeta^{(t)}$.

Figure 6.2, $\hat{W}_{IW}$ consists of a vector of connection weights and bias units between the input-hidden layers, and $\hat{W}_{LW}$ consists of a vector of connection weights and bias units between the hidden-output layers. Through mapping as shown in Figure 6.3 and Figure 6.4, $\hat{W}_{IW} = \{\hat{w}_1^1, \hat{w}_2^1, \ldots, \hat{w}_{hn}^1, \ldots, \hat{w}_{HN}^1\}$ and $\hat{W}_{LW} = \{\hat{w}_1^2, \hat{w}_2^2, \ldots, \hat{w}_{on}^2, \ldots, \hat{w}_{ON}^1\}$. The denotations *in*, *hn* and *on* refer to neurons in input, hidden and output layers, with $1 \leq in \leq IN$, $1 \leq hn \leq HN$, $1 \leq on \leq ON$.

For each time step, the neural network uses the information on $\beta^{(t)}$, $d_{opp}^{(t)}$, $\Delta\beta^{(t)}$, $\Delta d_{opp}^{(t)}$, $d_{wall}^{(t)}$ as the inputs, and produces the outputs for $\theta_r^{(t)}$ and $\theta_b^{(t)}$. At this point, the planned travel angle of the *red* agent $\theta_r^{(t)}$ is determined and the *red* agent moves to its new location. At the same time, the *blue* agent also moves to its new location based on its own strategy. The actual travel angle of the *blue* agent $\hat{P}_b^{(t)}$ is compared with the predicted one $\hat{P}_{pb}^{(t)}$, and the difference is used to adjust the back-propagation of error to the connection weights between network layers. Using the network as the control system, the *red* agent will move in the environment until the game is terminated. The fitness values of each neural network is

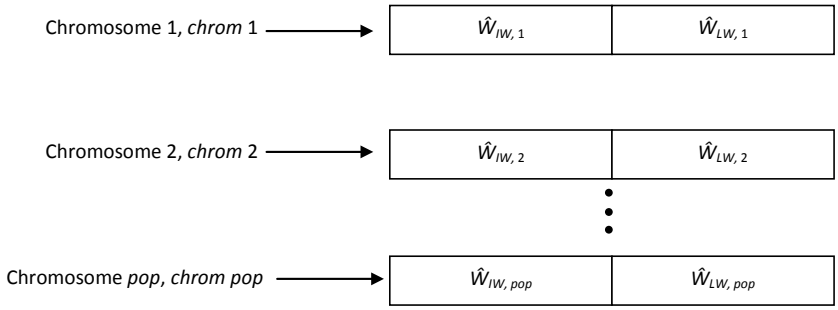


Figure 6.2: Weights for the fixed architecture neural network in chromosome representation.

evaluated based its performance in preventing the *red* agent from being caught by the *blue* agent for a number of repeated games, $N_G = 10$. The details of the fitness are explained in Section 6.3.2.

6.3.1 Genetic algorithm

In GA, we begin with a population of randomly generated individuals, i.e., $pop = 100$, each yielding a different set of connection weights for a neural network. The network architecture and relevant learning parameters are fixed and identical for all individuals. This is generation 0, $gen\,0$. The initial generation of networks are allowed to live 100 steps ($S = 100$) or until the *red* agent is caught by the *blue* agent in the game, depending on which condition is satisfied first. Therefore, each individual is responsible for a new game. At the beginning of each game, the locations of the *red* and the *blue* agents are placed randomly, with the following constraints:

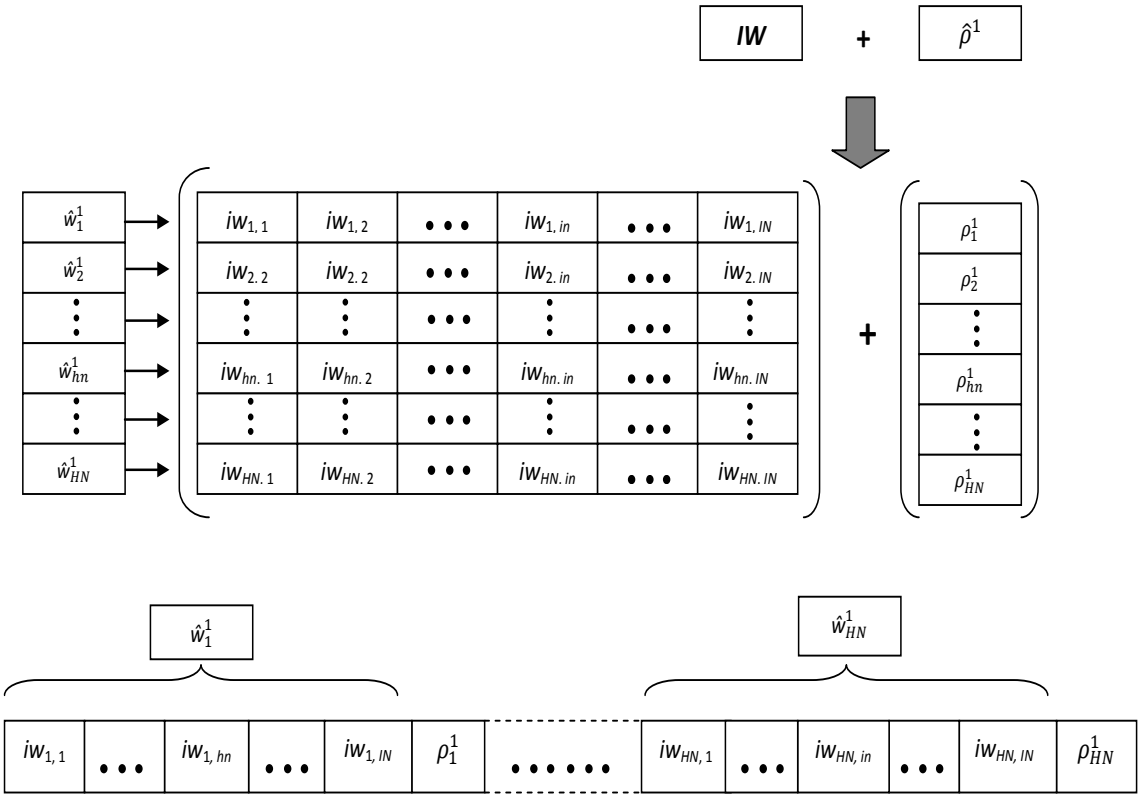


Figure 6.3: Mapping of the connection weights and bias units for both input-hidden layers.

$$\|\hat{P}_r^0 - \hat{P}_b^0\|^2 = 200\text{pixels} \tag{6.1}$$

with $\|\hat{P}_r^0 - \hat{P}_b^0\|^2$ referring to the *Euclidean* distance between the *red* and *blue* agents at time $t = 0$.

At the end of the game, individuals from the population that have accumulated the most fitness are allowed to reproduce based on the *binary tournament* selection method. Once the new population has been created by selective reproduction, *one point* crossover

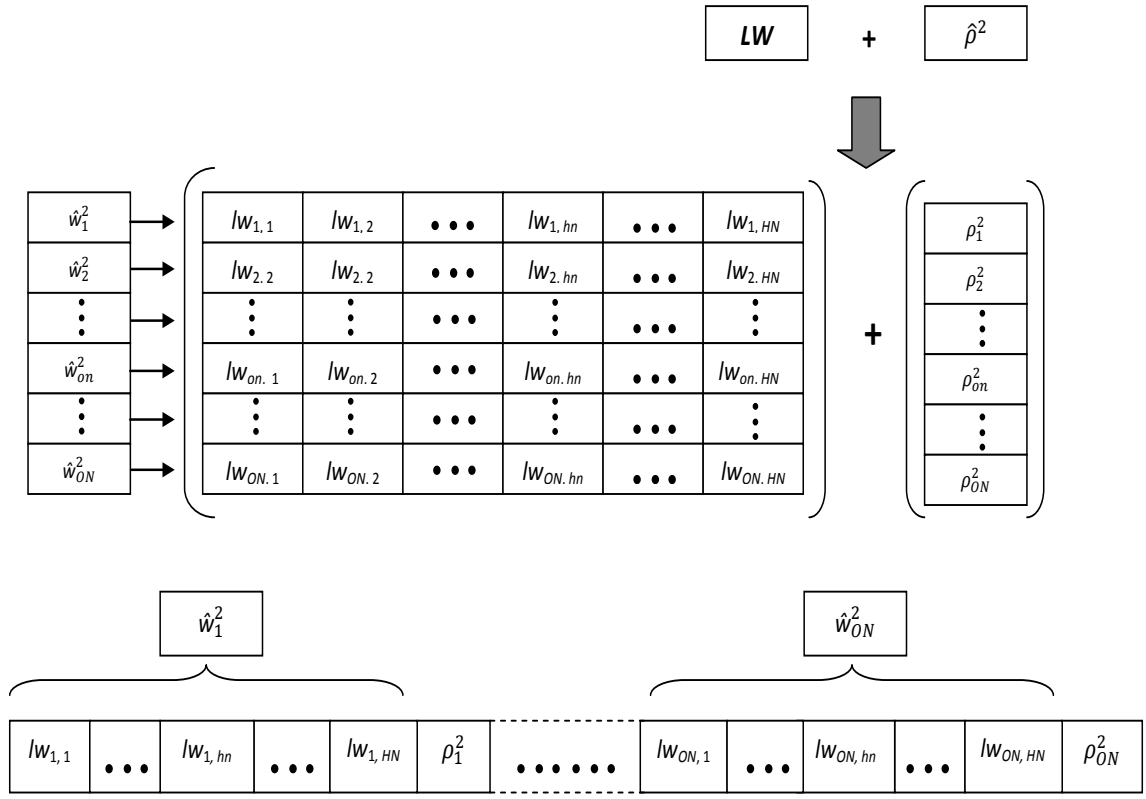


Figure 6.4: Mapping of the connection weights and bias units for both hidden-out layers.

and mutation are performed on the offspring. The individuals which have gone through selective reproduction, crossover and mutation are known as generation 1, $gen\,1$. Then, the fittest 10% of parents (from $gen\,0$) are copied to the offspring population ($gen\,1$) and replacing the bottom 10% of offsprings from $gen\,1$. The process is repeated for a number of generations, $gen = 200$.

6.3.2 Fitness Function

Since the main goal of the simulation is to evolve a *red* agent (controlled by a neural network) that is capable of autonomous movements in a partially unknown and unpredictable environment without human intervention, it would be desirable that the design of the fitness function will be behaviour-implicit [61], whereby the function is rated based on behavioural outcome of an evolutionary network and relies on a few variables and constraints only. Therefore, the proposed fitness functions are based on a variable directly measurable on the *red* agent at each time step. The fitness functions are as follows:

$$F = h \tag{6.2}$$

$$h = \begin{cases} 1 & \text{if } d_{opp} > D_{min} \\ 0 & \text{otherwise} \end{cases} \tag{6.3}$$

where h refers to the frequency of the *red* agent meeting the constraints ($d_{opp} > D_{min}$). The parameter d_{opp} refers to the actual distance between the *blue* and *red* agents, while D_{min} refers to the distance which declares the *red* agent is caught by the *blue* agent. The frequency of meeting the constraint is calculated as F and is accumulated for each step. Based on the above equations, the fitness function measures the performance of the *red* agent to prolong its lifetime in the game.

6.3.3 Neural Network

The neuro-evolutionary model as suggested by Nolfi *et al.* [66] is used in the experiment, which evolves the robot to accomplish a task at the population level and learn another

different task at the individual level. According to [34, 66], the emergence of a desired behaviour results from interaction between the innate knowledge and learning. Therefore, both evolution and learning need to be taken into account to evolve a control system. Even though what is learnt may not be the behaviour itself, it may support the emergence of the desired behaviour. In the neuroevolutionary model suggested in [66], learning has directionality and the learning task itself is evolved.

In our simulation, the so called *optimal* behaviours for the *red* agent are unclear, given that the *red* agent needs to face a partially unknown and unpredictable *blue* agent. Therefore, it would be preferable that the network itself is able to develop some useful behaviours rather than to learn a particular behaviour derived in advance by the experimenter. Therefore, we adopted the evolved neural network suggested by [66] into our experiment. The proposed approach involves the evolution of back-propagation neural networks, whereby a population of networks (individuals) reproduce selectively based on its performance in producing the desired behaviour. At the same time, the neural network also learns an additional task. However, the learning criterion is different from the evolutionary goal. Therefore, the performance of the neural network in the learning task does not affect the network's fitness. The important point to emphasise here is the neural network does not learn which behaviours are good or bad, it just learns that there are sensory consequences attendant on specific movements in the context of specific environmental inputs.

The network architecture is fixed and identical for all individual neural networks in the evolution. Therefore, the number of neurons in each layer and other relevant parameters such as learning rate and momentum rate are fixed for all neural networks. The learning rate, η for the network is set as 0.2 and the momentum rate, γ is set as 0. Each connection weight of the neural network ranges in the interval $[-1, 1]$.

6.4 Experimental Design for Machine *Red* Agent

The main objective of this experiment is to understand the behaviours exhibited by the *red* agent through evolving its strategies. Through the evolutionary process, we have a population of diversified strategies for the *red* agent to choose from. The population of strategies contains information which may reveal the biases and preferences of the *red* agent under the condition that it is placed within. Knowing the *red* agent's biases and preferences is crucial for helping us to understand the rationality of decision making, and thus, helping us to manage behaviours more effectively.

In this case, we would like to know how a *red* agent behaves facing a deceptive *blue* agent, when the *red* agent is exposed to different levels of perception about the *blue* agent. Two levels of perception are simulated in this experiment, called known-unknown and known-known. The difference between them lies in what the *blue* agent knows about the *red* agent. In the known-unknown scenario, *red* does not know what *blue* knows about *red*. In the known-known scenario, the *red* does know what *blue* knows about itself.

To answer the key questions, we need to fix the behaviour of the *blue* agent and evolve the *red* agent in each scenario. The behaviour of the *blue* agent is influenced by the information received by the *red* agent's intelligence as well as the deceptive efforts. Therefore, different scenarios based on the configurations of N_I, $\alpha^{(t)}$, N_D and $\zeta^{(t)}$ are created. By fixing the information received about the *red* agent's intelligence, we will have four combinations of setups involving N_D and $\zeta^{(t)}$. On the other hand, there are five combinations of setups which depend on N_I and $\alpha^{(t)}$, by fixing the deception efforts from the *blue* agent. In total, we have twenty different scenarios which reflect the differences of behaviours from the *blue* agent. Each scenario, depending on the configuration of N_I, $\alpha^{(t)}$, N_D and $\zeta^{(t)}$, is denoted as c_ϕ, with $1 \leq \phi \leq 20$.

Given that the *blue* agent mainly chases after the *red* agent, the *red* agent needs to move

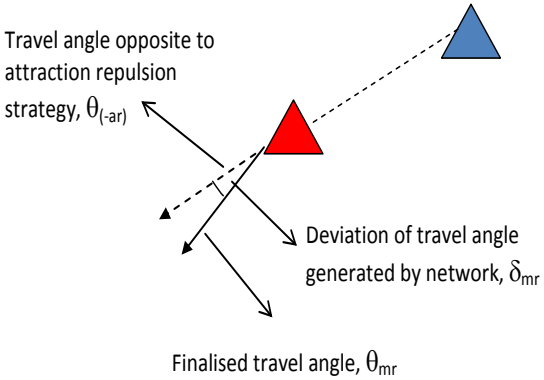


Figure 6.5: Generation of travel angle for the *red* agent.

in the opposite direction of the attraction repulsion strategy so that it won't be caught by the *blue* agent; the travel angle of the *red* agent under such condition is denoted as $\theta_{(-ar)}$. In this experiment, neuroevolution produces the deviation of travel angle of the *red* agent, $\delta\theta_{mr}$. Therefore, the finalized travel angle for the *red* agent, θ_{mr} is the sum of $\theta_{(-ar)}$ and $\delta\theta_{mr}$ ($\theta_{mr} = \theta_{(-ar)} + \delta\theta_{mr}$). For ease of understanding, both scenarios are illustrated in Figure 6.5.

The *red* agent is evolved in the direction of accomplishing its goal without any preferences on how it should achieve its goal. Initially, a population of diversified strategies is generated; each has its own strengths and limitations. Each strategy in the population refers to a neural network which acts as the *red* agent's decision-making model for its movements. Changes in the connection weights of the neural network represents the learning ability of the *red* agent in the decision-making process as it acts and interacts with the *blue* agent as well with the environment. Instead of evolving the strategies based on a detailed fitness function that may bias towards a specific characteristic, e.g., maximising the distance between the *red* and *blue* agents, the strategies are evolved to meet the *red* agent's goal only: to avoid being caught by the *blue* agent.

Along the evolutionary process, the strategies of the *red* agent are expected to survive by devising their own goals and finding the solutions to overcome the challenges that may

arise from the interaction with the environment. Here, the interaction with the environment includes the interaction with the $2D$ environment as well as the *blue* agent. At the end of the evolutionary process, we will have a population of survived strategies which manage to overcome the challenges in the environment. These strategies are rich in information that reveals the preferences of the *red* agent developing its behaviours for survival. By using data mining techniques to extract patterns in the strategies, we can identify the internal representation for the *red* agent's behaviours.

For each configuration, the *red* agent feeds the relevant input variables into the neural network and obtains the decision variables from the neural network to arrange its movements. The decision variables are the same for both scenarios, but there is a difference in input variables between them as shown below:

- Input variables:
  - Known-unknown scenario:
    1. the relative angle between the *blue* and *red*, $\beta^{(t)}$.
    2. the relative distance between the *blue* and *red* agents at time t, $d_{opp}^{(t)}$.
    3. the relative change of β at time t, $\Delta\beta^{(t)}$.
    4. the relative change of d_{opp} at time t, $\Delta d_{opp}^{(t)}$.
    5. the relative distance to the wall that the *red* agent faces at time t, $d_{wall}^{(t)}$.
  - Known-known scenario:
    1. the relative angle between the *blue* and *red*, $\beta^{(t)}$.
    2. the relative distance between the *blue* and *red* agents at time t, $d_{opp}^{(t)}$.
    3. the relative change of β at time t, $\Delta\beta^{(t)}$.
    4. the relative change of d_{opp} at time t, $\Delta d_{opp}^{(t)}$.
    5. the relative distance to the wall that the *red* agent faces at time t, $d_{wall}^{(t)}$.

6. the level of noise in the information received by the *blue* agent, $\hat{\alpha}^{(t)}$.

- Decision variables:

  1. deviation of travel angle for the machine *red* agent, $\delta\theta_{mr}^{(t)}$
  2. travel angle of the *blue* agent, $\theta_{pb}^{(t)}$

The input and decision variables are represented by a vector of real numbers. When the input information is received by the red agent's sensor, the input variables are scaled into the range of [0, 1] before they are fed into the neural networks to produce the output variables. The output variables are also in the range of [0, 1] and they are scaled to their actual ranges before they are used by the *red* agent. The strategies used by the *red* agent are evolved at the population level and learnt at the individual level at the same time.

To be more specific, the neural network that is selected to help the *red* agent to escape from the *blue* agent needs to learn another task; to predict the sensory consequences of the *red* agent's actions during its lifetime span. The learning task for the neural network is to predict the travel angle of the *blue* agent $\theta_{pb}^{(t)}$, which is different from the evolutionary goal. The difference between the predicted ($\theta_{pb}^{(t)}$) and actual travel angle $\theta_{b}^{(t)}$ of the *blue* agent is used to adjust the connection weights of the network through back-propagation.

Even though the neural network also determines the *red* agent's action based on the produced $\delta\theta_{mr}^{(t)}$, it does not learn to determine which actions are good or bad. Instead, it just learns that there are consequences relying on specific actions. At each time step, the neural network receives information about the *blue* agent's whereabouts and the surrounding environment, and produces as output the next planned travel angle for itself and a prediction of the *blue* agent's travel angle after the execution of its own current travel angle. Even though the network learning is based on the prediction on the $\theta_{pb}^{(t)}$ only, the execution of $\delta\theta_{mr}^{(t)}$ will influence the next received information about the *blue* agent's where-

abouts and the surrounding environment at time $t+1$. From here, we can see that there is no evaluation of whether the execution of $\delta\theta_{mr}^{(t)}$ is right or wrong. Instead, the evaluation of the *red* agent's actions is carried out at the population level.

For each strategy (neural network) from the population, the *red* agent will use it as its decision-making model to determine its movements in a game. The movements for both agents are updated simultaneously until the game is terminated.

Since the neuroevolutionary model for the *red* agent is a stochastic model, experiments were repeated with 30 independent random seeds for each of the experimental configurations discussed above. Each game is repeated 10 times for the same scenario (configuration), i.e., $N_G = 10$. The total number of repeated games for each scenario is 300 (30×10). This means the initial starting points of the *red* agent in these 300 games are the same between different scenarios. For instance, the starting point in game 1 based on random seed 1 is the same for the 20 different scenarios, and so on. Even though the distance from the *red* agent to the wall may be different at the start of the games, the large number of repetitions are able to reduce the random effect of the distance to the wall of the *red* agent. Besides that, all of the configurations are evaluated fairly because they all have the same game setups.

The fitness of each strategy is evaluated according to the average fitness of 10 games for the same scenario. The above procedures are repeated for 20 scenarios, which are based on different strategies from the *blue* agent. The objective of a scenario is to identify the preferable strategies of the *red* agent to achieve its goal given that it is placed with an intelligent *blue* agent.

The goal of the *red* agent is to survive as long as possible. As long as the *red* agent is not caught by the *blue* agent, it is considered to succeed in achieving its goal. The longer the *red* agent can survive, the more likely it is that it will win in the game. The fitness

function is defined as follows:

$$F = \frac{1}{N_G} \sum_{g=1}^{N_G} h_g \tag{6.4}$$

$$h_g = \begin{cases} 1 & \text{if } d_{opp} > D_{min} \\ 0 & \text{otherwise} \end{cases} \tag{6.5}$$

where h_g refers to the frequency of the *red* agent meeting the constraints ($d_{opp} > D_{min}$) in the g^{th} game, with $1 \leq g \leq N_G$, where N_G refers to the number of games that are repeated for each scenario (we use $N_G = 10$ in this experiment; thus results are averaged over 10 games). The frequency of meeting the constraint is calculated as F, and is accumulated for each step.

Based on the above equations, the fitness function measures the performance of the *red* agent to extend its life span.

At the end of the evolutionary process, a population of diversified strategies are obtained for each scenario. By comparing the strategies of the *red* agent for different scenarios, we can identify its preferences in strategies for different scenarios. There may be some sort of patterns exhibited in the strategies and the patterns are reflected/hidden in the actions taken by the *red* agent. To extract the patterns exhibited in the strategies, we need to use data mining techniques to analyse the *red* agent's actions. Given that each strategy will produce a sequence of actions for the *red* agent in a scenario, we will have a set of action sequences of different lengths for each scenario. A good strategy will help the *red* agent to get away from the *blue* agent for longer, while a bad strategy will cause the *red* agent to be caught by the *blue* agent faster. Consequently, there will be a difference of length in

both action sequences. Pre-processing as suggested in Section 4.2.2 is used to standardise the lengths of action sequences.

6.5 Result and Analysis

In this section, the effects of information and deception on the machine's behaviours in known-unknown and known-known scenarios are discussed in terms of evolution, action distribution and action similarity. For the figures related to the distributions of ϑ and $\Delta\vartheta$, they are plotted in such a way that:

- The sequence of information in the same row from left to right: frequent-accurate, frequent-noisy, infrequent-accurate, infrequent-noisy;
- The sequence of deception in the same column from top to bottom: none, infrequent-low, infrequent-high, frequent-low, frequent-high.

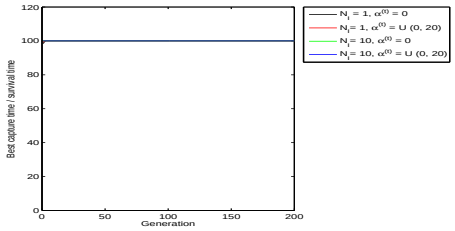
6.5.1 Evolution

For ease of discussion, Table 6.1 summarises the combinations of perception and deception.

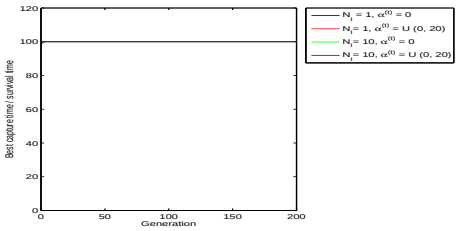
Table 6.1: Description about perception and deception

Definition	Parameter	Value	Description
Perception	Intel frequency	$N_I = 10$	Infrequent
		$N_I = 1$	Frequent
	Intel noise	$\zeta = 0$	Accurate
		$\zeta = U(0,\ 20)$	Noisy
Deception	Deception frequency	$N_D = 10$	Infrequent
		$N_D = 5$	Frequent
	Deception degree	$\zeta^{(t)} = U(-15°,\ 15°)$	Low
		$\zeta^{(t)} = U(-30°,\ 30°)$	High
	Combination	$N_D = 1,\ \zeta^{(t)} = 0$	No deception

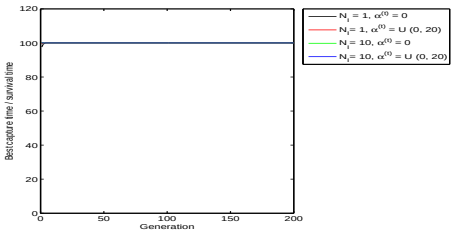
Figures 6.6 and 6.7 show the best and average capture times (averaged over 30 runs) for *blue*, for different combinations of information and deception. The sub-figures on the



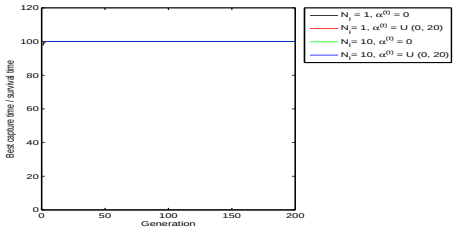
(a) $N_D = 1, \zeta^{(t)} = 0$



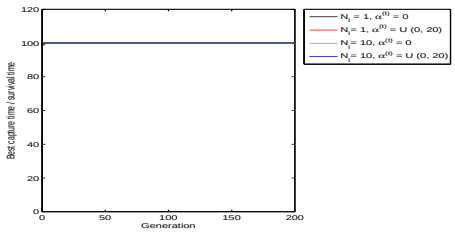
(b) $N_D = 1, \zeta^{(t)} = 0$



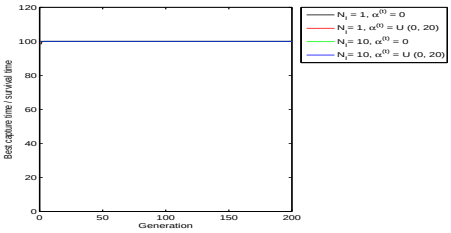
(c) $N_D = 10, \zeta^{(t)} = U(-15°, 15°)$



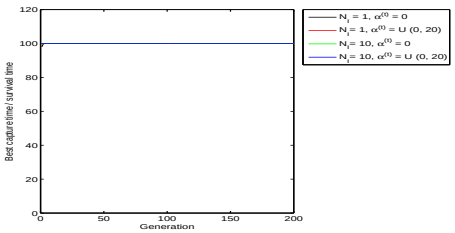
(d) $N_D = 10, \zeta^{(t)} = U(-15°, 15°)$



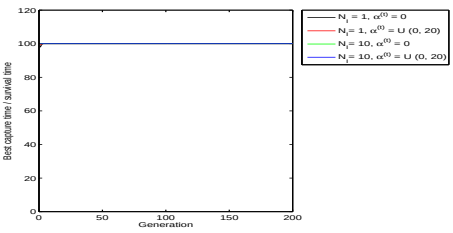
(e) $N_D = 10, \zeta^{(t)} = U(-30°, 30°)$



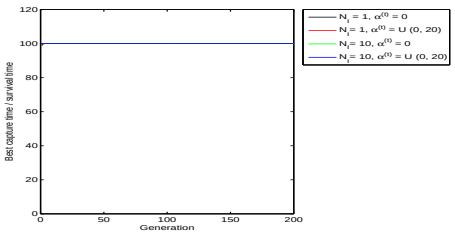
(f) $N_D = 10, \zeta^{(t)} = U(-30°, 30°)$



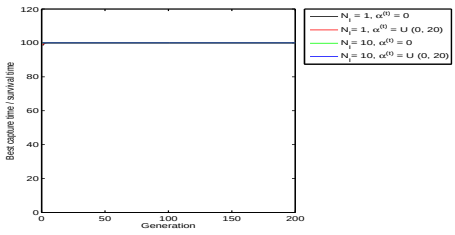
(g) $N_D = 5, \zeta^{(t)} = U(-15°, 15°)$



(h) $N_D = 5, \zeta^{(t)} = U(-15°, 15°)$

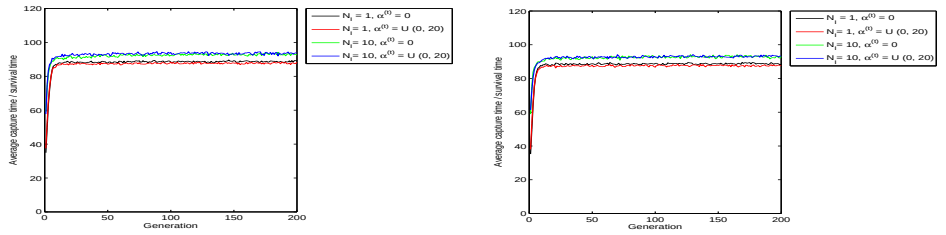


(i) $N_D = 5, \zeta^{(t)} = U(-30°, 30°)$

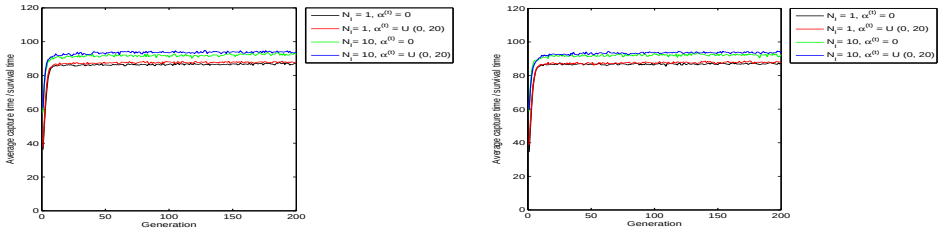


(j) $N_D = 5, \zeta^{(t)} = U(-30°, 30°)$

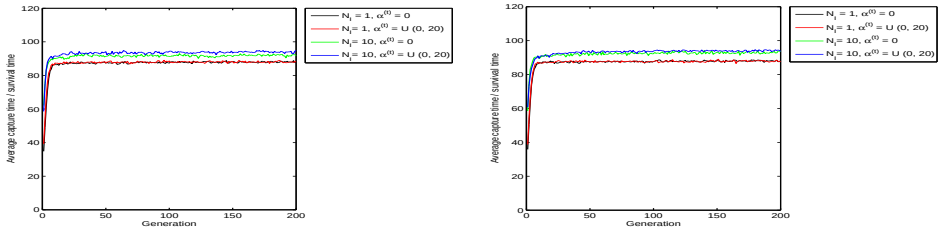
Figure 6.6: The plots of *best capture time / survival time* by fixing N_D, $\zeta^{(t)}$ and varying N_I, $\hat{\alpha}^{(t)}$ in the same sub-figure.



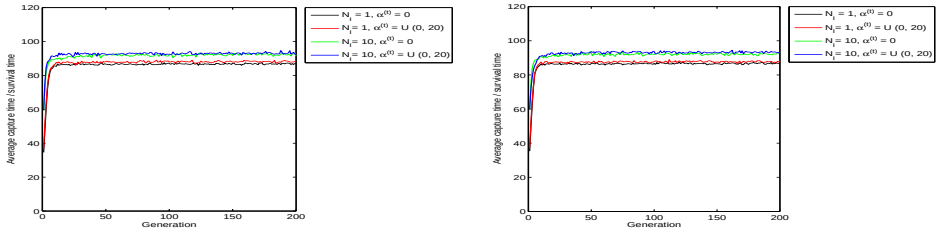
(a) $N_D = 1, \zeta^{(t)} = 0$ (b) $N_D = 1, \zeta^{(t)} = 0$



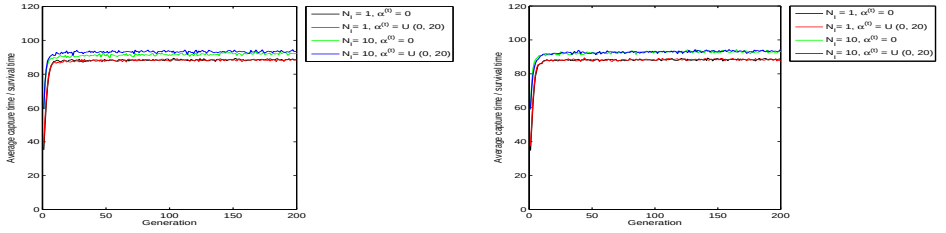
(c) $N_D = 10, \zeta^{(t)} = U(-15°, 15°)$ (d) $N_D = 10, \zeta^{(t)} = U(-15°, 15°)$



(e) $N_D = 10, \zeta^{(t)} = U(-30°, 30°)$ (f) $N_D = 10, \zeta^{(t)} = U(-30°, 30°)$

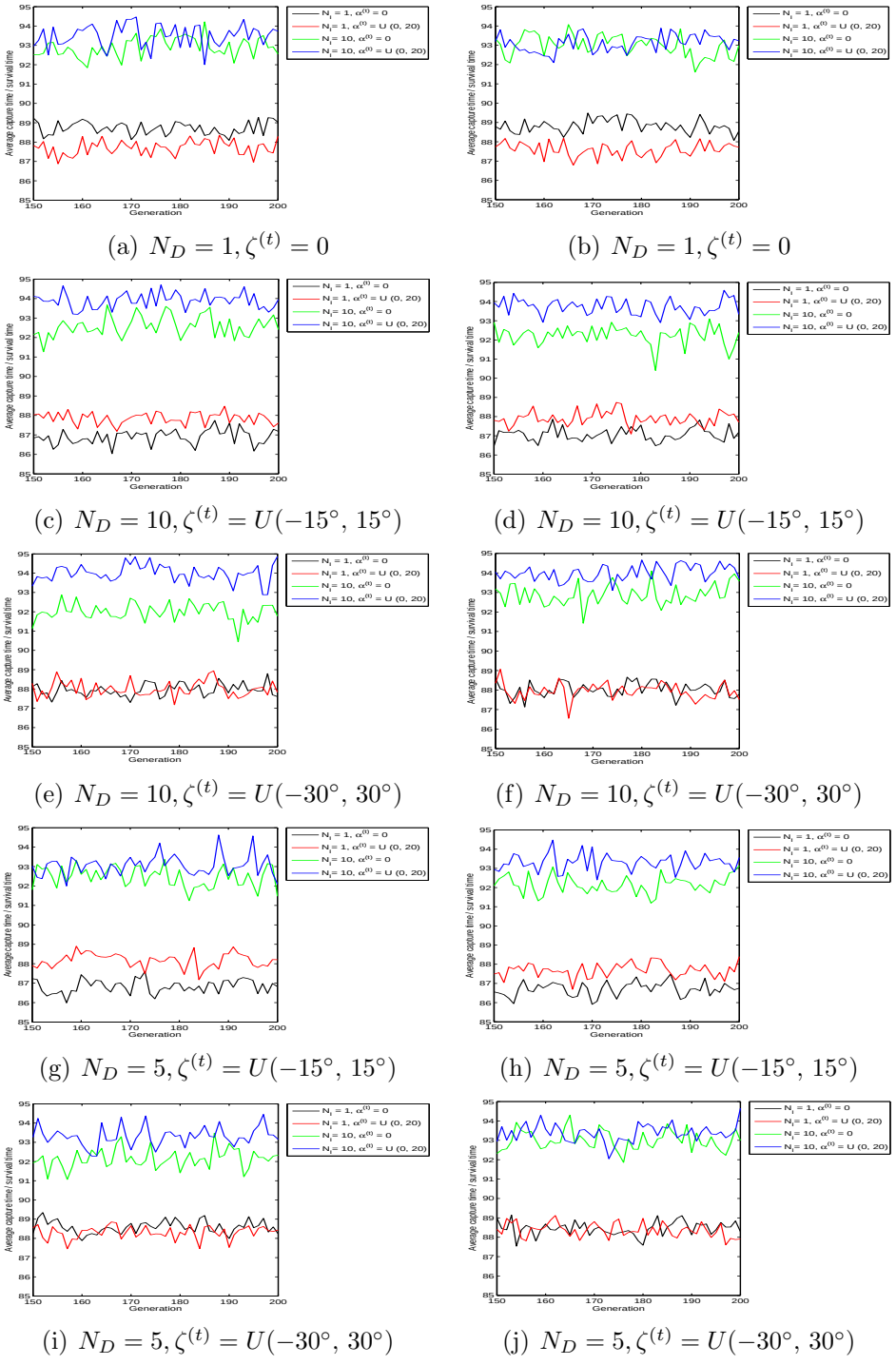


(g) $N_D = 5, \zeta^{(t)} = U(-15°, 15°)$ (h) $N_D = 5, \zeta^{(t)} = U(-15°, 15°)$



(i) $N_D = 5, \zeta^{(t)} = U(-30°, 30°)$ (j) $N_D = 5, \zeta^{(t)} = U(-30°, 30°)$

Figure 6.7: The plots of *average capture time / survival time* by fixing N_D, $\zeta^{(t)}$ and varying N_I, $\hat{\alpha}^{(t)}$ in the same sub-figure.



(a) $N_D = 1, \zeta^{(t)} = 0$ (b) $N_D = 1, \zeta^{(t)} = 0$

(c) $N_D = 10, \zeta^{(t)} = U(-15°, 15°)$ (d) $N_D = 10, \zeta^{(t)} = U(-15°, 15°)$

(e) $N_D = 10, \zeta^{(t)} = U(-30°, 30°)$ (f) $N_D = 10, \zeta^{(t)} = U(-30°, 30°)$

(g) $N_D = 5, \zeta^{(t)} = U(-15°, 15°)$ (h) $N_D = 5, \zeta^{(t)} = U(-15°, 15°)$

(i) $N_D = 5, \zeta^{(t)} = U(-30°, 30°)$ (j) $N_D = 5, \zeta^{(t)} = U(-30°, 30°)$

Figure 6.8: Zoom in of Figure 6.7 for the last 50 generations.

left and right sides show the plots for the known-unknown and known-known scenarios for the same configuration. From *red*'s perspective, the capture time of *blue* is the converse of

the survival time of *red*.

Based on Figure 6.6, it shows that the *best survival time* of *red* achieves maximum value after a few generations in both scenarios across different configurations. Figure 6.8 zooms in on the last 50 generations of Figure 6.7, in which the *average survival times* of *red* are shown. By comparing both scenarios for the same configuration in Figure 6.8, we can observe that the trends of *average survival times* for both scenarios are very similar. By referring to Figure 6.7, it is no surprise to observe that *red* has shorter *average survival times* given the *blue* agent receives frequent information rather than infrequent information. Besides that, continuous increments of the average survival time of *red* can be observed from Figures 6.7 and this observation means learning does occur within the *red* agent.

An interesting finding in the figure can be observed from the following paired comparisons between different levels of deception.

- frequent-accurate information and frequent-noisy information;
- infrequent-accurate information and infrequent-noisy information.

When the *blue* agent is not deceptive, we can observe that there is not much difference in *average survival times* between infrequent-accurate information and infrequent-noisy information. However, there differences become clearer when the *blue* agent changes from being non-deceptive to deceptive. From the comparisons made between the infrequent-accurate and infrequent-noisy information, we can observe that the *average survival times* of the *red* agent become closer to each other as the same degree of deception (for high or low degrees of deception) changes from being infrequent to frequent. Since the survival time of *red* is the converse of the capture time of *blue*, this means having higher frequency of deception can counterbalance the effect of noise in the delayed information, and thus, helps the *blue* agent to have similar *average capture time* regardless of whether the received

information is noise free or not.

From the comparisons between frequent-accurate information and frequent-noisy information, the *average survival times* of the *red* agent becomes closer to each other as the deception degree increases for the same level of deception frequency. This means that higher degree of deception can counterbalance the effect of noise in the frequently received information, which causes the *blue* agent to have similar *average capture times* as those upon receiving accurate information. This can be supported by the observation between the following paired figures:

- infrequent-low deception and infrequent-high deception: Figures 6.8(c)-6.8(e); 6.8(d)-6.8(f);
- frequent-low deception and frequent-high deception: Figures 6.8(g)-6.8(i); 6.8(h)-6.8(j)

6.5.2 Action Distribution

There are 20 different combinations of information and deception involved in the experiment, and the evolution is repeated by using 30 different seeds for each combination. For each seed number, the best solution is used by *red* in 10 repeated games. Therefore, there are 6000 (30×20×10) different action sequences generated in the experiment. This will need a lot of space to display all of the trajectories. Instead, we only show a few interesting trajectories between *blue* and *red* across different configurations in both scenarios as shown in Figures 6.9 and 6.10.

To extract information from the trajectories between *red* and *blue* for each configuration, ϑ and $\Delta\vartheta$ are measured respectively from all trajectories. For the distributions of these 2 measurements, the maximum likelihood estimates for the parameters of a mixture of

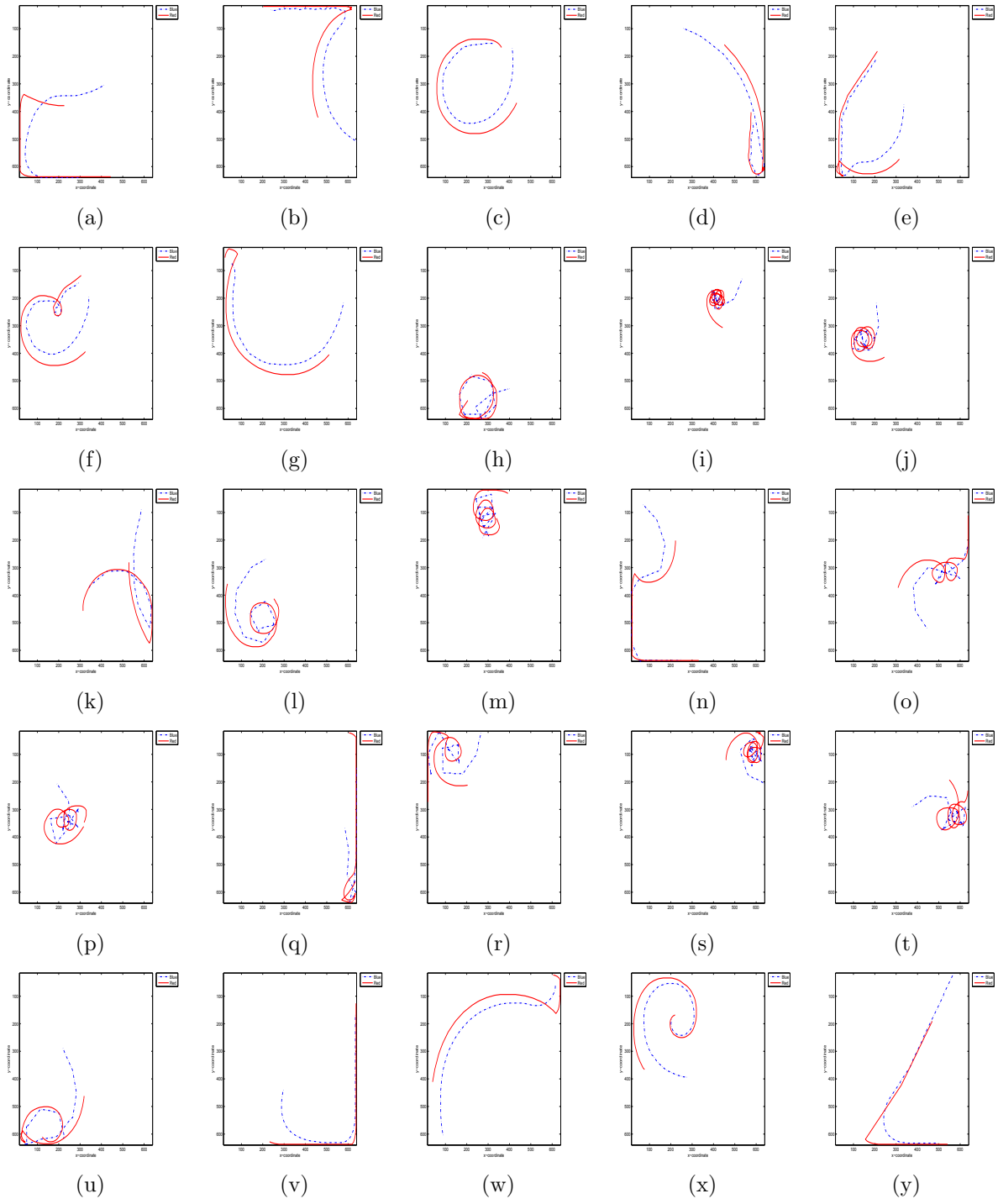


Figure 6.9: Examples of interesting trajectories between *blue* and *red* in the known-unknown scenario.

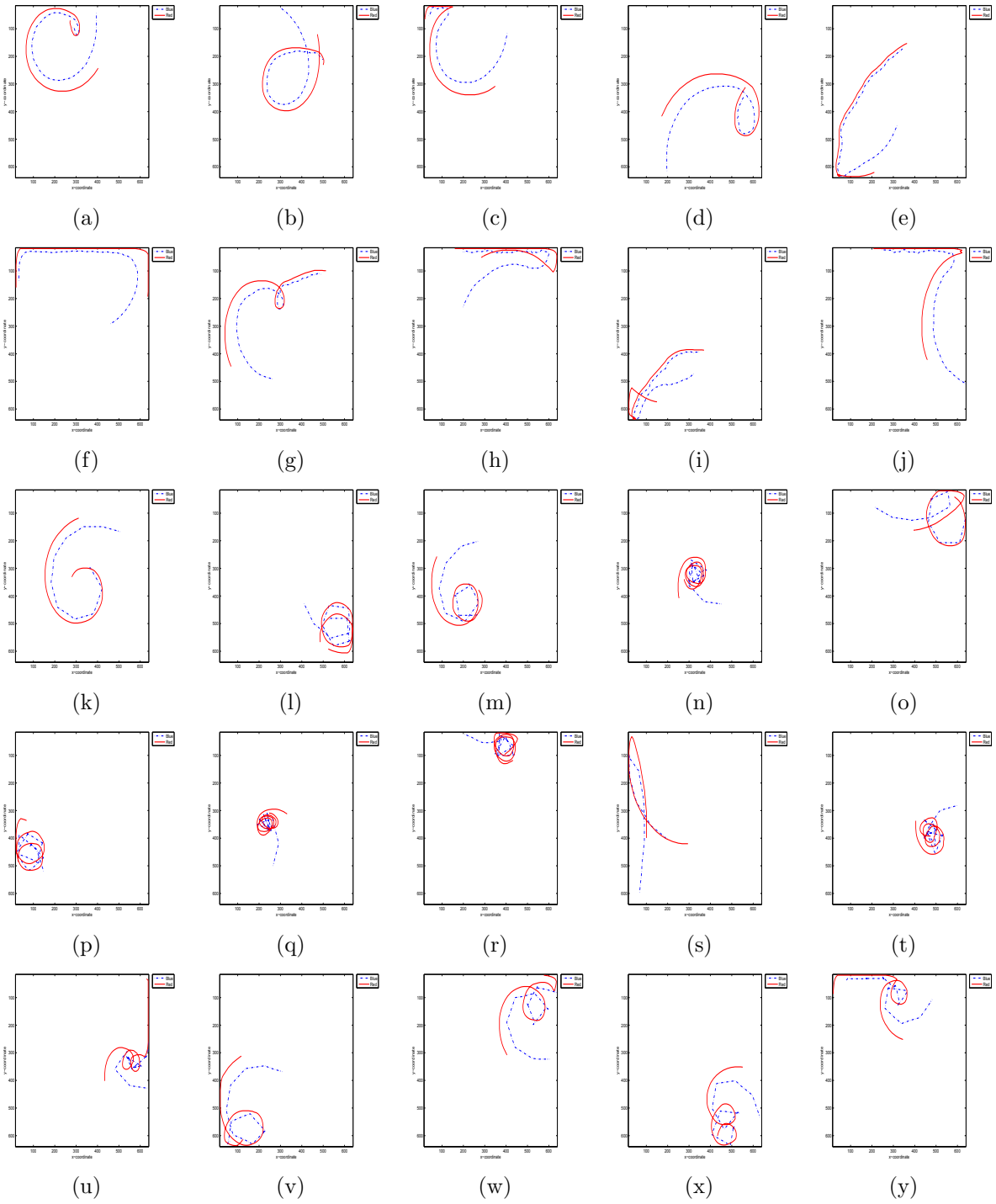


Figure 6.10: Examples of interesting trajectories between *blue* and *red* in the known-known scenario.

normal distributions based on 95% of confidence interval are conducted, i.e., ratio, mean and standard deviation (stdev). For ease of visualisation and analysis, probability density functions ("pdfs") for these measurements are shown in Figures 6.11 to 6.14 for both scenarios.

Figures 6.11 and 6.12 show pdfs for the measurements of ϑ for the known-unknown and known-known scenarios respectively. The pdfs based on $\Delta\vartheta$ for both scenarios are shown in Figures 6.13 and 6.14 respectively. For ease of explanation, we will call the distributions that are associated with deceptive *blue* and non-deceptive *blue* as deceptive distribution and non-deceptive distribution. The sub-figures in each figure are sorted in the following order:

- In the same row, the deception is fixed and the information changes in the sequence: frequent-accurate information, frequent-noisy information, infrequent-accurate information, infrequent-noisy information.

- In the column, the received information is fixed and the deception changes in the sequence: no deception, infrequent-low deception, infrequent-high deception, frequent-low deception, frequent-high deception.

By comparing the pdfs for both scenarios based on ϑ and $\Delta\vartheta$, we can observe that both scenarios have similar distributions for the same configuration, where ϑ has a mixture of normal distributions while $\Delta\vartheta$ has a unimodal distribution. Having similar distributions for the same configuration may suggest that neuroevolution manages to produce similar behaviours in spite of perception. This means the *red* agent, with and without having perception about the *blue* agent, is able to capture the dominant patterns in the *blue* agent's movements and produce similar actions. In other words, the quality of best solutions produced by neuroevolution are not affected even though it lacks perception about *blue*.

In the known-known scenario, the noise level $\hat{\alpha}^{(t)}$ provided to the *red* agent represents an extra task-relevant stimulus when it is viewed in a HRT. It is interesting to find out that the distributions of the *red* agent are not affected by the availability of the extra task-relevant stimulus.

Given that there is not much difference in action distributions between both scenarios, we are interested to know whether there is any significant difference of scores between them for the same combination of information and configuration or not. To investigate the effect of the the extra task-relevant stimuli, a t-test is carried out with 0.05 significance level to test the null hypothesis that the samples come from both scenarios with equal means, against the alternative that the means are unequal. Table 6.2 shows the result of t-test on the scores between the known-unknown and known-known scenarios based on the combinations of information and perception.

Table 6.2: t-test for scores between the known-unknown and known-known scenarios in CRT based on neuroevolution.

N_I	$\hat{\alpha}^{(t)}$	N_D	$\zeta^{(t)}$	Scores	
				Reject H_0	p-value
1	0	1	0	No	0.7478
		5	$U(-15°, 15°)$	No	0.9076
		5	$U(-30°, 30°)$	No	0.4485
		10	$U(-15°, 15°)$	No	0.4076
		10	$U(-30°, 30°)$	No	0.61967
1	$U(0, 20)$	1	0	**Yes**	**0.0428**
		5	$U(-15°, 15°)$	No	0.3704
		5	$U(-30°, 30°)$	**Yes**	**0.0445**
		10	$U(-15°, 15°)$	No	0.5516
		10	$U(-30°, 30°)$	No	0.2890
10	0	1	0	No	0.4828
		5	$U(-15°, 15°)$	No	0.4956
		5	$U(-30°, 30°)$	No	0.2683
		10	$U(-15°, 15°)$	No	0.2578
		10	$U(-30°, 30°)$	No	0.4554
10	$U(0, 20)$	1	0	Yes	0.0007
		5	$U(-15°, 15°)$	Yes	0.0010
		5	$U(-30°, 30°)$	Yes	0.0001
		10	$U(-15°, 15°)$	Yes	0.0102
		10	$U(-30°, 30°)$	Yes	0.0000

Based on the results shown in Table 6.2, 75% of the null hypotheses are not rejected or at the border of rejection except for those associated with infrequent-noisy information.

The results which are at the border of rejection are highlighted in bold, where they are associated with p-values of 0.0428 and 0.0445 respectively. This means that we do not have enough evidence to claim that there are significant differences between the known-unknown and known-known scenarios for most of the configurations. Indirectly, this finding indicates that having similar behaviours in most of the configurations is likely to cause the *red* agent to have similar outcomes as well.

Based on the observations in Figures 6.11 and 6.12, the distributions of ϑ are multimodal with four peaks. The obvious observation that we obtain from both figures is the distributions share high similarity of patterns if they are associated with the same level of information, regardless of deception. When the information changes from being frequent-accurate to infrequent-noisy, we can observe changes in heights for the four peaks and the patterns are consistent across different levels of deceptions. From the sub-figures associated with frequent-accurate information regardless of deception, we can observe that peaks in the distribution have the highest heights. This means that ϑ of the *red* agent are highly concentrated at certain values, given that the information is frequent and accurate. As the frequently received information changes from being accurate to noisy, there is a marginal reduction in the peaks' heights overall. When evolution occurs upon receiving infrequent-accurate information, the best solutions resulting from such conditions are no longer highly concentrated at certain values and this is shown by much lower peaks in the distributions. Consequently, the distributions of having infrequent-accurate information are almost uniformly distributed. However, the introduction of noise into the infrequent information has encourages the best evolved solutions to have high preferences at actions. As a result, the best evolved solutions have higher pdf on certain values of ϑ in such condition, thus leading to the formation of four obvious peaks in the distributions. When paired comparisons are made between frequent and infrequent information for accurate and noisy information respectively, we can see that the differences in peaks' heights are clearer in

accurate information than noisy information. The *red* agent has much lower preferences for certain actions as the frequency of information reduces in accurate information than those in noisy information. In other words, the frequency of information has higher impact in constructing the preferences of actions in accurate information than noisy information.

Based on pdfs of $\Delta\vartheta$ shown in Figures 6.13 and 6.14, all of the pdf plots are unimodal, which is highly concentrated at the values near to zero regardless of information and deception. This means that there are hardly changes in the *red* agent's actions produced by the best evolved solutions. Through learning and evolving in neuroevolution, the best evolved solutions produce movements for the *red* agent against different strategies of the *blue* agent in such a way that substantial changes are not necessary.

The findings from the action distributions show that the machine behaviours resulting from neuroevolution are less influenced by deception. Even though the *blue* agent tries to confuse the *red* agent by being deceptive, the *red* agent controlled by neuroevolution is able to discover the dominant patterns in *blue*'s movements, and thus uses similar strategies to survive. However, the quality of the information influences the *red* agent's actions. The patterns shown in the distributions of ϑ suggest that the preferences of the *red* agent's action at certain values reduces in the order: frequent-accurate, frequent-noisy, infrequent-noisy and infrequent-accurate. When the frequency of information is high, the preferences of *red* agent for certain actions are less affected by the noise. However, if there is a delay in the information, an introduction of noise in the information causes the *red* agent to prefer certain actions as compared to accurate information. Besides that, we also notice that the differences in action preference are clearer upon having accurate information than noisy information, when paired comparisons are made between frequent and infrequent information. Given that the machine behaviours of the *red* agent are produced by neuroevolution, the contingent production of multiple solutions in evolution are by quality of information, but not the deception. Furthermore, the finding from the distributions of $\Delta\vartheta$ suggests that

evolution is able to produces best solutions that enable the *red* agent to act with minimal changes in its movements in spite of information and deception.

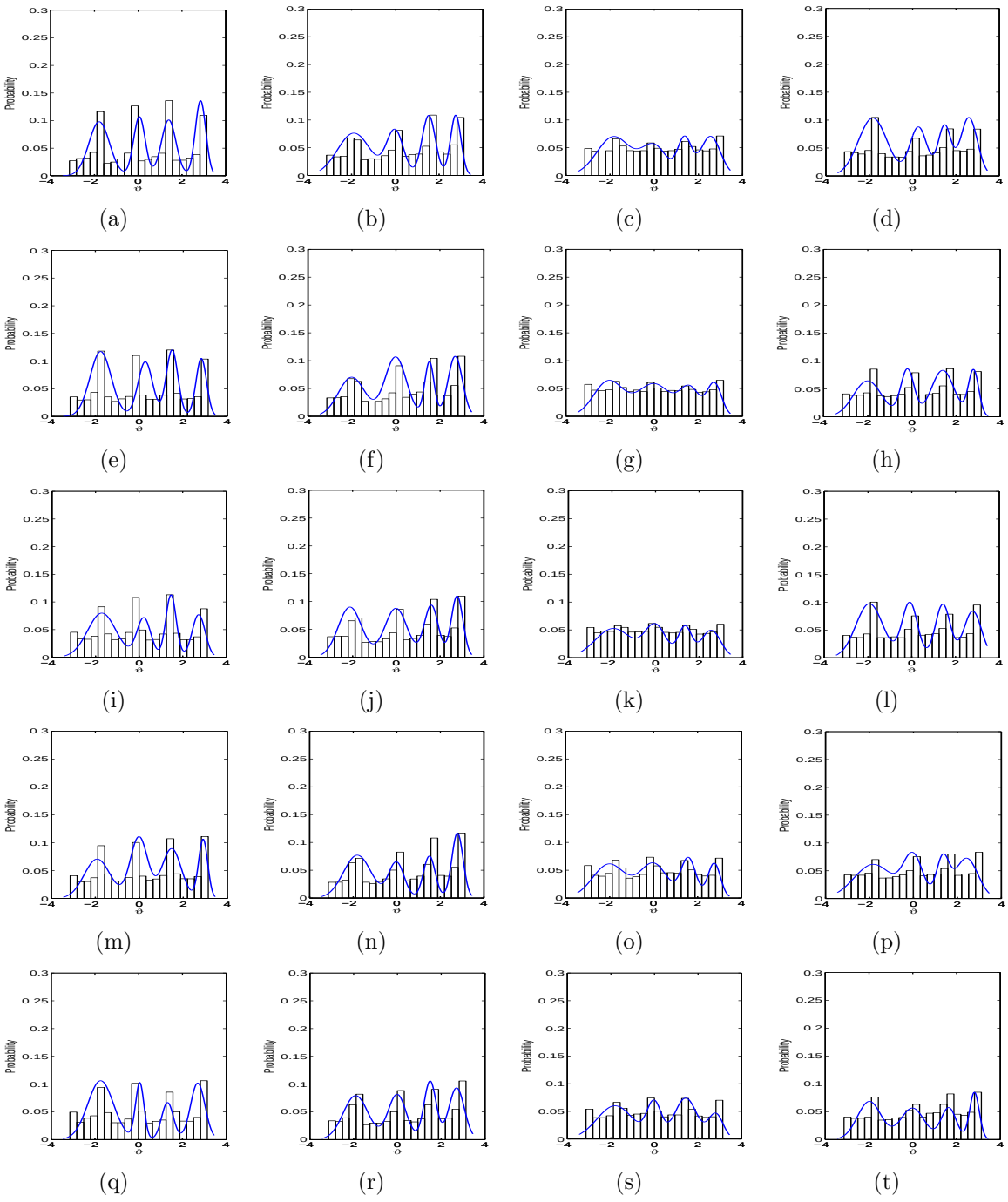


Figure 6.11: The plots of ϑ for various combinations of information and deception in known-unknown scenario.

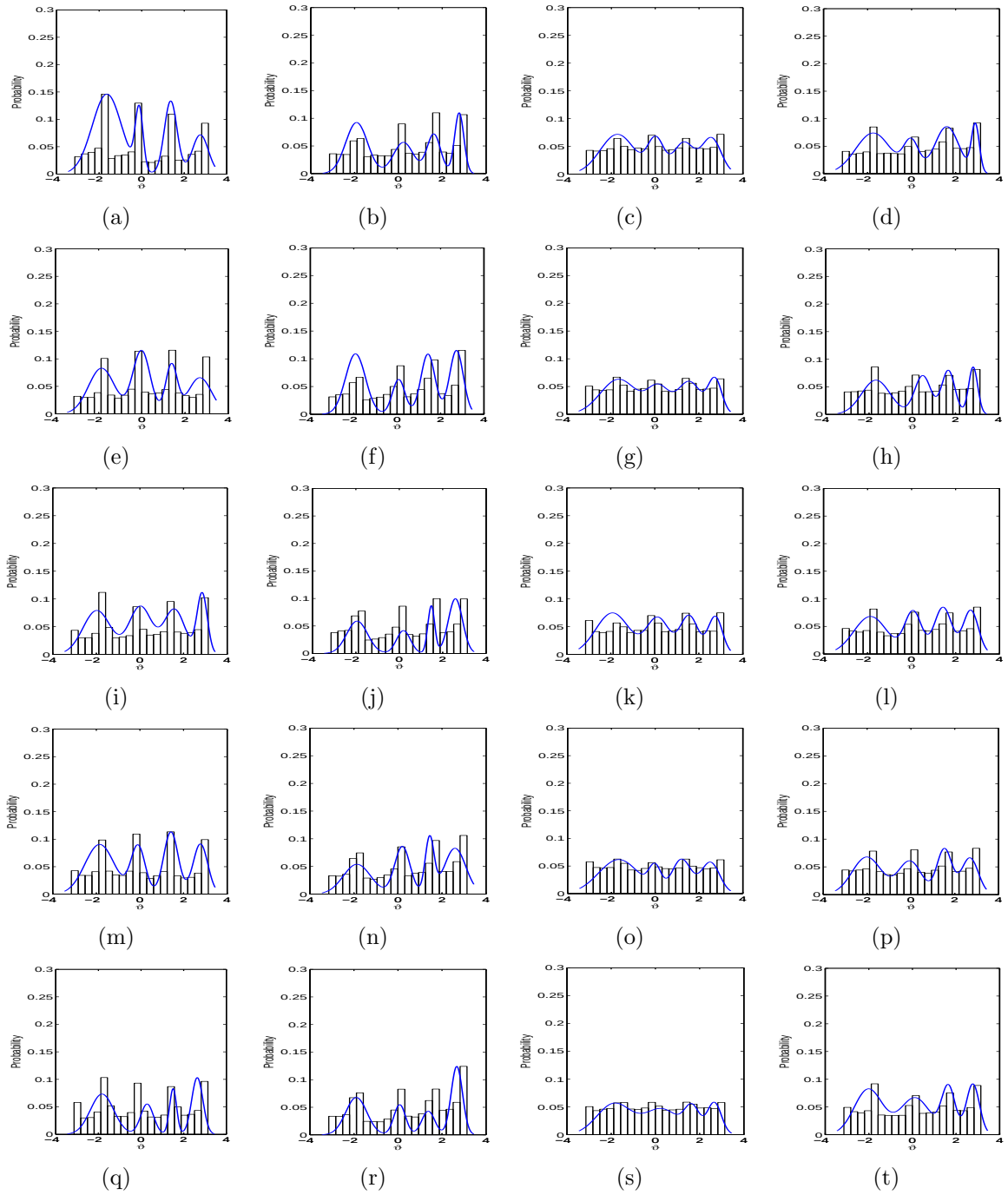


Figure 6.12: The plots of ϑ for various combinations of information and deception in known-known scenario.

6.5.3 Action Similarity

To understand the machine behaviours, actions taken by *red* and *blue* are described by the velocity of *red* relative to *blue* respectively, i.e., $V_{R\,rel\,B}$ and its change, $\Delta V_{R\,rel\,B}$. For each

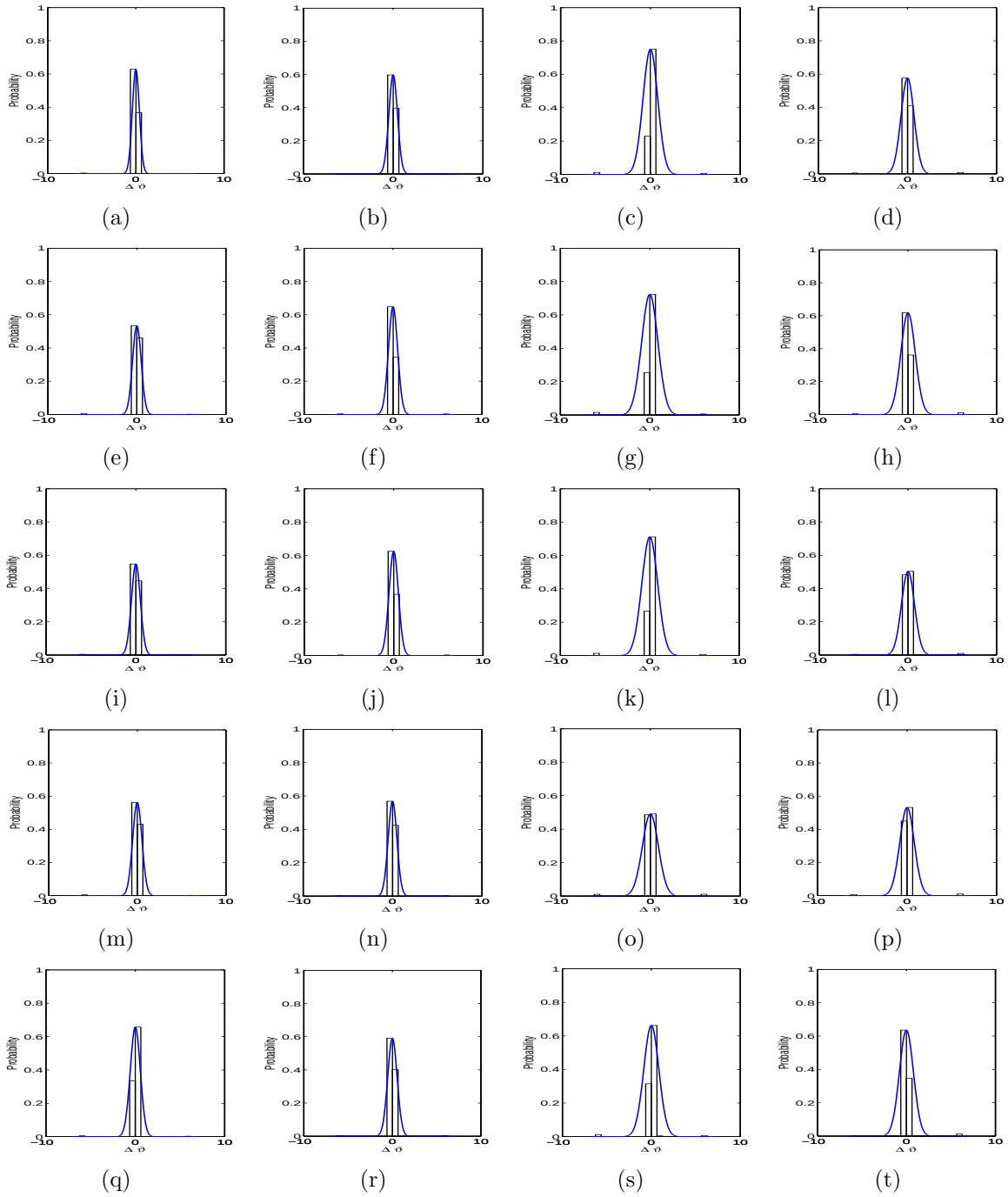


Figure 6.13: The plots of $\Delta\vartheta$ for various combinations of information and deception in known-unknown scenario.

combination of perception and deception, evolution is repeated by using 30 different seed numbers. The best solution for each seed is used by *red* facing *blue* for 10 repeated games.

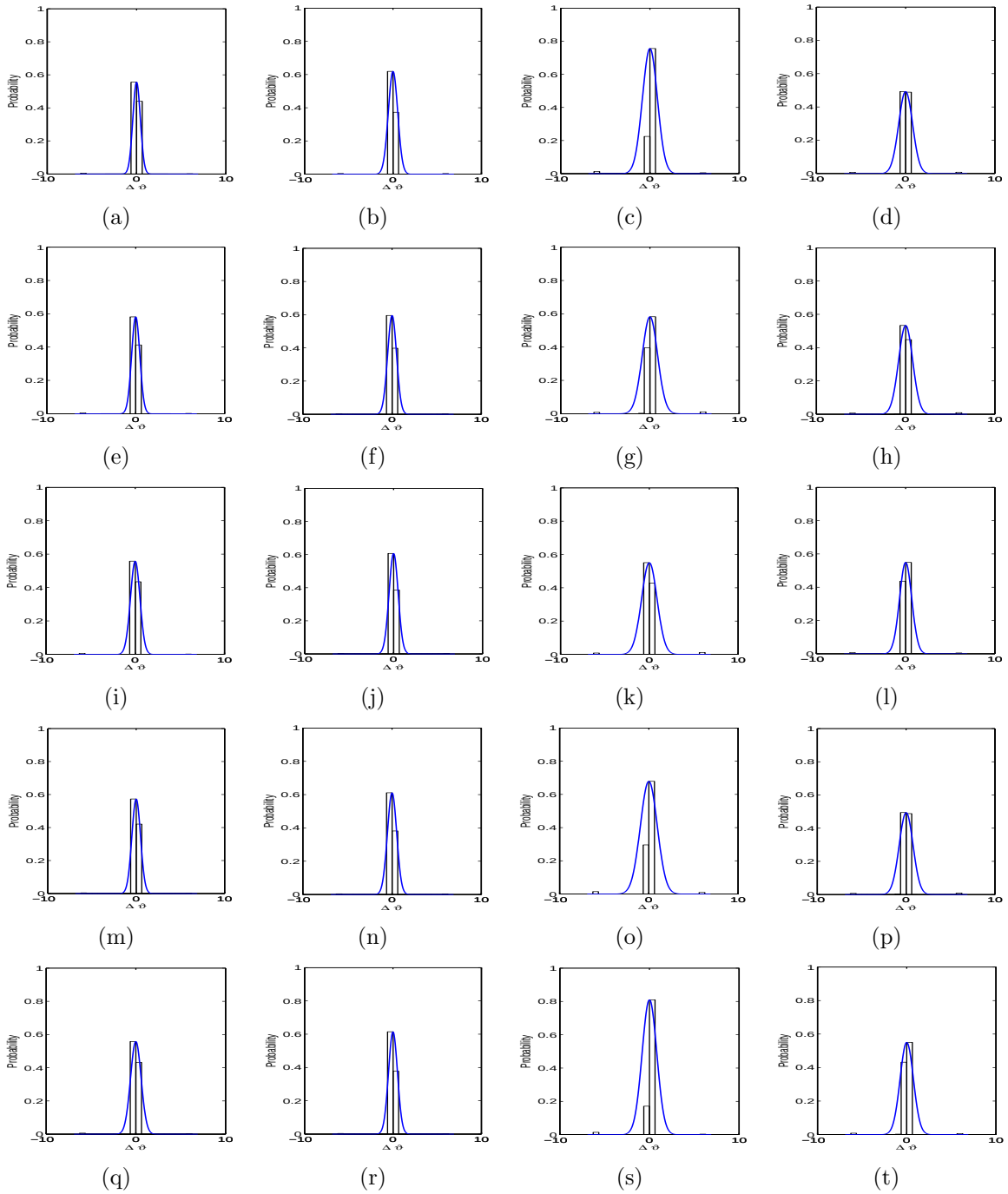


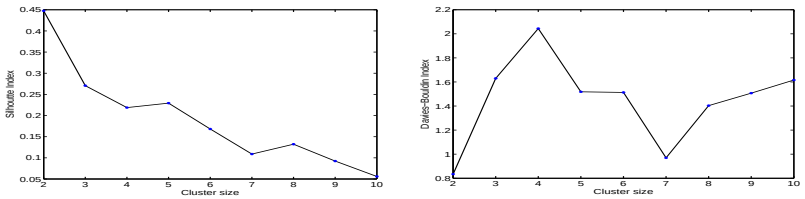
Figure 6.14: The plots of $\Delta\vartheta$ for various combinations of information and deception in known-known scenario.

Since the experiment involves 20 combinations of perception and deception, there are 6000 different action sequences in total.

Cluster analysis is used to categorise the action sequences so that we are able to extract similarities/differences for ease of understanding. However, the main drawback to perform cluster analysis is we need to determine in advance an appropriate cluster size. To solve this drawback, cluster validity indices can be a very useful as quantitative measurements for evaluating the quality of data partitions. These indices are endowed with particular features that make each of them most appropriate in specific classes of problem. Therefore, we use an ensemble that consists of several cluster validity indices instead of relying on one cluster validity index only. To form the ensemble, several well-known cluster validity indices based on internal criteria are used, i.e., Silhoutte, Davies-Bouldin, Calinski-Harabasz and Dunn. For all of the indices except Davies-Bouldin, higher index value indicates better cluster clustering performance. In contrast, lower index value means better clustering performance in the Davies-Bouldin method. The range of cluster size to be evaluated based on the methods is from 2 to 10.

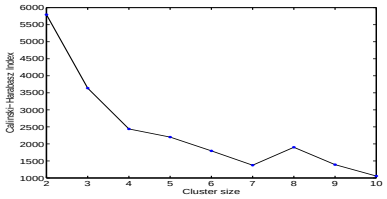
Figure 6.15 and 6.16 show the plots of cluster validity for the set of action sequences based on the proposed indices for both known-unknown and known-known scenarios. Both figures show that all of the cluster validity indices are associated with best values when the cluster size is 2. Therefore, we decide to perform clustering process by using cluster size of 2 in both scenarios and the cluster centroid is shown in Table 6.3.

Given that the cluster centroid describes the actions of *red* relative to *blue* based on velocity and acceleration, it may provide some clues on the strategies for both agents in both scenarios. Based on Table 6.3, the first to tenth features represent the differences of velocity between both agents, $V_{Rrel\,B}$ for ω_1 to ω_{10}. On the other hand, the eleventh to nineteenth features represent the differences of acceleration between both agents, $\Delta V_{Rrel\,B}$ for ω_2 to ω_{10}. For ease of interpretation, $V_{Rrel\,B}$ and $\Delta V_{Rrel\,B}$ versus window in the known-unknown and known-known scenarios are shown in Figure 6.17.

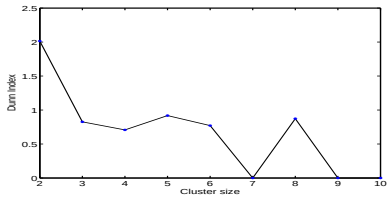


(a) Silhoutte Index

(b) Davies-Bouldin Index

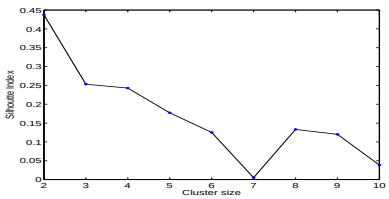


(c) Calinski-Harabasz Index

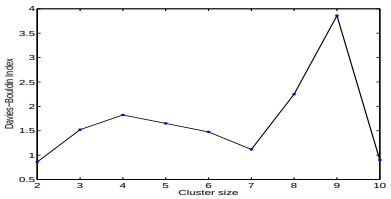


(d) Dunn Index

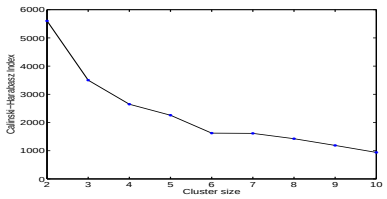
Figure 6.15: Cluster validity indices for the action sequences in the known-unknown scenario.



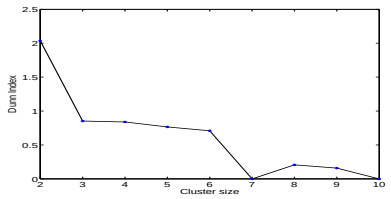
(a) Silhoutte Index



(b) Davies-Bouldin Index



(c) Calinski-Harabasz Index



(d) Dunn Index

Figure 6.16: Cluster validity indices for the action sequences generated in the known-known scenario.

Based on Figure 6.17, there are two different strategies in each scenario, which can be described by $V_{Rrel\,B}$ and $\Delta V_{Rrel\,B}$. It is interesting to find out that $V_{Rrel\,B}$ for both strategies in the known-unknown scenario are very similar to those in the known-known scenario. As for the measurement based on $\Delta V_{Rrel\,B}$, both strategies from the known-unknown and known-known scenarios also share high similarities, where strategy 2 from both scenarios are identical. As for strategy 1 in both scenarios, only some marginal differences can be

observed between them. Referring to Table 6.3, the difference based on Euclidean distance between the similar strategies from both scenarios are quite small, where the differences for strategies 1 and 2 are 0.2731 and 0.2458 respectively. Overall, the strategies in known-unknown and known-known scenarios are very similar to each other. This may suggest that the best evolved solutions produce similar machine behaviours regardless of knowledge availability of noise level. This again supports previous findings in action distribution, which suggests that the machine behaviours resultant from evolution are not affected by the extra task-relevant stimuli.

Table 6.3: Cluster centroid that represents the actions of *red* relative to *blue*.

Feature	Window	Centroid 1 / Strategy 1		Centroid 2 / Strategy 2	
		Known-unknown	Known-known	Known-unknown	Known-known
		1	2	1	2
$v_{Rrel\,B}$	ω_1	9.7350	9.6864	5.1761	5.2319
	ω_2	10.6030	10.5910	5.6861	5.6695
	ω_3	10.7830	10.6630	5.6966	5.8027
	ω_4	10.4380	10.4330	5.6434	5.8050
	ω_5	10.1430	10.2580	5.4967	5.5753
	ω_6	10.0850	10.1750	4.7920	4.8572
	ω_7	10.0440	10.0260	3.6245	3.6605
	ω_8	10.0020	10.0970	2.7445	2.8039
	ω_9	10.0510	10.0160	2.4442	2.4329
	ω_{10}	10.0690	9.9251	2.4856	2.5478
$\Delta v_{Rrel\,B}$	ω_2	0.1062	0.1107	0.0542	0.0471
	ω_3	0.0369	0.0325	0.0029	0.0165
	ω_4	-0.0158	-0.0052	-0.0045	0.0016
	ω_5	-0.0191	0.0018	-0.0145	-0.0223
	ω_6	0.0064	0.0087	-0.0682	-0.0698
	ω_7	0.0071	-0.0160	-0.1147	-0.1184
	ω_8	0.0137	0.0352	-0.0832	-0.0816
	ω_9	0.0067	-0.0356	-0.0256	-0.0326
	ω_{10}	0.0078	-0.0328	0.0110	0.0160
Difference	-	0.2731		0.2458	

Since the action sequences are actually produced by the best solutions from evolutionary process, we would like to know whether there is any significant difference of scores between these two strategies in both scenarios. Therefore, a t-test is carried out with 0.05 significance level to evaluate the null hypothesis that the samples come from both strategies with equal means, against the alternative that the means are unequal as shown in Table 6.4. The mean scores for strategies 1 and 2 are denoted as $\mu_{s1,\,ku}$, $\mu_{s2,\,ku}$ in the known-unknown scenario, and as $\mu_{s1,\,kk}$ and $\mu_{s2,\,kk}$ in the known-known scenario.

Based on Table 6.4, the result shows that there is enough evidence to reject the null

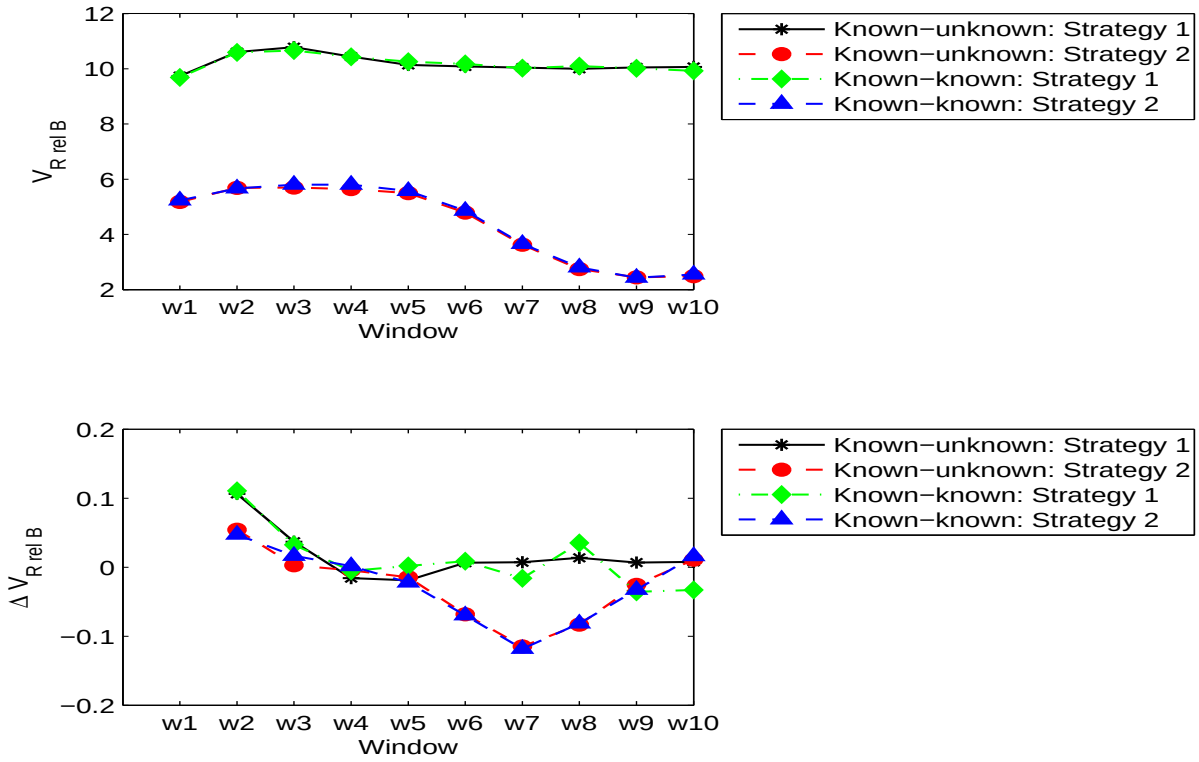


Figure 6.17: $V_{R\,rel\,B}$ and $\Delta V_{R\,rel\,B}$ for the action sequences generated in known-unknown and known-known scenarios.

hypothesis. Therefore, we can claim that the scores for both strategies are significantly different. The scores based on mean and standard deviation for both strategies are shown in Table 6.4. Since the the mean scores of strategy 2 are higher than strategy 1, we are further interested to test the null hypothesis that the score samples come from strategy 1 has greater or equal means as compared to strategy 2, against the alternative that the former strategy's mean is less than the later strategy's mean. Again, a t-test is carried out with 0.05 significance level to test the hypothesis and the significance test shows the null hypotheses for both scenarios are rejected. The results of the t-test support that the scores for strategy 1 are significantly lower than strategy 2. Despite different combinations of information and deception, the best evolved solutions can be categorised into two strategies, i.e., strategy 1 associated with lower score and strategy 2 associated with higher score. However, the scores for both strategies are still high, exceeding 90% in both scenarios.

Since the patterns in the actions taken by *blue* are influenced by the received information and its own deception, this means the evolution is also influenced by the effects of information and deception given that it learns from the patterns in *blue*'s actions. Despite the absence or presence of extra task-relevant stimuli in both scenarios, the best evolved solutions can be categorised into two main groups, which share high similarity between both scenarios. However, the main difference between these two strategies lies in the mean and standard deviation of their scores. In both scenarios, strategy 1 has lower mean and higher standard deviation, while strategy 2 is associated with higher mean and lower standard deviation. Besides that, we can observe that the frequency that better strategy generated by evolution is higher, and this is supported by the higher frequencies of strategy 2 in both scenarios.

Most of the strategies generated by the evolution are good and they share high similarity. The possible reason to explain high similarity among the strategies is that the evolution is able to extract the hidden patterns even though they may be affected by information and deception, and thus produces optimum solutions. For example, *blue*'s movement may become less obvious to *red* due to distraction in *blue*'s information and deception. Due to the interaction between *blue* and *red*, the evolution from the *red* agent uncovers the true intention of the *blue* agent and produces optimum solutions for the situation. Given that the true intention of the *blue* agent is to catch the *red* agent, similar strategies are generated by the evolution preventing the *red* agent from being caught. This means the evolution manages to capture the dynamics between both agents and produces strategies that are mostly still capable of selecting the correct target response. This enables the *red* agent to survive for longer.

Table 6.4: The results of t-test for the mean scores between strategies 1 and 2 in the known-unknown and known-known scenarios.

Scenario	Hypotheses	Reject H_0	p-value
Known-unknown	H_0: $\mu_{s1,ku} = \mu_{s2,ku}$, H_1: $\mu_{s1,ku} \neq \mu_{s2,ku}$	Yes	0.00
	H_0: $\mu_{s1,ku} \geq mu_{s2,ku}$, H_1: $\mu_{s1,ku} < \mu_{s2,ku}$	Yes	0.00
Known-known	H_0: $\mu_{s1,kk} = \mu_{s2,kk}$, H_1: $\mu_{s1,kk} \neq \mu_{s2,kk}$	Yes	0.00
	H_0: $\mu_{s1,kk} \geq \mu_{s2,kk}$, H_1: $\mu_{s1,kk} < \mu_{s2,kk}$	Yes	0.00

Table 6.5: Scores and frequencies for strategies 1 and 2 in the known-unknown and known-known scenarios.

Strategy	Known-unknown		Known-known	
	Score, $Mean \pm Stdev$	Frequency	Score, $Mean \pm Stdev$	Frequency
Strategy 1	93.2±20.5	2001 (33.4%)	90.3±24.1	2014 (33.6%)
Strategy 2	97.3±9.8	3999 (66.6%)	97.2±9.9	3986 (66.4%)

6.6 Conclusion

This study focuses on two central elements of decisions: the use of multiple strategies in CRT, and the factors that influence the use of strategies. In our study, the use of multiple strategies is demonstrated by the generation of diverse solutions by neuroevolution. On the other hand, the factors that influence the preferences of strategies are demonstrated by different configurations of information and deception. Two scenarios called known-unknown and known-known, are simulated to investigate the effect of perceptual load in CRT since it influences human behaviour in HRT.

The study shows that the machine behaviours in the known-unknown and known-known scenarios are very similar to each other, based on action distribution and action similarity. Besides that, both scenarios also have similar scores. This means there is no clear evidence to show that the availability of extra task-relevant stimuli, which is represented by the noise level, influences the construction of behaviour and outcome of behaviour in CRT. The analysis based on action distribution suggests that the contingent production of machine behaviour in neuroevolution depends on the quality of information but not the deception.

Despite different configurations of information and deception, evolution is able to learn the dominant patterns of *blue*'s movements and evolve the best solutions, where the actions generated by the solutions experience minimal changes. In other words, the strategies produced by the best evolved solutions are adequate to enable the *red* agent to survive without having to experience obvious changes in its movements. In short, characteristics of the task such as information can evoke strategies that at least partially determine the preferences of actions we observe in machine behaviour.

Chapter 7

Neural Networks Approximating Human Behaviour

7.1 Introduction

Neuroevolution is selected as the computational agent owing to its ability to approximate human behaviours, which depends on both evolution and learning. In this chapter, we would like to take the opportunity to understand the possible differences in machine behaviour given that only learning occurs in the computational model. This is because most of the defence systems used in adversarial learning or secure learning use this approach, where the systems are built through learning historical data. To perform this investigation, pure neural networks without evolution are used to learn from the data sets that have been collected from the HRT experiments. In this case, the neural networks can be viewed as fitting human behaviour. Besides that, we are interested to investigate the impact of information on machine behaviour by varying the data sets, which can be categorised into two types: general and specific data sets. In the general data, the neural networks learn

from a data set consisting of information about all players. On the other hand, the neural networks are fitting the behaviours of individual players if they are given a specific data set. In short, the neural network can be learning from either a collection of generalised or individualised human behaviours.

7.2 Experimental Designs

Similar to the experimental setup in previous chapters, the same 20 configurations of information and deception in the known-unknown and known-known scenarios are used in the experiments. Basically, two types of neural networks are built in the experiments, which can be known as generalised and specific neural networks. The difference between them lies in the types of data used in their learning phase and their experimental designs are explained in details in Sections 7.2.1 and 7.2.2.

7.2.1 Generalised Neural Networks

In HRT as shown in chapter 5, humans participated in the games as the *red* agent. The human players played against the *blue* agent deploying different strategies in both known-unknown and known-known scenarios. Given that the locations of both the *red* and *blue* agents are recorded in both scenarios, we can use the collected information to determine the relevant input and output variables in the CRT experiments, as follows:

- Input variables:
    1. the relative angle between the *blue* and *red*, $\beta^{(t)}$.
    2. the relative distance between the *blue* and *red* agents at time t, $d_{opp}^{(t)}$.
    3. the relative change of β at time t, $\Delta\beta^{(t)}$.

4. the relative change of d_{opp} at time t, $\Delta d_{opp}^{(t)}$.

5. the relative distance to the wall that the *red* agent faces at time t, $d_{wall}^{(t)}$.

- Decision variables:

1. travel angle of the human *red* agent from the *blue* agent, $\theta_{hr}^{(t)}$

Since there are 34 human players participating in the games, the above mentioned input variables are calculated for all players and combined into a data set for each configuration. For each configuration, a neural network with the same architecture and learning parameters are used to learn from the data set. A multilayer perceptron consisting of 5 input neurons, 7 hidden neurons, and 2 output neurons is used in the simulation. The learning rate η and the momentum rate γ are set to be 0.2 and 0 respectively. The neural network learns to predict the travel angle of the human players based on the provided inputs. The data set is further divided into training set and validation set based on ratio of 80% and 20%.

Mse of the validation set is used as the stopping criterion in the network training. For each 700 iterations, *mse* of the validation set is monitored and its connection weights are recorded until it reaches the maximum number of iterations 7,000,000. Based on monitoring, we would like to identify the iteration where the validation set produces the minimum *mse* and thus, its connection weights are used to build the neural network. For the same configuration, the trained network's performance is then evaluated by playing the games for 10 times. During the games, if the travel angle predicted by the neural network does not cause the *red* agent to get to an edge, the *red* agent will move according the predicted travel angle. Otherwise, an angle deviation will be added to the predicted travel angle so that the agent can escape from the edge. The assumption made here is a human will avoid to be trapped at an edge. The trajectories from previous HRT show such avoidance in

human movements. In this experiment, the neural networks learn from a collection of information consisting of a diversified of set of human behaviours. As a result, the behaviour model approximated by neural networks is generalised. Each combination of information and deception parameters was run 30 times using different seeds. The trajectories of the 10 repeated games for 30 runs using different seeds are analysed based on action distribution and action similarity. The above procedures are repeated for both scenarios, i.e., known-unknown and known-known.

7.2.2 Individualised Neural Networks

Different from the experimental setup in section 7.2.1, a neural network learns from a data set generated by an individual player instead of all players. Therefore, we will have 34 individual neural networks approximating the players' behaviours for each configuration. The same architecture and learning parameters are used to build the neural networks. Given the individual player has smaller data size as compared to those containing all players, the maximum iteration to be used to train neural networks is 200,000. Again, each data set belonging to an individual player is split into training set and validation set according to the ratio of 80% and 20%. For each 200 iterations, *mse* of the validation set is monitored and its connection weights are recorded until the network reaches the maximum iterations.

From the learning phase, the connection weights associated with the lowest *mse* for the validation set are used to build the neural network, which will be used to determine the *red* agent's travel angles in 10 repeated games. While playing the games, the same rule which prevents the *red* agent from being trapped at an edge is implemented. If the predicted travel angle doesn't cause the *red* agent to get stuck at an edge, it will be used. Otherwise, a small deviation of angle is added to the predicted travel angle. In this experiment, the behaviour model resulting from each neural network approximates each

individual player. For each configuration, 30 runs based on different seeds are carried out. The trajectories between the *red* and *blue* agents from these 300 runs are then analysed based on actions' distribution and actions' similarities. The procedures are implemented for the known-unknown and known-known scenarios.

Given that 300 trajectories are produced by each individual neural network, there are 12,600 trajectories produced for the 34 neural networks. The same analysis need to be performed for all the 20 configurations in both scenarios. Instead of analysing the behaviours produced by the networks fitting individual players, clustering analysis is performed on the neural networks based on the connection weights for each configuration. Cluster validity indices assist in determining the appropriate cluster size. As a result, the neural networks in each configuration are categorised into several clusters based on the similarity of the connection weights. The same clustering procedures are implemented in the known-unknown and known-known scenarios. Owing to this, both scenarios have a number of groups of neural networks for the same configuration and the centroid of the cluster can be viewed as the representation of a particular group of neural networks. Then, paired comparisons based on Euclidean distance are calculated for the centroid of the same configuration between both scenarios. Here, we would like to find the paired centroid associated with the highest distance, which means the neural networks in both scenarios are very different from each other for the same configuration. The assumption that we made here is the neural networks associated with the highest difference in connection weights will produce different behaviours and we would like to investigate it. Once the paired centroid have been identified, 30 neural networks located nearest to those paired centroid are selected to play 10 repeated games. In short, we have 30 neural networks from both scenarios for the same configuration. The trajectories between the *red* and *blue* agents resulting from the selected neural networks are then analysed based on actions' distribution and actions' similarities.

7.3 Result and Analysis

Machine behaviours resulting from the neural networks are analysed based on actions' distribution and actions' similarities. The discussions on the analysis are carried out using generalised and individualised neural networks respectively as shown in Sections 7.3.1 and 7.3.2. For the figures related to the distributions of ϑ and $\Delta\vartheta$, they are plotted in such a way that:

- The sequence of information in the same row from left to right: frequent-accurate, frequent-noisy, infrequent-accurate, infrequent-noisy;
- The sequence of deception in the same column from top to bottom: none, infrequent-low, infrequent-high, frequent-low, frequent-high.

7.3.1 Generalized Neural Networks

When comparisons are made between the neural networks which learn from both scenarios for the same configuration, we find out that there are not much differences in terms of ϑ and $\Delta\vartheta$. This means the human behaviours approximated by the neural networks in both scenarios are very similar to each other. Based on Figures 7.3 and 7.2, it is interesting to see that the heights of peaks for infrequent information are higher than those of frequent information regardless of deception. It seems the machine behaviours are influenced by the frequency of receiving information, which causes the distributions to be highly concentrated at certain values of ϑ given that there is delay in information. Having higher frequency of information means that the *blue* agent updates its knowledge about the *red* agent frequently, and thus, influences its movements frequently as well. In such condition, the *red* agent needs to adapt by varying its actions in order to survive. As a result, the values of ϑ are distributed more evenly as compared to those of having infrequent information.

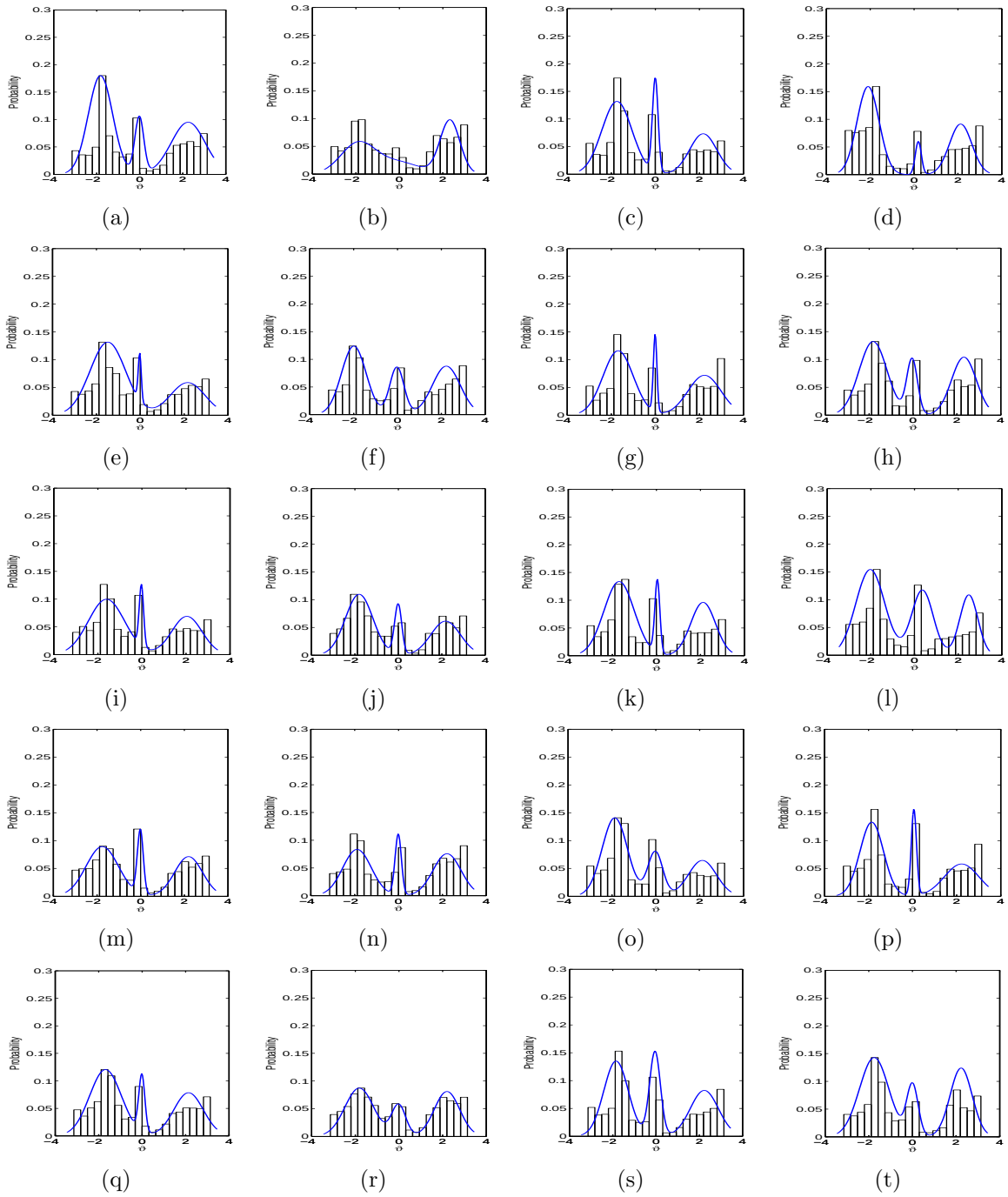


Figure 7.1: The plots of ϑ for various combinations of information and deception in known-unknown scenario based on the generalised neural networks.

A delay in the information received by the *blue* agent prevents it from catching the *red* sooner. Therefore, it is unlikely that the *red* agent needs to vary its actions so much, which

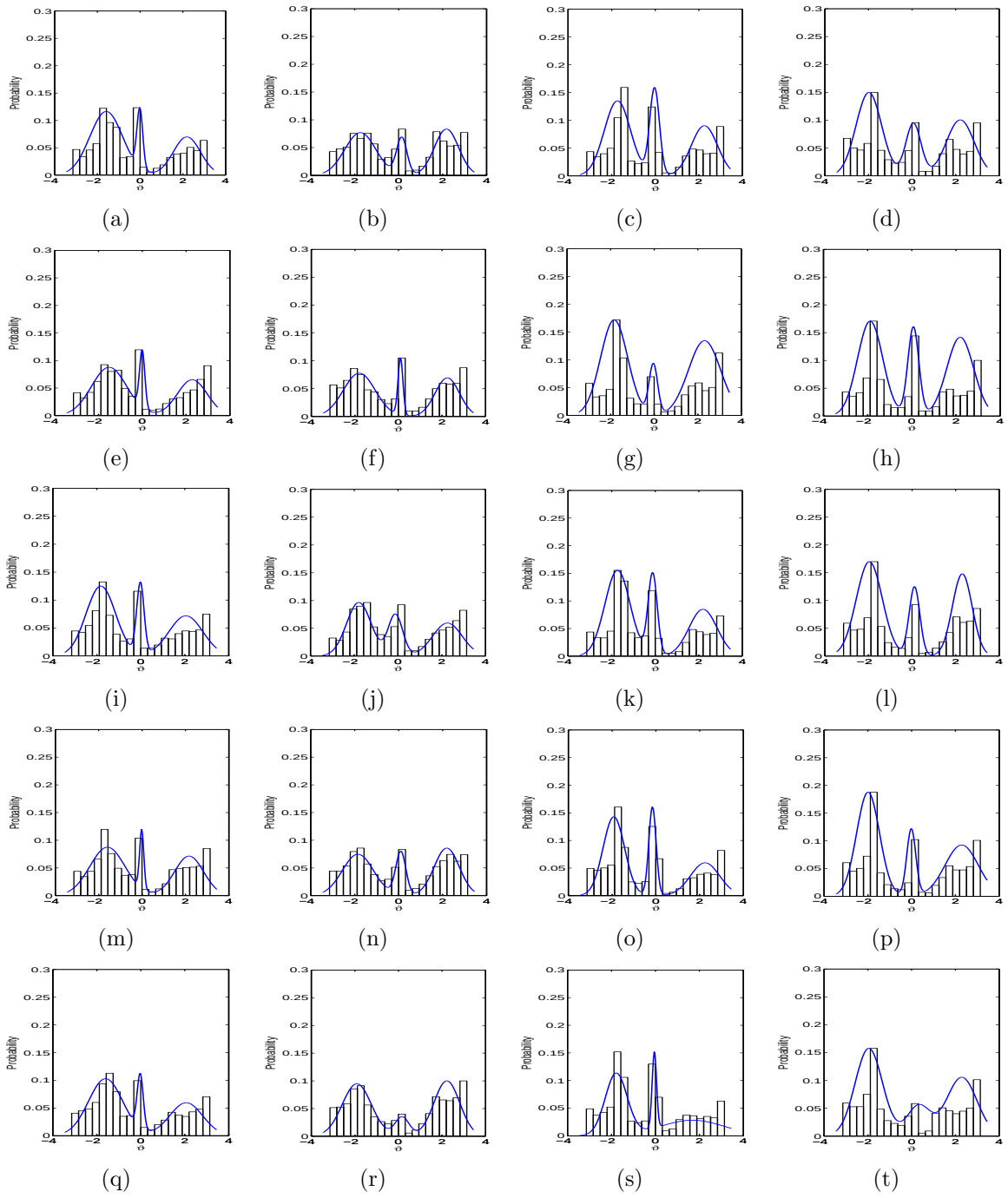


Figure 7.2: The plots of ϑ for various combinations of information and deception in known-known scenario based on the generalised neural networks.

also explains the observation of having high peaks in the distributions of ϑ. Besides that, the observation of having $\Delta\vartheta$ highly distributed around zero as shown in Figures 7.3 and

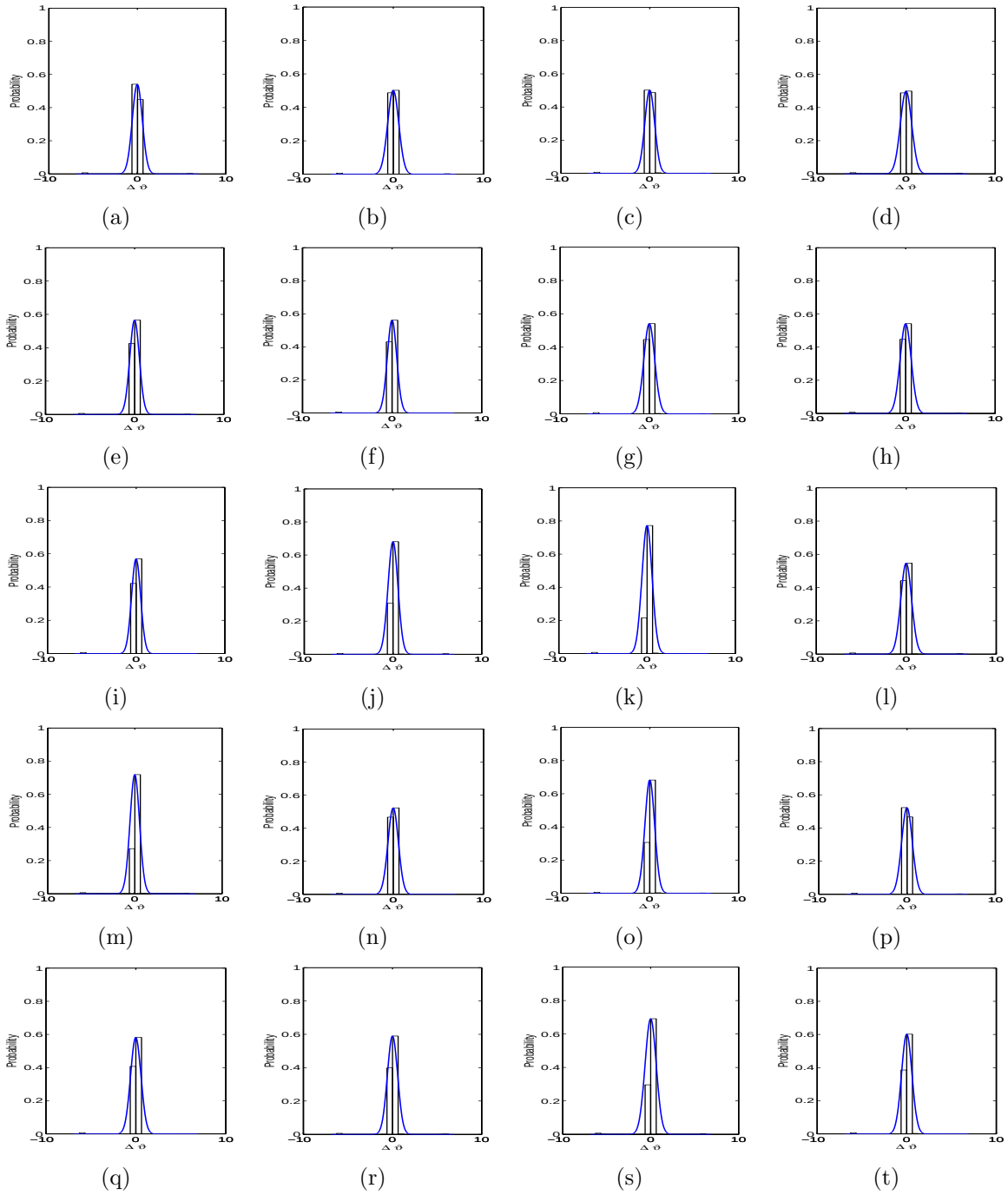


Figure 7.3: The plots of $\Delta\vartheta$ for various combinations of information and deception in known-unknown scenario based on the generalised neural networks.

7.4 indicate that the *red* agent makes marginal changes in its actions.

At the same time, we are interested to know whether there is any significant differ-

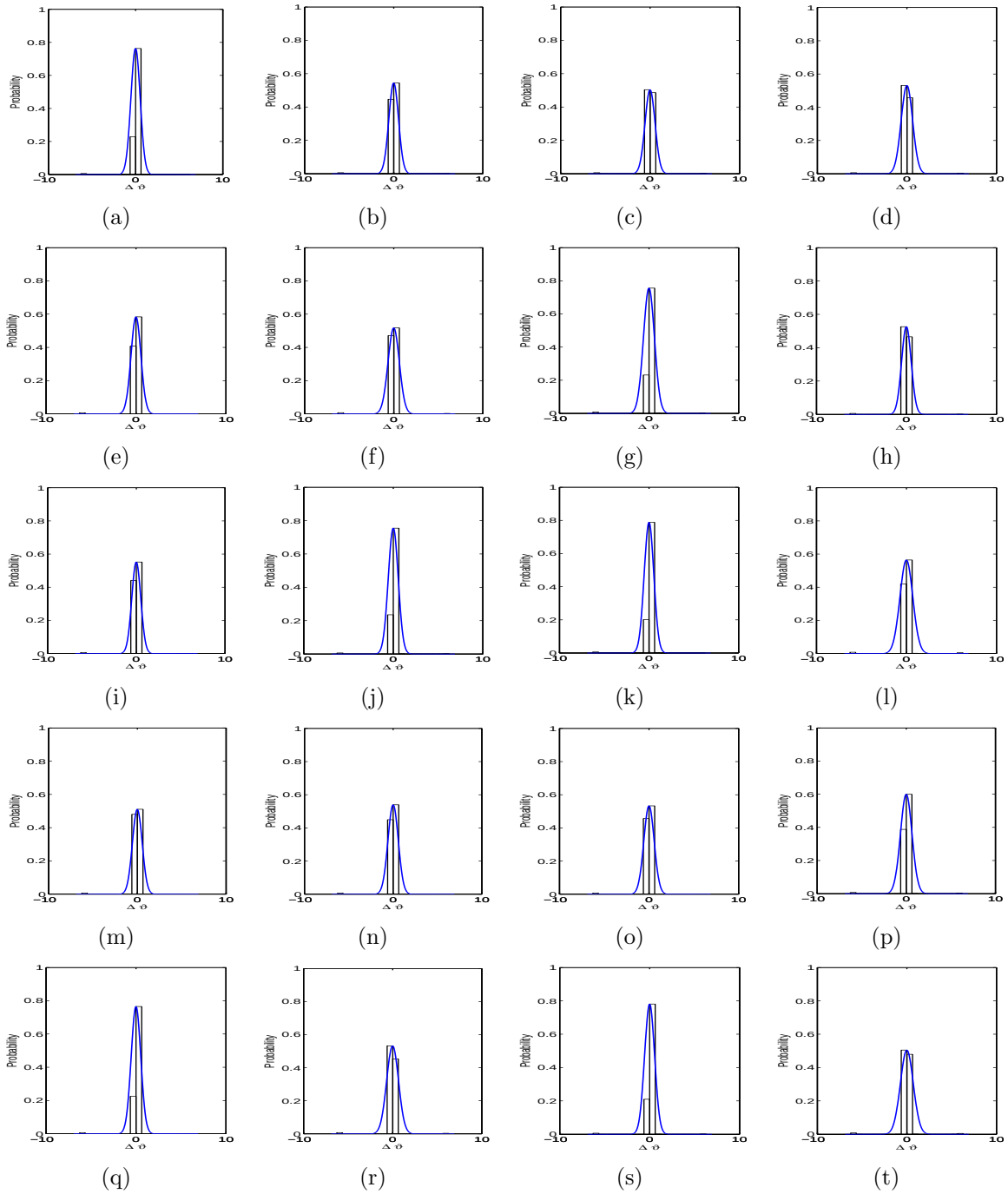


Figure 7.4: The plots of $\Delta\vartheta$ for various combinations of information and deception in known-known scenario based on the generalised neural networks.

ence of scores between them for the same combinations of information and configuration. Therefore, a t-test is carried out with 0.05 significance level to test the null hypothesis that

the samples come from both scenarios with equal means, against the alternative that the means are unequal. Table 7.1 shows the result of t-test on the scores between the known-unknown and known-known scenarios for the generalised neural networks. The significance tests in Table 7.1 show that 70% of the null hypotheses are not rejected. Since there is no significant difference of scores between both scenarios in most configurations, it is likely that the generalised neural networks producing similar behaviours in both scenarios, have similar outcomes as well.

Table 7.1: t-test for scores between the known-unknown and known-known scenarios in CRT based on generalised neural networks.

N_I	$\hat{\alpha}^{(t)}$	N_D	$\zeta^{(t)}$	Scores	
				Reject H_0	p-value
1	0	1	0	No	0.1533
		5	$U(-15°, 15°)$	No	0.6678
		5	$U(-30°, 30°)$	Yes	0.0012
		10	$U(-15°, 15°)$	No	0.0654
		10	$U(-30°, 30°)$	Yes	0.0093
1	$U(0, 20)$	1	0	No	0.7050
		5	$U(-15°, 15°)$	Yes	0.0075
		5	$U(-30°, 30°)$	No	0.8839
		10	$U(-15°, 15°)$	No	0.1345
		10	$U(-30°, 30°)$	No	0.6765
10	0	1	0	Yes	0.0003
		5	$U(-15°, 15°)$	Yes	0.0009
		5	$U(-30°, 30°)$	No	0.3505
		10	$U(-15°, 15°)$	No	0.2210
		10	$U(-30°, 30°)$	No	0.1733
10	$U(0, 20)$	1	0	No	0.1808
		5	$U(-15°, 15°)$	No	0.1543
		5	$U(-30°, 30°)$	Yes	0.0013
		10	$U(-15°, 15°)$	No	0.1857
		10	$U(-30°, 30°)$	No	0.1962

Clustering analysis is carried out using the machine behaviours in both scenarios to investigate their similarities. The cluster validity indices based on Silhoutte, Davies-Bouldin, Calinski-Harabasz and Dunn are used to determine the appropriate cluster size for the range of [2 10]. Then, the performances of the cluster sizes are ranked according to the indices. Cluster sizes associated with better clustering performance is given lower rank and vice versa. The ranking results for both scenarios are shown in Table 7.2. Based on the results, the appropriate sizes to be used in the known-unknown and known-known scenarios

are 2 and 4 respectively.

Table 7.2: Ranking of cluster size based on cluster validity indices in the known-unknown and known-known scenarios.

Scenario	Index	Cluster size								
		2	3	4	5	6	7	8	9	10
Known-unknown	Silhoutte	1	2	4	7	3	6	5	8	9
	Davis-Bouldin	6	5	8	3	2	9	1	4	7
	Calinski-Harabasz	1	2	3	6	4	7	5	9	8
	Dunn	1	2	4	3	6	5	7	8	9
	Average	**2.25**	2.75	4.75	4.75	3.75	6.75	4.5	7.25	8.25
Known-known	Silhoutte	1	3	2	6	7	9	4	5	8
	Davis-Bouldin	5	4	1	7	6	2	8	3	9
	Calinski-Harabasz	1	3	2	5	6	8	4	7	9
	Dunn	2	1	3	5	4	7	6	8	9
	Average	2.25	2.75	**2**	5.75	5.75	6.5	5.5	5.75	8.75

For ease of visualisation, the centroid for both scenarios are shown in Figure 7.5, where they are plotted based on $V_{Rrel\,B}$ and $\Delta V_{Rrel\,B}$ separately. According to the figure, each scenario is associated with strategies sharing very high similarities. Besides that, the comparison made between both scenarios also shows that their machine behaviours are similar, with some marginal differences in terms of $V_{Rrel\,B}$ and $\Delta V_{Rrel\,B}$. In both scenarios, $V_{Rrel\,B}$ reduces with the increase of window, which means the *blue* agent is moving towards the *red* agent with different speeds when it is viewed from the *red* agent.

7.3.2 Individual Neural Networks

Figures 7.6 to 7.9 belong to individual neural networks, fitting individual players who behave differently in both scenarios for the same configuration. There are only marginal changes that can be observed in $\Delta\vartheta$, as shown in Figures 7.8 and 7.9. In general, the human players hardly make substantial changes in their movements even though they behave differently. When a paired comparison is made between the distributions of ϑ between Figures 7.6 and 7.7, there is lack of consistency in the distributions for the same configuration. A similar observation is obtained across different configurations. This interesting finding

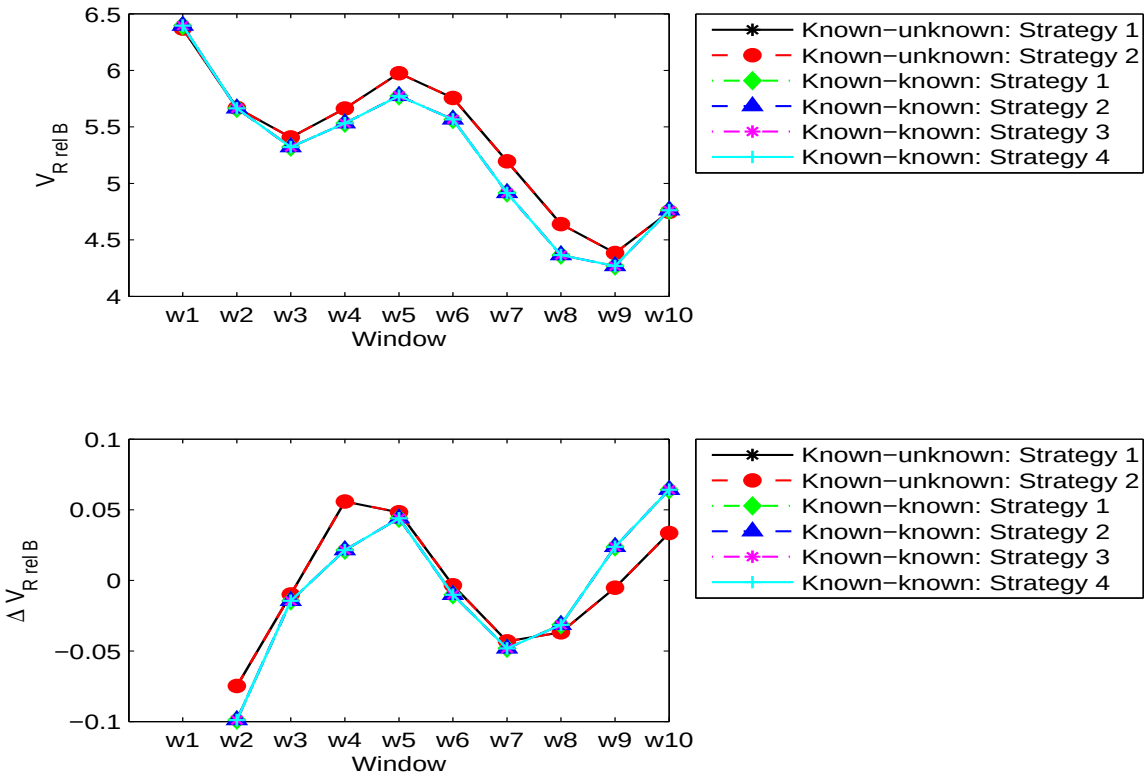


Figure 7.5: $V_{R\,rel\,B}$ and $\Delta V_{R\,rel\,B}$ in the known-unknown and known-known scenarios based on the generalised neural networks.

has demonstrated the diversity of human behaviours in the game environment. Given that each neural network learns based on individual player's data, the machine behaviours actually reflect the individual player's behaviours. In other words, we have a set of diversified neural networks, which behave differently as well.

Similar to the generalised neural networks, the difference between both scenarios for the individualised neural networks is made based on their scores for the same configuration. A t-test is carried out with 0.05 significance level to test the null hypothesis that the samples come from both scenarios with equal means, against the alternative that the means are unequal. Table 7.3 shows the result of t-test on the scores between the known-unknown and known-known scenarios for the individual neural networks.

The results shown in Table 7.3 show that 70% of the null hypotheses are rejected. This

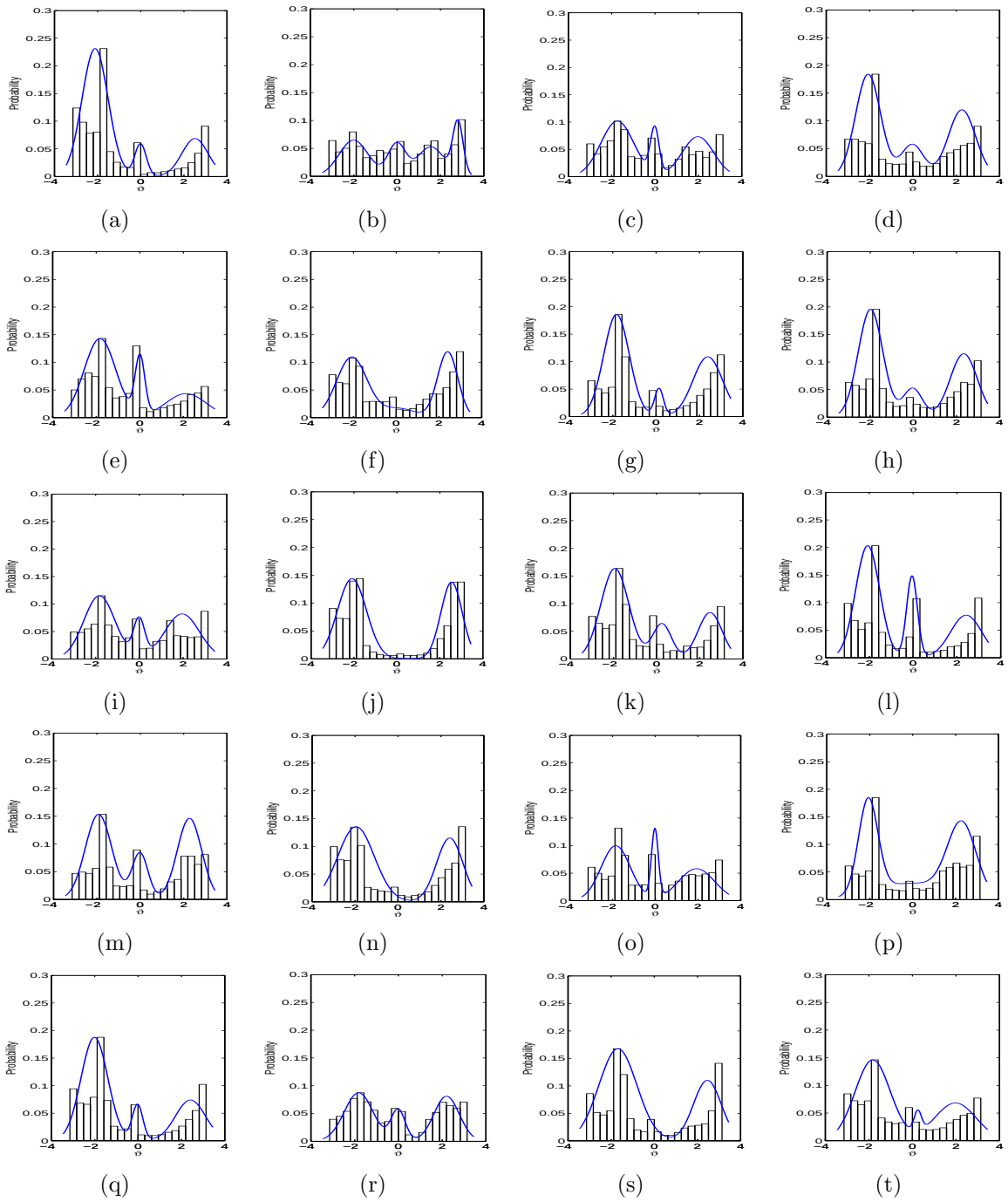


Figure 7.6: The plots of ϑ for various combinations of information and deception in known-unknown scenario based on the individual neural networks.

means in most of the configurations, there is a significant difference of scores between the known-unknown and known-known scenarios, which also suggests that different behaviours

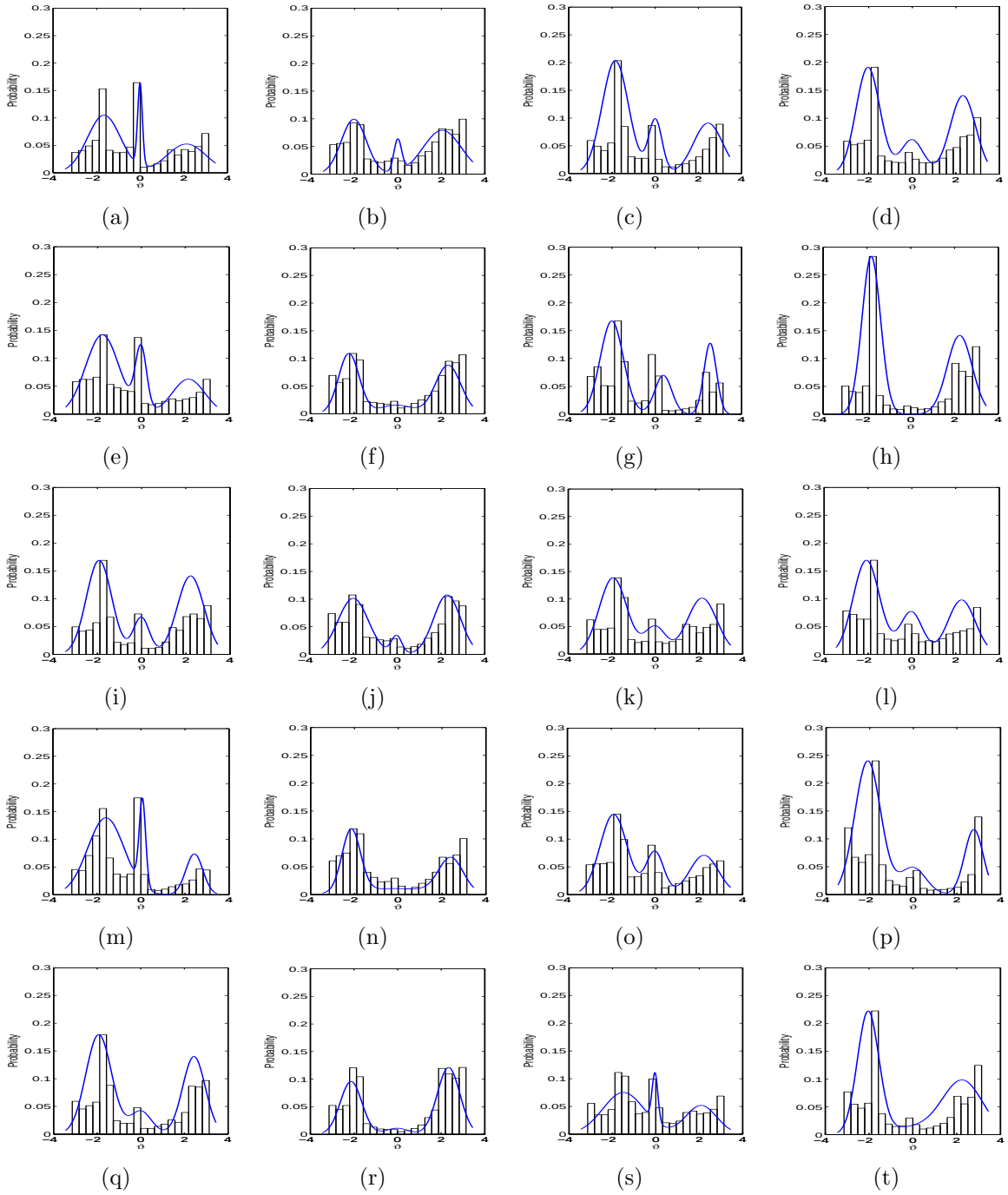


Figure 7.7: The plots of ϑ for various combinations of information and deception in known-known scenario based on the individual neural networks.

in the individual neural networks lead to the difference in outcomes.

Clustering analysis is carried out on the machine behaviours in both scenarios for the

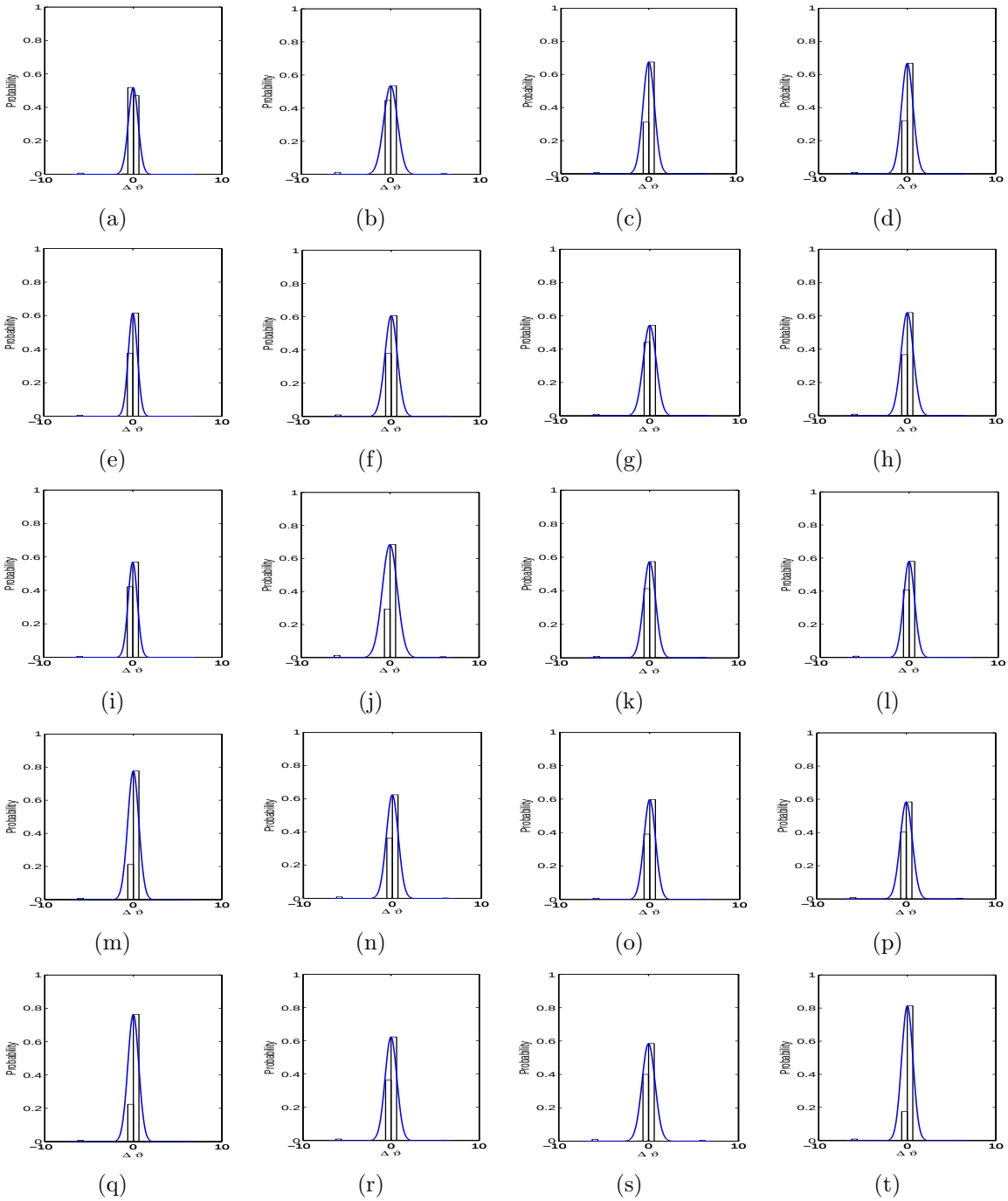


Figure 7.8: The plots of $\Delta\vartheta$ for various combinations of information and deception in known-unknown scenario based on the individual neural networks.

individual neural networks. The cluster validity indices based on Silhoutte, Davies-Bouldin, Calinski-Harabasz and Dunn are used to determine the appropriate cluster size for the range

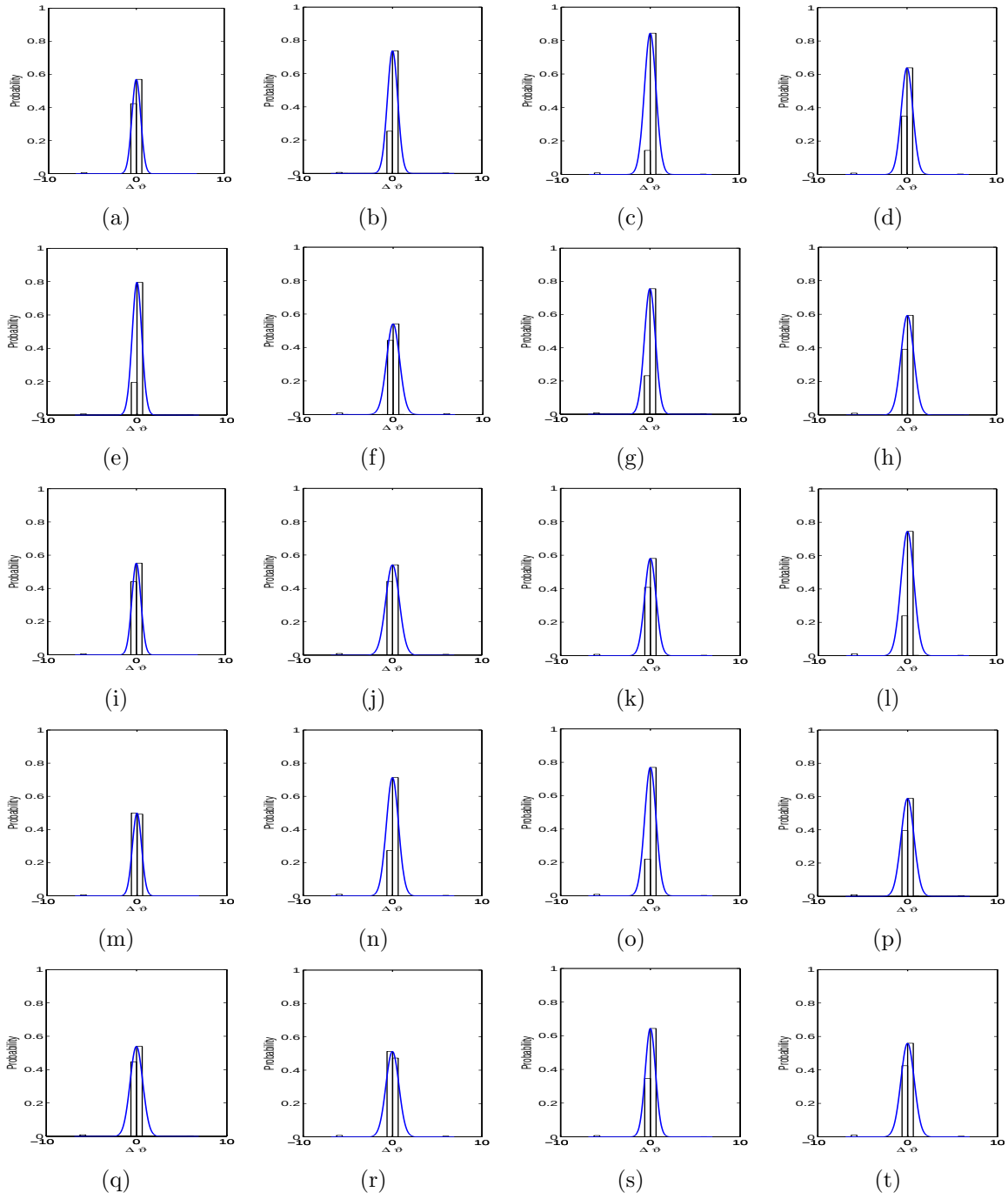


Figure 7.9: The plots of $\Delta\vartheta$ for various combinations of information and deception in known-known scenario based on the individual neural networks.

of [2 10]. Then, the performances of the cluster sizes are ranked according to the indices and the ranking results are shown in Table 7.4. Based on the results, the appropriate sizes

Table 7.3: t-test for scores between the known-unknown and known-known scenarios in CRT based on individualised neural networks.

N_I	$\hat{\alpha}^{(t)}$	N_D	$\zeta^{(t)}$	Scores	
				Reject H_0	p-value
1	0	1	0	Yes	0.0000
		5	$U(-15°, 15°)$	No	0.3947
		5	$U(-30°, 30°)$	Yes	0.0093
		10	$U(-15°, 15°)$	No	0.7046
		10	$U(-30°, 30°)$	No	0.2007
1	$U(0, 20)$	1	0	No	0.4045
		5	$U(-15°, 15°)$	No	0.6413
		5	$U(-30°, 30°)$	Yes	0.0001
		10	$U(-15°, 15°)$	Yes	0.0115
		10	$U(-30°, 30°)$	Yes	0.0003
10	0	1	0	Yes	0.0009
		5	$U(-15°, 15°)$	Yes	0.0432
		5	$U(-30°, 30°)$	Yes	0.0001
		10	$U(-15°, 15°)$	No	0.8921
		10	$U(-30°, 30°)$	Yes	0.0015
10	$U(0, 20)$	1	0	Yes	0.0011
		5	$U(-15°, 15°)$	Yes	0.0000
		5	$U(-30°, 30°)$	Yes	0.0069
		10	$U(-15°, 15°)$	Yes	0.0000
		10	$U(-30°, 30°)$	Yes	0.0000

to be used in the known-unknown and known-known scenarios are 2 and 3 respectively.

Table 7.4: Ranking of cluster size based on cluster validity indices in the known-unknown and known-known scenarios for the generalised neural networks.

Scenario	Index	Cluster size								
		2	3	4	5	6	7	8	9	10
Known-unknown	Silhoutte	1	2	7	3	4	5	8	6	9
	Davis-Bouldin	2	3	1	4	5	8	7	9	6
	Calinski-Harabasz	1	2	3	4	5	6	7	8	9
	Dunn	1	3	2	4	5	6	7	8	9
	Average	**1.25**	2.5	3.25	3.75	4.75	6.25	7.25	7.75	8.25
Known-known	Silhoutte	1	2	3	4	8	5	6	7	9
	Davis-Bouldin	9	3	7	8	5	1	2	4	6
	Calinski-Harabasz	1	2	3	4	5	6	7	8	9
	Dunn	1	2	3	4	5	6	7	8	9
	Average	3	**2.25**	4	5	5.75	4.5	5.5	6.75	8.25

The findings in the analysis of human behaviours based on $V_{Rrel\,B}$ and $\Delta V_{Rrel\,B}$ show that there are high similarities in the collection of individual neural networks in both scenarios. The explanation for this finding is the collection of machine behaviours resulting from individual neural networks can be viewed as an ensemble. Owing to this, a more generalised behaviour is formed as the diversified behaviours of the individual neural networks

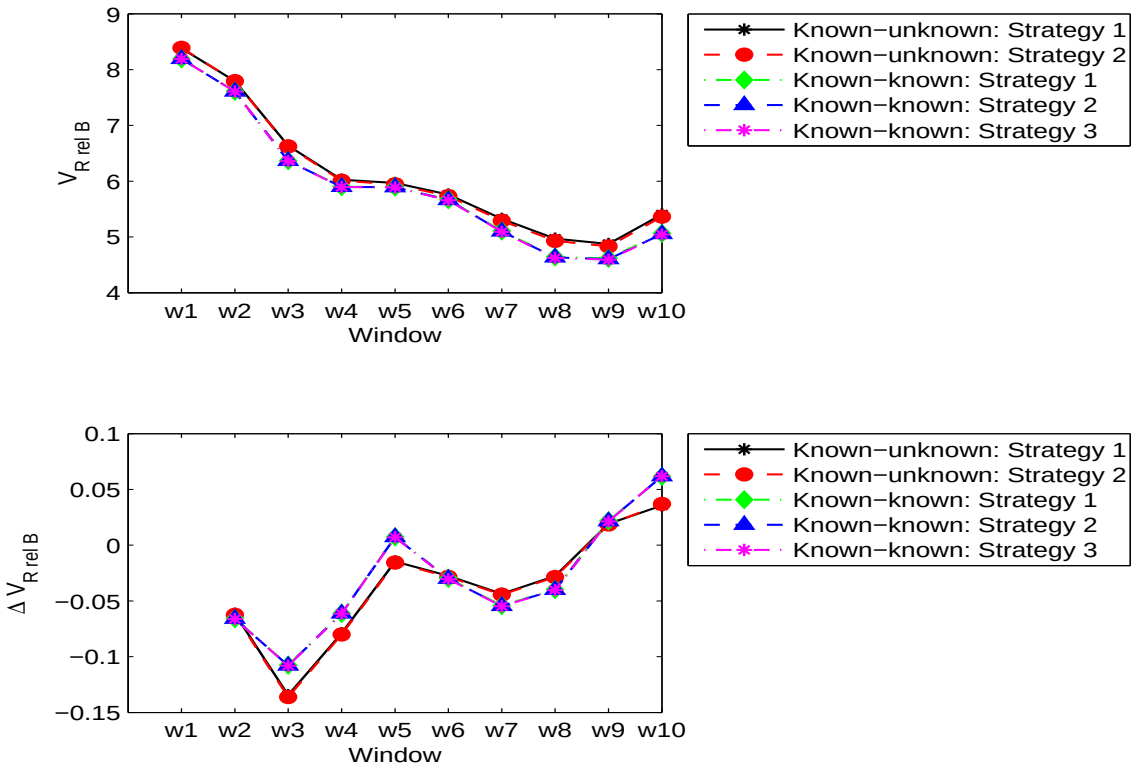


Figure 7.10: $V_{R\,rel\,B}$ and $\Delta V_{R\,rel\,B}$ in the known-unknown and known-known scenarios based on the individual neural networks.

are combined as an ensemble in each scenario. Therefore, there is not much difference in actions between both scenarios owing to the diversity in the collection of strategies.

7.3.3 Generalised and Individual Neural Networks

Based on the analysis from action distribution, it is found that generalised neural networks show pattern consistency in known-unknown scenarios, while the individual neural networks lack such consistency. The action distributions for the generalised neural networks are influenced more by the effect of information than deception. In contrast, the action distribution of the individual neural networks seems to be dependent on the network itself rather than on the effects of information and deception. Besides that, the action similarity for both types of neural network show that they are quite different from each other.

7.4 Conclusion

The results from this chapter show that the generalised neural networks are not affected by information and deception when they learn from a variety of human behaviours. Besides that, the machine behaviours produced by the generalised neural networks are similar in both known-unknown and known-known scenarios, which means they are not affected by the perceptual load. Having access to the extra task relevant stimuli does not influence the construction of machine behaviour in generalised neural networks. Unlike the generalised neural networks, there is lack of consistent and obvious patterns in the actions for the individual neural networks. This means the neural networks are able to approximate the behaviours of individual players, and the preferences of action are affected by the players rather than by the information they receive and/or deception. Even though the individual neural networks lack consistency in behaviours, the analysis based on action similarity shows that their strategies are similar in both scenarios. When clustering analysis is performed, the centroid actually refers to a general representation for the diversified behaviours. However, the behaviours produced by the generalised and individual neural networks are different from each other.

Chapter 8

Conclusion and Future Research

8.1 Thesis Summary

Red teaming is an approach to study a task by anticipating an adversary. Here, adversary refers to any entity affecting the objectives of the task. The objectives of the task and the adversary are known as *blue* and *red* respectively. A *blue* entity refers to an entity which would like to achieve the task, while a *red* entity refers to the circumstances and/or entities which may have an adverse impact on the task. In simple words, *blue* and *red* have conflicting interests. Even though red teaming as a concept can be traced back to 500 B.C., the concept especially CRT still has great potential to offer. Owing largely to the success of the concept, its implementation has expanded from military and physical realm-based to civilian and computer-based.

The focus of this thesis is to study the effects of different forms of *red* in an adversarial learning environment, which mainly depend on their manipulation approach. Here, we focus on four forms of red: one with direct access to stochastically manipulate the information received by *blue*; one based on machine learning with the ability to learn and evolve to

counteract *blue*'s behaviour; one that is a real human playing *red*; and one based on machine learning with the ability to fit human behaviour. These forms motivated our primary research question, where we are interested to understand the impact of information and behaviour on adversarial learning.

We attempted to address the issue by investigating the effect of *red* using an information centric design. A framework for adversarial attack simulation is developed, whereby the roles of *blue* and *red* are represented by the machine learning system and the adversarial attacks. The framework involves replication of a part of an existing adversarial taxonomy using data mining to simulate an adversarial environment to study the effect of *red*. The underlying assumption made in the simulation is that *red* has some sort of intelligence and is able to identify representative samples and subvert their inputs. Therefore, the adversarial attacks on these samples are based on sampling methods. Under the influence of *red*, the performances of *blue* represented by a single neural network and neural ensemble are evaluated in static and non-stationary environments. Based on the evaluation, it is shown that the vulnerability of the neural ensemble against the *red* effect is lower than the single neural network.

The simulation of adversarial attacks is based on the explicit assumption that *red* has intelligence to attack certain samples, and the effect of *red* is information driven. The main difficulty in previous work is we hardly know whether the assumption holds in a natural *red* or not, given that it is difficult to simulate the dynamics between *blue* and natural *red* in such simulation with the involvement of a human. Therefore, we need a good simulation environment which allows both the computational *red* and human *red* to perform the same task, so that the feasibility study can be carried out appropriately.

A red teaming game environment has been proposed in this thesis to enable both CRT and HRT to be carried out in the same task environment. A synthetic game environment

based on red teaming is developed to allow behavioural comparisons to be made between machine and human. The purpose of developing the game environment is to investigate the effect of *red* using a behaviour centric lens. In other words, we would like to understand the impact of behaviour in an adversarial learning environment. In the environment, the strategies of *blue* are pre-programmed and they are influenced by information and deception. Since the proposed forms of *red* face the same pre-programmed *blue*, the impact of information and deception on the *red* agents can be analysed and compared fairly.

The behaviours for three forms of *red* agents operating in a fixed *blue* agent's environment are analysed through a dynamic red having the ability to learn and evolve, a real human playing a *red* agent's role, and a *red* approximating the human players. A comparison study of behaviour between a natural and computational *red* needs to be performed in order to have better understanding of the possible differences between CRT and HRT, and thus establish the feasibility of using computational environments to play *red*. This became the main focus of this research, where the behaviours of the *red* agents are compared based on actions' distribution and actions' similarities.

The specific focus of the game simulation is to compare the differences in behavioural context between CRT and HRT. In our context, behaviour is defined as a sequence of actions taken by an entity. Based on this definition, several measurements are proposed to describe behaviour in the proposed game environment. Besides that, several methodologies are proposed to analyse and compare the machine behaviours and human behaviours. As far as we know, there are no standard measurements and methodologies for conducting behavioural analysis and behavioural comparisons in such simulation environments so we have proposed some of our own. The comparison in a behavioural context between CRT and HRT is carried out by fixing the strategies of the *blue* agent and comparing the dynamics of behaviour between the computational *red* and humans.

The strategies of *blue* are influenced by perception and deception. Perception refers to the access that *blue* has on *red*'s intelligence through observing the *red*'s actions. However, the perception of *blue* may be affected by the frequency of observing the *red* agent's actions, N_I and the noise which may be contained in the observations, $\hat{\alpha}^{(t)}$. Based on the received information about *red*, *blue* arranges its action to deceive *red* intentionally in order to disrupt *red*'s objective. The deception in *blue*'s action can be influenced by deception cycle length (deception frequency), N_D and deception degree, $\zeta^{(t)}$. This means the strategies of *blue* are modelled by perception and deception which are described by the combinations of the proposed parameters, i.e., N_I, $\hat{\alpha}^{(t)}$, N_D and $\zeta^{(t)}$. Thus, different scenarios for the *red* agent are generated by varying perception and deception in *blue*'s strategies.

HRT is conducted in two different scenarios, i.e., the absence and presence of access to *blue*'s perception, denoted as the known-unknown and known-known scenarios respectively. In both scenarios, the human players have very fast reaction times as well as high scores. Fast reaction times and high score reflect the engagement of humans to high perceptual load. The results of both scenarios in HRT suggest that humans' actions are independent of perception and deception. The behavioural analysis also shows that the strategies used in both treatments share high similarities. The interesting finding may actually represent the identification of task conditions that determine when the task-relevant stimuli is not used. If a task demands high perceptual load, it exhausts the perception capacity of human and leaves no spare capacity for perception of task-relevant stimuli. In orders words, humans have limited perception capacity. Owing to this, the perception capacity will be running out when humans engage to high perceptual load, preventing them from processing the task-relevant stimuli.

There were two approaches to develop the computational *red* in our work, i.e., neuroevolution and neural networks. Since a natural *red* is represented by a human, the selection of computational model for *red* needs to be able to mimic or reproduce human behaviours,

and this form of *red* is known as a neuroevolutionary *red*. Owing to this, neuroevolution based on a biologically-inspired approach is selected for behavioural reproduction. Highly complex systems such as those found in humans are extremely robust and resilient systems, which can adapt to environmental changes and constantly learn and evolve for their betterment. Therefore, a biologically-inspired approach such as neuroevolution borrows concepts from these systems and tries to provide robust and adaptive solutions to the involved task. Besides that, the abilities to learn and evolve in neuroevolution are important to reproduce optimality and rationality of human behaviour. Behaviour fitting based on neural networks is used to develop another type of computational *red*. Two types of neural networks, known as generalised and individual neural networks, are developed by feeding them the data from all human players and individual players respectively. This means the neural networks actually approximate the generalised behaviours and also individual behaviours for the human players. For both types of computational *red*, the known-unknown and known-known scenarios in HRT are simulated in CRT as well so that the impact of the perceptual load in CRT can be carried out.

8.2 A Reflection on the Research Questions

The common finding that we can obtain from the feasibility study conducted in CRT and HRT is the action distributions of the *red* agents seems not to be affected by the difference in scenarios. The observations that we have from the action distributions in both scenarios can either be very similar or lack consistency. Indirectly, this means that there is no clear evidence to show that the availability of extra task-relevant stimuli for *red*, represented by the noise level, influences the construction of behaviour and outcome of behaviour in both types of read teaming. Besides that, the finding from the action distribution indicates there is lack of changes in preference in both human and machine *red*. Once a constructed

behaviour enables *red* to achieve its objective, it is unlikely that it will make substantial changes in the action preferences.

In HRT, there is a lack of consistency in trends from the observation of action distribution in both known-unknown and known-known scenarios. A possible explanation for this observation is the human players react independently of information and deception. In both scenarios, the human players have shown goal-directed behaviours, as reflected by achieving a high score. Given that a human player needs to achieve his/her goal in a task environment which requires high attention at the same time, this type of high perceptual tasks may prevent the human player from processing the extra task-relevant stimuli even though it the stimuli was clearly communicated and displayed as shown in the known-known scenario. Due to humans' limited perception capabilities, there is no spare capacity for humans to process the extra task-relevant stimuli when they are engaged in a high perceptual load task. As a result, most of the human players use a general way to survive - run straight in the opposite direction of *blue* most of the time. This is supported by the analysis of action similarity, which shows the strategies in HRT share high similarity regardless of having access to *blue*'s perception or not.

The action distribution conducted on the neuroevolutionary *red* suggests that the contingent production of machine behaviour is influenced by the quality of information rather than the level of deception. It seems that the characteristics of task, such as information, can evoke strategies that partially determine the preferences of action in the neuroevolutionary *red*. For the neuroevolutionary *red*, its strategies across different combinations of information and deception can be categorised into two main groups based on the analysis of action similarity. Even though both strategies are associated with adequately high scores, one of them has a higher mean and lower standard deviation in term of scores than the other. Furthermore, the frequency of the better strategy is higher as well. This means neuroevolution is likely to generate strategies with better performance.

Neural networks are used to fit the human behaviour generally and individually. Based on the results in the action distribution, it is interesting to find that the generalised neural *red* is not affected by information and deception given that it learns from the data consisting of all players. On the other hand, the individual neural *red* fits the individual player's behaviour specifically. Therefore, we observe a variety of action distributions which lack consistency across different combinations of information and deception. It is not surprising to know that the strategies for the generalised neural *red* have high similarities since we observe similar action distribution in them. However, the analysis of action similarities also shows that the individual neural *reds* produce similar strategies as well. Each individual neural *red* can be viewed as a unique player which has his/her own behaviour, and the diversified behaviours complement each other to form a more generalised behaviour. Therefore, we observe that both scenarios show similar strategies as the cluster centroid refers to a general representation of a collection of diversified strategies.

From the comparisons made between different forms of *red* based on action distribution and action similarity, it is found out that the *red* agents have their own repertoire of ways for constructing their preferences in selecting actions, given that each of them show different action distributions for the same configuration of information and deception. In the analysis of action similarity, a collection of behaviours for each form of *red* is represented by a centroid, which also refers to a general representation of strategies.

The main differences that we have observed from previous chapters is the consistency of patterns in machine behaviours resultant from neuroevolution and generalised neural networks based on the above comparisons. When the deception is fixed, the neuroevolutionary *red* agent prefers certain actions, given that it receives frequent information, regardless of noise level or infrequent-noisy information. This means that the impact of noise on its action preference becomes clearer if there is a delay in the information. However, the action preferences for the neuroevolutionary agent are not affected by deception as long as the

quality of information is the same. As for the *red* agent produced by the generalised neural networks, it shows consistency in action preferences regardless of information and deception. Similar to human *red* agents, there is a lack of pattern consistency in the specialised neural networks since each network attempts to learn a different human pattern.

Another interesting finding is the actions in HRT are more evenly distributed than CRT, which causes HRT to have much lower peaks than CRT. This is interesting because the machine in CRT almost specialises on the task. It identifies areas whereby it can better exploit the task. The ensemble of humans, however, attempt to go through the experience from scratch; there is no transfer of knowledge and experience between humans as it is the case with the machine where evolution transfers knowledge from one generation to another.

Figures 6.17, 5.13, 7.5 and 7.10 show the measurements of $V_{R\,rel\,B}$ and $\Delta V_{R\,rel\,B}$ in the unknown-known. Despite different forms of *red* agents, all of them have shown a decreasing trend in $V_{R\,rel\,B}$. There are two very distinct strategies that can be observed in the neuroevolutionary designed *red* agent while the remaining *red* agents have a single strategy.

For the dynamic *red* agent, it is interesting to observe that $V_{R\,rel\,B}$ for one of the strategies is associated with high value and almost flat with the increase of window. In contrast, another strategy belonged to the dynamic *red* agent is associated with much lower values of $V_{R\,rel\,B}$. The decreasing trends in $V_{R\,rel\,B}$ means *blue* viewed from the *red* moves towards it with different speeds. This can be supported by the trends of $\Delta V_{R\,rel\,B}$ in Figures 6.17, 5.13, 7.5 and 7.10.

Unlike the computational *red* agents, the human *red* agents experience less change in $\Delta V_{R\,rel\,B}$, where they have almost constant $\Delta V_{R\,rel\,B}$. By considering both trends of $V_{R\,rel\,B}$ and $\Delta V_{R\,rel\,B}$, we can observe that the strategies for different forms of *red* agents are quite different from each other overall. Even though the *red* agents are exposed to the same

task environment, their behaviours are different and their action preferences are influenced differently by information and deception as well.

By scrutinising the diagrams with respect to the evolved agents and comparing them to the other agents, once can notice that the evolved agents are clustered in two groups with all other agents fall in between these two clusters. In other words, while the evolved agents clusters differently for the human agents, the evolved agents bound the human agents. This entails that evolution finds different patterns from humans but the same time finds similar patterns to humans. We can refer to this as creativity, where evolution generates much more diversity than humans.

The study on CRT and HRT has shed some light on understanding the machine behaviours and human behaviours in a red teaming environment. Characteristics of the task such as information can evoke strategies that at least partially determine the preferences of actions we observe in the neuroevolutionary *red* agent. On the other hand, the generalised neural networks are not affected by information and deception. Besides that, the comparison between the unknown-known and known-known scenarios suggests that the machine behaviours in neuroevolution and generalised neural networks are not affected by the perceptual load.

Unlike machine behaviours in neuroevolution and generalised neural networks, there is a lack of diverse patterns in the human behaviours and specialised neural networks across different combinations of information and deception. This leads us to believe that the humans do not take into account both perception and deception and their actions are independent of them. Given that the specialised neural networks learn from human players, it is not surprising that they share similar findings.

Cluster analysis shows an interesting finding: the strategies used by the human players are similar across different combinations of perception and deception. It seems that indi-

vidual differences, represented by different human players, do not influence the response to the task. Even though there is a lack of obvious patterns in the pdf plots, the cluster analysis suggests that the human uses similar strategies in the games. Both findings may be explained by the high perceptual load in the game environment requiring focusing attention from the human players to achieve their goals, and lead to the ignorance of task-relevant stimuli reflected in *blue*'s movements. Consequently, the condition exhausted the perception capacity of human and leaves no spare capacity to process the task-relevant stimuli. As a result, the actions taken by the humans are independent of both perception and deception, and also share high similarities. Consequently, the exhibition of machine behaviours in specialised neural networks was affected since each of them approximates individual players' behaviours. The conclusion that we can obtain from the work is a lack of variations in human decision behaviours due to high perceptual load.

Based on the comparisons that are made between the CRT and HRT, the results show that CRT encompasses HRT. It establishes a bound on what may arise from HRT. In short, CRT encompasses HRT but CRT generates more diversity (ie creativity) than HRT despite that the task and the environment are maintained constant. CRT can augment human behaviour. Reliance on CRT can be a real risk for HRT-only exercises. Not only an HRT would miss out on creative strategies, it won't evaluate them, which will open a hole in the risk assessment exercise [4].

In conclusion, this thesis found the following answers to the research sub–questions identified at the start.

1. What is the impact of intentional manipulation of information/data on a learning machine?

   We found that random manipulation of data can have a significant impact on a single

learning machine. However, by using an ensemble, this impact is reduced to minimum. Random manipulation of data becomes noise that most ensemble learners are able to filter out.

2. How to characterise and understand behaviour for a machine or a human?

   Behaviour of humans and machines is a difficult concept to be characterised objectively. Nevertheless, we introduced a number of metrics to capture these behaviours.

   The underlying actions of both humans and machines are mapped into different metrics. First, direction information of moves are normalized within fixed windows for comparisons. Second, different windows are analysed as a time series, with the velocity between windows used to capture trends in humans and machines behaviours.

3. How can we compare the differences and similarities between a machine and a human behaviour?

   We introduced a number of metrics to do the characterisation. First, cluster analysis was found to be an effective way to identify centre of mass in the behaviour of human and machines. By comparing the cluster prototypes (i.e., centre of masses), we were able to show the differences and similarities between human and machine.

   Second, we introduced a methodology by which actions are approximated using a mixture of Gaussian. The resultant distribution identifies points of interest in the human and machine actions. Peaks define most frequent actions, and centres and spreads of the Gaussian functions can be used to show the range of behaviours produced by humans and machines.

The main research question of the thesis was: "**What is the role of information and behaviour in adversarial learning**?" We have demonstrated that humans tend to ignore the information presented to them when the task is mentally demanding. Meanwhile, deception - or intentional slight deviation from goals - was found to be an essential strategy in environments with high noise in the information received by agents. In noise-free environments, deception was not needed.

The thesis demonstrated also that CRT produces a diverse set of behaviours that encompass human behaviours. The thesis provided concrete evidences that humans tend to behave in similar manners, and when focusing on a task, they miss out on information that is readily available to them. CRT overcomes these human biases and provide more diverse set of strategies.

8.3 Future Research

Our work shows the first attempt to study the feasibility of CRT through behavioural analysis between CRT and HRT. Several potential directions can be taken to extend this work.

- The task and fitness evaluation in the neuroevolution is designed without taking into account human preferences in the task environment. It would be interesting to incorporate human preference in the fitness evaluation so that the potential solutions are evolved in the direction of human preference instead. To do so, a survey based on a questionnaire can be carried out to collect relevant information on human preferences from a group of human subjects and the information will be used in the fitness

evaluation. Then, the same group of subjects will participate in the games and their actions will be compared with those generated by the evolved solutions.

- Introduction of measures of performance which are able to quantify the quality of human actions. Such measures of performance are important to ensure the generation of solutions according to the quality of human actions, rather than arbitrarily. This means the performances of neuronevolution or neural networks are guided by human's cognition since the corresponding decision making model reflects the capability of the agent in making decisions [4].

- The best evolved solutions across different configurations of information and deception can be combined to form an ensemble. Then, the performances of the neuroevolutionary ensemble can be evaluated by having it play against the pre-programmed *blue*. Besides that, its performances can be evaluated by replacing the *blue* agent as a human instead.

- Given that CRT can be used to explore the problem spaces as well as solution spaces, it offers a great potential in secure learning in two different layers. For example, in the exploration of problem spaces (the first layer), the use of appropriate measures in evolution by considering human reflection ensures that the generation of actions are guided by human's cognition. Therefore, a population of possible behaviours of *red* are generated in the evolution to form a database. For each *red*'s behaviour, CRT can be used to explore the solution spaces by using evolution to search for the potential defense solutions against it. Behaviour mapping based on data mining can be used to determine the behaviour of a real *red* in a given adversarial learning task from the behaviour database and its corresponding defence solution can be deployed to defeat the real *red*.

References

[1] H. A. Abbass. Computational red teaming and cyber challenges. Platform Technologies Research Institute Annual Symposium, PTRI 09, 14–15 July 2009. Invited Speech.

[2] H. A. Abbass, S. Alam, and A. Bender. MEBRA: multiobjective evolutionary-based risk assessment. *IEEE Computational Intelligence Magazine*, 4(3):29–36, August 2009.

[3] H. A. Abbass and A. Bender. The Pareto operating curve for risk minimization. *Artificial Life and Robotics*, 14(4):449–452, 2009.

[4] H. A. Abbass, A. Bender, S. Gaidow, and P. Whitbread. Computational red teaming: Past, present and future. *IEEE Computational Intelligence Magazine*, 6(1):30–42, February 2011.

[5] H. A. Abbass, A. Bender, and P. Whitbread. Computational red teaming. In *Defence Operations Research Symposium (DORS)*, Adelaide, Australia, 2010.

[6] H. A. Abbass, A. Bender, and P. Whitbread. Computational red teaming: Unravelling the pharaohs curse. In *Computational Intelligence in Security and Defence Applications Workshop*, WCCI 2010, Barcelona, Spain, 2010.

[7] S. Alam, H. A. Abbass, C. J. Lokan, M. Ellejmi, and S. Kirby. Computational red teaming to investigate failure patterns in medium term conflict detection. In *Proceed-*

ings of 8th Eurocontrol Innovation Research Workshop and Conference, Eurocontrol Experimental Research Center, Brtigny-sur-Orge, Paris, France, December 2009.

[8] S. Alam, K. Shafi, H. A. Abbass, and M. Barlow. An ensemble approach for conflict detection in free flight by data mining. *Transportation Research Part C: Emerging Technologies*, 17(3):298–317, 2009.

[9] S. Alam, W. Zhao, J. Tang, H. A. Abbass, C. Lokan, M. Ellejmi, and S. Kirby. Discovering delay patterns in arrival traffic with dynamic continuous descent approaches using co-evolutionary red teaming. *Air Traffic Control Quarterly*, 20(1), 2012.

[10] P. Angelov, E. Lughofer, and X. Zhou. Evolving fuzzy classifiers using different model architectures. *Fuzzy Sets and Systems*, 159(23):3160–3182, 2008.

[11] P. Angelov and X. Zhou. Evolving fuzzy systems from data streams in real-time. In *2006 International Symposium on Evolving Fuzzy Systems*, pages 29–35, 2006.

[12] T. Aven and O. Renn. The role of quantitative risk assessments for characterizing risk and uncertainty and delineating appropriate risk management options, with special emphasis on terrorism risk. *Risk Analysis*, 29(4):587–600, 2009.

[13] M. Barreno, P. L. Bartlett, F. J. Chi, A. D. Joseph, B. Nelson, B. I. P. Rubinstein, U. Saini, and J. D. Tygar. Open problems in the security of learning. In *Proceedings of the 1st ACM workshop on Workshop on AISec*, AISec '08, pages 19–26, New York, NY, USA, 2008. ACM.

[14] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.

[15] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, Computer*

and Communications Security (ASIACCS), pages 16–25, New York, NY, USA, 2006. ACM Press.

[16] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer, Norwell, MA, 1981.

[17] T. Caliáski and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics - Theory and Methods*, 3(1):1–27, 1974.

[18] J. Carpinter and R Hunt. Tightening the net: A review of current and next generation spam filtering tools. *Computers and Security*, 25(8):566–578, 2006.

[19] E. Caulcott. *Significance Tests*. Routledge and Kegan Paul, Ltd., USA, 1973.

[20] S. Chebrolu, A. Abraham, and J. P. Thomas. Feature deduction and ensemble design of intrusion detection systems. *Computers and Security*, 24(4):295–307, 2005.

[21] R. D. Clark. OptiSim: An extended dissimilarity selection method for finding diverse representative subsets. *Journal of Chemical Information and Computer Sciences*, 37(6):1181–1188, 1997.

[22] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 99–108, New York, NY, USA, 2004. ACM.

[23] M. Daszykowski, B. Walczak, and D. L. Massart. Representative subset selection. *Analytoca Chimica Acta*, 468(1):91–103, 2002.

[24] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 1979.

[25] J. C. Dunn. Well-separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4(1):95–104, 1974.

[26] H. J. Einhorn and R. M. Hogarth. Behavioral decision theory: Processes of judgement and choice. *Annual Review of Psychology*, 32(1):53–88, 1981.

[27] J. A. Fernandez-Leon, G. G. Acosta, and M. A. Mayosky. Behavioral control through evolutionary neurocontrollers for autonomous mobile robot navigation. *Robotics and Autonomous Systems*, 57(4):411–419, 2009.

[28] D. Floreano and J. Urzelai. Evolution and learning in autonomous robotic agents. In T. Mange and M. Tomassini, editors, *Bio-inspired Computing Systems*, pages 1–36. Lausanne, 1998.

[29] A. Frank and A. Asuncion. UCI Machine Learning Repository. <http://archive.ics.uci.edu/ml>, 2010. University of California, School of Information and Computer Sciences, Irvine, CA.

[30] S. B. Griffith. *Sun Tzu - The Art of War*. Oxford University Press, 1963.

[31] A. W. Grill and D. Grieger. Validation of agent based distillation movement algorithms. Technical Report DSTO-TN-0476, Defence Science and Technology Organization, Australia, 2003.

[32] T. S. Guzella and W. M. Caminhas. A review of machine learning approaches to Spam filtering. *Expert Systems with Applications*, 36(7):10206–10222, 2009.

[33] J. M. G. Hidalgo. Machine learning for spam detection resources. ftp://ftp.ics.uci.edu/pub/machine-learning-databases/spambase/, 2010.

[34] G. Hinton and S. J. Nowlan. How learning guides evolution. *Complex System*, 1:495–502, 1987.

[35] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.

[36] Z. Jorgensen, Y. Zhou, and M. Inge. A multiple instance learning strategy for combating good word attacks on spam filters. *Journal of Machine Learning Research*, 9:1115–1146, 2008.

[37] D. Jouan-Rimbaud, D. L. Massart, C. A. Saby, and C. Puel. Characterisation of the representativity of selected sets of samples in multivariate calibration and pattern recognition. *Analytica Chimica Acta*, 350(1-2):149–161, 1997.

[38] M. Keijzer and V. Babovic. Genetic programming, ensemble methods and the bias/variance tradeoff - introductory investigations. In *Genetic Programming*, volume 1802 of *Lecture Notes in Computer Science*, pages 76–90. Springer Berlin / Heidelberg, 2000.

[39] R. Kocjančič and J. Zupan. Modelling of the river flowrate: the influence of the training set selection. *Chemometrics and Intelligent Laboratory Systems*, 54(1):21–34, 2000.

[40] Y. Kou, C-T. Lu, S. Sirwongwattana, and Y. P. Huang. Survey of fraud detection techniques. In *Proceedings of the 2004 IEEE International Conference on Networking, Sensing and Control*, volume 2, pages 749–754, 2004.

[41] M. Lauder. Red dawn: The emergence of a red teaming capability in the canadian forces. *Canadian Army Journal*, 12(2):25–36, 2009.

[42] N. Lavie. Perceptual load as a necessary condition for selective attention. *Journal of Experimental Phsycology: Human Perception and Performance*, 21(3):451–468, 1995.

[43] N. Lavie. Distracted and confused?: Selective attention under load. *Trends in Cognitive Sciences*, 9(2):75–82, 2005.

[44] N. Lavie and S. Cox. On the efficiency of visual selective attention: Efficient visual search leads to inefficient distractor rejection. *Psychological Science*, 8(5):395–396, 1997.

[45] N. Lavie, A. Hirst, and J. W. D. Fockert. Load theory of selective attention and cognitive control. *Journal of Experimental Phsycology: General*, 133(3):339–354, 2004.

[46] S. L. Lima. Putting predators back into behavioral predator-prey interactions. *Trends in Ecology and Evolution*, 17(2):70–75, 2002.

[47] H. Liu and H. Motoda. On issue of instance selection. *Data Mining and Knowledge Discovery*, 6(2):115–130, 2002.

[48] D. Lowd and C. Meek. Adversarial learning. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 641–647, New York, NY, USA, 2005. ACM.

[49] B. F. J. Manly. *Multivariate Statistical Methods: A Primer*. Chapman and Hall, 2nd edition, 1994.

[50] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5(4):115–133, 1943.

[51] T. M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1997.

[52] F. Mondada and D. Floreano. Evolution of neural control structures: Some experiments on mobile robots. *Robotics and Autonomous Systems*, 16(2-4):183–195, 1995.

[53] A. L. Nelson, E. Grant, and T. C. Henderson. Evolution of neural controllers for competitive game playing with teams of mobile robots. *Robotics and Autonomous Systems*, 46(3):135–150, 2004.

[54] B. Nelson, M. Barreno, F. Jack Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia. Misleading a learner: Co-opting your spam filter. In *Machine Learning in Cyber Trust*, pages 17–51. Springer US, 2009.

[55] B. Nelson and A. D. Joseph. Bounding an attack's complexity for a simple learning model. In *In Proceedings of the First Workshop on Tackling Computer System Problems with Machine Learning Techniques (SysML)*, pages 1–5, 2006.

[56] J. Newsome, B. Karp, and D. Song. Paragraph: Thwarting signature learning by training maliciously. In *Recent Advances in Intrusion Detection*, pages 81–105. Springer Berlin / Heidelberg, 2006.

[57] S. Nolfi. Evolutionary robotics: Exploiting the full power of self-organization. *Connection Science*, 10(3-4):167–184, 1998.

[58] S. Nolfi. How learning and evolution interact: The case of a learning task which differs from the evolutionary task. *Adaptive Behavior*, 7(2):231–236, 1999.

[59] S. Nolfi and D. Floreano. Coevolving predator and prey robots: Do "arms races" arise in artificial evolution? *Artificial Life*, 4(4):311–335, 1998.

[60] S. Nolfi and D. Floreano. Learning and evolution. *Autonomous Robots*, 7(1):89–113, 1999.

[61] S. Nolfi and D. Floreano. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press, Cambridge, MA, USA, 2000.

[62] S. Nolfi and D. Floreano. Synthesis of autonomous robots through evolution. *Trends in Cognitive Sciences*, 6(1):31–37, 2002.

[63] S. Nolfi and D. Floreano. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. Bradford Company, Scituate, MA, USA, 2004.

[64] S. Nolfi and D. Parisi. Learning to adapt to changing environments in evolving neural networks. *Adaptive behavior*, 5(1):75–98, 1996.

[65] S. Nolfi and D. Parisi. Evolution of artificial neural networks. In M. A. Arbib, editor, *In Handbook of Brain Theory and Neural Networks*, pages 418–421. Bradford Books/MIT Press, Cambridge, MA, 2nd edition, 2002.

[66] S. Nolfi, D. Parisi, and J. L. Elman. Learning and evolution in neural networks. *Adaptive Behavior*, 3(1):5–28, 1994.

[67] K. D. Opp. The evolutionary emergence of norms. *British Journal of Social Psychology*, 21(2):139–149, 1982.

[68] D. Parisi and S. Nolfi. The influence of learning on evolution. In R. K. Belew and M. Mitchell, editors, *Adaptive Individuals in Evolving Populations: Models and Algorithms*, pages 419–428. Addison Wesley, Readings, MA, 1994.

[69] D. Parisi, S. Nolfi, and F. Cecconi. Learning, behavior and evolution. In F. Varela and P. Bourgine, editors, *Toward a practice of autonomous systems*, pages 207–216. MIT Press, Cambridge, 1992.

[70] J. W. Payne, J. R. Bettman, and E. J. Johnson. Behavioral decision research: A constructive processing perspective. *Annual Review of Psychology*, 43(1):87–131, 1992.

[71] C. Phua, V. Lee, K. Smith, and R. Gayler. A comprehensive survey of data mining-based fraud detection research. Available at http://www.bsys.monash.edu.au/people/cphua/, 2005.

[72] T. Reinartz. A unifying view on instance selection. *Data Mining and Knowledge Discovery*, 6(2):191–210, 2002.

[73] P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[74] V. K. Sharma. An easy method of constructing latin square designs balanced for the immediate residual and other order effects. *Canadian Journal of Statistics*, 3(1):119–224, 1975.

[75] H. A. Simon. A behavioral model of rational choice. *The Quarterly Journal of Economics*, 69(1):99–118, 1955.

[76] J. Teo and H. A. Abbass. Multi-objectivity for brain-behavior evolution of a physically-embodied organism. In R. Standish, M. Bedau, and H. A. Abbass, editors, *Artificial Life VIII: The 8th International Conference on Artificial Life (ICAL)*, pages 312–318. MIT Press, Cambridge, MA, USA, 2002.

[77] J. Teo and H. A. Abbass. Elucidating the benefits of a self-adaptive Pareto EMO approach for evolving legged locomotion in artificial creatures. In *The 2003 Congress on Evolutionary Computation, 2003*, volume 2 of *CEC '03*, pages 755–762. IEEE, December 2003.

[78] J. Teo and H. A. Abbass. Automatic generation of controllers for embodied legged organisms: A Pareto evolutionary multi-objective approach. *Evolutionary Computation*, 12(3):355–394, 2004.

[79] J. Teo and H. A. Abbass. Multiobjectivity and complexity in embodied cognition. *IEEE Transactions on Evolutionary Computation*, 9(4):337–360, 2005.

[80] J. Teo, L. D. Neri, M. H. Nguyen, and H. A. Abbass. Walking with EMO: Multi-objective robotics for evolving two, four, and six–legged locomotion. In L. T. Bui and S. Alam, editors, *Multi-Objective Optimization in Computational Intelligence: Theory and Practice*, pages 300–332. Information Science Reference, Hershey, PA, USA, 2008.

[81] V. Trianni, S. Nolfi, and M. Dorigo. Cooperative hole avoidance in a swarm-bot. *Robotics and Autonomous Systems*, 54(2):97–103, 2006.

[82] J. Urzelai and D. Floreano. Evolutionary robotics: Coping with environmental change. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO 2000, pages 941–948, 2000.

[83] A. Veloso and W. Meira Jr. Lazy associative classification for content-based spam detection. In *Proceedings of the Fourth Latin American Web Congress*, LA-Web '06, pages 154–161, October 2006.

[84] S. L. Wang, K. Shafi, C. Lokan, and H. A. Abbass. Adversarial learning: The impact of statistical sample selection techniques on neural ensembles. *Evolving Systems*, 1(3):181–197, 2010.

[85] S. L. Wang, K. Shafi, C. Lokan, and H. A. Abbass. Robustness of neural ensembles against targeted and random adversarial learning. In *Proceedings of the 2010 IEEE International Conference on Fuzzy Systems (FUZZ)*, pages 1–8, 2010.

[86] M. Wolf, G. S. van Doorn, and F. J. Weissing. Evolutionary emergence of responsive and unresponsive personalities. *Proceedings of the National Academy of Sciences*, 105(41):15825–15830, 2008.

[87] W. Wu, B. Walczak, D. L. Massart, S. Heuerding, F. Erni, I. R. Last, and K. A. Prebble. Artificial neural networks in classification of NIR spectral data: Design of the training set. *Chemometrics and Intelligent Laboratory System*, 33(1):35–46, 1996.

[88] A. Yang, H. A. Abbass, and R. Sarker. Landscape dynamics in multi-agent simulation combat systems. In G. Webb and X. Yu, editors, *AI 2004: Advances in Artificial Intelligence*, volume 3339 of *Lecture Notes in Computer Science*, pages 121–148. Springer Berlin / Heidelberg, 2005.

[89] A. Yang, H. A. Abbass, and R. Sarker. Characterizing warfare in red teaming. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 36(2):268–285, 2006.

[90] Standards Australia / Standards New Zealand. AS/NZ 4360: Risk Management, 1999.