

Saarland University
Faculty of Natural Sciences and Technology I
Department of Computer Science
Master's Program in Computer Science

Master's Thesis

Learning Game Theoretic Model Parameters Applied to Adversarial Classification

submitted by

Orhan Sönmez

on June 24, 2008

Supervisor

Tobias Scheffer

Advisor

Matthias Hein

To joyful Izmir and beautiful Istanbul,

Acknowledgement

Hereby, I would like to thank my supervisor, Prof. Tobias Scheffer, and my advisor Prof. Matthias Hein to let me work on this innovative topic. Furthermore, I would particularly thank Micheal Brueckner for never giving up on me and helping me with all the possible aspects of a master thesis.

I also would mindbogglingly like to thank Tunç Ozan Aydın to be my main and almost only support during my stay in Germany. Additionally, I thank E.İrem Arıkan to be the initial reason that I am here and Fulya Horozal for all of her academic helps.

Finally, I would like thank to İlker Tekbaşaran to be there for me when I was down, to my mom for being my mom, to my dear friend Burak Hasircioğlu for all the fun, to Batuhan Belik for his joyful accompany and punk rock band Greenday for keeping me up.

Statement

Hereby I confirm that this thesis is my own work and I have documented all the sources used.

Saarbrücken, June 24, 2008

Declaration of Consent

Herewith I agree that my thesis will be made available through the library of the Computer Science Department.

Saarbrücken, June 24, 2008

Abstract

In most of the machine learning approaches, it is commonly assumed that the data is independent and identically distributed from a data distribution. However, this is not the actual case in the real world applications. Hence, a further assumption could be done about the type of noise over the data in order to correctly model the real world. However, in some application domains such as spam filtering, intrusion detection, fraud detection etc. this assumption does not hold as there exists an opponent adversary that reacts the filtering process of the classifier and modifies the upcoming data accordingly. Hence, the performance of the classifier degrades rapidly after its deployment with the counter actions of the adversary.

When not assuming the independence of the data generation from the classification, arise a new problem, namely the adversarial learning problem. Now, the classifier should estimate the classification parameters with considering the presence of the opponent adversary and furthermore has to adapt itself to the activities of it.

In order to solve this adversarial learning problem, a two-player game is defined between the classifier and the adversary. Afterward, the game results are resolved for different classifier losses such as adversary-aware and utility-based classifier, and for different adversarial strategies such as worst-case, goal-based and utility-based. Furthermore, a minimax approximation and a Nikaido-Isado function-based Nash equilibrium calculation algorithm are proposed in order to calculate the resolved game results.

Finally, these two algorithms are applied over a real-life and an artificial data set for different settings and compared with a linear SVM.

Contents

1	Introduction	4
1.1	Motivation	5
1.1.1	Spam Filtering	5
1.1.2	Intrusion Detection	6
1.2	Related Works	6
1.2.1	Spam Filtering	6
1.2.2	Intrusion Detection	9
1.2.3	Feature Selection	10
1.2.4	Reverse Engineering	11
1.3	Remarks	11
1.3.1	Point of View	11
1.3.2	Adversarial Modification Type	12
1.3.3	Information Completeness	12
1.3.4	Game Step Count	12
1.4	Outline	13
2	Adversarial Learning Problem	14
2.1	Problem Definition	15
2.1.1	Initial Definitions	15
2.1.2	Game Theory Basics	17
2.1.3	Problem Settings	18
2.1.4	Classification Error	19
2.2	Adversarial Scenarios	20
2.2.1	Worst-Case Adversary	21
2.2.2	Goal-Based Adversary	21
2.2.3	Utility-Based Adversary	22
2.3	Assumptions	22
2.3.1	Complete Information	23
2.3.2	Linear Hypothesis	23
2.3.3	Loss Function	23
2.3.4	Regularization Function	23

2.3.5	Adversarial Modification	24
3	Algorithm	27
3.1	Prerequisites	27
3.1.1	Support Vector Machines	28
3.1.2	Minimax	29
3.1.3	Nikaido-Isado-type Functions	29
3.2	Algorithm Derivation	31
3.2.1	Game Result for the Classifier	31
3.2.2	Game Result for the Adversary	32
3.2.3	Game Result Calculation	34
3.3	Algorithm Flow	39
4	Experiments and Conclusions	41
4.1	Experiments	41
4.1.1	Data Sets	41
4.1.2	Implementation	42
4.2	Results	44
4.2.1	Worst-Case Scenario	44
4.2.2	Utility-based Scenario	46
4.3	Conclusions	47
4.3.1	Contributions	47
4.3.2	Remarks	48

List of Figures

1.1	Adversarial Learning Problem	5
1.2	Extensive Form of the Spam Game	9
2.1	Payoff Matrix of Prisoners' Dilemma Game	17
4.1	Worst-Case Scenario with $K = 1$ and $K = 2$	45
4.2	Worst-Case Scenario with $K = 3$ and $K = 5$	45
4.3	Worst-Case Scenario with $K = 10$	45
4.4	Worst-Case Scenario - Comparison of different K values	46
4.5	Worst-Case Scenario - Comparison of different K values	46

Chapter 1

Introduction

In most of the machine learning approaches, it is commonly assumed that the data is independent and identically distributed from a data distribution. Having this assumption, the model parameters of the assumed data distribution are estimated with using the training data. However, this is not the actual case in the real world applications and the algorithms that are trained under this assumption would even suffer when modeling a data distribution with non-random noise or missing values.

Hence, a further assumption could be done about the type of noise over the data in order to correctly model the real world. However, this is still not the case for some of the application domains, since the problem is still defined statically and it is approached with classical concept-based methods. In addition, the assumptions should be tuned in order to solve the dynamic world problems.

By defining a static problem, it is explicitly assumed that the data generating process is independent from the classification process. Such that, the data is not manipulated by a third party with considering the parameters of the classifier itself. However, in some application domains such as spam filtering, intrusion detection, fraud detection etc. this assumption does not hold as there exists an opponent adversary that reacts the filtering process of the classifier and modifies the upcoming data accordingly. Hence, the performance of the classifier degrades rapidly after its deployment with the counter actions of the adversary [5].

When not assuming the independence of the data generation from the classification, arise a new problem, namely the adversarial learning problem. Now, the classifier should estimate the classification parameters with considering the presence of the opponent adversary and furthermore has to adapt itself to the active modifications of it.

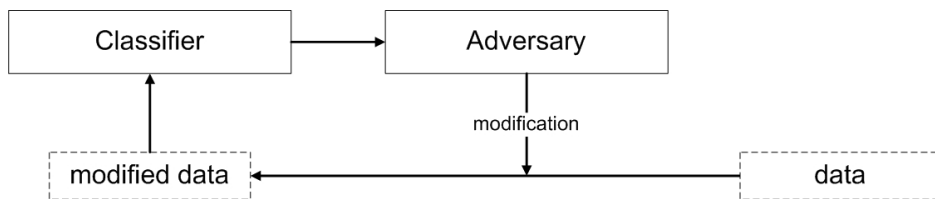


Figure 1.1: Adversarial Learning Problem

1.1 Motivation

Even though, the adversarial learning is a relatively recent research topic in machine learning, there are actually several real world application domains where the classifier tries to learn the data correctly in presence of an opponent adversary such as intrusion detection, fraud detection, shopping bots, captures etc. However, the main motivation of this research is to solve the adversarial learning problem between the spam filters and the evolving spam authors.

1.1.1 Spam Filtering

Constantly increasing amount of the spam makes the spam filtering one of the most important application domains in machine learning studies. Even though the exponential growth of the spam emailing in the recent years has slightly leveled of, as of April 2008, it is estimated that more than 100 billion spam emails are send every day by the spammers according to Wikipedia. Hence, explicit cost of email spamming to the network traffic and its implicit cost to the productivity is not disregardable. Furthermore, it is also estimated that %80 of that amount is actually send over the Internet by less than 200 spammers. However, due to dynamic IP allocations of the spam botnets makes it almost impossible to avoid spam emails by using only network rules. Now, it is commonly believed that a robust content classification technique is required in order to solve the email spamming problem.

Currently, all of the big email hosting services are using content classification methods to detect spams. Hence, every now and then several sophisticated classifiers are designed in order to prevent the newly generated complex spam attacks by the spammers. But, no matter how sophisticated the classifier is, the spammers would always come up with different types of attacks for which the classifier is not ready. Hence, the performance of the spam filters degrades instantly and new classifiers are needed to be deployed every time when such new spam attacks are discovered.

1.1.2 Intrusion Detection

After the global deployment of the Internet in the recent years, a lot of daily processes like shopping, booking, communication etc. and important processes like security, banking etc. have now been supplied over Internet by the web-based services. Thus, today the importance of the Internet and its security has gained huge importance and even this importance keeps on increasing steadily.

However, one of the main threats against Internet is network-based attacks and the very first step to secure it is to be able to detect such attacks. Meanwhile, the intruders do their best such as dynamic IP allocation, using proxy servers, generating artificial network etc. to hide from the network security systems. Since, the intruders are using different types of methods to avoid this detection, the most general way to uncover such attacks is to use classification-based methods that are aware of such enemy evolving parties.

1.2 Related Works

There had been several direct approaches in order to resolve the recently-emerging adversarial learning problem in the past years. Mainly, they consider the situation as a two-player game between a classifier and an adversary and then attempt to solve the optimization problem that is generated by the game theoretic methods.

1.2.1 Spam Filtering

Adversarial Classification

In order to solve spam problem, the process between the spam filters and the spam authors could be considered as a two-player game between a classifier and an adversary as in Dalvi et. al [5]. Such that, the new attacks of the spammers could be modeled as the modifications of the adversary to email instances, meanwhile classifier's choice of an appropriate hypothesis corresponds to deploying a new spam filter. Hence, instead of training a static filter, the classifier could then be adapted to the evolving spammer actions at each step of the game by choosing an appropriate strategy.

[5] firstly claims that the adversary learning problems can not basically be solved by the classical concept-based learning algorithms, as these algorithms can not handle the situations when the data generation process is modified with a function of the learner itself by a third party. Hence, instead of a usual learning scheme, after assuming the complete information which is

quite common in the game theory, a game between the classifier and the adversary is defined.

In this two-player game, when a new instance arrives, the classifier chooses the features to be measured among the feature vector and then decides the instance's classification. By doing so, it tries to maximize its utility of classifying the instances correctly and to minimize the cost of measuring the features. Meanwhile, the adversary could change some features of an instance in order to deceive the classifier. Its goal is to maximize its benefit by having the classifier generate false classifications and to minimize its cost of changing the features. Hence, both of the parties resolve a cost-sensitive optimization problem to obtain their optimal strategies.

In order to start the game, as a common approach in the evolutionary game theory, first the classifier assumes that the data is unmanipulated and trains its learner accordingly. Then, the adversary generates an optimal strategy with respect to the naive classifier. Furthermore, in the next step of the game, as all the information is complete, the classifier creates a new learner against the strategy of the adversary. And the game continues likewise.

At the very first step, a cost-sensitive naive bayes learner is chosen as the classifier for simplicity. Then, the adversary's optimal strategy is calculated with assuming the complete information and the classifier is unaware of adversary's existence. Here, the set of features with the minimum feature changing cost that would make the instance classified as negative, namely the minimum cost camouflage, is found as the optimal strategy of the adversary. And, if its utility is higher than its feature changing cost, the strategy is applied by the adversary.

The classifier assumes that the adversary modifies the test instance with using its optimal strategy while leaving the initial training set unmanipulated. Hence, the probability of seeing an instance x' is the sum of the probabilities that an instance x is generated then modified into x' by the adversary and an instance x' is generated directly by the distribution. Afterward, the classifier predicts the class that maximizes the conditional utility for each instance individually by comparing the probabilities for each possible class in a cost-sensitive manner. Finally, an efficient adversary-aware classifier algorithm is generated and even its performance is improved by pruning.

The performance of this classifier is experimented in spam domain against a classical naive bayes classifier under different scenarios such as boolean features, boolean features with different addition costs and multinomial features with synonyms. Also, in order to realize the dominance of the false positive errors in spam domain, the experiments are held with different utility matrices.

As a result, the adversary degrades the performance of the naive bayes classifier in all scenarios and with utility matrices. Meanwhile, the classifier mentioned in the paper still managed to label a large percentage of spam instances correctly with keeping the accuracy on non-spam high. Interestingly, modifications of the adversary tends to decrease the false positive error rate. Since the adversary would not send a spam mail unmodified, previously spam-labeled ham emails would be labeled as negative in the further steps as they would be remained unmodified.

Spam Game

Similarly, Androutsopoulos et. al [3] models the interaction between the spammers and the email readers as a two-player game. Here, the email readers decide to read or delete an email after it is labeled as legitimate or spam email by the spam filter, while the spam author generates its own strategy to insert spam content into a legitimate email or not. Furthermore, mixed strategies of both parties are introduced in order to find an appropriate equilibrium for the game.

Even though the spam problem is considered slightly different, in [3] a similar two-player adversarial game between the email readers and the spam authors is defined in order to resolve the problem. In this game, it is assumed that the spam author has the opportunity to insert spam content to every single legitimate email which is send to the user. Although, this assumption does not hold in the real world, the main aim is to let the spam author arrange the percentage of the spam emails received by the user which would definitely make sense.

Hence, implicitly the emails are firstly received by the spam author and according to its strategy, either spam content is inserted into the email or the email is left unmodified. Then, the emails are classified as spam " S " or legitimate " L " by the spam filter as usual. However, the problem is extended that the user also decides to read (R) or delete (D) the emails labeled by the spam filter. Hence, the user basically has four pure strategies such that reading all the emails (RR), deleting all the emails (DD), reading only legitimate labeled emails (RD) and reading only spam labeled emails (DR). The extensive form tree of the spam game is shown in Figure 1.2.

Also as shown in Figure 1.2, for each email, the user gains utility if it reads an ham email, has same amount of utility loss if it deletes the ham email and also has a smaller amount of loss when it reads a spam email. Deleting a spam email that is labeled as "spam" has neither no gain nor no loss of utility for the reader. Similarly, the spam author gains utility when the reader reads an email with spam content and has a smaller amount of

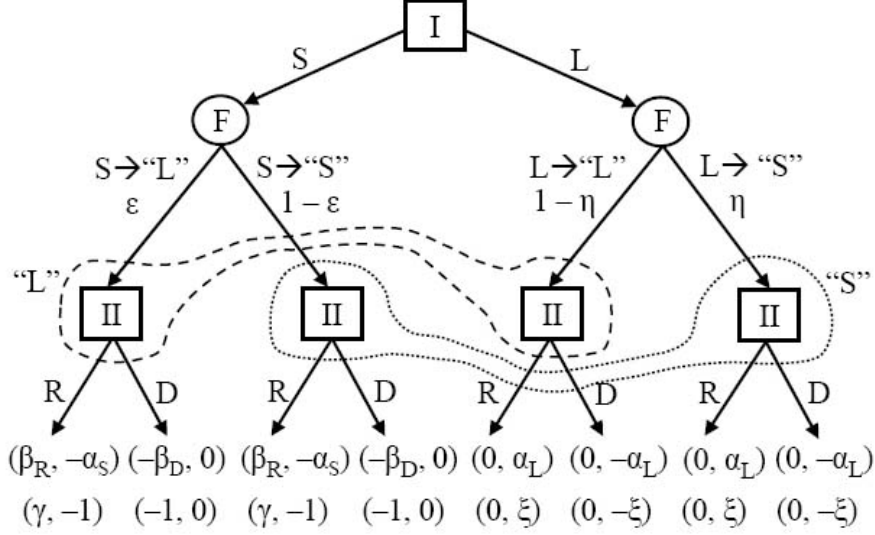


Figure 1.2: Extensive Form of the Spam Game

loss when the reader correctly deletes a spam email without reading it. The ham emails have no effect on the utility of the spam author.

Both the user and the spam author can adopt mixed strategies during the game. So, whenever the games is repeated, the user can choose one of the four pure strategies with probability $\sigma(RR)$, $\sigma(DD)$, $\sigma(RD)$ and $\sigma(DR)$ respectively, with supplying that their sum is equal to 1. Similarly, the spam author would simply determine the probability to insert spam content $\pi(S)$ and the probability of leaving an email legitimate $\pi(L)$ will then be equal to $1 - \pi(S)$.

Finally, after defining the two-player game and the actions of the players, spammer and email reader, the result of the game can be resolved by using game theoretic approaches. In [3], Nash equilibrium of the game is calculated by a graphical technique that is valid for two-player games and this equilibrium is proposed as the optimal game result for both of the parties.

1.2.2 Intrusion Detection

Additionally, adversarial learning algorithms or similar methods could be implemented in the intrusion detection systems against the altering intrusion techniques. In [9], instead of learning the novel attack, the classifier learns the attack-free network traffic as the normal behavior. When there is an anomaly

in the network traffic, it is detected by the intrusion detection system as a possible novel attack. Hence, even though the attack type is changed by the adversary, it would be detected by the classifier.

Even though the adversarial learning problem is a solid approach to detect the different behaviours as attacks, its main drawback is detecting the change of the normal behaviour as an attack. Hence, retraining of the classifier is required when the normal behaviour changes.

1.2.3 Feature Selection

Robust Feature Deletion

Also, in [10], without directly assuming a two-player game between the classifier and the adversary, a worst-case optimal algorithm, namely FDROP, is designed in order to minimize the negative effects of the possible future adversary actions. Then, the performance of the algorithm is experimented in the spam filtering domain and its robustness is shown against the evolving adversary.

In [10], robustness of the classifier is maintained by having precautions against the presence of a possible opponent adversary. Actually, the main idea of the paper is not to assign too much weight to any single input feature in order to keep robustness against non stationary feature distributions, input sensor failures and of course adversaries. Classical regularization methods could be thought as a tool to obtain robustness, however it is sufficient for this task and also useless against the latter.

Instead, the approach is to construct a classifier which is optimal in the worst case deletion scenario. Hence, the robustness problem is formalized as a two-player game where the classifier tries to choose its parameters robustly while the adversary deletes the feature whose removal would be the most harmful to the classifier. Having a predefined maximum number of possible deletion operations K by the adversary, the minimax problem of the two-player game with using a hinge loss as a loss function, is formulated as a quadratic and convex optimization problem, namely FDROP.

The solution of this optimization problem is a combination of weighted vectors for each sample where the maximal number of weights that can be set to zero is constrained by K . Hence, it could easily be interpreted as a solution of a feature selection problem. If so, the adversary can be thought as a sample-based feature selection algorithm that finds the set of K features that would yield the minimum generalization error when chosen alone. So, in order to maintain robustness, the classifier would disregard these best K features per sample basis. Intuitively, these features would also contain high

information one by one however it is not guaranteed.

The performance of the algorithm is tested with handwritten digit recognition when some of the pixels are removed. The error rate of FDROP seemed to remain stable when the probability of removing the highly informative pixels increases, meanwhile the performance of a classical SVM degraded rapidly. Also, FDROP outperformed SVM in another experiment as a spam filter when the spam authors react to the spam filter by changing the spam content.

Alternatively, uniform feature deletion scenario for all samples is also considered instead of the previous sample-based version. Although, the former is a less pessimistic subset of the latter problem, it is found that the optimization problem yield is harder and not efficiently solvable.

Convex Learning under Invariances

Finally in [7], a convex learning algorithm is proposed in order to learn under incorporative invariances for arbitrary loss functions. And then, the sample modifications of the adversary in the adversarial learning problem can be modeled as these invariances. Hence, the adversarial learning problem would then be transformed into learning under invariances problem to be solved. Then this method is applied to the adversarial spam filtering problem.

1.2.4 Reverse Engineering

There also exists a paper in the literature with the exact title of "Adversarial Learning" [2]. Here, the classifier also learns under the presence of an adversary like an adversarial learning problem. However, it is not assumed that the classifier information is completely known by the adversary. Hence, the adversary tries to model the parameters of the classifier with some reverse engineering techniques and then modifies the upcoming instances accordingly.

The paper mainly focuses on reverse engineering which is actually a slightly different problem, namely the adversarial classifier reverse engineering (ACRE) problem. However, it is also important to mention about this paper in order to avoid confusion due to its title.

1.3 Remarks

1.3.1 Point of View

Most of the previous related works such as [10],[3] and [7] consider the adversarial learning problem from the classifier's point of view. After making

assumptions about the adversary, resolve the strategy of the adversary with using the minimax theorem. Finally, they reach the optimal strategy of the classifier as they are aimed mainly.

However, [2] has a completely different approach to the problem. Instead of the classifier, main aim is to determine the optimal strategy for the adversary. This is motivated by the author such that when the strategies for the adversaries are studied, the strategies for the classifier could be design in a more conscious way.

In [5], the optimal strategies for both sides are calculated according to the previously known strategies of the opponent party.

1.3.2 Adversarial Modification Type

When we consider [5],[10] and [7], the modification of the adversary is defined as a transformation of a sample in the input space. In [10], this transformation is removal of features, [5] always modification of the value of features. Finally, any arbitrary transformation is allowed in [7].

However, main difference of [3] is that the adversary doesn't make its modification sample-based. Instead, it has the capability of manipulating the distribution that the samples are generated from.

1.3.3 Information Completeness

All of the previous work except [2] assumes the complete information principle. The algorithms in [5],[3],[10] and [7] need this assumption since they are using game theoretic techniques in their algorithms.

However, in [2] the information about the classifier is not complete for the adversary. As, the problem is defined as a reverse engineering problem, no game theoretic techniques are required. The retrieval of the classifier information is included in the generation process of the optimal strategy for the adversary.

1.3.4 Game Step Count

A two-player game between the classifier and the adversary is commonly defined for the approaches. But, most of the algorithms only calculate the one-shot game results for the very first game step such in [10] and [7].

However, in [5] the two-player game is defined as an evolutionary game such that both parties calculate their optimal strategy with considering the previous strategy of the other. Hence, the game would continue for many

steps one by one. But, even though an evolutionary game is defined, the result of the game are only available for the first steps of the parties.

1.4 Outline

After the introduction, in Chapter 2, detailed information about the adversarial learning problem is given and a mathematical model in order to represent the problem is proposed with corresponding assumptions. Chapter 3 starts with technical prerequisites and then the proposed algorithm to solve the adversarial learning problem is explained. In Chapter 4, different experiments settings and the results of them are revealed to the reader. Finally, in Chapter 5, contributions and future works are discussed.

Chapter 2

Adversarial Learning Problem

In machine learning problems, algorithms widely assume that the data is independent and identically distributed from a data distribution. Hence, the main goal of the learning algorithm is now to estimate the parameters of that corresponding data distribution with using the generated training data. However, this common assumption doesn't generally hold in the real world problems such that the data distribution could have some non-random noise and missing values.

Having an inappropriate assumption would make the algorithm suffer heavily, hence another assumption should be made about the model of the noise over the data in order to estimate the data distribution parameters correctly. However, to model some real world problems, it is still not sufficient. For the application domains in which the data distribution actually depends on the parameters of the algorithm itself, the assumption should be extended.

Using these classical machine learning assumptions explicitly assumes that the data generating process is independent from the learning process. Such that, the data distribution is not affected by the parameters of the learning algorithm. However, in some real world application domains such as spam filtering, intrusion detection, fraud detection etc. these assumptions don't hold as there exists an opponent adversary that reacts the filtering process of the classifier and modifies the upcoming data accordingly. Hence, the performance of the classifier degrades rapidly after its deployment with the counter actions of that opponent adversary [5].

When it is not assumed that the data generation process would depend on the parameters of the algorithm, it called an adversarial learning problem. Now, the machine learning algorithm have to estimate the data distribution parameters with considering the presence of the opponent adversary that actively modifies the generated data before revealed to the algorithm. Fur-

thermore, it even has to adapt itself to the activities of the adversary to keep its performance robust.

In the next section, the adversarial learning problem is explained formally and in more details. Additionally, a section is separated for the different adversary strategies that changes the adversarial optimization problem to be solved. Finally, the assumptions about information completeness, modification limit, positive modification and payload features are introduced to the reader in details.

2.1 Problem Definition

In adversarial learning problems, the classifier tries to generate a proper hypothesis with a minimal generalization error about the data distribution in the presence of an evolving opponent adversary. Meanwhile, the adversary also aims to maximize its own utility by causing the classifier generate classification errors with modifying the data instances.

The adversary gains utility if an instance that contains some special features that would make it classified as positive, namely the payload features, is classified as negative by the hypothesis generated by the adversary. Hence, it tries to hide these features from the classifier by adding a lot of extra features that would help the instance to be classified as negative, namely the chaff features.

In this manner, when considering adversarial learning problem in spam domain, the actual spam content in a spam mail corresponds to the payload, while all other innocent good words added by the spam author correspond to the chaff features. Also, in network attacks, the intruders add some random noise or some artificial network traffic as chaff features to avoid their actual novel attack, their payload, to be detected.

First of all, some required definitions are made to construct the main structure of the problem and then the actual problem is introduced in a formal manner with corresponding examples from the spam filtering domain.

2.1.1 Initial Definitions

Let X be the input space such that

$$X = R^d \tag{2.1}$$

and d is the number of input features. Also, let $x \in X$ denote an input sample from the input space X . Similarly, let Y represent the output space such

that

$$Y = \{-1, 1\} \quad (2.2)$$

and $y \in Y$ is an output label from the output space Y .

To summarize the data definitions, let us define Z as the whole data space such that

$$Z = X \times Y \quad (2.3)$$

and sample $z \in Z$ is a data sample from the data space Z .

In addition, for data generation from this data space, let us consider an arbitrary distribution D that is defined over data space Z .

After defining the data space and its corresponding data distribution, we now consider the functions that are defined over this data space.

Hypothesis Function

Let the hypothesis function space H be defined as

$$H = \{h|h : X \rightarrow Y\} \quad (2.4)$$

where hypothesis h of the classifier is the function that returns the estimation of output given the input.

Adversarial Modification

Also, let the modification function space M be

$$M = \{m|m : X \rightarrow X\} \quad (2.5)$$

where modification m is transformation of the actual input to an another valid value in the given input space.

Loss Function

Finally, the loss function space L is defined as

$$L = \{l|l : Y \times Y \rightarrow [0, 1]\} \quad (2.6)$$

where the loss function l returns the amount of loss given the output estimation and the actual output where 0 denotes the exact match for the estimation.

Regularization Function

Regularization function is the general name for the functions that are inserted into the optimization problems in order to prevent overfitting of data by the learning model. For minimization problems, it is a positive term such that the more the model gets complex, the more it penalizes the main term to be minimized. Further on we would use Ω to represent a regularization function.

2.1.2 Game Theory Basics

In the algorithm, initially a two-player game is defined and then the Nash equilibrium of this game is proposed as a possible solution of the problem. Finally, for the zero-sum case, the Nash equilibrium is calculated with using Minimax algorithm. Hence, required knowledge about these game theoretic aspects are introduced in this section.

Two-Player Strategic Games

Two-player strategic games are described as payoff matrices, where the strategies of both players serve as row and column labels respectively and the game payoffs or losses for both of the players are listed in the corresponding matrix square as pairs [8].

As defined in figure 2.1, Prisoners' Dilemma is one of the most famous examples of two-player strategic games. In this game, both players have two possible strategies, C as "confess" and D as "defect", and the game payoffs are given in the matrix according to the chosen strategies of both players.

$1 \backslash 2$	C	D
C	4,4	0,5
D	5,0	1,1

Figure 2.1: Payoff Matrix of Prisoners' Dilemma Game

Additionally, when the payoff/loss of the player 2 is equal to the negative of the payoff/loss for player 1 for all possible strategy combinations, the game is then called as a two-player zero-sum game. In the adversarial learning

problem, when the adversarial loss is equal to the negative of the loss of the classifier, the game is a two-player zero-sum game.

$$\theta_A = -\theta_C \quad (2.7)$$

where θ_A and θ_C are losses for the adversary and the classifier respectively.

Nash Equilibrium

Nash equilibrium is a strategy combination of the players where no player can get better payoffs by choosing a strategy other than the one in the strategy combination, given all the other players choose the corresponding strategies in the strategy profile [8]. In this sense, Nash equilibrium could be thought as an optimal strategic solution for all players.

For the adversarial learning problem, as defined in the previous chapter, the Nash Equilibrium (h^*, m^*) can be defined as,

$$\begin{aligned} \theta_C(h^*, m^*) &= \max_{h' \in H} \theta_C(h', m^*) \\ \theta_A(h^*, m^*) &= \max_{m' \in M} \theta_A(h^*, m') \end{aligned} \quad (2.8)$$

where $h^* \in H$ is the hypothesis function of the classifier and $m^* \in M$ is the adversarial modification function at the Nash equilibrium.

An important property of the Nash equilibrium is that its existence is guaranteed if the action spaces are compact and convex, and the cost functions for both of the parties are continuous and convex in these action spaces [1].

2.1.3 Problem Settings

After making the initial required definitions, now the adversarial learning problem can be explained in more details. Also, since the main motivation of this research is to resolve the adversarial learning problem in the spam filtering domain, the formal problem settings will be accompanied with their corresponding examples in the spam filtering problem.

Initially, an n -sampled data set z_1, z_2, \dots, z_n is generated from the data space Z with using a distribution D such that

$$z_i = (x_i, y_i) \quad \text{where } i = 1, 2, \dots, n \quad (2.9)$$

When considering the spam filtering problem, this data set represents a set of received emails by the spam filter and their actual labels. Hence, the input features x_i would represent the words and some statistical parameters

in the i^{th} email. Similarly, y_i is the label of the i^{th} email such that $y_i = 1$ when the email is actually spam and $y_i = -1$ when the email contains no spam content.

2.1.4 Classification Error

Normally, in a classical machine learning problem, the classifier would aim to minimize the total classification error E'_C by choosing an appropriate hypothesis function h on the data set z_1, z_2, \dots, z_n such as

$$E'_C = \sum_{i=1}^n l(h(x_i), y_i) + \frac{\lambda_c}{2} \Omega(h) \quad (2.10)$$

Hence, the optimization problem to determine the hypothesis corresponding h' would be,

$$h' = \arg \min_{h \in H} \sum_{i=1}^n l(h(x_i), y_i) + \frac{\lambda_c}{2} \Omega(h) \quad (2.11)$$

where $\lambda_c \in R$ is a predefined constant parameter, $l \in L$ is the chosen loss function and $\Omega(h)$ is the regularization function of h .

Adversary-aware Classifier

However, presence of an opponent adversary and its active sample modifications change this classical support vector machine optimization problem in Equation 2.11 into an adversarial learning optimization problem.

Now, for each sample $z_i = (x_i, y_i)$ in the data set, the adversary generates a modification rule m_i before it is revealed to the classifier and apply it to the input features x_i of the sample z_i . Hence, instead of x_i , modified $m_i(x_i)$ is given to the classifier for classification process.

Hence, the classifier tries to minimize the total classification error E_C over the data set z_1, z_2, \dots, z_n with respect to the a loss function $l \in L$. In order to do so, it chooses an appropriate hypothesis $h \in H$ with considering the possible manipulations of the adversary.

$$E_C = \sum_{i=1}^n l(h(m_i(x_i)), y_i) + \frac{\lambda_c}{2} \Omega(h, m) \quad (2.12)$$

So, to calculate the hypothesis h under the presence of an opponent adversary, the classifier should resolve the following optimization problem.

$$h = \arg \min_{h \in H} \sum_{i=1}^n l(h(m_i(x_i)), y_i) + \frac{\lambda_c}{2} \Omega(h, m) \quad (2.13)$$

where $\lambda_c \in R$ is a predefined constant parameter, $l \in L$ is the chosen loss function and $\Omega(h, m)$ is the corresponding regularization function for h and m .

Meanwhile, the adversary aims to maximize its own utility over the data set z_1, z_2, \dots, z_n by making the classifier generate classification errors. So, the adversary chooses appropriate manipulations $m_1, m_2, \dots, m_n \in M$ with respect to the hypothesis h of the classifier.

Utility-Based Classifier

If we consider a more generalized version of the adversary-aware classifier, such that the importance of each instance for the classifier differs. Then, we obtain the utility-based classifier that aims to minimize total classification error which is calculated as a sum of weighted sample-based errors according to their utilities.

$$E_C^{utility} = \sum_{i=1}^n u_i^c l(h(m_i(x_i)), y_i) + \frac{\lambda_c}{2} \Omega(h, m) \quad (2.14)$$

where u_i^c is the importance weight of the i^{th} sample for the utility-based classifier. Hence, the hypothesis $h^{utility}$ is calculated as,

$$h^{utility} = \arg \min_{h \in H} \sum_{i=1}^n u_i^c l(h(m_i(x_i)), y_i) + \frac{\lambda_c}{2} \Omega(h, m) \quad (2.15)$$

where $\lambda \in R$ is a predefined constant parameter, $l \in L$ is the chosen loss function and $\Omega(h, m)$ is the corresponding regularization function for h and m .

In spam filtering, these could correspond to the case that letting a phishing spam email into the mailbox could be more dangerous than letting a standard commercial spam email for the spam filter, or the size of the email would be considered when calculating the importance, since the more size the more storage space and more cpu usage would be required by the email server.

2.2 Adversarial Scenarios

The model to solve the problem differs as the presence of the adversary is regarded. For this case, we consider, three different general adversary scenarios. The model to solve the problem differs as the presence of the adversary is regarded. For this case, we consider, three different general adversary scenarios.

2.2.1 Worst-Case Adversary

In the worst-case scenario, the adversary does not directly try to gain benefit with modifying the instances, instead its main aim is harming the classifier. Hence, the goal of the worst-case adversary is to maximize the total classification error E_C of the classifier. It is explicitly equal to the minimization of its error E_A^{worst} (adversarial cost) which is the negative of the total classification error E_C .

$$E_A^{worst} = -E_C \quad (2.16)$$

So,

$$E_A^{worst} = - \sum_{i=1}^n l(h(m_i(x_i)), y_i) - \frac{\lambda_c}{2} \Omega(h) + \frac{\lambda_a}{2} \Omega(m) \quad (2.17)$$

Hence, worst-case modification m_{worst} could be determined by solving the following optimization problem.

$$m_{worst} = \arg \min_{m \in M} - \sum_{i=1}^n l(h(m_i(x_i)), y_i) - \frac{\lambda_c}{2} \Omega(h) + \frac{\lambda_a}{2} \Omega(m) \quad (2.18)$$

where $\lambda_a \in R$ is a predefined constant parameter, $l \in L$ is the chosen loss function and $\Omega(h)$ and $\Omega(m)$ are the corresponding regularization functions for h and m .

In the spam filtering problem, it can be thought as an opponent party that gains utility any time when the spam filter makes a misclassification.

2.2.2 Goal-Based Adversary

Goal-based adversary does not directly aim to maximize the total classification error E_C as in the worst-case. Instead, it tries to maximize the false negative classification error where a positive instance x_i is classified as negative. This goal of the adversary makes sense, since in the real world applications such as spam filtering and intrusion detection, the main aim of the adversary is to hide the positive instances, respectively spams and network attacks, by modifying them and to make the classifier label them as negative. So, the adversarial cost E_A^{goal} could be calculated as,

$$E_A^{goal} = - \sum_{i=1}^n l(h(m_i(x_i)), y_i) \cdot \frac{(y_i + 1)}{2} - \frac{\lambda_c}{2} \Omega(h) + \frac{\lambda_a}{2} \Omega(m) \quad (2.19)$$

Hence, the optimization problem in order to calculate the goal-based utility m_{goal} is,

$$m_{goal} = \arg \min_{m \in M} - \sum_{i=1}^n l(h(m_i(x_i)), y_i) \cdot \frac{(y_i + 1)}{2} - \frac{\lambda_c}{2} \Omega(h) + \frac{\lambda_a}{2} \Omega(m) \quad (2.20)$$

where the $\frac{(y_i+1)}{2}$ term captures the label of the sample such that it is equal to 1 when the sample label is 1 (positive) and it is equal to 0 when the sample label is -1 (negative).

In the spam filtering problem, it can be thought as a spammer that only gains utility when the spam filter classifies a spam email as ham and let it go into the mailbox of the email reader.

2.2.3 Utility-Based Adversary

Utility-based adversary is actually a more generalized form of the goal-based adversary and the main difference between them is that the each sample could have different utility gains for the adversary. Hence, weight parameters are added into the optimization problem in Equation 2.19 in order to capture this aspect of the utility-based adversary. Then, the adversarial cost $E_A^{utility}$ for the utility-based adversary could be as the following,

$$E_A^{utility} = - \sum_{i=1}^n u_i^a l(h(m_i(x_i)), y_i) \cdot \frac{(y_i + 1)}{2} - \frac{\lambda_c}{2} \Omega(h) + \frac{\lambda_a}{2} \Omega(m) \quad (2.21)$$

So, the modification $m_{utility}$ for the utility-based adversary is,

$$m_{utility} = \arg \min_{m \in M} - \sum_{i=1}^n u_i^a l(h(m_i(x_i)), y_i) \cdot \frac{(y_i + 1)}{2} - \frac{\lambda_c}{2} \Omega(h) + \frac{\lambda_a}{2} \Omega(m) \quad (2.22)$$

where $u_i^a \in R$ is the weight of the i^{th} sample according to the adversary that represents the utility gain of the adversary from the i^{th} sample.

In the spam filtering problem, it can be thought as a more sophisticated spammer with different types of spam contents which have different utility gains to the spammer when classified as "ham" by the spam filter and went into the mailbox of the email reader. Hence, it could be an appropriate model for the real world spam authors when the utility weights u_i are chosen correctly.

2.3 Assumptions

Before defining the two-player game and proposing the algorithm for the classifier, some assumptions are made about the problem environment and the opposing adversary.

2.3.1 Complete Information

It is assumed that the environment of the problem supplies the complete information principle such that both parties, the classifier and the adversary, have unlimited access to the current parameters of each other. This assumption is required to use the game theoretical properties of the zero-sum and non-zero sum games and the techniques such as minimax and Nash equilibrium in order to model and solve the adversarial learning problem.

Even though, this assumption would not hold in the real world case, it is a common assumption in the game theoretic approaches and the incomplete information scenario is out of the scope of this research.

2.3.2 Linear Hypothesis

In our approach, linear hypothesis space is chosen and

$$h = w \cdot x \tag{2.23}$$

Hence, weights w would be used instead of h during the optimization.

2.3.3 Loss Function

Even though, the problem definition is still valid for any loss function, without losing generality, our approach is based on a hinge loss function. Hinge loss function can be defined for $y_i \in -1, 1$ as follows,

$$l(y', y) = [1 - y' \cdot y]_+ \tag{2.24}$$

where y' is the estimation of y whose loss is considered and the subscript "+" denotes only the positive part of the outcome value such as

$$\forall a \in R \quad [a]_+ = \max(a, 0) \tag{2.25}$$

2.3.4 Regularization Function

For the rest of approach, we will use a common L2-norm regularization function, that is mostly used in SVMs. It is defined as follows,

$$\Omega(x) = \frac{1}{2} \|x\|^2 \tag{2.26}$$

where $\Omega(x)$ is the regularization function of x .

Also, it is important to notice that the distributivity property of our regularization function over the parameters such as,

$$\Omega(x, y) = \Omega(x) + \Omega(y) \tag{2.27}$$

2.3.5 Adversarial Modification

In spam problem for the real world, the spammer modifies the actual spam emails by adding some "good" words in order to avoid the detection. However, after retraining of the spam filters, they can now detect such "good" words attacks. Then, the spammers reply them by removing those detected "good" words and inserting new ones over the actual spam content. Again, the classifier can not filter these emails with spam content. This process of the adversary is modeled as adding and removing features in our approach.

In our approach a binary feature case is considered and the modification m_i from the i^{th} sample $x_i \in \{0, 1\}^d$ to x'_i is modeled as,

$$x'_i = m_i(x_i) = x_i + m_i \quad (2.28)$$

where $x'_i \in \{0, 1\}^d$ is the modified instance, $m_i \in \{-1, 0, 1\}^d$ is the modification vector such that,

$$\forall i \quad 0 \leq x_{i,j} + m_{i,j} \leq 1 \quad \text{where } j = 1, 2, \dots, d \quad (2.29)$$

Modification Limit

The classifier assumes that the adversary has a predefined modification budget for each sample x_i . Hence, the number of the features to be added or to be deleted by the adversary is constrained by a fixed constant K such that,

$$\forall i \in Z^+ \quad 0 \leq \|m_i\| \leq K \quad \text{where } K \in Z^+ \quad \text{and } 0 < i \leq n \quad (2.30)$$

such that, $\|m_i\|$ is the magnitude of the adversarial modification for the i^{th} sample which is the distance between x_i and $m_i(x_i)$.

For samples x_i with binary features, magnitude of the modification $\|m_i\|$ could be defined as the Manhattan distance between x_i and $m_i(x_i)$ which is also equal to the squared error loss for $m_{i,j} \in \{-1, 0, 1\}$ as in equation 2.31.

$$\|m_i\| = \sum_{j=1}^d m_{i,j}^2 \quad (2.31)$$

where $m_{i,j}$ is the corresponding modification bits of the j^{th} feature of the i^{th} modification m_i .

This assumption is made in order to avoid the modification of all possible features by the adversary which makes it even information theoretically impossible to classify the modified samples by the classifier. Furthermore, to

make the optimization problem simpler, it is also implicitly assumed that all the features have the same modification cost for the adversary.

Even though, this assumption would seem a bit strict, it would even hold in the real world when an appropriate problem-specific K is chosen.

Positive Modification

Considering the real life adversarial problems, it is common that the adversaries only modify the positive (+) instances, in order to make the classifier label them as negative (-). The negative (-) instances are generally not modified by the adversary as there is no benefit for the adversary when they are classified wrongly. Such that, the spammers only edit their spam emails but they never edit a legitimate email to fool the spam filter. Because when a legitimate email is labeled as spam by the spam filter, the spammer gains no utility.

Hence, it is assumed by the classifier that the adversary only modifies the positive (+) instances and the negative instances are left unmanipulated. Then, the modification limit inequality in 2.30 is updated as,

$$0 \leq \|m_i\| \leq K \frac{y_i + 1}{2} \quad \text{where } K \in \mathbb{Z}^+ \quad \text{and} \quad 0 < i \leq n \quad (2.32)$$

such that, $\|m_i\|$ is the magnitude of the adversarial modification for the i^{th} sample, K is the predefined modification limit and $y_i \in \{-1, 1\}$ is the actual label of the i^{th} sample.

Modification Costs

In adversarial learning problems, it is common that the adversary aims to hide some "bad" features among some "good" ones and make the classifier label the whole instance as "good". Such as, in spam filtering, spam authors try to avoid spam filters by inserting good words into an actual spam content which is namely known as good word attacks. Additionally, network intruders try to hide their actual attack from the intrusion detection systems by generating artificial stable network traffic.

Hence, this common real world case is also considered by the classifier. When considering "bad" samples as positive (+) and "good" samples as negative (-), the classifier assumes that the adversary does not want to remove the features that would normally make the sample classified as positive. Such features are named as "payload" features in our approach. Thus, the classifier assumes that the adversary has a higher cost for modifying the such features.

So, the inequality 2.33 for calculating the magnitude of a modification is extended as follows,

$$\|m_i\| = \sum_{j=1}^d c_j m_{i,j}^2 \quad (2.33)$$

where c_j is cost for adversary to modify the j^{th} feature.

Modification Cost Determination

In order to determine feature-based modification costs c_j for the adversary before the actual optimization problem, the classifier initially trains a linear learner just for this purpose. Then, with considering the weights of this initial learner, the classifier estimates the feature-based modification costs c_j for the adversary. These weights are rescaled and normalized to be in the interval $[0, 1]$ such that 0 corresponds to the smallest and 1 corresponds to the greatest weight.

Chapter 3

Algorithm

With motivation from the previous approaches [3] and [5], to resolve the adversarial learning problem, the interaction between the classifier and the adversary is defined as a two-player game. In order to make this game represent the real world adversarial learning problem appropriately and to make the corresponding optimization problems solvable, some general and problem-specific assumptions are made.

Then, under the presence of these assumptions, the payoff for the classifier is generated as equal to the negative of the total classification error E_C with using the equation 2.12. Similarly, for different adversarial scenarios such as worst-case, goal-based and utility-based, the payoffs for the adversary is obtained from the equations 2.17, 2.19 and 2.21 respectively.

In the following section, the prerequisites to understand the algorithm is introduced. Furthermore, in Section 2, the two-player game is formally defined and the losses for different types of classifiers and adversaries are generated with using the defined model under the assumptions. Finally, an algorithm to calculate the game result for different situations is proposed in Section 3.

3.1 Prerequisites

The algorithm in section 3 is based on an optimization problem that is coming from the Support Vector Machine (SVM) optimization problem, hence basics of SVMs are introduced in the next subsection. Furthermore, in addition to the machine learning techniques, basis for the game theoretic approach in the algorithm is introduced such as Two-Player Games, Nash Equilibrium and Minimax. Finally, Nikaido-Isado-type functions are explained in order to be used to calculate the Nash Equilibrium.

3.1.1 Support Vector Machines

SVMs are learning algorithms that aim to find the best linear boundaries with the biggest margins in the feature space if possible. Otherwise, the feature space could then be enlarged with using kernel methods in order to get better training-class separation. Then, these linear boundaries in the enlarged space could be mapped into nonlinear boundaries in the original input feature space [6].

In SVM algorithm, an hyperplane $f(x)$ is defined as,

$$\{x : f(x) = x^T \beta + \beta_0\} \quad (3.1)$$

where β is a unit vector such $\|\beta\| = 1$. Then, a corresponding classification rule $G(x)$ is induced by $f(x)$ as,

$$G(x) = \text{sign}[f(x)] = \text{sign}[x^T \beta + \beta_0] \quad (3.2)$$

When the input feature space is transformed into an appropriate space where the classes are linearly separable, a hyperplane $f(x) = x^T \beta + \beta_0$ could be found to separate the classes such that,

$$\forall i \quad y_i(x_i^T \beta + \beta_0) > 0 \quad (3.3)$$

where the class labels $y_i \in \{-1, 1\}$ and $1 \leq i \leq n$.

Hence, the aim of the SVM of finding the biggest margins could then be formalized as the following optimization problem,

$$\max_{\beta, \beta_0, \|\beta\|=1} C$$

$$\text{subject to} \quad y_i(x_i^T \beta + \beta_0) \geq C \quad i = 1, \dots, n \quad (3.4)$$

where $C \in \mathbb{R}^+$ and the margin that is the distance between the separated classes, is equal to $2C$. It can be easily shown that this maximization problem of C can be transformed into a minimization problem of β ,

$$\min_{\beta, \beta_0} \|\beta\|$$

$$\text{subject to} \quad y_i(x_i^T \beta + \beta_0) \geq 1 \quad i = 1, \dots, n \quad (3.5)$$

where C is chosen as $C = 1/\|\beta\|$.

Furthermore, this optimization problem with n constraints is then transformed into an optimization problem with no constraints but penalization [6].

$$\min_{\beta, \beta_0} \sum_{i=1}^n [1 - y_i f(x_i)]_+ + \frac{\lambda}{2} \|\beta\|^2 \quad (3.6)$$

where the subscript ”+” denotes only the positive part such as

$$\forall a \in \mathbb{R} \quad [a]_+ = \max(a, 0) \quad (3.7)$$

In this equation, the first sum actually corresponds to the total classification loss. Meanwhile, the second parameter that is also known as the regularization function, is the penalty for the chosen classifier parameters. In next part, more detailed information will be given about the regularization function.

3.1.2 Minimax

Minimax algorithm is a method that is commonly used in game theory to calculate the nash equilibriums in zero-sum games. According to [12], the Minimax theorem states that, if (a^*, b^*) is a Nash equilibrium of a two-player zero-sum game,

$$\theta(a^*, b^*) = \max_a \min_b \theta(a, b) = \min_b \max_a \theta(a, b) \quad a^* \in \arg \max_a \min_b \theta(a, b) \quad b^* \in \arg \min_b \max_a \theta(a, b) \quad (3.8)$$

where (a, b) is a valid action pair for two players and θ denotes the loss for player 1 θ_1 while the loss for player 2 θ_2 is implicitly equal to the negative of θ_1 , since the game is a zero-sum game.

In our problem settings, the minimax theorem would yield when the loss for the adversary equals to the loss for the classifier,

$$\theta_C(h^*, m^*) = \min_h \max_m \theta_C(h, m) \quad (3.9)$$

and as the game is a zero-sum game,

$$\theta_A(h^*, m^*) = - \min_h \max_m \theta_C(h, m) \quad (3.10)$$

such that the adversary aims to maximize the loss of the classifier and then the classifier aims to minimize the loss that adversary had already maximized with respect to its own parameters.

3.1.3 Nikaido-Isado-type Functions

A summand of the Nikaido-Isado function denotes sum of the reduction of the loss for each player when changing its strategy from a to b, while the rest of the player keep their strategies unchanged as in a.

For adversarial learning, a Nikaido-Isado-function summand can be calculated as,

$$\begin{aligned}\psi(a, b) &= [\theta_C(h_a, m_a) - \theta_C(h_b, m_a)] \\ &+ [\theta_A(h_a, m_a) - \theta_A(h_a, m_b)]\end{aligned}\quad (3.11)$$

where $a = (h_a, m_a)$ and $b = (h_b, m_b)$ are two strategy pairs for the classifier and the adversary such that $h_a, h_b \in H$ and $m_a, m_b \in M$.

Hence, the function actually represents the total decrease in the loss when changing the strategy from a to b . Thus, the maximum of this function is always non-negative for a given a . Since, when there exists a strategy combination b that has a smaller total loss, the function takes a positive value. In the other case, when there is no such strategy combination, b could be chosen as $b = a$, then the function value would equal to 0.

Furthermore, if the given a is the Nash equilibrium of the two-player game, then the Nikaido-Isado function is non-positive as none of the players can reduce its loss in that case.

Combining these two properties of the Nikaido-Isado function, the global optimum of the following problem is actually equal to the Nash equilibrium of the game [11].

$$a^* = \arg \min_a \max_b \psi(a, b) \quad (3.12)$$

where a^* is the normalized Nash equilibrium and $a = (h_a, m_a)$ and $b = (h_b, m_b)$ are two strategy pairs for the classifier and the adversary such that $h_a, h_b \in H$ and $m_a, m_b \in M$.

As differentiability of the normalized Nash equilibrium in equation 3.12 is not guaranteed, regularized Nash equilibrium $\psi_\alpha(a, b)$ is defined as the following,

$$\psi_\alpha(a, b) = \psi(a, b) - \frac{\alpha}{2} \|a - b\|^2 \quad (3.13)$$

Additionally α -optimal loss reduction $r_\alpha(a)$ defined as

$$r_\alpha(a) = \max_b \psi_\alpha(a, b) = \psi(a, b'_\alpha(a)) \quad (3.14)$$

where $b = (h_b, m_b)$ is a strategy pair such that $h_b \in H$ and $m_b \in M$ and $b'_\alpha(a)$ is the α -optimal response of action a which is unique for all $\alpha > 0$.

Additionally, as shown in [11], the solution of $\min_a r_\alpha(a)$ is equal to the solution of equation 3.12. Thus we obtain that,

$$a^* = b'_\alpha(a) \quad (3.15)$$

is a normalized Nash equilibrium.

Finally, $\alpha\beta$ -gap function is defined as,

$$\text{for } 0 < \alpha < \beta \quad r_{\alpha\beta}(x) = r_\alpha(x) - r_\beta(x) \quad (3.16)$$

This $\alpha\beta$ -gap function is constrained by,

$$\frac{\beta - \alpha}{2} \|x - b'_\beta(x)\|^2 \leq r_{\alpha\beta}(x) \frac{\beta - \alpha}{2} \|x - b'_\alpha(x)\|^2 \geq r_{\alpha\beta}(x) \quad (3.17)$$

considering the definitions of r_α and ψ_α [11].

Combining all the definitions and properties of the Nikaido-Isado function, vector a^* is a normalized Nash equilibrium of the game if and only if a^* is a global minimum of $r_{\alpha\beta}$ with $r_{\alpha\beta} = 0$ [11].

3.2 Algorithm Derivation

After making the assumptions, a one-shot two-player game between the classifier and the adversary is defined to model the adversarial learning problem. Both parties aim to minimize the individual losses after the game.

3.2.1 Game Result for the Classifier

Adversary-aware Classifier

Loss after the game for the classifier θ_C is equal to the total classification error E_C as in equation 2.12.

$$\theta_C = E_C \quad (3.18)$$

Hence, the classifier aims to minimize the loss in the following equation.

$$\theta_C = \sum_{i=1}^n l(h(m_i(x_i)), y_i) + \frac{\lambda_c}{2} \Omega(h) \quad (3.19)$$

Utility-Based Classifier

Similarly, loss after the game for the utility-based classifier $\theta_C^{utility}$ is equal to the total classification error $E_C^{utility}$ as in equation 2.14.

$$\theta_C^{utility} = E_C^{utility} \quad (3.20)$$

Hence, the loss for the utility-based classifier can be calculated as,

$$\theta_C^{utility} = \sum_{i=1}^n u_i^c l(h(m_i(x_i)), y_i) + \frac{\lambda_c}{2} \Omega(h) \quad (3.21)$$

3.2.2 Game Result for the Adversary

Worst-Case Adversary

Considering the worst-case adversary, loss after the game θ_A^{worst} is equal to the adversarial cost E_A^{worst} which is also equal to the negative of the total classification error.

$$\theta_A^{worst} = E_A^{worst} \quad (3.22)$$

So, the loss for the worst-case adversary θ_A^{worst} is determined as in equation 2.17,

$$\theta_A^{worst} = - \sum_{i=1}^n l(h(m_i(x_i)), y_i) - \frac{\lambda_c}{2} \Omega(h) + \frac{\lambda_a}{2} \Omega(m) \quad (3.23)$$

The important aspect for the worst-case adversary to consider is that loss for the worst-case adversary is exactly the negative of the loss for the classifier.

$$\theta_A^{worst} = -\theta_C \quad (3.24)$$

Goal-Based Adversary

Loss after the game for the goal-based adversary θ_A^{goal} is equal to the goal-based adversarial cost E_A^{goal} in equation 2.19.

$$\theta_A^{goal} = E_A^{goal} \quad (3.25)$$

Hence, if we substitute the equation 2.19 in this equation,

$$\theta_A^{goal} = - \sum_{i=1}^n l(h(m_i(x_i)), y_i) \cdot \frac{(y_i + 1)}{2} - \frac{\lambda_c}{2} \Omega(h) + \frac{\lambda_a}{2} \Omega(m) \quad (3.26)$$

As y_i can only take values in the two-membered set $\{1, -1\}$, we can distribute the sum over these values for each y_i . Then the loss for the goal-based adversary would equal to,

$$\begin{aligned} \theta_A^{goal} &= - \sum_{y_i=1, 1 \leq i \leq n} l(h(m_i(x_i)), y_i) \cdot \frac{(y_i + 1)}{2} \\ &\quad - \sum_{y_i=-1, 1 \leq i \leq n} l(h(m_i(x_i)), y_i) \cdot \frac{(y_i + 1)}{2} \\ &\quad - \frac{\lambda_c}{2} \Omega(h) + \frac{\lambda_a}{2} \Omega(m) \end{aligned} \quad (3.27)$$

If we substitute the values of y_i appropriately and reduce the equation we obtain that,

$$\theta_A^{goal} = - \sum_{y_i=1, 1 \leq i \leq n} l(h(m_i(x_i)), y_i) - \frac{\lambda_c}{2} \Omega(h) + \frac{\lambda_a}{2} \Omega(m) \quad (3.28)$$

Also, for $y_i = -1$ negative (-) instances, according to the assumptions, the adversary wouldn't do any modifications. Hence, the optimization sum can be extended for the negative (-) instances as they would only act as a constant factor. Finally, the loss for the goal-based adversary θ_A^{goal} is then equal to,

$$\theta_A^{goal} = - \sum_{i=1}^n l(h(m_i(x_i)), y_i) - \frac{\lambda_c}{2} \Omega(h) + \frac{\lambda_a}{2} \Omega(m) \quad (3.29)$$

Combining this equation with equation 3.23,

$$\theta_A^{worst} = \theta_A^{goal} \quad (3.30)$$

Hence, loss for the goal-based adversary is equal to the lost for the worst-case adversary which is also exactly equal to the negative of the payoff for the classifier according to the adversary.

$$\theta_A^{goal} = -\theta_C \quad (3.31)$$

Utility-Based Adversary

Finally, loss after the game for the utility-based adversary $\theta_A^{utility}$ is equal to the utility-based adversarial loss $E_A^{utility}$ which is also the negative of weighted sum of the sample-based classification loss.

$$\theta_A^{utility} = E_A^{utility} \quad (3.32)$$

So,

$$\theta_A^{utility} = - \sum_{i=1}^n u_i^a l(h(m_i(x_i)), y_i) \cdot \frac{(y_i + 1)}{2} - \frac{\lambda_c}{2} \Omega(h) + \frac{\lambda_a}{2} \Omega(m) \quad (3.33)$$

For the utility-based adversary, it is important to remark that the loss is no more equal to the negative of the loss for the classifier as previously in the worst-case and goal-based adversary scenarios.

3.2.3 Game Result Calculation

Minimax Approximation

The result of the minimax algorithm is stated in [10],

$$\begin{aligned} \min_w \quad & \sum_i [1 - y_i(w \cdot x_i) + t_i]_+ + \frac{\lambda}{2} \|w\|^2 \\ \text{s.t.} \quad & t_i \geq Kz_i + \sum_j v_{i,j} \\ & v_i \geq 0 \\ & z_i + v_i \geq (y_i(x_i \cdot w)) \end{aligned} \tag{3.34}$$

$$\tag{3.35}$$

where t_i is the amount of increase in the total classification error when the i^{th} sample is modified and z, v are the variable used to constraint the adversarial modification and the increase in the error.

Due to these constraints of the optimization, the problem becomes quadratic. Even though, solving the quadratic optimization can be done efficiently, for huge amounts of data with a lot features, the practical calculation of the result would be cumbersome. Hence, we propose an algorithm to approximate the minimax solution, starting from an initial guess and converges the optimum. An iteration of the approximation algorithm is defined as the following.

When none of the classifier and the adversary is utility-based, since the loss of the adversary is exactly the negative of the loss of the classifier, the defined two-player game between the adversary and the classifier is turned into a zero-sum game. Hence, the optimal result of the game for both of the parties can be found by using minimax theorem as stated in equation 3.8.

When using the Minimax theorem over the loss of the classifier θ_C to find the Nash equilibrium (h^*, m^*) , we obtain,

$$\begin{aligned} (h^*, m^*) &= \arg \min_h \max_m \theta_C(h, m) \\ \text{s.t.} \quad & 0 \leq \|m_i\| \leq K \frac{y_i + 1}{2} \quad \text{for } 0 < i \in Z^+ \leq n \end{aligned} \tag{3.36}$$

As assumed in the previous chapter, hinge loss is chosen as the loss function and an L2-norm function is used as a regularization function. When using a linear classifier with weights w as hypothesis, the loss of the linear classifier would then equal to,

$$\theta_C(h = (w), m) = \sum_{i=1}^n [1 - y_i w \cdot (x_i + m_i)]_+ + \Omega(w, m) \tag{3.37}$$

, we obtain the following optimization problem.

$$\begin{aligned} \min_w [\max_m \sum_{i=1}^n [1 - y_i w \cdot (x_i + m_i)]_+ - \frac{\lambda_a}{2} \Omega(m)] + \frac{\lambda_c}{2} \Omega(m) \\ \text{s.t. } \forall i \quad 0 \leq \sum_{j=1}^d c_j m_{i,j}^2 \leq K \frac{y_i + 1}{2} \quad \text{for } 0 < i \in Z^+ \leq n \end{aligned} \quad (3.38)$$

where c_j is the corresponding adversarial modification cost for the j^{th} feature and $\Omega(w)$ and $\Omega(m)$ are the regularization functions for w and m .

Now, we will reduce this optimization problem. If we separate the constraint for the modification for different for values of $y_i \in \{-1, 1\}$ such as $y_i = -1$ and $y_i = 1$ and substitute the values of y_i we obtain,

$$\forall i \quad \sum_{j=1}^d c_j m_{i,j}^2 \leq K \quad \text{for } y_i = 1, 0 < i \in Z^+ \leq n, \quad (3.39)$$

$$\forall i \quad \sum_{j=1}^d c_j m_{i,j}^2 \leq 0 \quad \text{for } y_i = -1, 0 < i \in Z^+ \leq n \quad (3.40)$$

Hence, we obtain,

$$\forall i \in \{i | y_i = -1\} \quad m_{i,j}^2 = 0 \quad \text{for } 0 < i \in Z^+ \leq n \quad (3.41)$$

as assumed. Hence, we can now rewrite the optimization problem only for the positive cases of y_i .

$$\begin{aligned} \min_w [\max_m \sum_{y_i=1, 1 \leq i \leq n} [1 - y_i w \cdot (x_i + m_i)]_+ - \frac{\lambda_a}{2} \Omega(m)] + \frac{\lambda_c}{2} \Omega(m) \\ \text{s.t. } \forall i \quad \sum_{j=1}^d c_j m_{i,j}^2 \leq K \quad \text{for } y_i = 1, 0 < i \in Z^+ \leq n \end{aligned} \quad (3.42)$$

When only considering the initial maximization in order to determine the minimax result for the adversary, we can remove the constant parts of the optimization with respect to m and reformulate it as a maximization problem

such as,

$$\begin{aligned} & \max_m \sum_{y_i=1, 1 \leq i \leq n} [1 - w \cdot (x_i + m_i)]_+ - \frac{\lambda_a}{2} \Omega(m) \\ \text{s.t. } & \forall i \sum_{j=1}^d c_j m_{i,j}^2 \leq K \quad \text{for } y_i = 1, 0 < i \in Z^+ \leq n \end{aligned} \quad (3.43)$$

As the summands are independent from each other, minimization of the sum is equal to the minimization of its summands.

$$\begin{aligned} & \max_m \forall i \quad [1 - w \cdot (x_i + m_i)]_+ + \frac{\lambda_a}{2} \Omega(m_i) \\ \text{s.t. } & \forall i \sum_{j=1}^d c_j m_{i,j}^2 \leq K \quad \text{for } y_i = 1, 0 < i \in Z^+ \leq n \end{aligned} \quad (3.44)$$

If we rewrite element-wise multiplication between the weights and the modified instance,

$$\begin{aligned} & \max_m \forall i \quad [1 - \sum_{j=1}^d w_j (x_{i,j} + m_{i,j})]_+ - \frac{\lambda_a}{2} \Omega(m_i) \\ \text{s.t. } & \forall i \sum_{j=1}^d c_j m_{i,j}^2 \leq K \quad \text{for } y_i = 1, 0 < i \in Z^+ \leq n, \end{aligned}$$

Now, with disregarding the constraints and the regularization function, let us consider to maximize the feature-based summands for each feature j for different values of w_j and $x_{i,j}$. For the features with $w_j \geq 0$, $m_{i,j} = -1$ maximizes the equation. However, when $x_{i,j} = 0$ already, due to the constraint $m_{i,j} = 0$ should be chosen. Similarly, for the features with $w_j < 0$, choosing $m_{i,j} = 1$ is the maximizer. But, if $x_{i,j} = 1$, $m_{i,j} = 0$ should be chosen.

Without losing generality, we consider that, the features are ordered in an ascending order according to their costs.

$$c_1 \leq c_2 \cdots \leq c_d \quad (3.45)$$

Thus, the whole minimization would be supplied by choosing the maximum number of features j^* such as,

$$\forall i \in \{i | y_i = 1\}, j' \leq d$$

$$j_i^* = \arg \max_{j'} \sum_{j=1}^{j'} c_j [1(w_j \geq 0)x_{i,j} + 1(w_j < 0)(1 - x_{i,j})] \leq K \quad (3.46)$$

where function $1(\cdot)$ returns 1 when the given condition true and 0 otherwise.

Then, the optimal strategy of the adversary is,

$$m_{i,j} = -1 \quad \text{if} \quad y_i = 1 \wedge j \leq j_i^* \wedge x_{i,j} = 1 \wedge w_j \geq 0 \quad (3.47)$$

$$m_{i,j} = 1 \quad \text{if} \quad y_i = 1 \wedge j \leq j_i^* \wedge x_{i,j} = 0 \wedge w_j < 0 \quad (3.48)$$

As the classifier is not trained for these feature modification, its total classification error would increase rapidly. Motivated from [10], the response of the classifier is assigning weight of 0 to those features that are modified by the minimax adversary.

Thus, as the minimax strategy approximation of the classifier, it could minimize the effect of adversarial modification over the total classification error by removing the features that would be modified by the optimal minimax adversary. Here, our approach converge to a similar solution as in [10] for the worst-case or goal-based adversary against a adversary-aware classifier where the loss of the both parties are negative of each other. So, the minimax approximation of the classifier is given in algorithm 1.

Input: Function Spaces H and M, Loss Function l , Data set x, y ,
parameter λ_c

Output: w^* - Minimax Approximation Weights

Train initial hypothesis h_{init}

$$h_{init} = \arg \min_h \sum_{i=1}^n l(h(x_i), y_i) + \frac{\lambda_c}{2} \Omega(h)$$

Get w_{init} from h_{init}

Compute adversarial costs c_j using w_{init}

Compute adversarial minimax strategy with respect to c_j

Generate classifier's response w^*

return w^*

Algorithm 1: Minimax

Finding Nash Equilibrium

As the hypothesis space and the modification space are compact and convex and also the hinge loss is continuous and convex over these spaces, the Nash

equilibrium exists for this problem setting [1]. Hence, the Nash equilibrium can be calculated with using the Nikaido-type functions [11].

We use the same algorithm that is proposed in [4] to calculate the Nash equilibrium. However, as the problem settings are different, the objective functions and the gradients should be calculated for the optimization problems in the algorithm. Hence, the required parameters are generated as given below.

Let us first give the calculation of $r^{\alpha\beta}$ for our problem setting of which global minimum we will try to converge in order find the Nash equilibrium.

$$\begin{aligned}
r^{\alpha\beta}((w, m), (w^*, m^*)) &= r^\beta((w, m), (w^*, m^*)) - r^\alpha((w, m), (w^*, m^*)) \\
&= \theta_A(w, m_{alpha}^*) - \theta_A(w, m_\beta^*) + \theta_C(w_\alpha^*, m) - \theta_C(w_\beta^*, m) \\
&\quad + \frac{\beta}{2} \|w - w_\beta^*\|^2 + \frac{\beta}{2} \|m - m_\beta^*\|^2 - \frac{\alpha}{2} \|w - w_\alpha^*\|^2 - \frac{\alpha}{2} \|m - m_\alpha^*\|^2
\end{aligned} \tag{3.49}$$

Now, let us calculate α -optimal response $(w_{alpha}^*, m_{alpha}^*)$ for $\psi(w, m)$ such as,

$$m_{alpha}^* = \arg \min_m \theta_A(w, m) - \frac{\alpha}{2} \|m - m'\|^2 \tag{3.50}$$

$$w_{alpha}^* = \arg \min_w \theta_A(w, m) - \frac{\alpha}{2} \|m - w'\|^2 \tag{3.51}$$

In order to be used during the convergence to the optimum, we define the gradient $\nabla r^{\alpha\beta}$ of $r^{\alpha\beta}$

$$\begin{aligned}
\nabla_{w,m} r^{\alpha\beta} &= \nabla_w \theta_A(m_\alpha^*) - \nabla_w \theta_A(m_\beta^*) + \nabla_m \theta_C(w_\alpha^*) - \nabla_m \theta_C(w_\beta^*) \\
&\quad + \beta(w - w_\beta^*) + \beta(m - m_\beta^*) \\
&\quad - \alpha(w - w_\alpha^*) - \alpha(m - m_\alpha^*)
\end{aligned} \tag{3.52}$$

The derivative of the adversarial loss function with respect to classifier weights $\nabla_w \theta_A(m)$ is

$$\nabla_w \theta_A(m) = \sum_{i=1}^n \nabla_{w_i} \theta_A(m_i) \tag{3.53}$$

Such that,

$$\begin{aligned}
\nabla_{w_i} \theta_A(m_i) &= -u_i^a y_i (x_i + m_i) \quad \text{where } 1 - y_i w(x_i + m_i) < 0 \\
&= 0 \quad \text{otherwise} \\
&= \forall_i \in 1, 2, \dots, n
\end{aligned} \tag{3.54}$$

Similarly, to calculate the derivate of the classifier's loss function with respect to the adversarial modification $\nabla_m \theta_C(w)$, we have to calculate $\nabla_{m_i} \theta_C(w)$ for all $1 \leq i \leq n$

$$\begin{aligned} \nabla_{m_i} \theta_A(w) &= -u_i^c y_i w \quad \text{where } 1 - y_i w(x_i + m_i) > 0 \\ &= 0 \quad \text{otherwise} \\ &\quad \forall_i \in 1, 2, \dots, n \end{aligned} \tag{3.55}$$

Finally, the algorithm has defined as follows as in [4],

Input: Function Spaces H and M, $0 < \alpha < \beta$, Loss Functions θ_C and θ_A , $0 < \epsilon$

Output: w^*, m^* - Nash Equilibrium

Select arbitrary initial point such $h \in H, m \in H$

Get w from h

repeat

 Compute α -optimal response w_α^*, m_α^* , Compute β -optimal response w_β^*, m_β^*

 Compute $\alpha\beta$ -gap function $r^{\alpha\beta}(w, m)$

 Compute gradient of $\alpha\beta$ -gap function $\nabla r^{\alpha\beta}(w, m)$

 Perform line search over $\alpha\beta$ -gap function with using its gradient

 Update (w,m)

until $r^{\alpha\beta}(w, m) \leq \epsilon$ or $\|\nabla r^{\alpha\beta}(w, m)\| \leq \epsilon$;

return w, m

Algorithm 2: Nash equilibrium Calculation

3.3 Algorithm Flow

Initially, the algorithm checks the loss functions of both the adversary and the classifier. It decides upon the result, to branch the minimax approximation or the Nash equilibrium calculation algorithm. If the loss functions are complement of each other, the algorithm will jump to the minimax approximation routine. Otherwise, the Nash equilibrium calculation algorithm is called to get the corresponding weights at the equilibrium.

For the worst-case and the goal-based adversaries, the utilities of the classifier and the adversary is exactly the negative of each other. Hence, the defined two-player game is a zero-sum game which can be solved with simply using Minimax theorem.

However, when the adversary is utility-based, the game between these two parties is no more a zero-sum game. Hence, a possible result of the game

Input: Function Spaces H and M , Loss Functions θ_C , and θ_A , Data set x,y
Output: w^* - Linear hypothesis
 Check θ_A, θ_C .
if $\theta_A = -\theta_C$ **then**
 Compute w^* using minimax approximation
else
 Compute w^* using Nash equilibrium calculation
end
return w^*

Algorithm 3: Adversarial Learner

should be calculated as an equilibrium. Here, algorithm calculates the Nash equilibrium, similar to [3], of the game with using a routine based on the properties of the Nikaido-Isoda-type functions as mentioned in [11].

Chapter 4

Experiments and Conclusions

4.1 Experiments

4.1.1 Data Sets

Temporal Spam Data Set

There are two spam data sets that are actually generated from real-life mails. First data set has a total of 55000 labeled ham and spam emails and similarly the second one has a total of 70000 labeled spam and ham emails. In order to capture the evolution of the adversarial strategy against the classifier, both of those data sets are sorted according to their timestamps in an ascending order. Furthermore, feature vectors that are corresponding to the data samples are generated using a bag-of-words technique over whole feature set.

Finally, both of them are separated into groups of 5000 emails without disregarding the initial order. Such that, the first groups of the both data sets are corresponding to the training sets and the rest represents a number of test sets that are in order of their timestamps.

Artificial Spam Data Set

In artificial spam data set, the features are separated into three parts, two sets of good features that are used in legitimate emails and a set of bad features that are actual spam content.

During the generation of the training set, for the positive samples, the "bad" features has the highest probability to be included. Additionally, the "good" features have a higher probability than the ones in the second set to be selected. Meanwhile, for the negative instances, the "bad" features have an almost-zero probability and all of "good" features in both sets have the some probability to be included in the sample.

Table 4.1: Distribution Probabilities for the Artificial Training Set

Features	Ham (Training)	Spam (Training)
"bad" words	0.001	0.4
"good" words in set 1	0.05	0.001
"good" words in set 2	0.05	0.2

Table 4.2: Distribution Probabilities for the Artificial Test Set

Features	Ham (Test)	Spam (Test)
"bad" words	0.001	0.4
"good" words in set 1	0.05	0.2
"good" words in set 2	0.05	0.001

However, in the test set, the distribution differs for the positive samples. Now, the "good" features second set has a higher probability to be in a generated positive sample, meanwhile the "bad" features still keep the highest probability. And the negative samples are generated similar to the training set.

The probabilities used for the data set generation are given in tables 4.1 and 4.2.

Here, the generation process could be interpreted as, the spammer has generated an amount of spam emails with a "good" word attack. Then, as the classifier would be trained against these good words, spammer has changed it "good" word attack by differing the "good" words in its spam emails to some other ones while keeping the spam content still.

4.1.2 Implementation

SVMperf

All the sources for the implementation other than the code required for the SVM training method, is coded in Matlab. SVM in the algorithm is trained by using the third party SVM application called SVMperf. The configuration

of the application is modified according to the needs of our approach. Such,

```
-t 0 parameter to choose linear kernel
-l 0 parameter to choose hinge loss
-b 1 parameter to use a bias
```

parameters are used during the training of the initial SVM.

Matlab Optimization Toolbox

The optimization problems that are solve for the calculation of the Nash Equilibrium, are solved by built-in functions from Matlab Optimization Toolbox. *fminunc()* is selected from the toolbox as it is capable of solving multivariate unconstrained optimization problem. It has been called by the given parameters.

```
GradObj on gradient of the function is given
TolFun 0.01 tolerance on the value of the objective function
TolX 0.01 tolerance on the value of the input
```

Basic Feature Selection

In the temporal spam data set, every word in every email corresponds to a feature. Considering 55000 emails with hundreds of words would yield a huge dimension in the feature space. And even, most of this features would be noisy features that are only included in a very small amount of email, hence they could be obviously omitted. Furthermore, due to properties of the English language, some words are redundant since they are so common that do not give any information about the legitimacy of an email. Such as "the", "and", "this", etc. Finally, because of the protocol of the emailing, some words are also used almost in every email, hence they are far from giving any information. The best example for this case is the word "subject" which every mail starts with.

Thus, a basic feature selection method is implemented and applied to data sets prior to the training phase. It simply consists of two thresholds. One of them is an upper bound on the frequency of words in the data set in order to remove redundant features and the other is an lower bound on the frequency of the words in the data set for clearing noisy features. Hence,

only the features with frequency between these bounds are selected as the features of the training set.

Frequency of a word $freq_j$ in the data set is calculated as,

$$freq_j = \frac{\sum_{i=1}^n 1(x_{i,j} > 0)}{\sum_{i=1}^n 1} \quad (4.1)$$

where function $1(\cdot)$ returns 1 if the given condition is true and false otherwise. Then feature selection is simply done by the following rule,

$$select \quad j \quad if \quad lower \leq freq_j \leq upper \quad (4.2)$$

where *lower* and *upper* denotes the lower and upper bound on the frequency of word in the data set respectively.

In our implementation the upper bound of frequency is chosen 0.5 and the lower bound is decided as 0.005. After applying this basic feature selection method to the temporal spam data set, the feature has been reduced from 511340 to 3910 without significantly affecting the total classification error on the initial SVM.

4.2 Results

4.2.1 Worst-Case Scenario

In the worst-case scenario, the losses for the adversary and the classifier are complement of each other. This condition holds when the classifier is an adversary-aware classifier and the adversary is either a worst-case or a goal-based adversary as defined in the problem definition. Hence, the algorithm calculates the game result using the minimax rule.

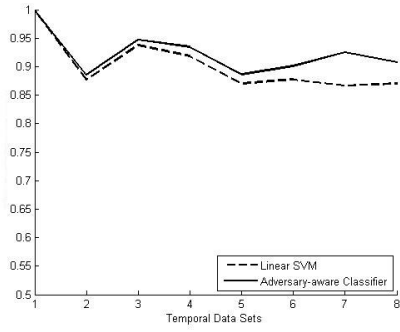
Temporal Spam Data Set

During the experiments, the performance of a linear SVM and the proposed adversary-aware classifier is compared over the temporal spam data set for different values of K .

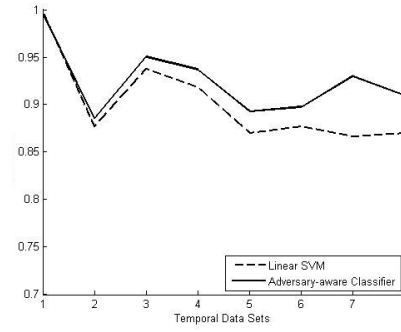
Such comparison is made for $K = 1$, $K = 2$, $K = 3$, $K = 5$ and $K = 10$. Total classification errors for the training set and the temporal test sets are given in the figures 4.2(a), 4.2(b), 4.3(a), 4.3(b) and 4.3 respectively.

Result of the experiments for different values of K are combined together in figure 4.4 in order to be able make comments on the selection of the K value.

Figure 4.1: Worst-Case Scenario with $K = 1$ and $K = 2$

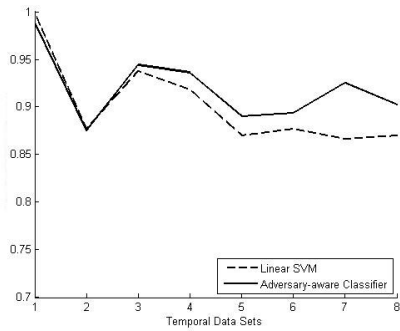


(a) $K = 1$

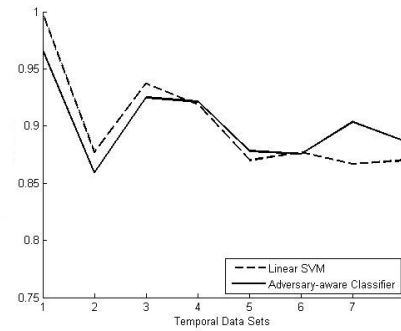


(b) $K = 2$

Figure 4.2: Worst-Case Scenario with $K = 3$ and $K = 5$



(a) $K = 3$



(b) $K = 5$

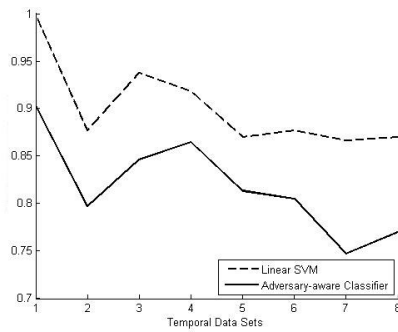


Figure 4.3: Worst-Case Scenario with $K = 10$

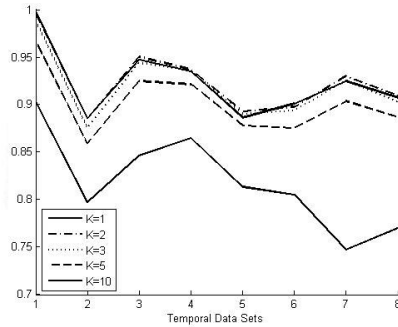


Figure 4.4: Worst-Case Scenario - Comparison of different K values

Artificial Data Set

After comparing the linear SVM and the proposed adversary-aware classifier over the temporal data set, another experiment is held on the artificial data set. Both of the learners are trained over the training set and their performance is considered under the test which has a different distribution than the training set.

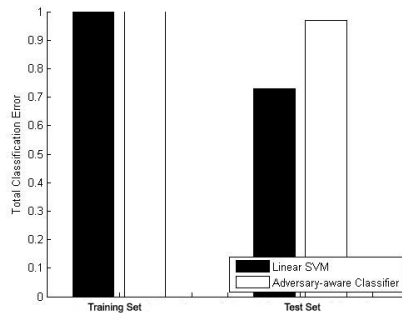


Figure 4.5: Worst-Case Scenario - Comparison of different K values

As seen in 4.5, both of the classifiers had really good performances on the test set from the first distribution. However, when the distribution has been altered to the second one, the overall accuracy of both of the learners had decreased significantly.

4.2.2 Utility-based Scenario

As the calculation of the Nash Equilibrium with using the Nikaido-Isado function and Matlab Toolbox takes a huge amount time, the experiments are only held on the smaller artificial data set.

Table 4.3: Performances for Utility-based Scenario

Algorithm	Performance	Standart Deviation	Error Band
Linear SVM	51.30	0.0350	0.0110
Adversary-aware Classifier	51.70	0.0350	0.0110
Utility-based Classifier	64.06	0.0275	0.0087

Artificial Data Set

After the generation of the artificial data set, performances of the linear SVM, adversary-aware classifier and utility-based classifier is compared in the experiments. Sample-based utilities are introduced in this setting in order the make the two-player game between the adversary and the classifier non-zero sum game.

After training over the same training set, all the algorithms have been run for several times over the test set and the performances are selected according the average of their performance. Furthermore, with using the standart deviation of the results, an error band has been calculated for significance checks.

As seen in table 4.3, in the test set linear SVM and adversary-aware classifier acted almost like uniform random because of their unavailability to take the utilities into account. However, utility-based classifier manage to keep its robustness somehow by keeping the decrease in the performance in a small level.

4.3 Conclusions

4.3.1 Contributions

Feature Addition

By introducing the adversarial modification function in equation 2.28, as an extension to [10], addition of the features by the adversary is included in the model in addition to the feature removal.

Different Sample Utilities

In previous approaches, when calculating the total classification error and the adversarial utilities, the possible importance difference in-between the

samples is disregarded. However, we propose a more general model that has the capability to capture this aspect.

Furthermore, since introducing these new parameters changes the two-player game into a non-zero sum game which is no more solvable with using minimax-based approaches. Hence, the Nash equilibrium of the game is proposed as the optimal result of this new game.

Nash Equilibrium Calculation

Even though, on previous approaches on adversarial learning Nash equilibrium has been suggested as an optimal solution when the loss functions of the adversary and the classifier differs. However, only in [3], the actual Nash equilibrium of the game has been calculated. Also, this calculation only allows one variable per player for the two-player game as it uses a graphical method that is only applicable on two dimensional settings.

In our approach, with the help of Nikaido-Isado functions, we propose an algorithm to calculate the Nash equilibrium of the two-player for arbitrary number of variables. Thus, the algorithm is actually suitable for the real world problems where a lot of variables are available per player to manipulate.

4.3.2 Remarks

Distribution Alteration

The difference between the training and the test distributions would always cause problems during the classification. Meanwhile, the adversary-aware classifier manage to keep its robustness by keeping the decrease in the classification performance at small amounts as seen in 4.5. However, the performance of linear SVM has decreased rapidly due to the alteration of the data distribution as thought in the motivation of the algorithm.

Modification Limit

It is important to notice that the values of the K is not directly equal to the number of features to be modified. Since, the adversarial costs are scaled and normalized versions of the initially trained weights, the smallest ones have a value much smaller than 1. Hence, the modification limit K can only be considered proportionally and a comparison can only be made between the different values of K . Such as, during the experiment with $K = 2$ the classifier assumes a modification limit that is two times the modification limit of the classifier with $K = 1$. However, the number of features removed are not directly double.

Also, as seen in figure 4.4, it is important to choose the appropriate modification Limit K for the problem and the adversary. Choosing a K value greater than the required would increase the total classification error significantly. Since, the more the assumed value of K by the classifier increases, the more informative features are removed during the determination of the weights. However, similarly choosing a lower value of K than the appropriate one, the classification error increases again as the robustness of the classifier to the adversarial modifications decreases.

Regularization Coefficients

One of the reasons that Nikaido-Isado functions are used to calculate the Nash equilibrium is that the yielded optimization problem is without any constraints. Hence, the equilibrium can be calculated faster. However, the actual optimization could have some constraints such as the modification limit constraint in our problem settings.

Actually, these constraints are implicitly added into the optimization problem derived with Nikaido-Isado functions. For instance, the modification limit K for our case, can be implicitly manipulated by changing the regularization coefficient. As we, increase the adversarial regularization coefficient λ_a , the penalty from the complexity of the modification increases and vice versa. Hence, the adversary tends to make modifications with less magnitudes which is actually has the same effect as decreasing the modification budget limit K .

Similarly, any constraints other than the minimization of the loss, can be inserted into the corresponding regularization function and can be tuned by changing the corresponding regularization coefficient.

Parameter Selection

During the experiments, it was also seen that utility-based classifier optimization with high λ_a takes much shorter than the ones with small ones. The more λ_a increases the more the regularization function penalizes the optimization. Hence, for the greater values of λ_a , the optimization mostly focus on the complexity of model that is far more easier problem than minimizing the total classification error.

Also, it was remarked that choosing an appropriate assumption about the adversarial modification is important for the performance of the classifier. Hence, an intuitive algorithm to choose the appropriate K could be proposed as follows.

Input: initial parameter λ_0 , tolerance ϵ , decrease multiplier
 $0 < step < 1$, loss function θ
Output: λ^* - Regularization Coefficient
 Set i to 0
repeat
 Increase i
 $\lambda_i = \lambda_{(i-1)} * step$
 Calculate θ_{λ_i} - Loss with λ_i
until $\theta_{\lambda_i} - \theta_{\lambda_{(i-1)}} < \epsilon$;
return w^*

Algorithm 4: Parameter Selection

Adversarial Minimax Strategy

ADversarial minimax strategy can be interpreted as the adversarial scanning of the features in an ascending order according to their costs. Afterward, the adversary inserts them to the sample if they don't exist for features with $w_j < 0$ and deletes them if they are already in the sample for features with $w_j \geq 0$ until it reaches its modification limit.

However, deleting a feature with $w_j \geq 0$ would only occur in our problem setting when all other $w_j < 0$ that are in the sample, are already added. Hence, for an acceptable modification limit K , the adversary would only add to sample features with $w_j < 0$, since features with $w_j \geq 0$ have always higher modification costs than the features with $w_j < 0$.

It absolutely makes sense, when considering the spam filtering problem where the adversary adds "good" words to the emails without removing any of its actual spam content. Hence, minimax result for the adversary is to add "good" words that are already not in the sample, with the lowest costs.

Bibliography

- [1] Olsder G.J. Basar T. *Dynamic Noncooperative Game Theory*. Society for Industrial and Applied Mathematics, 1999.
- [2] Lowd D. Meek C. Adversarial learning. *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2005.
- [3] Androutsopoulos et. al. A game theoretic model of spam e-mailing. Technical report, Athens University of Economics and Business, 2005.
- [4] Brueckner et. al. Nash-optimal solution of adversarial classification games. Technical report, Max Planck Institute for Computer Science, 2008.
- [5] Dalvi et. al. Adversarial classification. *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.
- [6] Hasti T. et. al. *The Elements of Statistical Learning*. Springer series in statistics, 2001.
- [7] Teo C. H. et al. Convex learning with invariances. *Twenty-First Annual Conference on Neural Information Processing Systems (NIPS)*, 2007.
- [8] Nash J. Non-cooperative games. *Annals of Mathematics* 54, 1951.
- [9] Chan P. K. Mahoney M. V. Learning nonstationary models of normal network traffic for detecting novel attacks. *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.
- [10] Globerson A. Rowies S. Nightmare at test time: Robust learning by feature deletion. *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

- [11] Kanzow C. von Heusinger A. Optimization reformulations of the generalized nash equilibrium problem using nikaido-isoda-type functions. *Computational Optimization and Applications* DOI 10.1007/s10589-007-9145-6, 2007.
- [12] Morgenstern O. von Neumann J. *The Theory of Games and Economic Behaviour*. Princeton University Press, 1944.