CZECH TECHNICAL
UNIVERSITY
IN PRAGUE

F3

Faculty of Electrical Engineering
Department of Cybernetics

Bachelor's Thesis

Strategic Games in Adversarial Classification Problems

Tomáš Kasl
kasltoma@fel.cvut.cz

May 2020
Supervisor: doc. Ing. Tomáš Kroupa, Ph.D

Acknowledgement / Declaration

Firstly, I would like to thank my supervisor, doc. Tomáš Kroupa, for being patient with me. I would also like to thank my other teachers for preparing me for this personal milestone. Next, I would like to thank my colleagues for letting me focus on the thesis and spending a considerable amount of time working on it. Most importantly, however, I would like to express my gratitude towards my friends, family and fellow CTU students, for they have been a great support not only along making this project but also for the whole study period.

I declare that the presented work was developed independently and that I have listed all sources of information used within in accordance with methodical instructions for observing ethical principles in the preparation of university theses.

Pilsen, 22 May 2020

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Plzni, 22. 5. 2020

Abstrakt / Abstract

Tato práce se soustředí na strojovou klasifikaci protivníka. Prezentuje důvody pro použití teorie her v této klasifikaci a jak takové hry řešit. Dále se soustředí na jeden specifický model hry, jenž má základ ve statistickém testování hypotéz. Kromě rozebírání modelu, jeho možných nedostatků a návrhů možných řešení, budu také zkoumat výsledky, které hra nabízí. Hlavním bodem práce potom bude experimentování s řešením tohoto modelu pomocí tzv. *Double oracle algorithm* - algoritmem Dvou proroků. Zatímco použití tohoto algoritmu pro řešení her s konečným prostorem strategií je prokazatelně validní, o jeho použitelnosti pro nekonečné hry žádné důkazy zatím nejsou. Nicméně kvality tohoto algoritmu, když je použit zamýšleným způsobem, stojí za pokusy o rozšíření domény jeho použitelnosti, třeba i do her se nekonečným prostorem strategií. Kromě toho se zaměřím na klasifikační chybovost klasifikátoru, který z vyřešení této hry plyne.

Klíčová slova: teorie her, spojité hry, nekonečné hry, strojová klasifikace protivníka, algoritmus Dvou proroků, testování hypotéz

This thesis focuses on an adversarial classification. It presents a motivation for a game-theoretic approach to the classification, and how that can be tackled. It then focuses on a specific game model for the classification based on statistical hypothesis testing. Besides discussing the model's possible shortcomings, I will experiment with their possible solutions and also generally examine the results it presents. The main point of interest is experimenting with solving the model by the Double oracle algorithm. While the algorithm has been proved to be a valid solution to finite strategy-space games, its usage in a continuous strategy-space game model is not backed by any exact proofs. However, its excellent performance in common use cases appeals for experimenting with a possible extension of its domain, perhaps showing that the application to continuous-space games is justifiable. I will also test the classification error resulting from the game model.

Keywords: game theory, continuous games, infinite games, adversarial classification, Double oracle algorithm, hypothesis testing

Contents /

Chapter 1
Introduction

1.1 State of the art in game-theoretic adversarial classification

1.1.1 Adversarial Classification

It has always been the case that a certain portion of people is willing to break laws and rules whenever it allows them to reach some form of personal gain. This statement applies to a wide range of systems, situations, relations, et cetera.

Naturally, the ability to identify such individuals with malicious intent has been sought after. Being able to pinpoint these individuals allows responsible supervisors to limit these malicious activities by appropriate means, whether that is banning the individual from the system, locking him away, or any other possible way. Doing that, the general well-being of said system can be protected from unnecessary harm.

The ability to recognize individuals (called *adversaries*) with harmful intentions is commonly known as the *adversarial classification* [1].

The few previous decades have witnessed a rise in computing technology, and with that, the Internet. That brought with itself an ever-increasing amount of various systems and environments, which now have to face mentioned adversaries. But as computing technology brought us an increasing need for the adversarial classification, it also provided us with more and more powerful tools to do so.

Over the decades, multiple approaches to tackle this difficulty have appeared. While there were considerable advancements in possibilities to monitor specifically these malicious behaviors, they proved to be insufficient.

That is because the partial actions taken by the adversary do not have to be necessarily perceived as harmful. More general observation of the system, therefore, was needed. And with that, also more sophisticated approaches to classification of an individual as an adversary. Naturally, statistical [2] and machine-learning [3] techniques received much of the interest. While they reached various degrees of success, eventually, they face similar limitations.

Approaches based on machine learning utilize commonly known classification techniques, which are trained to classify vectors from feature space.

On the other hand, classifiers based on a statistical hypothesis test expect a list of samples. It is then examined on which probability distribution is it more likely based, benign or adversarial.

None of these, however, reflects the interests of attackers nor their will to exploit the defense system.

1.1.2 Game theory

Their limitations are the reason why the game-theoretic approach to attack detection has seen an enormous increase in popularity in recent years. When discussing adver-

sarial classification, there is generally a common concern. Once an attacker seeking to exploit a particular system discovers how the classifier is designed, he can modify his strategies to avoid detection. He can do so by purposely choosing attacks, which are likely to be misclassified as harmless.

The idea is to employ the game theory, which models interactions between multiple entities (formally called agents) in a shared environment. Here, instead of predefining what attack is and what is not, this method tries to model the system as a game between two agents: An attacker, who seeks to reach a personal gain by malicious activities while overcoming the defense, and a defender, whose goal is to discover him and stop him.

However, there are present also other agents, benign users of the system. While they are not an explicit part of the mathematical game model, their presence in the system is essential, as the defender has to classify any agent as either attacker or non-attacker.

To model their interactions within the model, there is usually an estimation of the cost and utility gained for the agents' actions and behaviors. They represent the agents' gain or loss for their chosen strategies in numerical values.

Equipped with these tools, we should be able to compute the optimal strategy for the attacker. And having that estimation, we can use it also to calculate the optimal strategy for the defender, which is the ultimate goal.

The solution gained from examining the game model then should reduce the amount of both false positive and false negative errors when used in a real-world application. Also, it should provide us with a more general defense strategy when it comes to constraints, by which is the observed behavior classified.

The topic of adversarial classification is closely related to two other general game models. One of them is *security games*, where the defender tries to allocate system defenses and monitoring sensors optimally.

The second one is the so-called *anomaly detection games*. The key difference is that the defender tries to identify, also in the presence of non-attackers, anomalies in behaviors. Agents, whose behavior is labeled as an anomaly, are then on their own examined for possibly adversarial intentions (using further techniques), rather than being proclaimed as adversaries right away.

The study of deploying the game theory to the adversarial classification has been split into two main directions:

In the first one, there is a game where the attacker chooses an attack vector from a feature space. The defender then tries to classify the unknown agent based on observing the vector inside the feature space and estimating an attacker's possible gain. Here, more conventional machine-learning algorithms may be used for the defender to learn individual strategies [1].

The second direction models a game where the attacker (and also the non-attacker) chooses a probability distribution. Their samples are then chosen from the distribution independently. This approach leads to more statistical-oriented solving methods, as the defender is then usually performing a modified hypothesis testing based on a number of observed samples of the unknown agent [4].

This thesis focuses on the second case.

1.2 Applications

Not surprisingly, wide possibilities for adversarial classification are in the domain of information technologies, where becoming an attacker is often simple, and price for detection negligible. The amount of attackers is therefore significant. Recall, for example, a very familiar problem of fake emails and spam-filtering [5].

Another domain is a monitoring system of a social network. Here, attackers might want to exploit the accessibility and wide audience, as they spread so-called fake news for personal profit. Either it could be advertising of a product based on false claims, or it could be a propagation of certain political parties with the vision of altering the election results. The company behind the social network would, in return, try to identify these (fraudulent) accounts, whose published content conflicts with the rules or interests of the company behind the network [6].

One of the frequently suggested applications is a security system of an app store. While the majority of the apps available there are harmless, a certain number of malicious apps may try to sneak in. When installed, these apps can exploit the device and possibly even the user. For example, by accessing and sharing the user's private data, sending the user's location to a third party, et cetera. (Proposed, for example, in a paper [7].)

One of the main conditions for these models to work is having a possibility to estimate the cost of different kinds of attacks. That is how much effort there was in developing and maintaining the attack. It is needed to correctly estimate the attacker's gain or loss under chosen strategies. Apparently, this condition tends to limit the application of some game theory models. In some models, though, this cost estimate can be constructed without much struggle [1].

Another possible domain is multimedia forensics. Here, the defender tries to decide whether a presented image is legitimate, or if it has been purposefully modified. While the picture modification can be done by an attacker to mislead a living person seeing the picture, for whichever reason, there is another huge motivation. That is the ability to alter a decision of a neural network-based classifier (presented, for example, in a paper [8]).

Yet another domain is checking the values scanned from a set of sensors. Here, the defender's goal is again checking that data coming from the sensors were not somehow altered (also proposed in [7]). Notice that while the real-world situation is quite different when compared with the multimedia forensics case, the model structure remains similar. Thus, game-theoretic models for adversarial classification stay in a very general setting.

However, the adversarial classification is not limited to the domain of the Internet. Imagine a driver on a highway being registered by some of the speed radars as slightly overspeeding. The police's time is limited; therefore, not all overspeeding drivers can be pursued - a strategy for selecting only some of them must be created. Is it beneficial to somehow classify the driver based on the registered speed values and then decide whether this driver belongs to the group reasonable drivers facing a serious urge to reach the destination as soon as possible? Or does the driver actually intends breaking the rules, either because he is trying to escape the place of more serious crime (or just because of the thrill), and should, therefore, be pursued?[1]

[1] `https://www.youtube.com/watch?v=kI712FeoPSM`

1.3 Summary of the thesis

In this thesis, I will try to resolve a game model with an infinite action space, which is suggested in a study paper [9], by a different approach. While the authors of the model settle for a specific setting of the model and then discretize it, I will experiment with resolving the model by an iterative algorithm called Double oracle.

More specifically, I am interested in whether this algorithm, which was not purposed for this use case, can achieve similar results in the same setting of the game, as the model's authors did. That means I will examine how does the algorithm perform when it comes to speed and reliability of convergence, as well as whether the algorithm converges to correct solutions.

While the algorithm's convergence is evident in finite-space games, for which it is designed, its convergence in infinite-space games is only expected intuitively. More explanation on this is given in Section 2.5.2.

If the Double oracle deployment succeeds, I will examine the classification error for various values of n, comparing it with the expected error rate proposed in the paper. Does the error rate actually converge to zero?

I will also try to modify the game slightly in a few ways in order to improve upon its shortcomings and argue about possible improvements. Those will be tested by cross-comparing the error rates.

Also, I want to point out that I will use, whenever referring to any abstract agent or person, masculine form. There is no hidden intention other than making formal things as simple as possible.

Chapter 2
Strategic games

2.1 Game Theory

2.1.1 Introduction

Game theory is a mathematical discipline, which tries to model strategic interactions between multiple agents (often called players) with varying degrees of rationality. In the game models, every participating agent tries to maximize his individual gain from the game among the individuals. This implies that every agent is trying to find his own best (optimal) strategy, which consequently depends on the strategies of other agents. The optimal strategies are, therefore, found by optimizing partial functions inside the model.

Nowadays, game theory is successfully used to explain various phenomena in a range of areas. Besides defeating humans in actual games [10], it also helps to understand phenomena in economics [11], biology [12], and even politology [13]. Basically, game theory can find its place wherever we are interested in multi-agent competition, and also often when we are interested in multi-agent cooperation. And in recent years, its use case has also been expanded to the adversarial classification problem.

2.1.2 Evolution

Mathematical examination of games and looking for the best strategies is quite a natural field of interest. And so with the rise of mathematics, games have been the target of focus at least since the beginning of the 18th-century [14]. However, game theory as a separate field of science has been established a bit later, in the early 20th century. It has been expanded most notably by the research done by John von Neumann and Oskar Morgenstern [11]. What is important is that they expanded the usability and focus of the game theory outside classic board games, making the new theory a useful tool for economics. Also, an important term - an equilibrium (later named Nash Equilibrium - NE) was defined in this book.

Another crucial point in the game theory development was the year 1951, when John Forbes Nash Jr. proved the existence of NE in any finite game, along with his further research done on non-cooperative games [15]. This event commonly marks the beginning of modern game theory science.

Over the years, game theory as a general approach to solving problems has only seen an increase in various topics.

2.1.3 Game theory disciplines

The game theory is now split into two main fields of interest:

Cooperative games, where the agents form a coalition. Inside this coalition, agents aim to maximize their collective utility gain. Here, the basic unit of strategic decision-making is often the whole group of agents [16][Chapter 12].

However, here we focus on the other one, non-cooperative games. Here, the agents interact with each other, seeking to maximize their own gain. That could mean the agents' gain is not correlated with the utility gain of others in any obvious way, or it could mean they explicitly compete with each other [16][Chapter 3].

2.2 Basic setup for finite games

To define a game, multiple things have to be specified.

Naturally, we are interested in which agents are present in the game. Note that just the number of agents is not sufficient; we need the whole set of unique agents specified, as each of them might have different possibilities, objectives, and even rewards associated with their actions.

That is exactly what needs to be specified next - the so-called action profile of each agent. That is the set of all the actions each agent can do. These sets might not be the same for all the agents.

The last thing that is necessarily needed is a mapping from the actions, which the individual agents could choose, into the utility they gain from selecting it.

The following definitions and principles are roughly based on a textbook [16][Chapters 3 and 4].

Usually, it is clearly stated in the game model whether the game is *simultaneous* (also *one-shot*); that is, each agent chooses his action in the beginning and then just observes the game outcome. Or whether the game is *sequential*, that means agents play in order, turn after turn, while also watching others. Games modeled for the adversarial classification are usually in the first group. Also, there can be more specific game rules outside the scope of the three mentioned sets.

The three essential sets give us a tuple:

- N - set of agents denoted n_i
- A - set of individual strategy profiles denoted A_i, which consist of possible actions a_i^x
- u - set of utility functions giving each players reward/cost for chosen action denoted as u_i

Definition 2.1. (Normal form of a finite game) *These three entities together form a tuple* $G = (N, A, u)$*, that is known as the* normal form of *a finite game.*

2.2.1 Strategies

As the modeled game proceeds, individual agents provided with their action sets need to settle on a strategy. Intuitively, said agents could choose to always play one specific action, perhaps the one that generates the highest average reward. Or they could choose between two specific actions, based on an arbitrary condition. This intuition fulfills what is called a *pure strategy.*

Definition 2.2. (A pure strategy) *A pure strategy is an element* a_i^x *of* A_i*, under which, based on the state of the game, a single specific action is deterministically chosen by an agent* n_i*.*

However, pure strategies are often not sufficient for agents' optimal behavior precisely because of their deterministic nature. Since the outcome of the game depends on the chosen actions, whenever any agent settles on a pure strategy, other agents can exploit

it by adjusting their own. What is missing then is some way of unpredictability. That would be critical, for example, if an agent needed to switch unpredictably between the action with the highest average reward, and the action with the least possibility to be countered.

Definition 2.3. (A mixed strategy) *A* mixed strategy *S_i is a probability distribution over the agent's strategy profile A_i, from which the final action is chosen randomly.*

Note that a pure strategy is a special of a mixed strategy, where one action has a 100% probability of being played, while all of the rest have 0%.

Another useful definition, which will be later needed, is a dominating strategy:

Definition 2.4. (A dominating strategy & A dominated strategy) *For a player n_i, a strategy S_i is called a* dominating strategy *and a pure strategy S_i' is called a* dominated strategy*, if $u_i(S_i) > u_i(S_i')$ for every possible state of the game.*

Now, when it is clear what a strategy is and how a clear superiority looks... For which strategies can actually an agent aim? Trivially, an agent can randomly choose a mixed strategy (*dummy* agent). Or he can choose a strategy with the potential for the highest value of utility gain (*greedy* agent).

For each combination of actions taken by other agents, an agent n_i could deterministically choose a specific pure strategy with the maximal reward. This pure strategy is called the *best response.*

Definition 2.5. (Best response) *The* Best response *is a pure strategy maximizing the utility against a specific combination of strategies chosen by other agents.*

Previously it was stated that an agent could seek to reach some kind of optimal strategies, perhaps the one which, on average, yields the biggest reward. Defining these in any useful way, however, proved to be tricky. That is because each utility depends on strategies picked by other agents. Thus they can be ultimately countered.

For each strategy an agent n_i can choose, other agents might respond by choosing strategies, which result in the smallest, minimal gain for n_i. Naturally, an agent might want to suppress this counterplay by choosing the strategy which offers the maximum of these minimal gains.

Definition 2.6. (Maximin Strategy) *A* maximin strategy *is a strategy that maximizes a minimal possible reward.*

2.2.2 Game outcomes

First, let's address what exactly is meant by a game outcome:

Definition 2.7. (A game outcome) *A game outcome is a tuple (S, U), where S is the set of all agents final mixed strategies, and U is the set of their individual utility gain.*

In other words, how much utility each agent ended with, and which strategies were needed for reaching such utility redistribution.

It is rational to talk about game outcomes whenever we are interested in what strategies have to be chosen to reach a specific utility redistribution. Another occasion is, conversely, when we want to observe the final utility distribution after choosing particular strategies. What happens, for example, if all the agents play the maximin strategy?

Expanding on the idea of defining an optimal strategy, a significant game outcome exists - the aforementioned Nash Equilibrium (NE). NE is such a state of the game, where all agents have chosen a strategy and, at the same time, no single agent can choose

a better strategy in regards to what has been chosen by others. That is, no agent has neither incentive nor reason to deviate from the strategy he has already chosen. As a result, NE is a stable outcome of a game, which, without external influence, will not be abandoned by participating agents.

Definition 2.8. (Nash Equilibrium (NE)) *A game outcome, in which no single agent can benefit from changing his own strategy, is called a* Nash equilibrium.

Nash has also proved such equilibrium exists in every finite-space game. In infinite games, however, no NE may exist.

While some games may have a single NE, it is more typical that there are infinitely many of them.

You can imagine the principle of a NE like this. There is a game between 2 agents, Alice and Bob. Both of them have a two-sided coin, which they (per turns) put on a table. If both Alice and Bob put the coin with the same side upwards, Alice wins. If the coins' orientations differ, Bob wins.

If either of the agents sticks to continuously playing a specific side of the coin, the other one can adapt, which creates an infinite circle of mutual adaption.

The NE for both agents means choosing their side of the coin randomly with a 50%/50% chance, making the action unpredictable. The best response to the randomizing, therefore, is also randomizing. Often, the strategies that lead to NE are what is meant by the label 'optimal' .

2.3 Continuous / infinite space games

While most of the setup (N, A, u) remains similar, the strategy profiles of individual agents are now infinite. That means, for example, some subset on $\mathbb{R}^n$.

This presents certain complications in the computation of the game's outputs. Since we are now unable to select the best response by comparing the outcomes of all possible actions, as there is now an infinite amount of them, other mathematical techniques must be deployed. They have various disadvantages, but a common one is a much higher computational complexity when compared to a simple finite-space model.

An obvious solution to this complication is the discretization of the strategy space. If the utility function is predictable, we can sample its values at a certain, finite amount of points, making the game model finite-space again. The problem is that the actual game optima most probably are not at the sampled points. The solution found is only an estimation, with the error depending on the predictability of the function, and resolution of the sampling.

Another option would be an iterative algorithm based on the gradient descend method. That one is, however, also dependent on the form of the utility function - the existence of multiple local minima, as well as points without a clear direction of descent, present complications.

Therefore, new techniques often have to be used for continuous-space games.

2.4 Zero-sum games and why we can solve them

Zero-sum games are games, where the sum of utility gained by all agents is equal to 0. This can be interpreted as a competition of participating agents over limited resources.

While this idea yields the potential for an arbitrary amount of agents, we are now interested in two-player zero-sum games only. Notice that as agent n_1 receives arbitrary utility gain or lose u_1, the zero-sum constraint implies agent n_2 must receive utility value $u_2 = -u_1$.

Clearly, the gain of one agent is equal to the loss of the other.

A crucial result of this equality is that we can focus on utility values of one of the agents only, as values of the other one can be obtained trivially. Moreover, an equivalent model can be obtained by changing the second agent's goal. While agent $n_1 = n_{max}$ seeks to maximize his gain, agent $n_2 = n_{min}$ seeks only to minimize n_1's gain. That simplifies the model significantly - for each combination of both agents' strategies, only one utility gain value may be considered.

With this modification in mind, a maximin strategy analogy can be defined for the agent n_{min} :

Definition 2.9. (Minimax Strategy)

A minimax strategy *is a strategy that minimizes a maximal possible reward of his zero-sum game opponent.*

The existence of maximin and minimax strategies is guaranteed by this fundamental theorem:

Theorem 2.10. (Minimax theorem (von Neumann, 1928))[11] *In any finite, two-player, zero-sum game, in any NE each player receives a payoff that is equal to both his maximin and his minimax outcome utility value.*

This is very important, as we now know that a NE exists, and also that it is achieved whenever n_{max} plays his *maximin strategy* while n_{min} plays *minimax strategy.* And since the utility gain values are the same for both agents, we can compute these strategies (and therefore the NE) by utilizing *linear programming* (LP).

The *maximin strategy* for agent n_{max} can be solved by LP as:

$$
\begin{array}{lll}
\text{maximize} & U_{max}* & \\
\text{subject to:} & \sum_{j \in A_{max}} u_1(a^j_{max}, a^k_{min}).s^j_{max} \geq U_{max} * & \forall k \in A_{min} \\
& \sum_{j \in A_{max}} s^j_{max} = 1 & \\
& s^j_{max} \geq 0 & \forall j \in A_{max}
\end{array}
$$

Similarly, the *minimax strategy* for agent n_{min} is the dual LP program, which can be written as:

$$
\begin{array}{lll}
\text{minimize} & U_{max}* & \\
\text{subject to:} & \sum_{k \in A_{min}} u_1(a^j_{max}, a^k_{min}).s^k_{min} \leq U_{max} * & \forall j \in A_{max} \\
& \sum_{k \in A_{min}} s^k_{min} = 1 & \\
& s^k_{min} \geq 0 & \forall k \in A_{min}
\end{array}
$$

The two LP's then enable us to find a NE in zero-sum games. However, their formulation needs to be refined.

We can define a matrix, where each line represents a possible action one agent (*Row agent*), and each column represents a possible action the second agent (*Column agent*). Usually, n_{max} is assigned as the row player, and n_{min} as the column one. Then we can create a matrix M, where each element in row j and column k represents the utility gain

$u_1(a^j_{max}, a^k_{min})$. If we also introduce vectors of mixed strategies, the dual LP problem is simplified into:

Primal task:		Dual task:	
maximize	x_0	minimize	y_0
subject to:	$M^T\mathtt{x} - \mathtt{1}x_0 \geq 0$	subject to:	$M\mathtt{y} - \mathtt{1}y_0 \leq 0$
	$\sum_{j \in A_{max}} x_j = 1$		$\sum_{k \in A_{min}} y_k = 1$
	$\mathtt{x} \geq 0$		$\mathtt{y} \geq 0$

For a more general overview of the game-theoretical definitions, see chapters 3 and 4 in the textbook [16].

2.5 Double oracle algorithm for continuous games

2.5.1 Algorithm presentation

McMahan, Gordon, and Blum [17] propose a new algorithm for computing the NE in two-player zero-sum games defined by a payoff matrix.

It is assumed that both agents have an *oracle*, an entity that can deduce the best pure strategy against a chosen mixed strategy of his opponent.

In finite-space zero-sum games, these *oracles* corresponds to choosing the optimal column against a distribution over a chosen row of the payoff matrix and vice versa.

The algorithm is initiated by creating a new matrix M^* by taking an element at the intersection of an arbitrary row and column from the original game-defining matrix M. Then the algorithm iterates over the following steps:

- Compute optimal mixed strategies for both agents in a game defined by a matrix M^*.
- The *oracle* of the column agent chooses the optimal column from M against the mixed strategy of the row player.
- The *oracle* of the row agent adds a new row analogously.
- If both the row and the column proposed in this iteration are already present in M^*, the algorithm terminates and outputs the final optimal mixed strategies.

In the paper, where this algorithm is presented, its main purpose is performance improvement, as it allows to compute NE strategies by deploying linear programming on much smaller payoff matrices when compared to the original ones. The worst-case scenario, obviously, is solving the LP for M.

It is easily shown that the algorithm converges to the optimal solution in finite strategy-set games. This is true for the very reason that there is only a finite amount of pure strategy combinations.

2.5.2 Continuous-space games adaptation

We can imagine a zero-sum game with continuous strategy spaces (on subsets of R, for example) as a payoff matrix with an infinite amount of rows and columns. There, both *row agent* and *column agent* can choose an arbitrary value from their respective strategy sets, which collide in a specific point. At this point, we can compute the utility function. A matrix M^* consisting of utility function values in chosen pure strategies of the agents can be maintained. Each newly chosen pure strategy will then be added to this matrix as a new column or row.

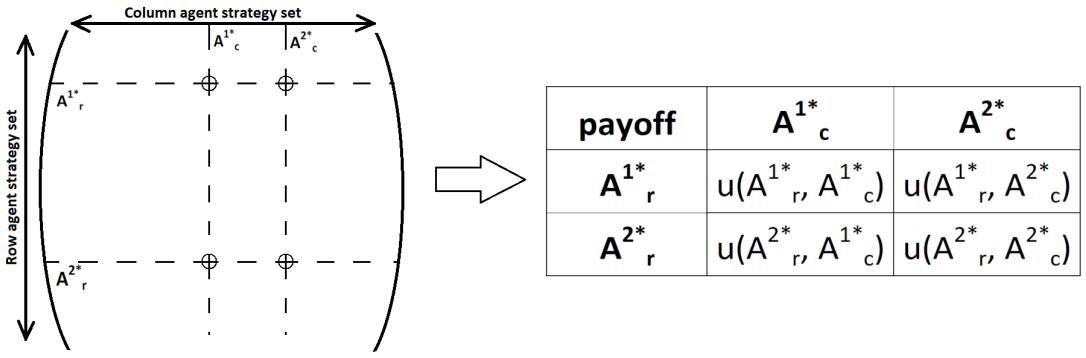


Figure 2.1. Creating the Double Oracle payoff matrix (M^*) from an infinite one

However, the *oracles* now have to be able to identify the best response in a continuous space, rather than a simple comparison of a finite amount of choices. That can be done by minimizing/maximizing the cross-sections of the utility function. However, this optimization usually must be done numerically, which raises its own questions about convergence and accuracy.

While the authors of the Double oracle algorithm justify deploying it in finite-strategy set games only, the performance of this algorithm in continuous games is worth experimenting with. As far as I know, as of now, there has not been a proof of this algorithm convergence in an infinite strategy set games. Its possible convergence, however, is intuitively conceivable, as the *oracles* of both agents gradually learn the possibilities and dominant strategies of each other, representing the learning processes of the agents. Both agents then ultimately settle on a strategy withstanding the dominant strategies of the opponent.

Nonetheless, deploying this algorithm in continuous strategy set games, I believe, is intuitively superior to some other methods, such as discretizing the strategy sets of the game. That is because the Double oracle conserves the fundamental continuity of both players' pure strategy spaces. Therefore, the exact optimum should not be skipped.

Chapter 3
Adversarial hypothesis testing

3.1 The model

3.1.1 Elementary presentation

In the bachelor thesis, I build upon the model and the theory developed in the paper [9] by Yasodharan and Loiseau.

Therein, Sarath Yasodharan and Patrick Loiseau propose a nonzero-sum continuous-space game model. Both Bayesian and Neyman-Pearson frameworks are taken into consideration. Let's describe the model under the Bayesian framework first:

Suppose there is a finite alphabet set X of actions of an *external agent*, who is either a non-attacker (with the probability θ) or an attacker (with the probability $1-\theta$). Also, suppose there are two distinctive distributions over X, distribution q for adversaries, and distribution p for non-attackers. Then $q, p \in M_1(X)$, which is the space of all possible probability distributions on X.

The defender is then permitted to observe a realization of a sequence of i.i.d. variables $X_1, ..., X_n$ following an unknown distribution over X. His task is to test a hypothesis:

$$H_0 : X_1, X_2, ..., X_n \sim p$$
$$H_1 : X_1, X_2, ..., X_n \sim q$$

where:

- $n \in N$, that is the size of the random sample.
- $p(x^n)$ is the probability of observing an $n-length$ word x^n under the distribution p.
- $q(x^n)$ is the probability of observing an $n-length$ word x^n under the distribution q.
- $\varphi : X^n \to [0, 1]$ is the defender's decision function, which returns a probability the observed word will be proclaimed as adversarial

Under the classical Bayesian hypothesis testing, where a priori probabilities are known, the decision criterion would be a likelihood ratio of $\frac{q(x^n)}{p(x^n)}$, which is then compared with the found threshold to make a decision. Here, the attacker's utility gain also must be considered, the test, however, remains a likelihood ratio and the threshold comparison [Proposition 4.1][9].

The attacker's strategy set is

$$Q \subset M_1(X)$$

and the defender's one is the set of all possible decision rules, denoted as

$$\Phi_n = \varphi : X^n \to [0, 1]$$

Note that even the pure strategies of the attacker are merely probability distributions over X. Hence considering the mixed strategy of this game means randomizing over such distributions.

3.1.2 Utility functions

The *utility function* of the attacker is

$$u_n^A(q, \varphi) = \sum_{x^n}(1 - \varphi(x^n))q(x^n) - c(q)$$

where $c(q) \in R_+$ is the cost for choosing a distribution $q \in Q$.

That is, in other words, the attacks' probability of being undetected minus the corresponding cost.

The *utility function* of the defender is

$$u_n^D(q, \varphi) = -(\sum_{x^n}(1 - \varphi(x^n))q(x^n) + \gamma \sum_{x^n} \varphi(x^n)p(x^n))$$

where $\gamma > 0$ is a constant representing the penalty balance of false positive/negative errors as well as the value of θ. So u_n^D represents the scaled penalty for both false alarms and false negatives.

It is, however, important to note that adding

$$\gamma \sum_{x^n} \varphi(x^n)p(x^n)$$

to $u_n^A(q, \varphi)$ does not alter the attacker's choice, since it does not depend on the value of q. Analogously, subtracting

$$c(q)$$

from $u_n^D(q, \varphi)$ does not change the defender's choice.

Doing this allows us to define a zero-sum equivalent of this game. The modified utility function for the attacker then is

$$u_n^{eq}(q, \varphi) = \sum_{x^n}(1 - \varphi(x^n))q(x^n) + \gamma \sum_{x^n} \varphi(x^n)p(x^n) - c(q)$$

and its negation is the defender's utility function. Following the Theorem 2.10, the NE can be obtained by solving a LP, for example, using these utility functions.

3.1.3 Model's assumptions

This model has multiple assumptions [A1-A4, Section 3.1][9]:

- Q is a closed subset of $M_1(X)$, and $p \notin Q$
- $p(i) > 0$, $q(i) > 0$; for $\forall$i $\in X$ and $\forall$q $\in Q$
- $c(q)$ is a continuous function on Q, this function must be predefined
- there exists unique $q* \in Q$ s.t. $q* = \text{argmin}_{q \in Q} c(q)$
- the benign distribution p is distant from the set Q relative to the point $q*$

The final clarification needed is this: the classification error of the classifier computed in this framework decreases as $n \to \infty$, and n should, therefore, be large.

3.1.4 Neyman-Pearson adaptation

Here the a priori probabilities of an agents' presence (that is θ) are unimportant. However, there is a limit for the false positive error rate. This limit is denoted ϵ, and its value is fixed. The defender's strategy set is then modified to reflect this limit as

$$\Phi_n = \{\varphi : X^n \to [0,1] : \sum_{x^n} \varphi(x^n)p(x^n) < \epsilon\}$$

While the initial utility function of the attacker remains unchanged, that is

$$u_n^A(q,\varphi) = \sum_{x^n}(1-\varphi(x^n))q(x^n) - c(q)$$

defender's utility function can be simplified. Because the type I error rate is already bounded by the strategy set Φ_n, his utility function can now reflect only the type II error. Thus:

$$u_n^D(q,\varphi) = -(\sum_{x^n}(1-\varphi(x^n))q(x^n))$$

Similarly as in the Bayesian framework, because the expression $c(q)$ does not depend on the value of φ, a zero-sum equivalent game can then be defined - using the attacker's utility function

$$u_n^{eq}(q,\varphi) = \sum_{x^n}(1-\varphi(x^n))q(x^n) - c(q)$$

and its negation as the defender's one.

3.2 Comparison with existing models

3.2.1 Game theory models

The discussed adversarial classification model [9] keeps the more traditional approach of letting the defender learn in a safe environment, where the attacker is not allowed to perform so-called *poisoning* of his learning data set. We can see the learning set as the distribution p, in this case. Instead, the model's deployer is responsible for figuring out the correct value of p.

It also does not handle the defender's obstructed vision. Instead, it is expected that he can fully observe all the agents' actions. However, this can be added as some kind of module to the game, unlike the previous characteristic, which would need a rework of the model's fundamental specifications. Some models, which address these topics, can be found here[1].

The model, though, addresses two common shortcomings of game-theoretical classifications models: Both strategy spaces are arbitrary distributions, which makes the model very general. It does not limit the attacker's options to a predefined enumeration of attacks; any action from the continuous space can be taken into consideration. It does, however, require a robust cost function. Also, the principle of distribution over distributions feels very un-intuitive.

Another advantage of this specific model is that it also allows agents to have different utility gains, as their goals do not have to be exactly opposite. The attacker is not concerned with the defender's false positive rate, and the defender is not concerned with the attacker's action costs. While it does not limit agents' goals by constraints of zero-sum models, it is also shown how this model can still be turned into a zero-sum game equivalent for computation purposes.

3.2.2 Statistical hypothesis testing models

As discussed earlier, the adversarial classification game model [9] shares similarities with classical hypothesis testing [2] in both Bayesian and Neyman-Pearson frameworks.

Under the Bayesian framework, the defender/classifier performs the hypothesis test very similarly as it is done in the classical setting, which is comparing the likelihood ratio to a threshold. The big difference, however, is that the vectors to be classified are not generated by *nature*. Instead, the defender must reflect on the presence and strategy of an attacker. The adversarial's attack cost and his goals push the defender's optimal strategy from what it would look like if he were an agent in a single-player game.

Notice, under the Neyman-Pearson framework, the difference in which error type is being limited by ϵ. In classical Neyman-Pearson, classification error done on positive cases is bounded, that means ϵ is the limit of false negatives. In this game model, however, ϵ limits the false positive error rate.

The rest of the specific characteristics from the Bayesian framework apply.

3.2.3 Anomaly detection models

The attacker, in the presented game model, is concerned only about the chance of being caught and his attacks' cost. And his goal is to minimize them both. His ultimate direction, therefore, is to maintain his attack position stable for as long as possible. In anomaly detection models, on the other hand, instead of cost, more focus is targeted at a specific potential reward (amount of the damage done) of individual attacks.

Here, I would like to briefly compare this model to a specific anomaly detection model called simply *Generalized [anomaly] Classification Game* [18]. The model is surprisingly similar to the adversarial classification one.

This paper proposes a zero-sum game model. In this model, an agent (either an attacker or a non-attacker) chooses one vector from a feature space denoted $f \in F$. The feature space has dimension n, and each real-valued axis is bounded from both sides. Vectors chosen by a non-attacker are generated by a probability distribution denoted P_D, and each un-inspected vector chosen by an attacker (denoted f_a) yields a non-negative reward $R(f_a) \geq 0$.

The defender tries to identify anomalies in agents' behavior, so their actions can be put under deeper inspection. He does so, just as in our model, with a decision function $c : F \to [0; 1]$. However, there is an upper bound on the false-positive error rate, which corresponds to the definition of our game under the Neyman-Pearson framework. The defender does not observe the attack vectors directly. Instead, he observes them transformed by a transformation observation function T, which must be defined. The probability that vector f_a transformed by T, will be classified by c as an anomaly is denoted by $\rho_c(f_a)$.

The utility function is defined as

$$u^{eq}(c, f_a) = (1 - \rho_c(f_a)) \cdot R(f_a)$$

The anomaly detection model appears to comply with the required assumptions, as F is a closed subset. Let's take now into consideration $T = identity$. A specific setting could be designed in a way that complies with the rest of the assumptions, too - for example, having a specific single f_a* with the highest reward and having a continuous reward function R. And doing so appears to be reasonably achievable.

Even though not equivalent, the zero-sum utility functions of the agents are very similar between the models. The utility function from anomaly detection [18] differs only by multiplying the probability of defense-trigger instead of substracting an attack cost from it.

This comparison shows how game-theoretic models may, even with different purposes, become very similar in terms of their mathematical definition.

3.3 Specific setting

3.3.1 Setting presentation

The authors of the model chose a specific setting of this model [9][Chapter 5], in which experiments were done. I will, therefore, use the same setting, too.

The defender observes a binary vector of various length n, which is the subject of examination. A non-attacker has a fixed probability $p = 0.5$ for choosing both 1 and 0.

That can be seen, for example, as making suspicious actions. In the domain of a social network, an ordinary user might, with some rate (0.5, for example), try to reply to a deleted post or encounter a time-out when sending a message. And then there could be a malicious crawler (the attacker), that is trying to click itself through all URI links it can find to gather as much information as possible. And that would make him more likely to trigger the social network's backend exception. This binary vector setting can be projected onto a wide amount of domains.

Since the alphabet is just binary, the attacker chooses probability for picking 1 from a range on $Q = (p; 1]$. More specifically, $[0.7, 0.9]$ is used here. That means q is the probability the attacker chooses 1, and $1 - q$ is the probability he chooses 0. Recall one of the requirements of the model is that p is distant from Q relative to the point $q*$ [9][Assumption A4, Section 3.1].

The defender, on the other hand, can only choose from deterministic ($\varphi : x^n \to \{0, 1\}$) decision functions. That means he chooses a threshold for the number of ones present in the vector. In the binary setting, that is equal to setting a threshold on $\frac{q(x^n)}{p(x^n)}$.

The cost for choosing a specific q is set as $|0.8 - q|$. That also implies the value of $q*$, which is the attack with the lowest cost, is equal to 0.8.

When it comes to the Neyman-Pearson framework, $\epsilon = 0.1$ is chosen. Note that this binary setting also simplifies the computation of both $p(x)$ and $q(x)$, which is further elaborated upon in the implementation section.

3.3.2 Summary

- $X = \{0, 1\}$ (the alphabet is binary)
- $d = 2$ (size of the alphabet)
- $p = 0.5$ (a non-attacker has equal probabilities between choosing 0 and 1)
- $Q = [0.7, 0.9]$ (the attacker's space is a range of the real axis)
- $q* = 0.8$
- and $c(q) = |q* - q|$
- $\varphi : x^n \to \{0, 1\}$ (only deterministic, threshold based decision functions are considered)
- $\theta = 0.5$ for the Bayesian framework
- $\epsilon = 0.1$ for the Neyman-Pearson framework

3.4 Expected results

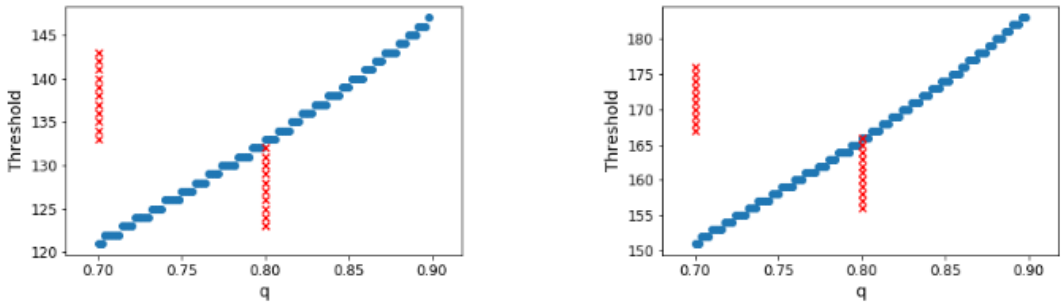
3.4.1 Convergence

As mentioned before, it is proven, in the paper, the classification error decreases as $n \to \infty$, when the defender's strategy is in the NE.

It is also shown that the optimal strategy for the attacker converges to a distribution more and more resembling $q*$. In this setting, that means the attacker's choice converges to the pure strategy $q = 0.8$.

Similarly, the defender's strategy tends to converge to a threshold somewhere near $\frac{2}{2}n$. Remember that its precise value is dependent on factors such as the value of γ (favoring one type of classification error over another), and n (observed vectors represent better the underlying distribution because of the *law of large numbers*).

To be exact, this is which best responses the model's authors found and what we want to replicate using the Double oracle algorithm. Red crosses are the attacker's and blue dots are the defender's best responses in Figure 3.1. The plots are taken from the paper [9].



(a) $n = 200$ (b) $n = 250$

Figure 3.1. Best response plots for $c(q) = |0.8 - q|$ under the Bayesian framework, taken from the referenced work [9][Appendix B.12]

Under the Neyman-Pearson framework, the threshold tends to be much lower. While, for this framework, no analogy of plots in Figure 3.1 is provided in the paper, a different plot is provided - the attacker's optimal strategy in dependence on the value of n.

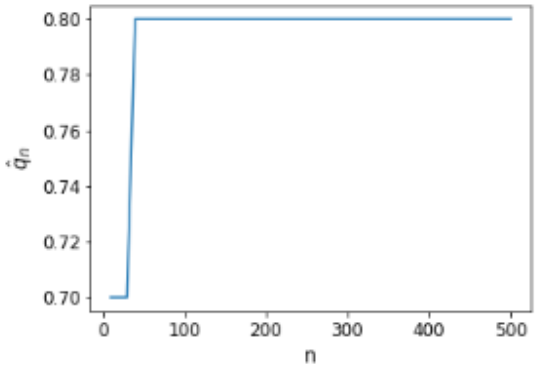


Figure 3.2. Attacker's strategy as a function of n under the Neyman-Pearson framework, taken from the referenced work [9][Appendix C.2]

We are interested in a question whether, when using the Double oracle algorithm, we can achieve the same results as the authors of the original paper did (using discretization).

3.4.2 Error rate and error exponent

Besides computing the strategies, we are also interested in the question of whether the solution gained by solving the model leads to a decreasing error rate as $n \to \infty$. Authors of the game model defined error rates in NE's as $e_n(q, \varphi) = -u_n^D(q, \varphi)$, which is basically the expected, weighted average of both types of errors. The *error exponent* plots were constructed by computing the NE under various n, getting the value of $e_n(q, \varphi)$ and transforming the value into an *error exponent* inside the Bayesian framework as

$$err\ exp^{Bayes} = -\frac{1}{n} log(e_n(q, \varphi)) = -\frac{1}{n} log(\sum_{x^n}(1 - \varphi(x^n))q(x^n) + \gamma \sum_{x^n} \varphi(x^n)p(x^n)\ )$$

and inside the Neyman-Pearson framework as

$$err\ exp^{Neyman} = -\frac{1}{n} log(e_n(q, \varphi)) = -\frac{1}{n} log(\sum_{x^n}(1 - \varphi(x^n))q(x^n)\ )$$

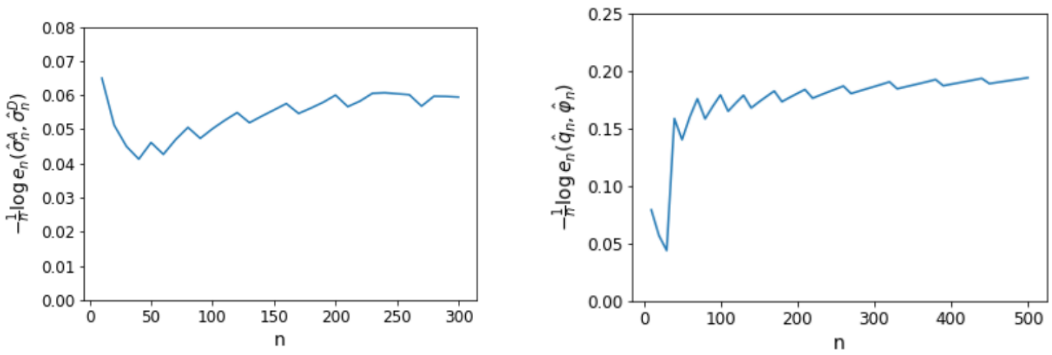


Figure 3.3. Error exponents in the Bayesian framework (left) and the Neyman-Pearson framework (right), taken from the referenced work [9][Appendix C.2]

Experimenting with the computation of this theoretical *error exponent* is not interesting, as this expected error rate is not dependent on a solving method. The Double oracle algorithm, therefore, cannot modify these plots in any way.

Instead, I will classify random data samples to test how the value of n influences the real-world error rate.

Chapter 4
Experiments and implementation

4.1 Basic environment and implementation information, Bayesian framework

4.1.1 Environment and transformation into binary setting

The project is implemented using the Scipy 1.3.3 and Numpy 1.17.4 inside the environment of Python 3.8. All of the implementation code is available at Git[1].

I have decided to start by implementing the game as a whole unit, and execute the experiments on it afterwards. That is in contrast to the approach of experimenting only with a specific section of the game model. Recall that

$$u_n^{eq}(q,\varphi) = \sum_{x^n}(1-\varphi(x^n))q(x^n) + \gamma \sum_{x^n} \varphi(x^n)p(x^n) - c(q)$$

Because of the binary setting, some parts of the computation model can be considerably simplified (using the NumPy symbolics) like this:

- as $q(x^n)$ depends only on the amounts of ones and zeros in the vector, the likelihood that the vector was generated by q can be implemented as $q(x^n) = q^{\sum x^n}(1-q)^{n-\sum(x^n)}$
- since $p = 0.5$, $p = (1-p)$, the likelihood, that the vector was generated by p, can be implemented as $p(x^n) = p^n = 2^{-n}$
- $\varphi(x^n, thr) = 1$ if $\sum x^n \geq thr$, 0 otherwise, as used in the original paper

Even after unfolding the game model in this way, implementing the game consistent with the model proved to be problematic. While the basic implementation worked, it was painfully slow for $n > 12$. That was mainly because iterating over all possible x^n for large n takes simply far too many steps. Since in every position of the vector could be either of the two binary values, there exist 2^n binary vectors of the length n. Computing the utility function thus reaches the time complexity $O(n \cdot 2^n)$

A critical realization here, again, is that the expression inside each sum depends on the number of ones $(= k)$ and zeros $(= n-k)$ only. Thus a vast amount of specific x^n iterations are redundant.

Have a look at the first sum inside $u_n^{eq}(q,\varphi)$, that is $\sum_{x^n}(1-\varphi(x^n))q(x^n)$. Both $\varphi(x^n)$ and $q(x^n)$ depend only on the number of ones and zeros in the vector (or the vector's length n, as we can compute any third number from the remaining two).

For all vectors of size n with k ones in it, the member $(1-\varphi(x^n))q(x^n)$ equals to the same number. We can compute the value for just one such vector x_k^n, and multiply it by the amount of all vectors x_k^n, instead of computing the function for all of them. And as the number represents the amount of possibilities of rearranging k ones in n positions, that number is $\binom{n}{k}$, the binomial coefficient.

[1] `https://gitlab.fel.cvut.cz/kasltoma/bachelorthesis/`

Thus, we can see that:

$$u_n^{eq}(q,\varphi) = \sum_{x^n}(1-\varphi(x^n))q(x^n) + \gamma \sum_{x^n} \varphi(x^n)p(x^n) - c(q) =$$

$$= \sum_{k=0,\dots,n} \binom{n}{k}(1-\varphi(x^n))q(x^n) + \gamma \sum_{k=0,\dots,n} \binom{n}{k}\varphi(x^n)p(x^n) - c(q)$$

Notice that the game model, in a binary setting like this, is quite in resemblance with the classical binomial distribution, only expressions representing the agents' strategies are added:

$$\binom{n}{k}q^k(1-q)^{n-k} = \binom{n}{k}q(x^n)$$

Also, instead of computing with the actual binary vectors, we can define a structure, which holds the information

- n - the length of the vector
- k - the count of ones

and iterate over instances of this structure instead.

4.1.2 Implementation confirmation

By reducing the time complexity to $O(n)$ like this, we can look at how these individual functions look inside this model's implementation (compared to the reference):

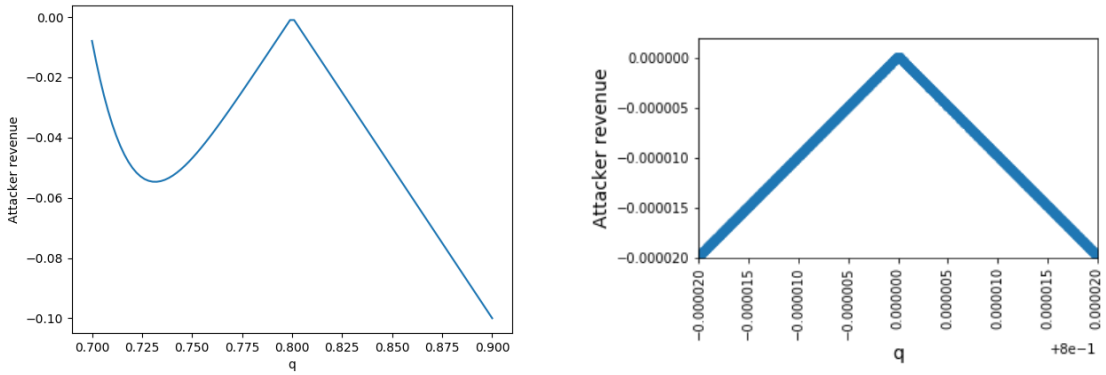


Figure 4.1. Attacker's cross-section utility function with the defender's threshold set. Output of this implementation (left), and the reference plot around $q*$ provided in the paper (right) [9][Section B.12] .

The defender's threshold for φ is set to the expected optimum. As we can see in Figur 4.1, near $q* = 0.8$, the function resembles the cost function (which is $c(q) = |q - 0.8|$). That outcome is consistent with the reference. Notice that as q decreases, the attacker's probability to be detected does so too, and that pushes his expected utility up again. Consistently with an intuition, whenever the defender lowers the threshold too much, q_{min} becomes the preferred strategy.

We can also plot the analogous figure (Figure 4.2) for the defender:

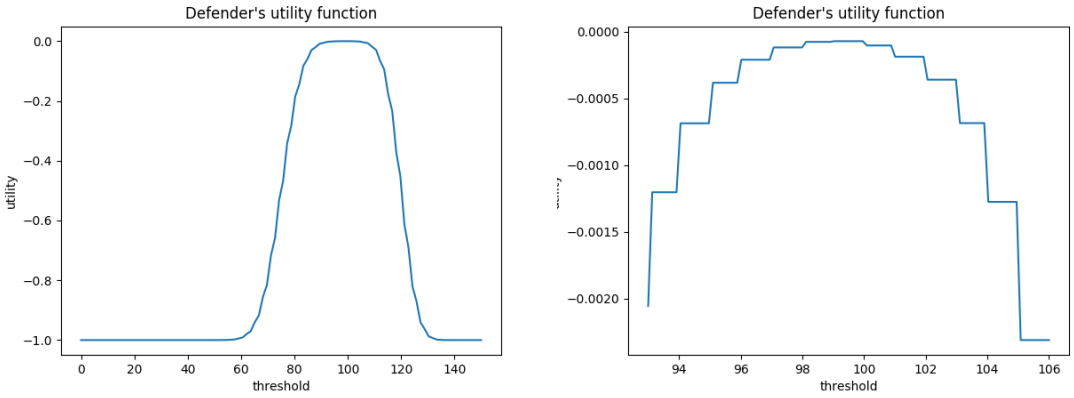


Figure 4.2. Defender's cross-section utility function with the value of q set to $q*$. The output of this implementation (left) and zoomed plot of the maximum (right).

The expected utility remains constant outside the range between $p \cdot n$ and $q_{max} \cdot n$, reaching the maximum at, as expected, 99, that is $0.66 \cdot 150$. Notice that this is almost exactly the average value of p and $q*$. Recall that the model has set $\gamma = 1$, which makes both types of errors equally costly. That raises a question of how big influence γ has on the defender's strategy and if, potentially, could his strategy be simplified to a (possibly linear) function of γ. This question will be addressed in Section 4.4.

4.2 Deploying the Double oracle

The algorithm is implemented, as it is explained in Section 2.5.

Finding the optimal pure strategies is done by appropriate SciPy optimization functions, depending on the number of variables and whether boundaries are needed. Brent's method is used for single-variable optimization, as it reliably finds optimum within specified boundaries (even when there are multiple local minima). For unbounded optimization, the Nelder-Mead method is used. Sequential Least Squares Programming is used for multi-variable optimization, as its NumPy implementation provides convenient optimization performance. Also, finding optimal mixed strategies from $M*$ is done by SciPy LP-solving function *linprog*, using the interior-point algorithm.

An important question here is a floating-point machine precision: These optimization functions might return slightly different values, because of numerical reasons, for the same problem and its solution. The algorithm could then endlessly iterate, with no justifiable reason, over strategies with ridiculously small differences. How many decimals must two numbers share to proclaim them as identical?

Since nothing specific is known about the Double oracle's convergence and its dependence on the machine precision, I chose these, as they are sufficient and also beyond a border of precision-consideration of individual agents:

- threshold for φ has precision set to 0.1
- q has precision set to 10^{-6}
- for Chapter 4.5, the precision is set to 0.01 for sigmoid φ and 0.1 for linear φ

More information about the impact of changing the machine precision can be found in Section 4.3.1.

Even then, the optimizing functions occasionally (depending on the initial randomly chosen strategies) output an error, stating that no definitive solution can be found, because of numerical reasons. Results are found most of the time, though.

We can now test whether, in this setting, these optimizing functions give the expected results for agents' best responses (Figure 4.3):

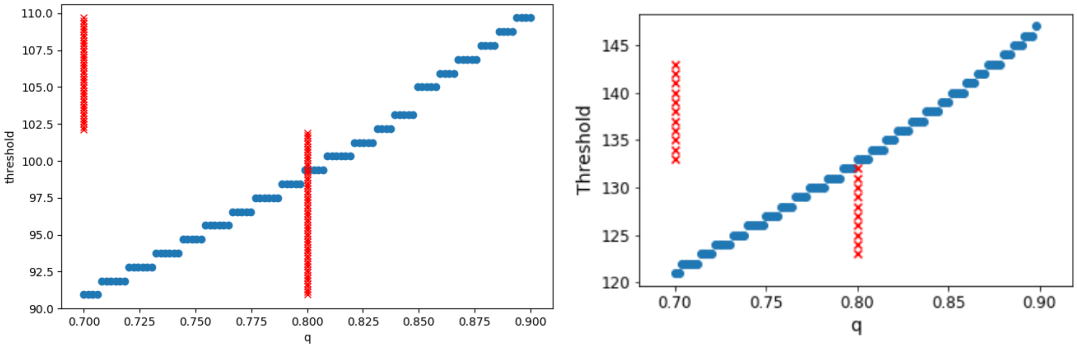


Figure 4.3. Mutual best responses: my implementation (left) and the original implementation [9] (right) - the same picture as in Figure 3.1

4.3 Double oracle results

4.3.1 Convergence and speed

Let's answer the essential question: Does the Double oracle algorithm, applied to this particular setting, converge?

We can illustrate the behavior of DO algorithm with the following Figure 4.4:

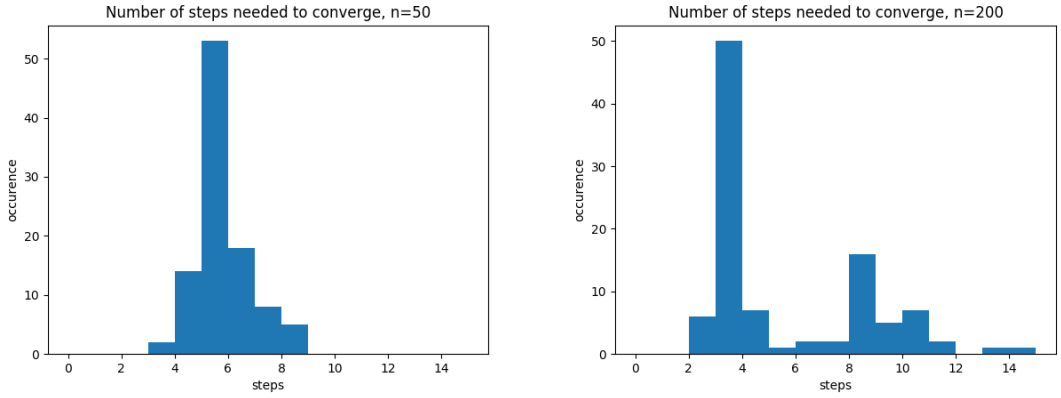


Figure 4.4. Speed of convergence expressed by iteration steps required for different n. Sampled from 100 game runs for each value of n.

As we can see, the algorithm converges to a solution in just a few steps, most of the time under only five iterations. And practically always under 15 iterations, making it a surprisingly reliable and fast method.

An interesting note here is that the convergence time (on average) drops down as n increases. The *law of large numbers* can explain that: As n increases, the percentage of ones inside a vector generated by the probability distribution q more and more resemble the actual value of q, and less oscillation happens.

As an example, I have also compared, on my personal computer, the runtime of solving the game by the Double oracle for different floating-point precision of the agents' pure strategies. For each amount of decimals, the DO algorithm was initiated 100 times (with a value of n set to 150).

Decimals (attacker;defender)	(2;0)	(6;1)	(8;3)	(10;5)	(14;14)
Time [s]	17	25	38	111	239
Threshold	99	99.4	99.375	99.375	99.375

Table 4.1. Time of convergence for different amounts of rounding decimals, $n = 150$.

By settling for the precision proposed in Section 4.2, the same outcome of the game is computed in just a fraction (on average) of the time needed when compared to the maximal precision the 64-bit machine can provide.

4.3.2 Optimal strategies

Another question is whether the claim, that both optimal mixed strategies (in NE) converge to pure strategies (Section 3.4.1), holds. The expected pure strategies are, as already stated, $q*$ for the attacker, and around $\frac{2}{3}n$ as the threshold for the defender.

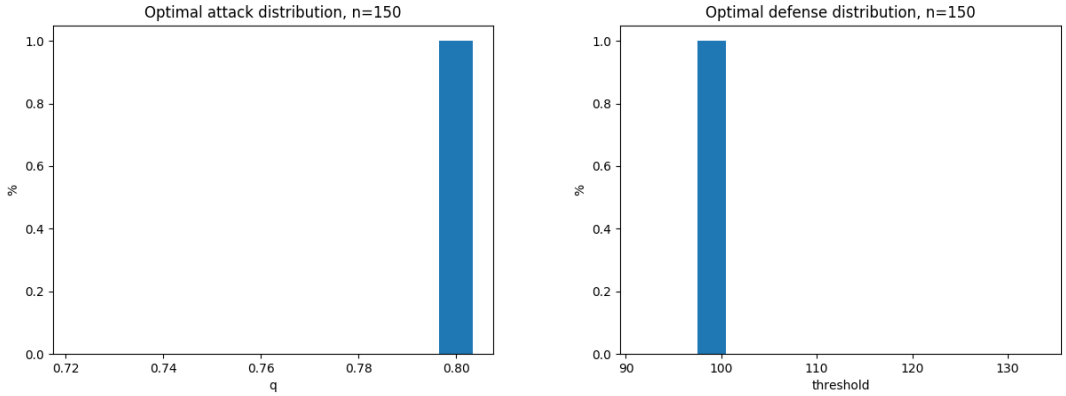


Figure 4.5. Optimal mixed strategies, to which both agents converge - using the Double oracle, $n = 150$

Indeed, both agents converge to always choosing a pure strategy when using the Double oracle. Moreover, they converge to the values stated in the paper.

Now, when it has been shown that the algorithm converges to pure strategies, the question remains whether the algorithm actually converges to the expected value even for different values of n.

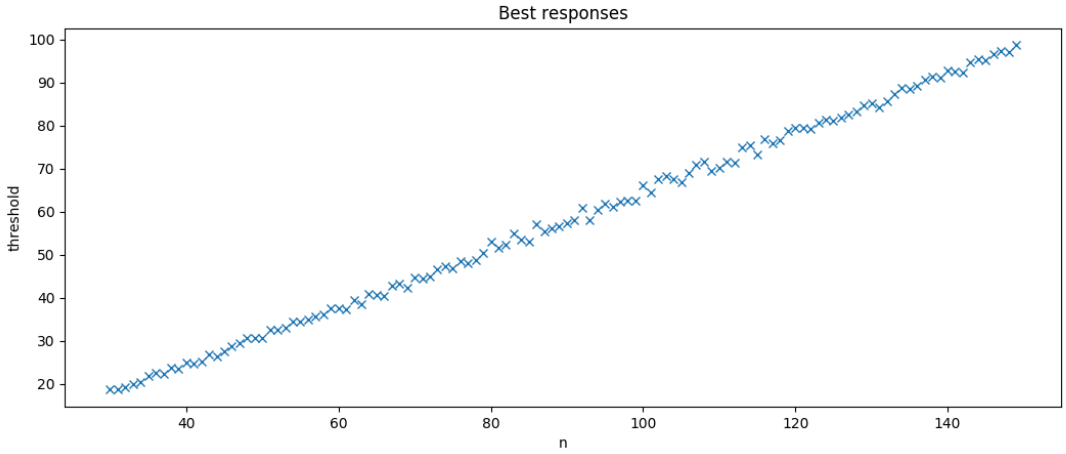


Figure 4.6. Defender's optimal strategy as a function of n.

In the case of this game, the Double oracle algorithm reaches the correct optima. It shows that the Double oracle algorithm can perform reasonably well for this continuous game.

4.3.3 Data classification and error rates

The original definition of error rate as $e_n(q, \varphi) = -u_n^D(q, \varphi)$ cannot be used for examining the real error rate of real-world data classification. Recall that

$$u_n^D(q, \varphi) = -(\sum_{x^n}(1 - \varphi(x^n))q(x^n) + \gamma \sum_{x^n} \varphi(x^n)p(x^n))$$

Since the error rate is defined as a weighted sum of expected average error rates of both error types, it cannot be used for measuring the error rate of real-world classification. Therefore, I will redefine it (for real-world classification) as the weighted sum of occurrences of the classification errors:

False negative rate is

$$FN(X^n, labels) = \frac{\sum_{x^n \sim q}[[\varphi(x^n) == 0]]}{\sum_{x^n \sim q} 1}$$

and analogously false positive rate is

$$FP(X^n, labels) = \frac{\sum_{x^n \sim p}[[\varphi(x^n) == 1]]}{\sum_{x^n \sim p} 1}$$

which gives the final, balanced error rate:

$$E_n^{Bayes}(X^n, labels, \gamma) = \frac{FN(X^n, labels) + \gamma FP(X^n, labels)}{1 + \gamma}$$

Naturally, the classification error exponent then is

$$Err\ Exp^{Bayes} = -\frac{1}{n} log(E_n^{Bayes}(X^n, labels, \gamma))$$

Recall that we have set $\gamma = 1$, i.e., the prior probability of an adversary's presence is $\theta = 0.5$, and both kinds of errors have the same weight.

Values of n are between 10 and 300, and its value is always increased by 10. For each value of n, NE strategies are computed. The resulting classifier then classifies $5 \cdot 10^5$ vectors, of which $2.5 \cdot 10^5$ generated under $q*$ and $2.5 \cdot 10^5$ generated under p.

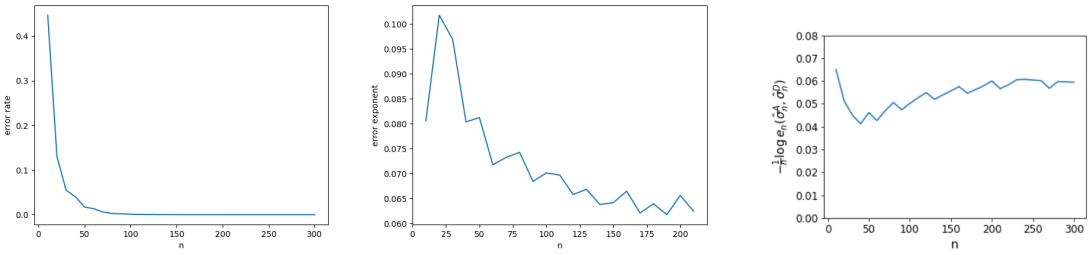


Figure 4.7. Error rate for different values of n: classification error rate (left), classification error exponent (middle) and expected error exponent (right). The last plot is taken from the paper [Section 6][9].

The classification error rate, as shown (Figure 4.7), goes to 0 as $n \to \infty$. Also, the error exponent converges to a value of around 0.06, exactly as it was proposed. Error exponent for $n > 200$ is missing from the plot because the error rate was exactly 0, which renders the logarithm uncomputable. The Double oracle algorithm, therefore, can provide us with a reliable classifier.

4.4 Influence of γ on the utility function

An objection might be raised that γ (which is the scalar used for balancing the defender's cost for making either of the two types of classification error) might have an overly strong influence on the defender's optimal strategy. Could this game model possibly be simplified to a simple function of γ? To answer this, let's see the defender's utility function (with q set to $q*$) when the value of γ is changed (Figure 4.8).

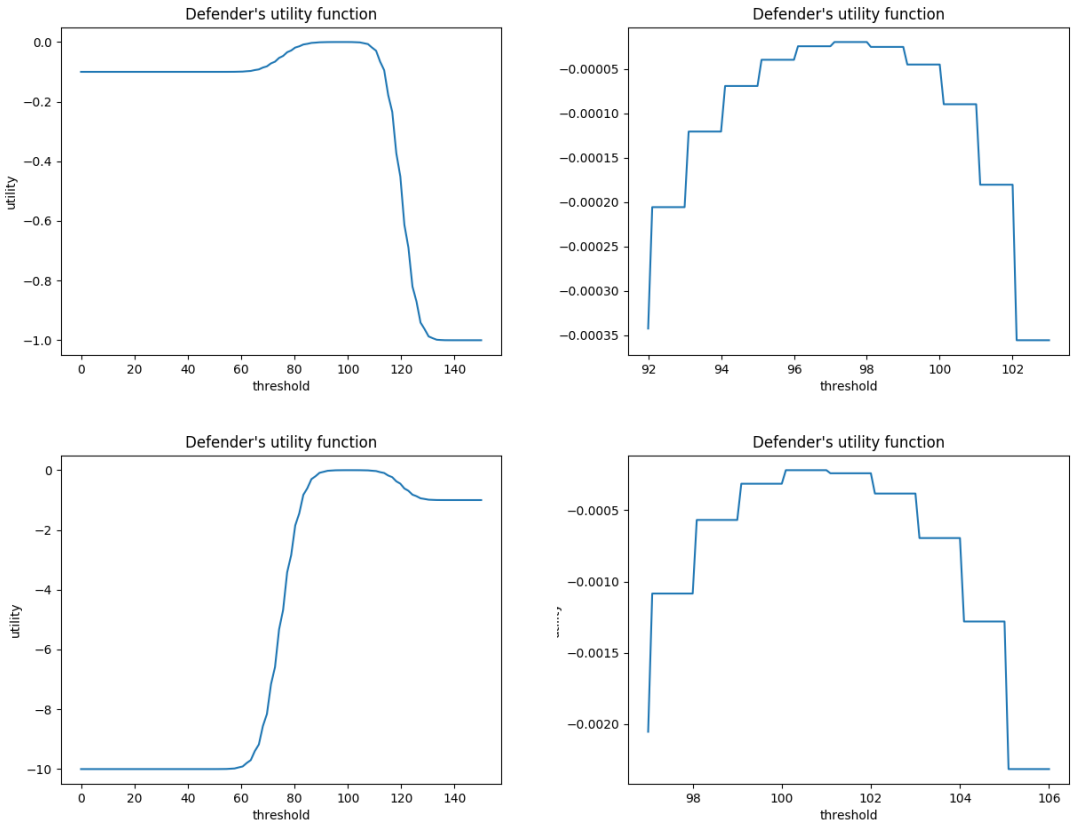


Figure 4.8. Defender's utility function when γ is changed from 1 to 0.1 (left) and 10 (right)

The value of γ has a noticeable influence on the utility function outside the essential range of $(0.5n; 0.9n)$. However, changing its value, and thus rebalancing the cost of the two types of error, has just a minor effect on the optimal defense strategy, and the optimal attack strategy does not change at all.

As the change to the optimum is negligible, we might abandon the idea. It is not possible to simplify the defender's optimal strategy as just a function of γ, (therefore skipping the attacker's role in the game model) in a reasonable manner.

4.5 Expanding upon φ

4.5.1 Argument

One of the design choices of the model is the decision function φ. The threshold-based binary function is the reason why utility functions look as piece-wise constant. More importantly, when the attacker knows the trigger threshold, he can choose appropriate q and just cap the percentage of ones in the binary vectors (by delaying the next action whenever the cap has been reached, for example). This would let him exploit the system as well as remain undetectable by such a defense system.

Recall that the initial model specifies the function φ as a map to the range $[0; 1]$, that is to a probability to classify the agent as an adversary. Here, for modifying the decision function, two options naturally arise:

- a sigmoid function, that is $\varphi(x^n) = \frac{1}{1+exp(-\alpha\frac{p(x^n)}{q(x^n)}+\beta)}, \quad \alpha, \beta \in \mathbb{R}$
- a linear function $\alpha\frac{k}{n} + \beta$, that is

$$\begin{aligned} \varphi(x^n) &= \alpha\tfrac{k}{n} + \beta, && \text{if } (\alpha\tfrac{k}{n} + \beta) \in [0; 1], \\ \varphi(x^n) &= 0, && \text{if } (\alpha\tfrac{k}{n} + \beta) < 0, \\ \varphi(x^n) &= 1 && \text{otherwise}, \quad \alpha, \beta \in \mathbb{R} \end{aligned}$$

An important implication of modifying the game model this way is that, while there is no change for the attacker, the algorithm must now optimize the defender's strategy by iterating over the two new variables. That means the defender now looks for the best tuple $(\alpha; \beta)$ instead of $threshold \in N$. Since we moved from optimizing a single-input function into optimizing a two-input function, the model is now far more computationally demanding.

Surely enough, the algorithm (after modifying the implementation accordingly) converges for both of these modifications. Let's compare the plots of these different φ with their respective optimal values:

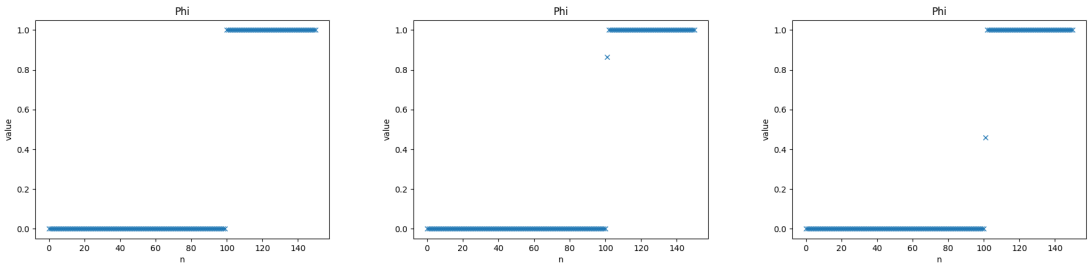


Figure 4.9. decision function φ functions: original (left), sigmoid (middle) and linear (right)

These are the plots (Figure 4.9) for the different possible φ functions when their parameters are set to the values the Double oracle algorithm converges to.

Observe that both the sigmoid and linear functions are very close in shape to what the threshold-based φ, which might not be quite intuitive.

4.5.2 Results of using the sigmoid decision function

Firstly, this model also converges to playing pure strategies only, as shown here.

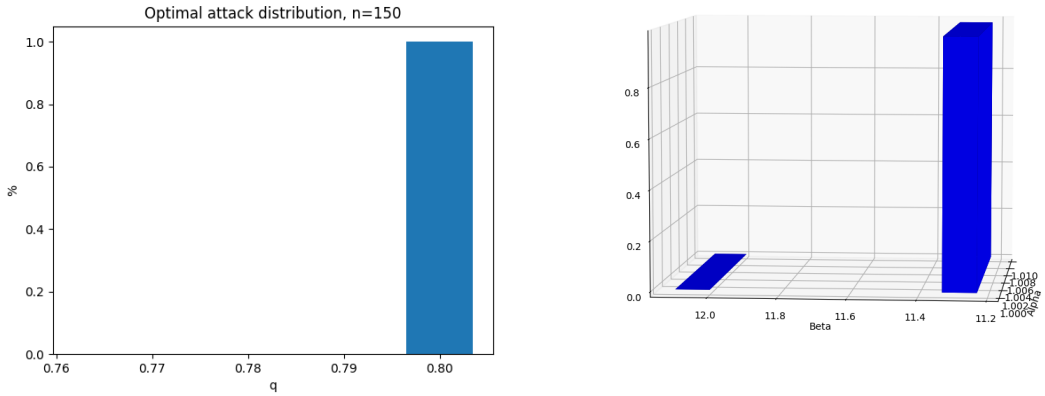
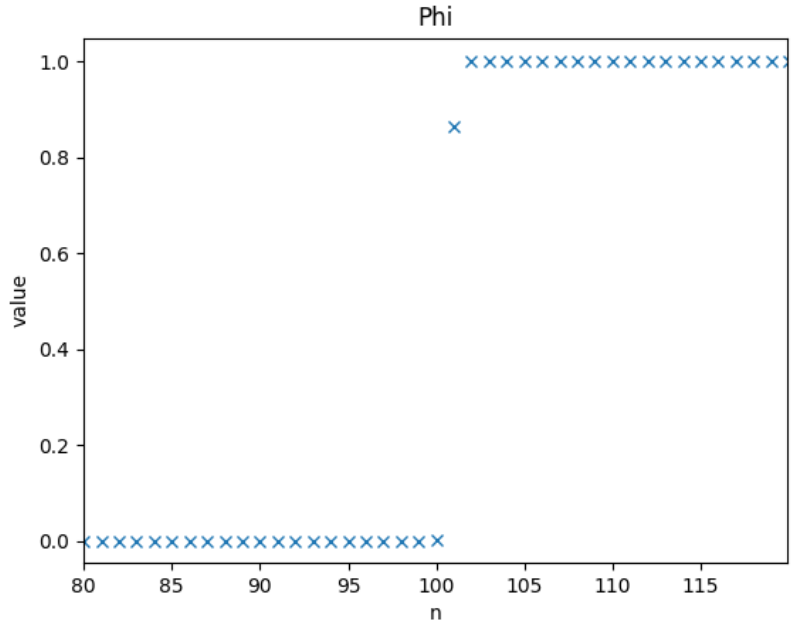


Figure 4.10. Optimal distribution for $\varphi(x^n) = \frac{1}{1+exp(-\alpha\frac{p(x^n)}{q(x^n)}+\beta)}$

Final attacker's strategy ends up being identical when compared to the original model. The optimal defense strategy converges to $(\alpha;\beta) = (1\ ;\ 11.23)$. That means his optimal decision function is $\varphi(x^n) = \frac{1}{1+exp((-1)\frac{p(x^n)}{q(x^n)}+11.23)}$. Let's see how that translates into the actual mapping $x^n \to [0;1]$. For that have a look at the zoomed φ plot again:



The mapping remains almost the same, as huge majority of x^n ends up (with negligible rounding) in the binary set $\{0;1\}$. Three only exceptions, in our case (where $n = 150$), are:

- $\varphi(x^{n=150}_{k=99}) = 0.00003$
- $\varphi(x^{n=150}_{k=100}) = 0.00035$
- $\varphi(x^{n=150}_{k=101}) = 0.8657$

Therefore, $k = 100$ is still practically allowed for the attacker by this new model in comparison with the original. That is redeemed by the fact that the unknown agent's

classification as an adversary is uncertain for the higher value of $k = 101$. Nonetheless, an adversary cannot stay undetected while choosing these distributions, indefinitely - even when settling for $k = 100$, the attacker will (eventually) be classified as an adversary.

Since we have now moved (in the defender's case) from optimizing over one variable into optimizing two variables, we might also be interested in how the speed convergence changes. A natural expectation would be, in this modified model, that the number of steps needed for convergence increases.

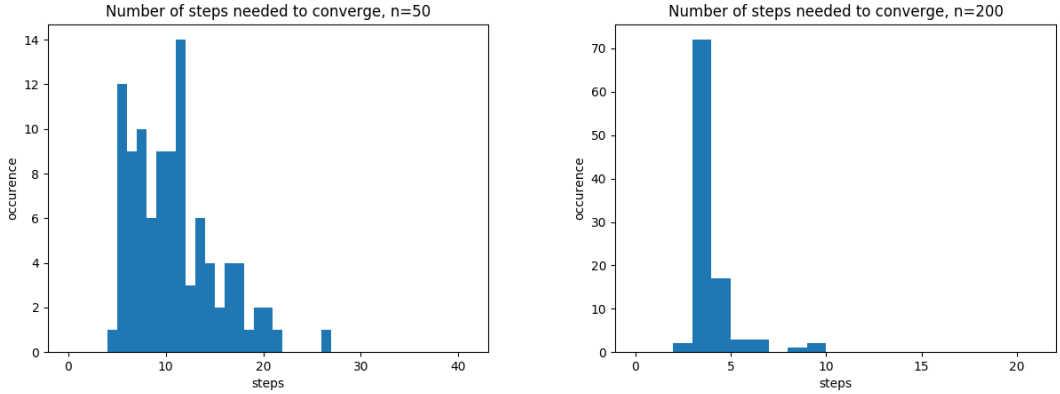


Figure 4.11. Speed of convergence, with the sigmoid decision function, expressed by iteration steps required for different values of n. Sampled from 100 game runs for each n.

As shown in Figure 4.11, while the convergence speed for a lower value of $n = 50$ decreased drastically, as the majority of samples needed over ten iteration steps to converge, it did not decrease for the case of higher n. Actually, for $n = 200$, the convergence appears to be faster than when using the threshold-based φ.

Another interesting thing to note then is that as the value of n increases, not only does the classifier become more precise, the convergence speed increases as well. One would then think increasing n is a win-win situation. However, it proved not to be the case, as ever-increasing n is also met with growing numerical difficulties, and the chance of encountering an optimization error rises.

Should you, therefore, find a way of mitigating these numerical issues, increasing the value of n as high as possible appears to be optimal. Remember, however, that classification of the unknown agent should be performed fast enough so that an attacker does not have much time to cause harm before he is inspected.

Also, we can compare the classification error rate of the model with a sigmoid decision function with the original, threshold-based one. Specifics of the classification are the same, n is between 10 and 300, and it is always increased by 10. For each value of n, exactly $5 \cdot 10^5$ samples are classified.

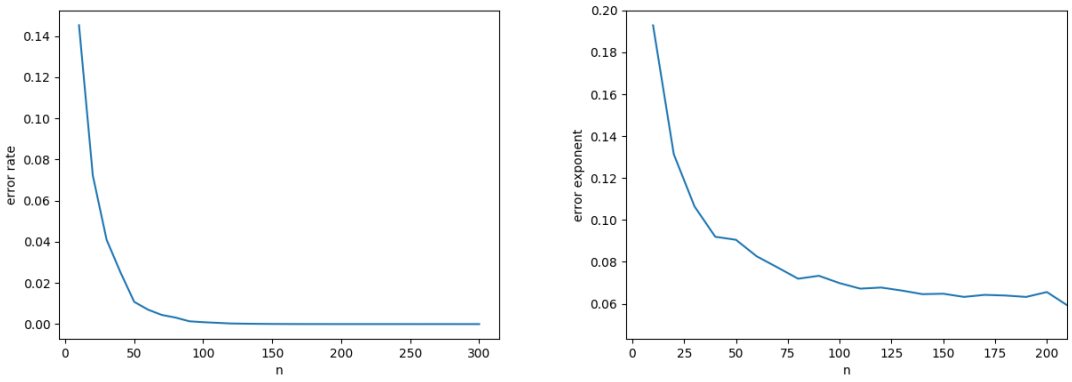


Figure 4.12. The classification error rate with a sigmoid decision function (left) and corresponding classification error exponent (right).

We can see (Figure 4.12) that the classification error rate converges to zero faster with the sigmoid decision function than the original game model. Nonetheless, it does converge to zero, as well as the error exponent converges to 0.06.

4.5.3 Results of using the linear decision function

Perhaps surprisingly, the algorithm still successfully converges to a strong resemblance with the threshold-based function. And also this time, both agents settle on always playing a specific pure strategy.

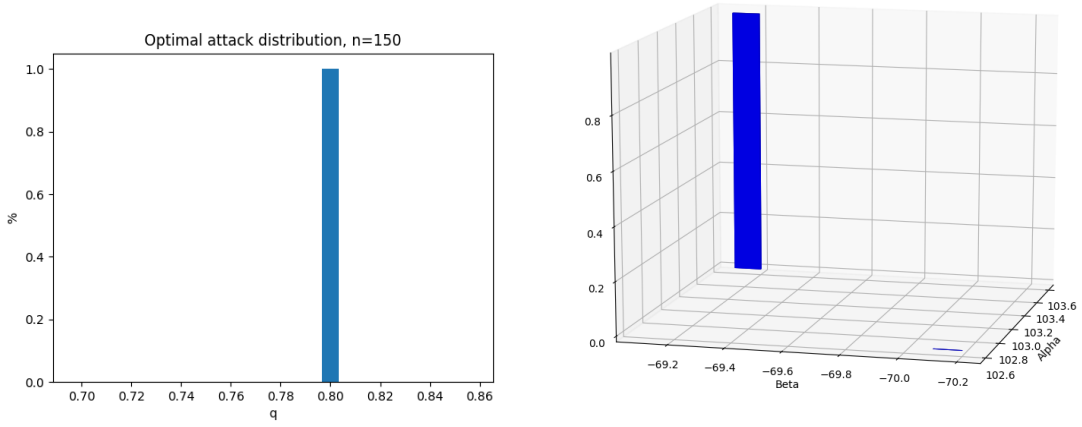


Figure 4.13. Optimal strategy distributions for bounded $\varphi(x^n) = \alpha\frac{k}{n} + \beta$

As shown in Figure 4.13, the optimal strategy of the attacker is again unchanged. When it comes to the defender, the optimal strategy now is $(\alpha; \beta) = (103.6; -69.2)$. The decision function in this case (for $n = 150$) then is $\varphi(x^n) = \frac{103.6 \cdot 101}{150} - 69.2$.

An important note here is that this time, the algorithm does not always converge to the same value. Instead, depending on the random initial pure strategies chosen, the DO algorithm converges to a value of α in the range $[103, 104]$, and value of β in the range $[-69.5, -68.5]$

Let's now have a look at the zoomed linear φ (where the values are $(103.6; -69.2)$) plot again: (Figure 4.14)

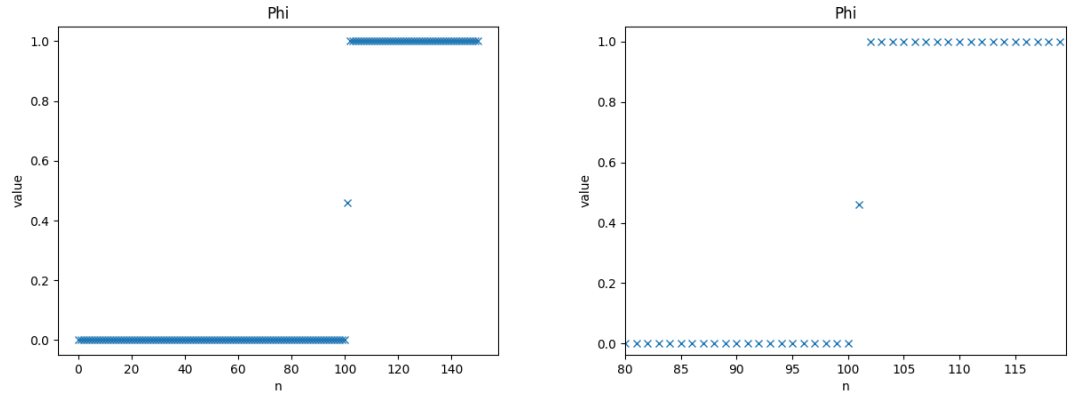


Figure 4.14. Plot of bounded $\varphi(x^n) = \alpha\frac{k}{n} + \beta$

On closer inspection, however, the slight changes in values of variables α and β do not change the final shape of the function at all. With the exception of $k = 101$, all values of k lead to output 0 or 1, only $\varphi(x^{n=150}_{k=101}) \sim 0.5$. Multiple different tuples of $(\alpha; \beta)$ lead to the same mapping of the finite vector set.

As it seems, this simpler model (when compared to the sigmoid) is perhaps the closest to fulfilling the basic adversarial classification intuition. That is, there should not be an attacker's *sweet spot* right below the threshold, in which he is undetectable by the system. But while the value of $k = 101$ is 50/50 on par, lower values of k again lead to the certainty of labeling the vector as not harmful.

Here, we can also compare the classification error rate of the classifier gained from the model with a linear decision function with the other ones. Specifics of the classification are the same:

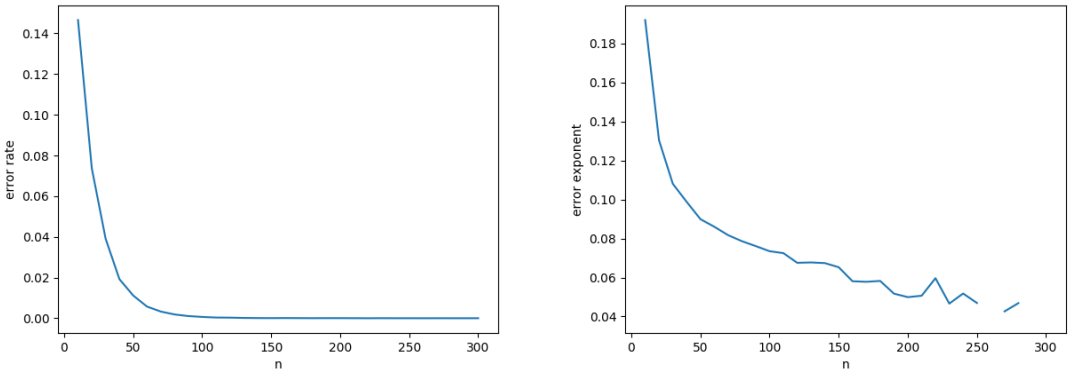


Figure 4.15. The classification error rate with a linear decision function (left) and corresponding classification error exponent (right).

Also this time, the error rate converges to zero, and the error exponent converges to a value close to 0.06. It appears to converge slightly faster in comparison to the sigmoid function, but the difference is negligible. The two missing parts of the second plot in Figure 4.15 are caused by the fact that no incorrect classification was encountered for the specific values of n. It should be noted that this specific model faces a bit more numerical difficulties.

4.5.4 Error exponent comparison

Here, we can compare the classification error rates of all three models side by side (Figure 4.16).

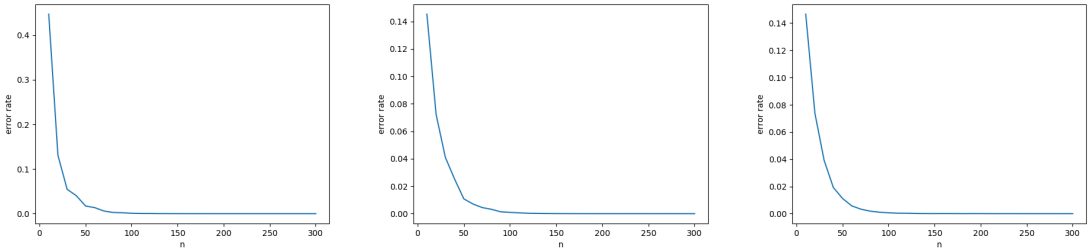


Figure 4.16. The classification error rate with a threshold-based decision function (left), sigmoid-based (middle) and linear (right) decision function

Perhaps unexpectedly, the game model with a linear decision function appears to perform the best, and also (for $k = 101$) is the only one to truly provide uncertainty. Nonetheless, the optimal non-deterministic decision functions converge to a form, which oddly resembles the deterministic one.

It seems clear, however, that continuous decision functions can provide considerably better accuracy if chosen reasonably.

4.6 Expanding upon the cost function

4.6.1 Argument

Let's start by reminding the utility function:

$$u_n^{eq}(q, \varphi) = \sum_{x^n}(1 - \varphi(x^n))q(x^n) + \gamma \sum_{x^n} \varphi(x^n)p(x^n) - c(q)$$

The authors of this model propose multiple various definitions of c, for example $c(q) = (q - 0.8)^2$. All of them are constrained by the prerequisite that there is a single optimum denoted as $q*$. But since around $q*$ is the utility function indistinguishable from $-c(q)$, the local minimum will always be at $q*$. Therefore, I think experimenting with the definition of the cost function itself is not interesting.

There might be, however, a different topic to be addressed. Notice that the attacker's cost for choosing a specific q does not depend on the length of the observed vector. That can be interpreted as considering only the initial cost of inventing an attack, while its usage is regarded as free.

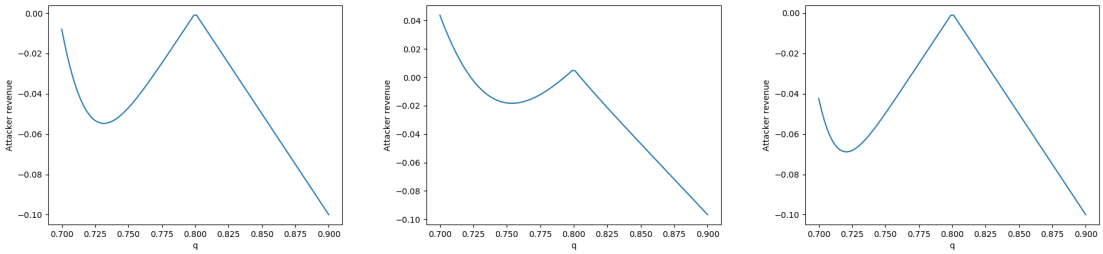


Figure 4.17. Dependence of (attacker's) utility function with various values of n, $n = 150$ (left), $n = 50$ (middle) and $n = 250$ (right).

These plots (Figure 4.18) of the attacker's cross-sections of the utility function shows how much the value of n affects the influence of c on the (attacker's) utility function. The reason, why under higher values of n the utility function resembles $-c$ more and more, is again the law of large numbers. More confidence can be put by the defender into the selected threshold.

The optimal strategy for both agents is not affected by n at all.

Intuitively, one would think that except for an initial cost for choosing a specific distribution q, there could be a cost for maintaining this q, too. However, since the value of n is set for the game, adding or multiplying the cost function by any function of n would always be the same, as if it was only a constant. Thus, the shape of the function nor the outcome would change, and no alteration of it is worth experimenting with.

4.7 Neyman-Pearson framework

4.7.1 Convergence

In the paper, there is also examined the classification model under the Neyman-Pearson framework.

The Neyman-Pearson framework implies limiting the false positive error count by the value of ϵ. That is in our case:

$$u_n^{eq}(q, \varphi) = \sum_{x^n} (1 - \varphi(x^n))q(x^n) - c(q)$$

subject to:

$$\Phi_n = \{\varphi : X^n \to [0, 1] : \sum_{x^n} \varphi(x^n)p(x^n) < \epsilon\}$$

Recall that we have set the value of $\epsilon = 0.1$.

Also in this framework the Double oracle converges to pure strategies (Figure 4.18):

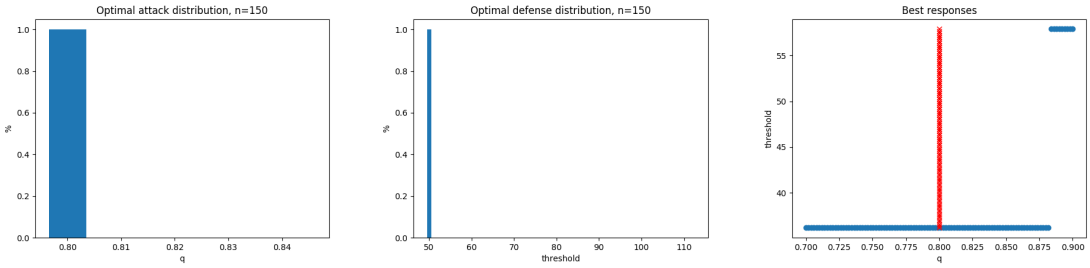


Figure 4.18. Optimal distributions under the Neyman-Pearson framework for the attacker (left) and defender (middle). Also the mutual best responses (right)

While the attacker's optimal strategy stays the same as in the Bayesian framework, the defender's threshold decreases. It is now much closer to $\frac{1}{3}n$.

4.7.2 Dependence on the vector length n

Naturally, the interest then shifts onto the question, how the optimal strategies depend on the vector length n. Remember, that one of the original questions was whether the Double oracle algorithm could replicate the relation between n and optimal q. The direct comparison is shown in Figure 4.19:

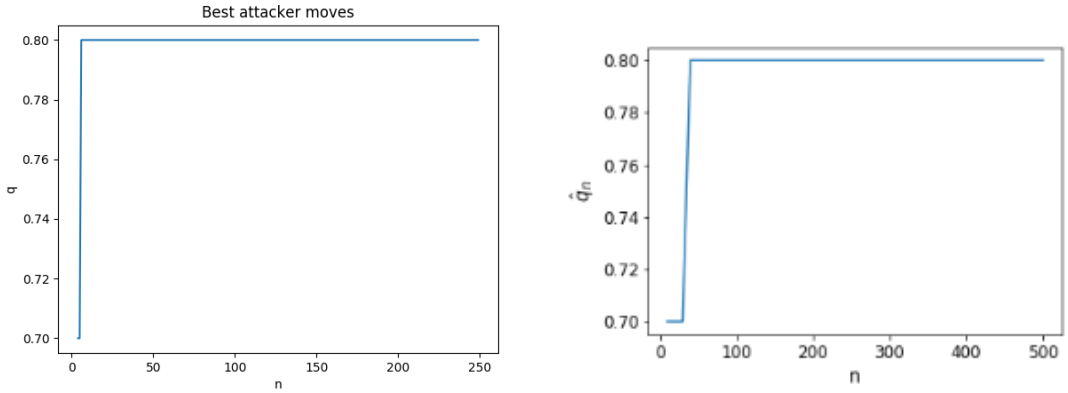


Figure 4.19. Optimal attacker strategies: in this model implementation (left) and reference (right) - the same picture as in Figure 3.2. Taken from the paper [9]

The basic plot structure is the same, and for the lowest possible values of n, the attacker chooses q_{min}. However, in contrast to the reference, the optimal q rises, in my implementation, to the value of $q*$ much faster. I am not sure what the reason for this is. But since the algorithm converges to the correct q for reasonable values of n, it doesn't impose a problem.

Figure 4.19 looks very reasonable when the attacker's utility function (Figure 4.20) is also considered, as it is indistinguishable from $-c$ here.

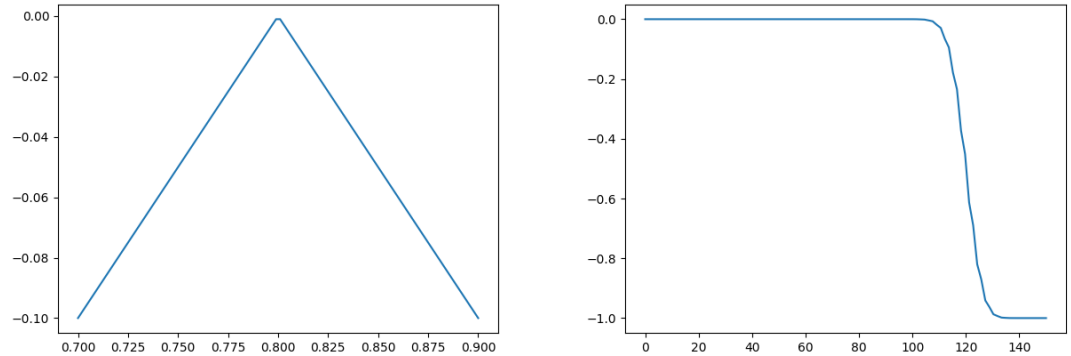


Figure 4.20. Attacker's (left) and defender's (right) utility cross-section functions under the Neyman-Pearson framework, $n = 150$.

We can also have a look at how the optimal defense strategy evolves as n increases. (Figure 4.21)

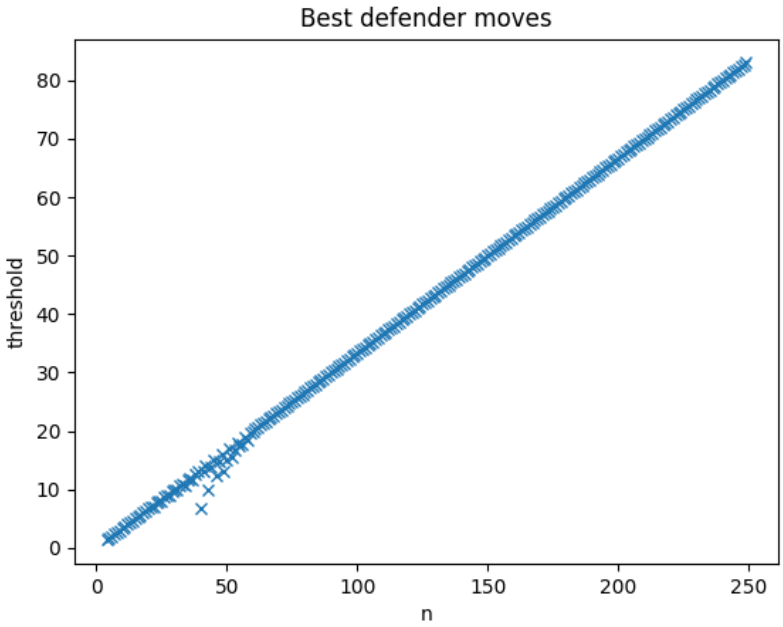


Figure 4.21. Optimal defender strategies for varying n

The optimal threshold happens to be a fraction of n this time, too. This time quite smaller, however, which appears to be fairly reasonable. In order to keep the false-positive count under a limit, the defender has to behave with much bigger caution, forcing his trigger threshold much lower.

4.7.3 Data classification and error rate

We are also interested in the error rate of a classifier created by solving this game model. First, let's start again by defining the error rate and error exponent. The expected error rate once again is $e_n(q, \varphi) = -u_n^D(q, \varphi)$.

In the Neyman-Pearson framework that means

$$e_n(q, \varphi) = -u_n^D(q, \varphi) = \sum_{x^n}(1 - \varphi(x^n))q(x^n)$$

To create its real classification error rate adaptation, we remind the definition of false negative error rate:

$$FN(X^n, labels) = \frac{\sum_{x^n \sim q}[[\varphi(x^n) == 0]]}{\sum_{x^n \sim q} 1}$$

Since the false-positive error type does not figure in the defender's utility function, it is not included in the classification error rate either. The classification error rate therefore is

$$E_n^{Neyman}(X^n, labels) = FN(X^n, labels) = \frac{\sum_{x^n \sim q}[[\varphi(x^n) == 0]]}{\sum_{x^n \sim q} 1}$$

Naturally, the classification error exponent then is

$$Err\ Exp^{Neyman} = -\frac{1}{n} log(E_n^{Neyman}(X^n, labels))$$

Values of n are again between 10 and 300, and its value is always increased by 10. For each value of n, NE strategies are computed. And 250000 adversarial vectors are again classified:

Here, we can see (Figure 4.22) the classification error rate of the model under the Neyman-Pearson framework, with the modified definition of what error rate is.

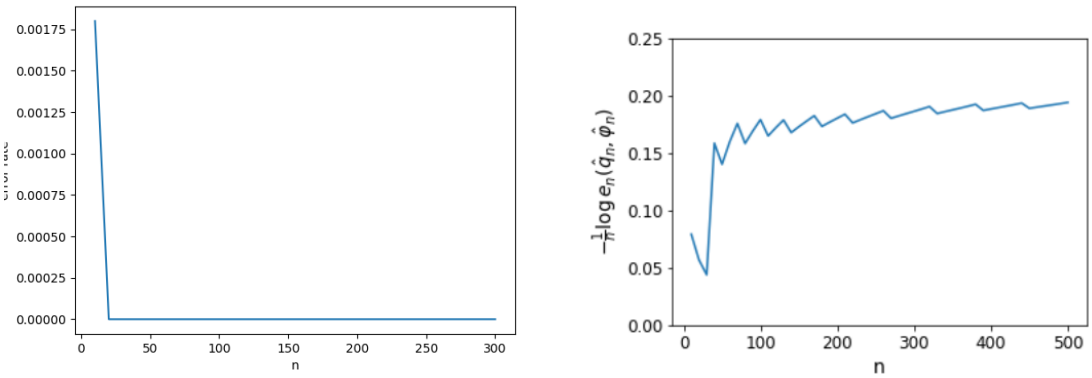


Figure 4.22. The classification error rate (left) and error exponent reference plot (right) in the Neyman-Pearson framework, from [9][Appendix C.2]

The error rate converges to zero with the increasing value of n this time, too. Because only one error type (the less likely one, actually, given the lower threshold) is considered, the error rate converges much faster. No reasonable error exponent plot can be provided for comparison with the reference. The speed of convergence renders the error exponent plot uncomputable, and so we cannot say the error exponent converges to the expected value. Perhaps, the real-world classification performs better than what the expected error rate suggests.

Chapter 5
Conclusion

5.0.1 Adversarial classification

The so-called adversarial classification is a domain that, along with the evolution of the Internet, is now graced with ever-increasing importance. New and more sophisticated ways of exploiting online environments are found continually, and so novel approaches to identifying adversaries are needed too. Because of the exploitative nature of attackers, static and artificial classifiers are increasingly difficult to justify, and so game-theoretic models gain popularity. As discussed in the thesis, though, they still have much of the needed progress ahead.

Together with the game-theoretic approach to the adversarial classification, game theory is gaining popularity also in related fields, like security games and anomaly detection games. These games are solved for optimal outcomes using similar methods and can be used together to construct complex online security systems.

5.0.2 Double oracle

It has been shown that using the Double oracle algorithm for continuous/infinite-space games is a reasonable and justifiable approach. The algorithm quickly converges to the correct solutions. Most of the time, only a few iterations are needed. The LP solved in each iteration is calculated fast, because its time complexity is bounded, and the matrices are small. The only difficulty, therefore, is optimizing over a section of the utility function to find the best responses.

It is also shown that the algorithm can handle optimizing over multi-variable strategy spaces when it is provided with a finely tuned optimization function.

One does, however, need to choose the right floating-point machine precision carefully. Otherwise, the algorithm might lose its efficiency.

5.0.3 Model discussion

Even though the basic principle of the model, the definition of strategies as distributions over distributions, is very unintuitive, it can provide us with a well-performing classifier. As tested, continuous decision function (and the uncertainty it presents) yields much better results when compared to deterministic ones.

Also, pairing this model with the Double oracle algorithm appears to favor high values of n for both classification accuracy and computation time.

An implementer of this model, however, is still required to come up with a robust cost function for an arbitrary attack, which can be a difficult task.

References

[1] Prithviraj Dasgupta, and Joseph B. Collins. *A Survey of Game Theoretic Approaches for Adversarial Machine Learning in Cybersecurity Tasks*. 2019. https://arxiv.org/pdf/1912.02258.pdf. In: AI Magazine, Vol. 40, N^o. 2, pages 31-43.

[2] Micaela Troglia Gamba, Minh Duc Truong, Beatrice Motella, Emanuela Falletti, and Tung Hai Ta. *Hypothesis testing methods to detect spoofing attacks: a test against the TEXBAT datasets*. 2017. https://doi.org/10.1007/s10291-016-0548-7. In: GPS Solut 21, pages 577–589.

[3] Antoine Delplace, Sheryl Hermoso, and Kristofer Anandita. *Cyber Attack Detection thanks to Machine Learning Algorithms*. 2019. https://arxiv.org/pdf/2001.06309.pdf. May 17.

[4] Mauro Barni, and Benedetta Tondi. *Binary Hypothesis Testing Game with Training Data*. Aug. 2014. https://arxiv.org/pdf/1304.2172.pdf. In: IEEE Transactions on Information Theory, Vol. 60, N^o. 8, pages 4848-4866.

[5] Ion Androutsopoulos, Evangelos F. Magirou, and Dimitrios K. Vassilakis. *A Game Theoretic Model of Spam E-Mailing*. 2005. http://nlp.cs.aueb.gr/pubs/ceas2005_paper.pdf. Conference: CEAS 2005 - Second Conference on Email and Anti-Spam, July 21-22, 2005, Stanford University, California, USA.

[6] Sixie Yu, Yevgeniy Vorobeychik, and Scott Alfeld. *Adversarial Classification on Social Networks*. July 2018. https://arxiv.org/pdf/1801.08159.pdf. Conference: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems(AAMAS 2018), Pages 211–219.

[7] Lemonia Dritsoula, Patrick Loiseau, and John Musacchio. *A Game-Theoretic Analysis of Adversarial Classification*. 2017. https://arxiv.org/pdf/1610.04972.pdf. In: IEEE Transactions on Information Forensics and Security, Vol. 12, N^o. 12, pages 3094-3109.

[8] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. *Adversarial Machine Learning at Scale*. 2017. https://arxiv.org/pdf/1611.01236.pdf. Conference: Published as a conference paper at ICLR 2017.

[9] Sarath Yasodharan, and Patrick Loiseau. *Nonzero-sum Adversarial Hypothesis Testing Games*. 2019. https://arxiv.org/pdf/1909.13031.pdf. In: Advances in Neural Information Processing Systems, pages 7310 – 7320.

[10] David Silver, Thomas Hubert1, Julian Schrittwieser, Ioannis Antonoglou1, Matthew Lai1, Arthur Guez, Marc Lanctot, Laurent Sifre1, Dharshan Kumaran,

Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. *A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play*. 2018.
https://arxiv.org/pdf/1712.01815.pdf. In: Science 07 Dec 2018: Vol. 362, Issue 6419, pages 1140-1144.

[11] John von Neumann, and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1953. ISBN 9780691041834.
http://jmvidal.cse.sc.edu/library/neumann44a.pdf.

[12] Jonathan Newton. *Evolutionary game theory: A renaissance*. 2018.
http://dx.doi.org/10.3390/g9020031. In: Games, ISSN 2073-4336, MDPI, Basel, Vol. 9, Iss. 2, pages 1-67.

[13] Nolan McCarty, and Adam Meirowitz. *Political Game Theory*. Princeton University Press, 2008. ISBN 9780521841078.
http://www.princeton.edu/~nmccarty/Political_Game_Theory.pdf.

[14] David Bellhouse. *The Problem of Waldegrave*. 2007.
http://www.jehps.net/Decembre2007/Bellhouse.pdf. In: Electronic Journal for History of Probability and Statistics, Vol. 3, N^o. 2, December 2007.

[15] John Nash. *Non-Cooperative Games*. September 1951.
https://www.cs.vu.nl/~eliens/download/paper-Nash51.pdf. In: Annals of Mathematics, Second Series, Vol. 54, N^o. 2 , pages 286-295 .

[16] Yoav Shoham, and Kevin Leyton-Brown. *Multiagent systems*. Cambridge University Press, 2008. ISBN 9780521899437.

[17] H. Brendan McMahan, Geoffrey J Gordon, and Avrim Blum. *Planning in the Presence of Cost Functions Controlled by an Adversary*. 2003.
https://www.aaai.org/Papers/ICML/2003/ICML03-071.pdf. In ICML, pages 536–543.

[18] Olga Petrova, Karel Durkota, Galina Alperovich, Karel Horak, Michal Najman, Branislav Bosansky, and Viliam Lisy. *Discovering Imperfectly Observable Adversarial Actions using Anomaly Detection*. 2020.
https://arxiv.org/pdf/2004.10638.pdf. Conference: Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems(AAMAS 2020), Auckland, New Zealand.

Appendix A
Assignement

CTU CZECH TECHNICAL UNIVERSITY IN PRAGUE

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Kasl Tomáš** Personal ID number: **474747**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Branch of study: **Computer and Information Science**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Strategic Games in Adversarial Classification Problems

Bachelor's thesis title in Czech:

Strategické hry v problémech strojové klasifikace s protivníkem

Guidelines:

1. The student will study the elements of game theory [4] with particular attention to the computation of Nash equilibria in two-person games over possibly infinite strategic spaces.
2. The main goal of the thesis is to investigate selected game-theoretic approaches to adversarial machine learning problems; see [1] and [3] for a recent survey. The main emphasis will be on adversarial hypothesis testing games developed in [2]. The student will investigate this model and evaluate its performance using simulations. Specifically, he will:
a) Compare the game-theoretic model of adversarial attacks with the usual Neyman-Pearson or Bayesian framework for hypothesis testing.
b) Run a series of experiments showing the convergence to equilibria for large sample sizes.
c) Evaluate the behavior of error exponents.

Bibliography / sources:

[1] P. Dasgupta and J. Collins. A survey of game theoretic approaches for adversarial machine learning in cybersecurity tasks. AI Magazine, 40(2):31–43, 2019.
[2] S. Yasodharan and P. Loiseau. Nonzero-sum adversarial hypothesis testing games. In Advances in Neural Information Processing Systems, pages 7310–7320, 2019.
[3] L. Dritsoula, P. Loiseau, and J. Musacchio. A game-theoretic analysis of adversarial classification. IEEE Transactions on Information Forensics and Security, 12(12):3094–3109, 2017.
[4] Y. Shoham and K. Leyton-Brown. Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge University Press, New York, NY, USA, 2008.

Name and workplace of bachelor's thesis supervisor:

doc. Ing. Tomáš Kroupa, Ph.D., Artificial Intelligence Center, FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **08.01.2020** Deadline for bachelor thesis submission: _____

Assignment valid until: **30.09.2021**

doc. Ing. Tomáš Kroupa, Ph.D.	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt

Student's signature