Classifier Evaluation and Attribute Selection against Active Adversaries

by

M. Kantarcioğlu
University of Texas

B. Xi and C. Clifton
Purdue University

Technical Report #09-01

Department of Statistics
Purdue University

February 2009

Classifier Evaluation and Attribute Selection against Active Adversaries

Murat Kantarcıoğlu and Bowei Xi and Chris Clifton

University of Texas at Dallas and Purdue University

Abstract: Many data mining applications, ranging from spam filtering to intrusion detection, are faced with active adversaries. In all these applications, initially successful classifiers will degrade easily. This becomes a game between the adversary and the data miner: The adversary modifies its strategy to avoid being detected by the current classifier; the data miner then updates its classifier based on the new threats. In this paper, we investigate the possibility of an equilibrium in this seemingly never ending game, where neither party has an incentive to change. Modifying the classifier causes too many false positives with too little increase in true positives; changes by the adversary decrease the utility of the false negative items that aren't detected. We develop a game theoretic framework where the equilibrium behavior of adversarial classification applications can be analyzed, and provide a solution for finding the equilibrium point. A classifier's equilibrium performance indicates its eventual success or failure. The data miner could then select attributes based on their equilibrium performance, and construct an effective classifier.

Key words and phrases: Adversarial Classification, Game Theory, Attribute Selection, Simulated Annealing.

1 Introduction

Many data mining applications, both current and proposed, are faced with an active adversary. Problems range from the annoyance of spam to the damage of computer hackers to the destruction of terrorists. In all of these cases, statistical classification techniques play an important role in distinguishing the legitimate from the destructive. There has been significant investment in the use of learned classifiers to address these issues, from commercial spam filters to research programs such as those on intrusion detection [9]. These problems pose a significant new challenge not addressed in previous research: The behavior of a class (the adversary) may adapt to avoid detection. Traditionally a classifier is constructed from a training dataset, and future datasets will come from the same population as the training dataset. A classifier constructed by the data miner in such a static environment will not maintain its optimal performance for long, when faced with an active adversary.

An intuitive approach to fight the adversary is to let the classifier adapt to the adversary's actions, either manually or automatically. Such a classifier was proposed in [3], which left open the following issue. The problem is that this becomes a never-ending game between the classifier and the adversary. Or is it never-ending? Will we instead reach an equilibrium, where each party is doing the best it can and has no incentive to deviate from its current strategy? If so, does this equilibrium give a satisfactory result for those using the classifier? Or does the adversary win?

Our approach is *not* to develop a learning strategy for the classifier to stay ahead of the adversary. We instead predict the end state of the game – an equilibrium state. We model the problem as a two-player game, where the adversary tries to maximize its return and the data miner tries to minimize the misclassification error. We examine under which conditions an equilibrium would exist, and provide a stochastic search method and a heuristic method to estimate the classifier performance and the adversary's behavior at such an equilibrium point (e.g., the players' equilibrium strategies). Furthermore, for any given set of attributes, we can obtain the equilibrium strategies of all the subsets of attributes. Such information will enable us to select the most effective attributes to build a classifier. When none of the subset's equilibrium performance is satisfactory, data miner will have to change the rule of the game (for example, to come up with new attributes or increase the penalties for existing ones).

Spam filtering is one such application where classifier degradation is clearly visible and where we can easily observe the actions taken by the adversary (spammer) and the classifier (spam filter). The email in Figure 1.1 shows an example of what an adversary will do to avoid detection by a filter. This email barely looks like what it actually is: An advertisement for mail-order pharmaceuticals (the HTML version uses strange font and style commands to try to be slightly more readable to humans; the text version shows just how far spammers go to avoid filters.) Thus we see that the transformations the adversary makes to defeat the data miner come with a cost: reduced readability. Combining the fact that the reward to the adversary decreases as they try to defeat the data miner, with the data miner's interest in avoiding false positives as well as false negatives, will lead us to an equilibrium where both are best served by maintaining the status quo. (Please see [14] for extensive study of adversarial behaviour evolution in spam filtering.)

Our paper addressed a general statistical problem: classification in an environment where future data

```
From: "Ezra Martens" <ezrabngktbbem...
To: "Eleftheria Marconi" <clifton@pu...
Subject: shunless Phaxrrmaceutical
Date: Fri, 30 Sep 2005 04:49:10 -0500

Hello,
Easy Fast =
Best Home Total
OrdeShipPrricDelivConf
ringpingeseryidentiality
VIAAmbCIALevVALXan
GRAienLISitraIUMax
$  $ $$
3.33 1.21 3.75
Get =additional informmation attempted to
```

Figure 1.1: Spam email with text modified to avoid filters.

and training data are no longer from the same populations. Statistical classification plays important role in such adversarial applications, but it needs further development. These applications have attracted many researchers, but so far people are only working on case-specific solutions. Our model is trying to provide a different perspective to tackle these very difficult problems.

Predicting the eventual equilibrium state has two primary uses. First, the data miner can determine if the approach used will have long term value, before making a large investment into deploying the system. Second, this can aid in *attribute selection*. While it may seem that the best solution is simply to use all available attributes, there is often a cost associated with obtaining the attributes. For example, in spam filtering "white listing" good addresses and "black listing" bad IP addresses is effective, but in addition to blocking some good traffic (a trade-off handled in the classifier learning process), creating and maintaining such lists demands effort. Our approach enables the data miner to predict if such expensive attributes will be effective in the long run, or if the long-term benefit does not justify the cost. In Section 6 we show experimentally that an attribute that is effective at first may not be effective in the long-term.

The paper is organized as follows: In Section 2 we present a game theoretic model. In Section 3 we propose a stochastic search method to solve for the equilibrium. We demonstrate that penalty costs can affect the equilibrium bayes error in interesting ways in Section 4. In Section 5 we provide a computationally efficient heuristic solution, which enables us to deal with multiple attributes. Section 6 presents a simulation study, where we evaluate the equilibrium performance of multiple combinations of attributes, demonstrating the effect of different combinations of attributes and penalties without transformation and in equilibrium. We conclude with a discussion of future work. First, however, we will discuss related work in adversarial classification.

1.1 Related Work

Learning in the presence of an adaptive adversary is an issue in many different applications. Problems ranging from intrusion detection [11] to fraud detection [5] need to be able to cope with adaptive malicious adversaries. As discussed in [3], the challenges created by the malicious adversaries are quite different from those in concept drift [7], because the concept is maliciously changed based on the actions of the classifier. There have been applications of game theory to spam filtering. In [8], the spam filter and spam emails are considered fixed, the game is if the spammer should send legitimate or spam emails, and the user decides if the spam filter should be trusted or not. In [10], the adversary tries to reverse engineer the classifier to learn the parameters. In [3], the authors applied game theory to produce a Naïve Bayes classifier that could automatically adapt to the adversary's expected actions. They concentrated on a single-shot version of the game. While recognizing the importance of an equilibrium state, they simplified the situation by assuming the adversary bases its strategy on the Naïve Bayes classifier rather than their proposed adaptive strategy.

We take a different approach, directly investigating the equilibrium state of the game, at which point all parties will stick to their current strategies. We aim at providing a guide for how to construct classifiers that could lead to the data miner's eventual success in the game.

2 A Game Theoretic Model

In this section we present a game theoretic model for adversarial classification applications. The adversarial classification scenario can be formulated as a two class problem, where class one (π_1) is the "good" class and class two (π_2) is the "bad" class. q attributes will be measured from a object coming from either classes. Denote the vector of attributes by $\mathbf{x} = (x_1, x_2, ..., x_q)'$. Assume the attributes of an object $\mathbf{x}$ will follow different distributions for different classes. Let $f_i(\mathbf{x})$ be the probability density function of class π_i, $i = 1, 2$. The overall population is formed by combining the two classes. Let p_i denote the proportion of class π_i in the overall population. Note $p_1 + p_2 = 1$. The distribution of the attributes $\mathbf{x}$ for the overall population can be considered as a mixture of the two distributions, with the density function written as $f(\mathbf{x}) = p_1 f_1(\mathbf{x}) + p_2 f_2(\mathbf{x})$.

Assume that the adversary can control the distribution of the "bad" class π_2. In other words, the adversary can modify the distribution by applying a transformation $\mathbf{T}$ to the attributes of an object $\mathbf{x}$ that belongs to π_2. Hence $f_2(\mathbf{x})$ will be transformed into $f_2^{\mathbf{T}}(\mathbf{x})$. Each such transformation will come with a cost; the transformed object is less likely to benefit the adversary (although more likely to "get by" the classifier.) When a "bad" object (from π_2) is misclassified as a "good" object (π_1), it generates profit for the adversary. A transformed object from $f_2^{\mathbf{T}}(\mathbf{x})$ will generate less profit than the original one. We assume that the values of p_1 and p_2 will not be affected by the transformation, meaning that adversary will transform the distribution of π_2, but in a short time period will not significantly increase or decrease the number of "bad" objects. Here we examine the case where a rational adversary and a rational data miner play the following game:

1. Given the initial distribution and density $f(\mathbf{x})$, the adversary will choose a transformation $\mathbf{T}$ from

the set of all feasible transformations S, the action space.

2. After observing the transformation $\mathbf{T}$, the data miner will create a classifier h.

Consider the case where data miner wants to minimize its misclassification cost. Given transformation $\mathbf{T}$ and the associated $f_2^{\mathbf{T}}(\mathbf{x})$, the data miner responds with a classifier $h(\mathbf{x})$. Let $L(h,i)$ be the region where the objects are classified as π_i based on $h(\mathbf{x})$, $i = 1,2$. Let the expected cost of misclassification be $C(\mathbf{T},h)$, which is always positive. Define the payoff function of data miner as $u_g(\mathbf{T},h) = -C(\mathbf{T},h)$. In order to maximize its payoff u_g, the data miner needs to minimize the misclassification cost $C(\mathbf{T},h)$.

Note that adversary will only profit from the "bad" objects that are classified as "good". Also note that the transformation may change the adversary's profit of an object that successfully passed the detection. Define $g(\mathbf{T},\mathbf{x})$ as the profit function for a "bad" object $\mathbf{x}$ being classified as a "good" one, after the transformation $\mathbf{T}$ being applied. Define the adversary's payoff function of a transformation $\mathbf{T}$ given h as the following:

$$u_b(\mathbf{T},h) = \int_{L(h,1)} g(\mathbf{T},\mathbf{x}) f_2^{\mathbf{T}}(\mathbf{x})\ d\mathbf{x}$$

Within the vast literature of game theory, the *extensive game* provides a suitable framework for us to model the sequential structure of adversary and data miner's actions. Specifically, the *Stackelberg game* with two players suits our need. In a Stackelberg game, one of the two players chooses an action a_b first and the second player, after observing the action of the first one, chooses an action a_g. The game ends with payoffs to each player based on their utility functions and actions. In our model, we assume all players act rationally throughout the game. For the Stackelberg game, this implies that the second player will respond with the action a_g that maximizes u_g given the action a_b of the first player. The assumption of acting rationally at every stage of the game eliminates the Nash equilibrium with non-credible threats and creates an equilibrium called the *subgame perfect equilibrium.* We assume that each player has perfect information about the other. Here in this context, "perfect information" means that each player knows the other player's utility function. Further more, player two observes a_b before choosing an action. This assumption is not unreasonable since data and other information are publicly available in many applications, such as spam filtering.

The game theoretic framework we propose is very different than the well known strategic games, such as zero-sum game. In a strategic game, each player is not informed about the other player's plan of action. Players will take "simultaneous" actions. Consequently if one player chooses the equilibrium strategy while the other doesn't, the result can be arbitrarily bad for him. In online learning such as [2], a strategic game has been used to learn a concept in real time or make a prediction for the near future by seeing instances one at a time. To the best of our knowledge, those work do not try to deal with situations where an adversary changes the distribution of the underlying concept.

Compared with strategic games, we choose the Stackelberg Game to emphasize on the sequential actions of the two players: Adversary acts first by transforming the population under its control; then data miner responds to adversary's action by adjusting the classifier parameters. Examining the classifier's performance at the equilibrium, where adversary maximizes its gain after the classifier being optimized to defeat its action, is a sensible choice in Stackelberg game.

Hence we define the **Adversarial Classification Stackelberg Game** $G = (N, H, P, u_b, u_g)$: *$N = \{$adversary, data miner$\}$. Set of sequences $H = \{\emptyset, (\mathbf{T}), (\mathbf{T}, h)\}$ s.t. $\mathbf{T} \in \mathcal{S}$ and $h \in \mathcal{C}$, where $\mathcal{S}$ is the set of all admissible transformations for adversary, and $\mathcal{C}$ is the set of all possible classification rules given a certain type of classifier. Function P assigns a player to each sequence in H: $P(\emptyset) =$adversary and $P((\mathbf{T})) =$data miner. Equivalently there exists an corresponding function A that assigns an action space to each sequence in H: $A(\emptyset) = \mathcal{S}$, $A((\mathbf{T})) = \mathcal{C}$, and $A((\mathbf{T}, h)) = \emptyset$. Payoff functions u_b and u_g are defined as above.*

Our model can accommodate all types of classifiers. However we assume the data miner will stick to one type of classifier while the adversary can choose from all the transformations in the action space. For example if the data miner chooses a Bayesian classifier, it will adjust the Bayesian classifier with new weights in order to defeat the adversary's transformations. The data miner won't use a Bayesian classifier facing certain transformations and a decision tree against other transformations. This is a realistic assumption because of development costs: adjusting parameters on a model (or even retraining) is much less expensive than switching to a new type of model.

We notice that the action space of adversary $\mathcal{S}$ can be quite complex. However in reality people have to deal with every transformation employed by adversary. It is well documented and can be observed from data. In such cases, a formal model will provide a systematic approach in a seeming chaos.

In this game, we assume the adversary will act by first applying a transformation $\mathbf{T}$. After observing $\mathbf{T}$ being applied to the "bad" class ($f_2^{\mathbf{T}}(\mathbf{x})$), the optimal classification rule becomes $h_{\mathbf{T}}(\mathbf{x})$. $h_{\mathbf{T}}(\mathbf{x})$ is the best response of data miner facing a transformation $\mathbf{T}$. Let $L(h_{\mathbf{T}}, 1)$ be the region where the objects are classified as π_1 given $h_{\mathbf{T}}$. Define the adversary gain of applying transformation $\mathbf{T}$ as:

$$W(\mathbf{T}) \;=\; u_b(\mathbf{T}, h_{\mathbf{T}}) \;=\; \int_{L(h_{\mathbf{T}},1)} g(\mathbf{T},\mathbf{x}) f_2^{\mathbf{T}}(\mathbf{x})\, d\mathbf{x} \;=\; E_{f_2^{\mathbf{T}}}\left(I_{\{L(h_{\mathbf{T}},1)\}}(\mathbf{x}) \times g(\mathbf{T},\mathbf{x})\right). \tag{2.1}$$

$W(\mathbf{T})$ is the expected value of the profit generated by the "bad" objects that will pass detection under transformation $\mathbf{T}$ and the data miner's optimal classification rule against $\mathbf{T}$. When both parties are rational players, each will attempt to maximize their payoff. Therefore we can write the subgame perfect equilibrium as $(\mathbf{T}^e, h_{\mathbf{T}^e})$, where

$$\mathbf{T}^e = \operatorname{argmax}_{\mathbf{T}\in\mathcal{S}} \;(\; W(\mathbf{T})\;). \tag{2.2}$$

Game theory [13] established that the solution of the above maximization problem is a subgame perfect equilibrium. Furthermore if the action space $\mathcal{S}$ is compact and $W(\mathbf{T})$ is continuous, the maximization problem has a solution.

Another important aspect of the Adversarial Classification Stackelberg game and its subgame perfect equilibrium is that once an equilibrium point is reached, even if the game is repeated, both parties will not have an incentive to change their actions: *Assume that the Adversarial Classification Stackelberg game is played n times. Assume p_1, p_2 and $f_1(\mathbf{x})$ stay the same for the n rounds, and the adversary applied $\mathbf{T}^e$, the subgame perfect equilibrium transformation defined by Equation 2.2, in the k^{th} game. Since both parties will change their actions only to increase their payoffs, the adversary will not change $f_2^{\mathbf{T}^e}(\mathbf{x})$ in the j^{th} round where $k < j \leq n$. Then the data miner will stick to $h_{\mathbf{T}^e}(\mathbf{x})$ in the j^{th} round where $k < j \leq n$.*

We will argue by contradiction. When $\mathbf{T}^k = \mathbf{T}^e$, let us assume that adversary finds another transformation $\mathbf{T}^*$ that increases its payoff. This implies that

$$u_b(\mathbf{T}^{k+1}, h_{\mathbf{T}^{k+1}}) = u_b(\mathbf{T}^* \circ \mathbf{T}^e, h_{\mathbf{T}^* \circ \mathbf{T}^e}) \ > \ u_b(\mathbf{T}^e, h_{\mathbf{T}^e})$$

However $u_b(\mathbf{T}^e, h_{\mathbf{T}^e})$ is already the maximum value. Therefore, $\mathbf{T}^{k+1} = \mathbf{T}^e$ if u_b has a unique maximum value. And $h_{\mathbf{T}^{k+1}} = h_{\mathbf{T}^k} = h_{\mathbf{T}^e}$. Following the argument above, $\forall j > k+1$, $\mathbf{T}^j = \mathbf{T}^e$. and $h_{\mathbf{T}^j} = h_{\mathbf{T}^e}$.

The above formulation can accommodate any well defined set of transformations $\mathcal{S}$, any appropriate distributions with densities $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$, and any meaningful profit function $g(\mathbf{T}, \mathbf{x})$.

We solve the above equations by exploiting the structure of the game: To search for the equilibrium in Stackelberg game is equivalent to solve an optimization problem. We present a general solution based on stochastic search in Section 3, and a heuristic solution based on an approximation of the classification region for minimal cost Bayesian classifier in Section 5, for large scale tasks.

3 Solving for the Equilibrium

To compute the subgame perfect equilibrium, the underlying problem is always converted to an optimization problem similar to the one defined in Equation 2.2 [1]. Although there are computational game theory tools to find subgame perfect equilibrium for *finite* games [12], unfortunately computing subgame perfect equilibrium is a hard problem in general [1]. Therefore several heuristic optimization techniques, such as genetic algorithms, have been applied to compute subgame perfect equilibriums [16]. To the best of our knowledge, none of the existing computational game algorithms can be applied to our case due to the special structure of the adversary gain $W(\mathbf{T})$. Since the domain of the integration $L(h_{\mathbf{T}}, 1)$ for the adversary gain $W(\mathbf{T})$ is a function of the transformation $\mathbf{T}$, finding an analytical solution to the maximization problem is very challenging. In addition, even calculating the integration analytically for a specific transformation is not possible for high dimensional data. We have to numerically evaluate $W(\mathbf{T})$. Because of the difficulties, we consider stochastic search algorithms for finding an approximate solution. A typical stochastic search algorithm for optimization problems works as follows: The algorithm starts with a random initial point and then searches the solution space by moving to different points based on some selection criterion. This process involves evaluating the target function at the selected points in the solution space. Clearly, this implies a computationally efficient method for calculating $W(\mathbf{T})$ is required, since the function will be evaluated at thousands of transformations in $\mathcal{S}$. Furthermore a stochastic search algorithm with the ability to converge to the global optimal solution is highly desirable. In the rest of this section, the Monte Carlo integration method is introduced to compute $W(\mathbf{T})$ and a simulated annealing algorithm is implemented to solve for the subgame perfect equilibrium.

3.1 Monte Carlo Integration

The Monte Carlo integration technique converts a given integration problem to computing an expected value. Assume that we would like to calculate $\int g(\mathbf{x})dx$. If we can find a probability density function

$f(\mathbf{x})$ ($\int f(\mathbf{x})d\mathbf{x} = 1$) that is easy to sample from, then

$$\int g(\mathbf{x})d\mathbf{x} = \int \frac{g(\mathbf{x})}{f(\mathbf{x})} \times f(\mathbf{x})d\mathbf{x} = E_f[\frac{g(\mathbf{x})}{f(\mathbf{x})}].$$

The integration is equal to the expected value of $g(\mathbf{x})/f(\mathbf{x})$ with respect to the density $f(\mathbf{x})$.

The expectation of $g(\mathbf{x})/f(\mathbf{x})$ is estimated by computing a sample mean. Generate m samples $\{\mathbf{x}^i\}$ from $f(\mathbf{x})$ and calculate $\mu_m = 1/m \times \sum_1^m (g(\mathbf{x}^i)/f(\mathbf{x}^i))$. When the sample size m is large enough, μ_m provides an accurate estimate of $\int g(\mathbf{x})d\mathbf{x}$.

The adversary gain $W(\mathbf{T})$ can be written as:

$$W(\mathbf{T}) = \int \left(I_{L(h_{\mathbf{T}},1)}(\mathbf{x}) \times g(\mathbf{T},\mathbf{x})\right) f_2^{\mathbf{T}}(\mathbf{x})d\mathbf{x}$$

In the above formula, $I_{L(h_{\mathbf{T}},1)}(\mathbf{x})$ is an indicator function. It returns 1 if $\mathbf{x}$ is classified into π_1, else it returns 0. $f_2^{\mathbf{T}}(\mathbf{x})$ is naturally a probability density function. Therefore $W(\mathbf{T})$ could be calculated by sampling m points from $f_2^{\mathbf{T}}(\mathbf{x})$, and taking the average of $g(\mathbf{T},\mathbf{x})$ for the sample points that fall into $L(h_{\mathbf{T}}, 1)$. The pseudo-code for Monte Carlo integration is given in Algorithm 3.1.

Algorithm 3.1 Monte Carlo Integration

```
{Evaluating $W(\mathbf{T})$ for a given transformation $\mathbf{T}$}
Generate $m$ samples $\{\mathbf{x}^i\}$ from $f_2^{\mathbf{T}}(\mathbf{x})$
$sum = 0$
for $i = 1$ to $m$ do
  if $\mathbf{x}^i \in L(h_{\mathbf{T}}, 1)$ then
    $sum = sum + g(\mathbf{T}, \mathbf{x}^i)$
  end if
end for
return $sum/m$
```

3.2 Simulated Annealing

Simulated annealing is a stochastic search method that is based on an analogy taken from physics [4]. Physical systems that have many interacting components can be in any of the possible states based on some probability distribution. For high temperatures, a system can be in any one of the possible states with roughly equal probability. As the temperature decreases, the system will choose a low energy state with higher probability. Similarly, when the temperature is high, our simulated annealing algorithm will accept nearly all new points. As the temperature gets lower later in the search, the algorithm will converge to a global optimal solution.

Our version of the simulated annealing algorithm first selects a few random transformations and tries to get a good starting transformation. After the selection of the initial transformation, for each temperature, a few hundred transformations are selected from the neighborhood of the current transformation. A new transformation replaces the current transformation if it gives a larger value of $W(\mathbf{T})$. In case the new transformation is not better than the current one, simulated annealing algorithm may accept it with some probability. This probability is calculated using the value of the new transformation, the value of the current transformation and the current temperature. This probabilistic step enables the algorithm to escape local maxima [4]. The current temperature is reduced by multiplying it by a reduction rate constant r, where $0 < r < 1$. The whole process is repeated until the algorithm freezes. The pseudo-code is given in Algorithm 3.2.

Algorithm 3.2 Simulated Annealing Algorithm to Solve for Equilibrium

```
Require: $TempMin$, $TempMax$, $ReductionRate \in (0,1)$, $SampleSize$
1: Select random $\mathbf{T}$ and evaluate $W(\mathbf{T})$
2: Let $\mathbf{T}_c$ be the starting transformation with value $evalc = W(\mathbf{T}_c)$
3: Let $\mathbf{T}_g$ be the best transformation seen in the search with value $evalg = W(\mathbf{T}_g)$
4: $\mathbf{T}_g = \mathbf{T}_c$, $evalg = evalc$
5: $TempCurrent = TempMax$
6: while $TempCurrent \geq TempMin$ do
7:   for $i = 1$ to $SampleSize$ do
8:     Randomly select $\mathbf{T}_n$ in neighborhood of $\mathbf{T}_c$
9:     Let $evaln = W(\mathbf{T}_n)$ for $\mathbf{T}_n$
10:    if $evaln > evalc$ then
11:      $\mathbf{T}_c = \mathbf{T}_n$, $evalc = evaln$
12:      if $evalg < evaln$ then
13:        $\mathbf{T}_g = \mathbf{T}_n$, $evalg = evaln$
14:      end if
15:    else if $rand(0,1) \leq exp\{\frac{evaln - evalc}{TempCurrent}\}$ then
16:      $\mathbf{T}_c = \mathbf{T}_n$, $evalc = evaln$
17:    end if
18:  end for
19:  $TempCurrent \times= ReductionRate$
20: end while
```

There are practical issues about searching for the subgame perfect equilibrium. First of all, we need a sample of transformed objects, for both the construction of the classification rule and Monte Carlo integration. We can solve this problem by either simulating a dataset from the transformed population, or creating bootstrap samples from a real life data set. Then we need to re-train the classifier and construct the classification region $L(h_{\mathbf{T}}, 1)$ for a selected transformation $\mathbf{T}$ during the search process. For the classifiers that require the knowledge of the density functions, such as minimal cost Bayes classifier, we can re-train the classifiers by leveraging the established and ongoing research about multivariate density estimation. For other types of classifiers, such as k-nearest neighbor and decision tree, we can build the classification regions without relying on the knowledge of the densities by using bootstrap samples or simulated data. Therefore, our formulation could work with any type of classifier.

Given enough time for the simulated annealing algorithm to converge, it is guaranteed to reach the equilibrium transformation for the adversary. Afterward we can obtain the data miner's equilibrium classification rule. We have done experiments to test simulated annealing's effectiveness. The results showed that simulated annealing algorithm performed up to expectation. They are omitted from the paper due to limited space. None of the simulations in Section 4 use simulated annealing, for such a search algorithm demands excessive computational time. Hence later in Section 5 and 6, we focus on Bayes classifier and provide a simple heuristic solution. The heuristic solution also avoids estimating the multivariate density function, which needs a large sample in high dimensional space.

4 Equilibrium Performance

We have done simulations to examine various equilibrium strategies, showing that there exist "interesting" equilibria for this problem. The simulation results show the effect of various penalties and cost matrices. Attributes carrying heavy penalties will force the adversary to stop. Yet after a certain point, further increasing the penalty of an attribute won't have a significant impact. It remains a challenge to discover the penalty imposed on the adversary for any given attribute, as well as the effect on classification accuracy. The data miner could examine the attributes one by one in simulations similar to the one below, given the attribute's initial distributions and the misclassification costs. This gives a rough estimate of how heavy the penalty needs to be for the attribute to have value. In real life applications, attributes with penalties smaller than the effective values will not have a good long term performance.

All the simulations in this section involved only one attribute. The results are obtained from exhaustive search on fine grids in the action space. With one attribute in the model we were able to obtain very accurate estimates of the equilibrium transformations without a time consuming stochastic search. Gaussian distributions and minimal cost Bayesian classifier were used in the experiments. Gaussian distributions have a particularly helpful property: after a linear transformation of the attributes, we still have a Gaussian distribution and an explicit expression for the density. This combination as a simple example gave us important insight about the equilibrium strategies. Next we will define the profit function and explain the set-up for the simulations.

4.1 Profit Function and Gaussian Mixture

First define the one-dimensional profit function $g(T, x)$ as:

$$g(T, x) = max(k - a\left|T^{-1}(x) - x\right|, 0), \tag{4.1}$$

where x is the transformed "bad" object, $T^{-1}(x)$ is the original one, and k and a are positive constant numbers. To quantify the difference of the "bad" object $T^{-1}(x)$ before and after transformation T, we compute the absolute value of $T^{-1}(x) - x$. k is the constant profit generated by original "bad" objects if it is not detected, which is also the maximum profit that a "bad" object could possibly generate. In our simulation study, we assume the profit will decline linearly according to the extent of the transformation. Here a is the reduction rate. And the minimum profit is 0. A "bad" object won't generate negative profit. This definition of the profit is based on the following intuition: The more the original distribution changes, the higher the cost for the adversary. Although more "bad" objects can avoid being detected, each object will generate less profit for the adversary. Hence it is possible to reach a point where adversary stops modifying the objects. Then the equilibrium is established. Further assume that each class π_i, $i = 1, 2$, follows a Gaussian distribution. $f_i(x)$ is the density function of $N(\mu_i, \sigma_i^2)$.

Consider the set of linear transformations $\mathcal{L}$. Let T be a real number, and the transformed object x is simply $T \times T^{-1}(x)$. Here $\mathcal{L}$ will be limited to a certain interval, not the entire real line, because the equilibrium transformation will not be too far away from the identity transformation when there is a penalty. Under transformation T, $f_2^T(x)$ becomes the density of $N(T \times \mu_2, T^2 \times \sigma_2^2)$, which is the new distribution for the "bad" class π_2.

The expected cost of misclassification can be written as [6]:

$$C(T,h) = \int_{L(h,1)} (c(1,1)p_1 f_1(x) + c(1,2)p_2 f_2^T(x))\, dx + \int_{L(h,2)} (c(2,1)p_1 f_1(x) + c(2,2)p_2 f_2^T(x))\, dx,$$

where $c(i,j)$ is the cost of classifying an object into class π_i given that it is actually in class π_j. The optimal minimal cost Bayesian classification rule under transformation T is:

$$h_T(x) = \begin{cases} \pi_1 & (c(1,2) - c(2,2))p_2 f_2^T(x) \leq (c(2,1) - c(1,1))p_1 f_1(x) \\ \pi_2 & \text{otherwise} \end{cases}$$

$h_T(x)$ is the decision rule that minimizes the expected misclassification cost of the data miner $C(T,h)$. Let $e_1(\mathbf{T})$ be the percentage of misclassified "good" objects and $e_2(\mathbf{T})$ be the percentage of misclassified "bad" objects. The Bayes error $e(\mathbf{T}) = p_1 \times e_1(\mathbf{T}) + p_2 \times e_2(\mathbf{T})$. They are functions of the transformation $\mathbf{T}$.

Rewrite the subgame perfect equilibrium transformation using the above specifics as follows:

$$T^e = \text{argmax}_{T \in \mathcal{L}} \left(\int_{L(h_T,1)} max(k - a\left|\frac{x}{T} - x\right|, 0) \times f_2^T(x)\, dx \right). \tag{4.2}$$

4.2 Adversary Gain and Bayes Error

In the experiments we will examine the adversary gain $W(\mathbf{T})$ as a function of $\mathbf{T}$. Furthermore we will see how the equilibrium will change for increasing penalties and different misclassification costs.

4.2.1 Effect of Penalty and Misclassification Cost

The general set-up of the experiments was explained in Section 4.1. The parameter values are given in Table 4.1. There are three experiments. In each experiment we fixed a cost matrix c and the initial distributions of the two classes, and gradually increased the penalty a. The only difference among the three experiments is the cost matrix in the Bayesian classifier. First the cost of misclassifying a "good" object is equal to the cost of misclassifying a "bad" one: $c(2,1)/c(1,2) = 1$. In the second experiment misclassifying a "good" object costs twice as much: $c(2,1)/c(1,2) = 2$. Then in the third experiment misclassifying a "good" object costs ten times as much: $c(2,1)/c(1,2) = 10$. From the three experiments we are able to observe the effect of classifier's cost and penalty of transformation on the equilibrium.

When $T = 1$, π_2 was not transformed. The *initial adversary gain* $W(1)$ and the *initial Bayes error* $e(1)$ are the gain and the error rate under identity transformation, i.e. without transformation. Notice

Table 4.1:

	k	Cost Matrix c	π_1	p_1	π_2	p_2
Experiment 1	1	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	$N(5, 0.64)$	0.65	$N(1, 0.36)$	0.35
Experiment 2	1	$\begin{bmatrix} 0 & 1 \\ 2 & 0 \end{bmatrix}$	$N(5, 0.64)$	0.65	$N(1, 0.36)$	0.35
Experiment 3	1	$\begin{bmatrix} 0 & 1 \\ 10 & 0 \end{bmatrix}$	$N(5, 0.64)$	0.65	$N(1, 0.36)$	0.35

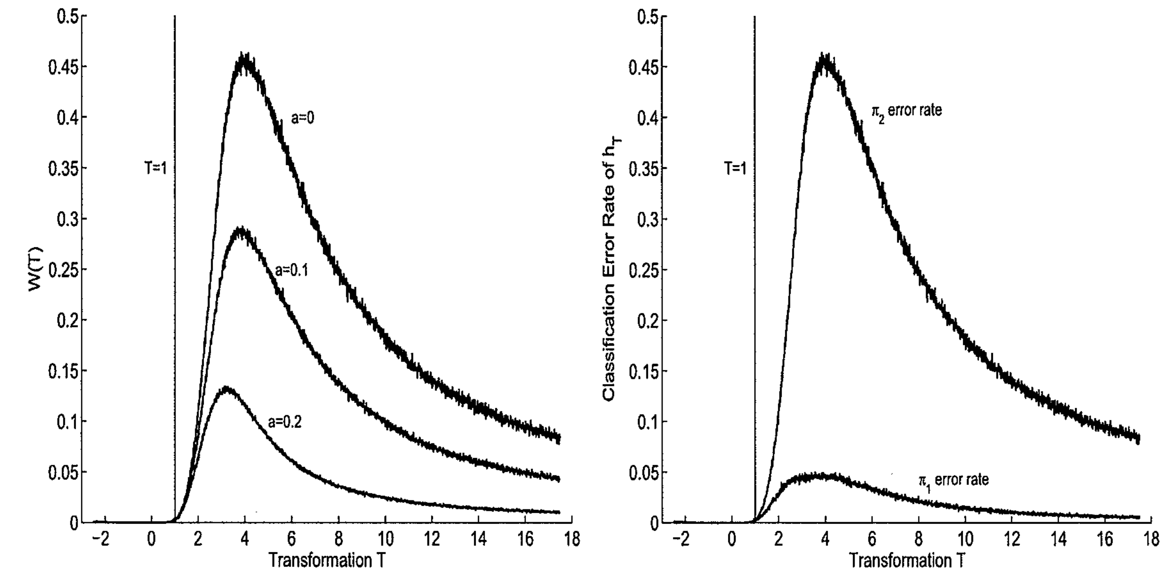

Figure 4.1: Left: the adversary gains for increasing penalty; Right: the Bayes error. Both from Experiment 1.

that they are not affected by the penalty a under the current set-up, since $T^{-1}(x) - x = 0$. $W(1)$ and $e(1)$ are only related to the cost matrix c. Without any transformation, the classifier was very successful, as shown in both Figure 4.1 and Table 4.2. Increasing the cost of misclassifying a "good" object slightly increased the error rate from 0.0022 to 0.0041, thus increased the adversary gain from 0.0045 to 0.0113.

Table 4.2: Initial adversary gain and Bayes error

	$W(1)$	$e(1)$
Experiment 1	0.0045	0.0022
Experiment 2	0.0059	0.0027
Experiment 3	0.0113	0.0041

However when the adversary starts to transform the "bad" objects, the classifier after adjustment failed to block a significant proportion of the transformed "bad" objects while making some mistakes with the "good" ones. Since we set $k = 1$, when $a = 0$, we get $W(T) = e_2(T)$. Also notice that the

Bayes error (the overall misclassification error rate) as a function of T was not affected by the penalty a. Increasing the penalty a reduced the adversary gain for all the transformations because a "bad" object generated less profit, as shown in the left panel of Figure 4.1.

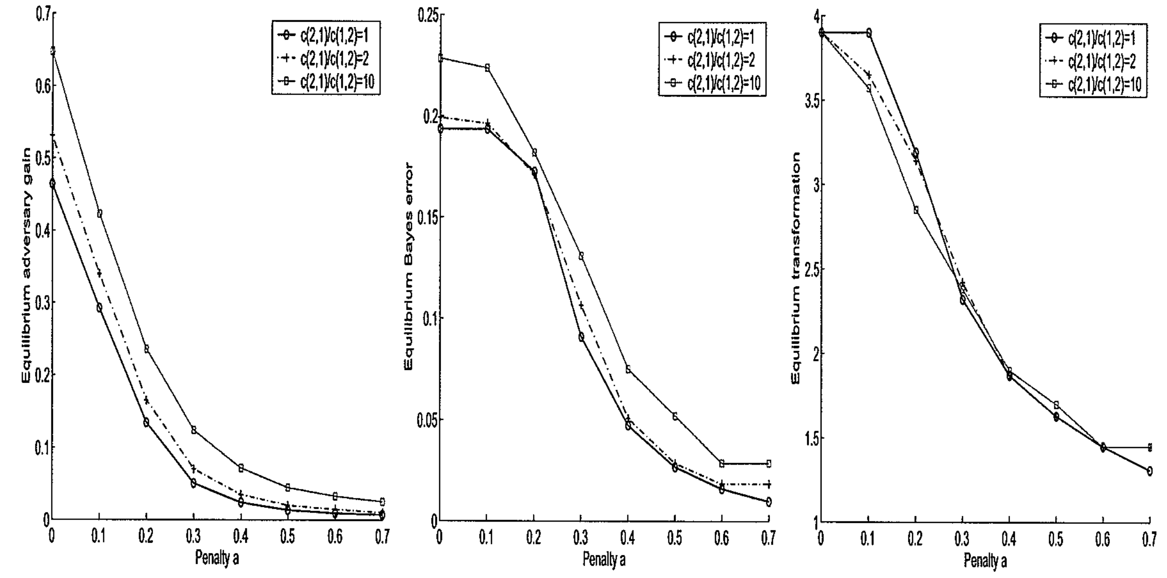

Figure 4.2: Left: the equilibrium adversary gains; Middle: the equilibrium Bayes errors; Right: the equilibrium transformations. Penalty a increased from 0 to 0.7 by 0.1.

Figure 4.2 illustrates the effect of penalty a more clearly. As a increased from 0 to 0.7, the equilibrium transformation was pushed toward 1, the identity transformation. Both the equilibrium adversary gain and the equilibrium Bayes error were dropping down to near 0. Although the Bayes error, which is a function of T, was not affected by penalty a, as the equilibrium transformation moved toward the identity transformation, the equilibrium Bayes error were closer to the initial Bayes error.

The left panel of Figure 4.2 shows that $a = 0.2$ is an elbow point of the equilibrium adversary gain $W(T^e)$. For a heavy penalty $a \geq 0.2$, $W(T^e)$ was close to 0, and the equilibrium Bayes error $e(T^e)$ started declining sharply. Misclassification costs also affect the equilibrium performance. As the cost of misclassifying a "good" object $c(2,1)$ increased, the equilibrium Bayes error was larger, and $W(T^e)$ increased too.

To more carefully examine the effect of penalty and cost matrices on the equilibrium Bayes error and the equilibrium transformation, we gradually increased a from a small (0) to a moderate (0.2) penalty in 0.01 increments. In Figure 4.3 $W(T^e)$ was decreasing as a increased for all three cost matrices. But the equilibrium transformation T^e started moving notably toward 1 only when a increased to a certain level. The same phenomenon occurred for the equilibrium Bayes error $e(T^e)$. Furthermore when the cost of misclassifying a "good" object $c(2,1)$ was equal to $c(1,2)$, it required a heavier penalty for T^e to significantly move toward identity transformation, and for $e(T^e)$ to drop.

When the data miner chooses to monitor certain attributes in order to block the "bad" objects, it may achieve great success at the beginning. However when facing an active adversary, the distribution of one class will be modified constantly. The performance of a classifier depends heavily on how the adversary

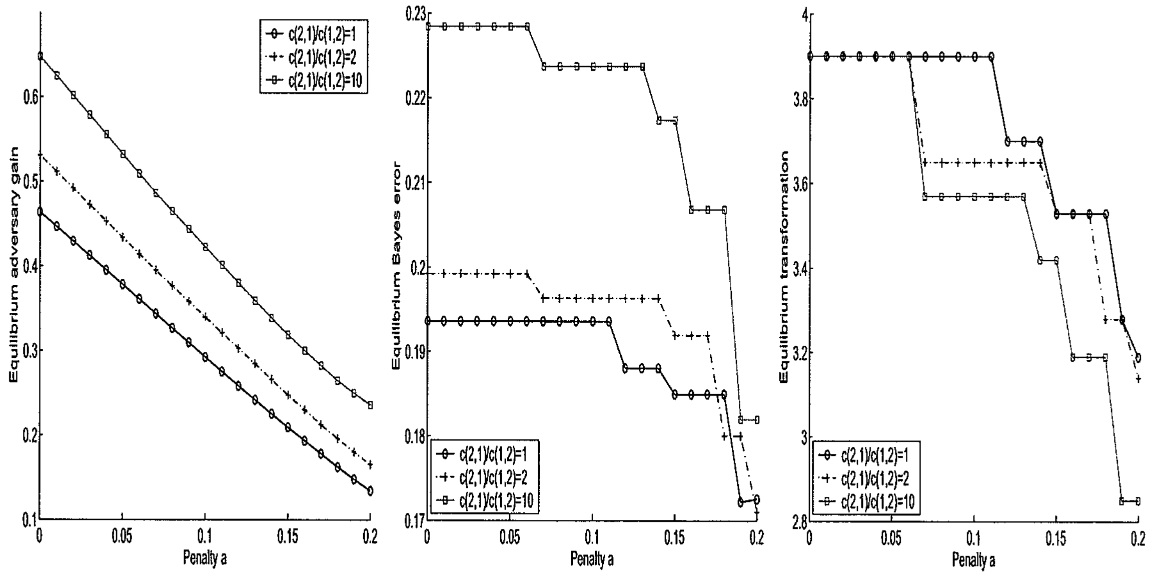

Figure 4.3: Left: the equilibrium adversary gains; Middle: the equilibrium Bayes errors; Right: the equilibrium transformations. Penalty a increased from 0 to 0.2 by 0.01.

will transform the "bad" class. *Therefore we should focus on a classifier's equilibrium performance, where the adversary maximizes its gain, instead of its initial success.* Our game theoretic model has the ability to project a classifier's eventual effectiveness without waiting for the adversary to apply all sorts of transformations.

When facing an active adversary, all the following three quantities can be used to evaluate a classifier's equilibrium performance, which in turn indicates a classifier's long term success or failure. These three quantities can be stated as follows: 1) Its equilibrium Bayes error and misclassification cost; 2) Adversary's equilibrium transformation; 3) Adversary's equilibrium gain. Clearly these three quantities are correlated. At the first glance, it seems like if adversary's equilibrium transformation is close to identity transformation, then the data miner wins the battle. However when the cost of misclassifying "good" objects is huge, classifier will pass more objects from both classes. The adversary does not have to transform much to maximize its gain. This is not a favorable scenario for the data miner. Figure 4.3 shows that adversary's equilibrium transformation might be close to identity transformation, yet the adversary managed to have more "bad" objects pass the classifier, and receive big profit from each "bad" object. Therefore in Section 5, we only use classifier's equilibrium Bayes error and expected misclassification cost to examine the long term effectiveness of a given set of attributes. After all a classifier is built to block the "bad" objects without making too many mistakes with the "good" ones.

4.2.2 A Discontinuous Point

Here we present a special case. If there exists a transformation $\mathbf{T}^0$ that makes two classes indistinguishable (i.e., $f_2^{\mathbf{T}^0}(\mathbf{x}) = f_1(\mathbf{x})$), $W(\mathbf{T})$ is discontinuous at $\mathbf{T}^0$. The following simple one-dimensional example shows such a discontinuous point. The parameter values for this example is in Table 4.3. In this simple example,

Table 4.3: Parameter values where $W(T)$ is discontinuous

				π_1	p_1	π_2	p_2
$k=1$	$a=0$	$c=$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	$N(0.2160, 0.3168)$	0.5	$N(0.36, 0.88)$	0.5

there is no cost for correctly classifying the objects and there is equal cost for misclassification. Also there is no penalty for transformation ($a = 0$). Further more, the number of "good" objects is the same as the number of "bad" objects (i.e., $p_1 = p_2$).

In our example, $T^0 = 0.6$ would transform $f_2(x)$ into $f_1(x)$. The adversary gain $W(T)$ would reach a maximum value 1 at T^0. Note that the transformation $T^0 = 0.6$ and the corresponding Bayesian classification rule form a subgame perfect equilibrium. In Figure 4.4, the value of the adversary gain $W(T)$ is plotted against the transformation T. $W(T)$ is continuous everywhere else except for the equilibrium transformation $T^0 = 0.6$. Furthermore $W(T)$ has local minimal and maximal regions.

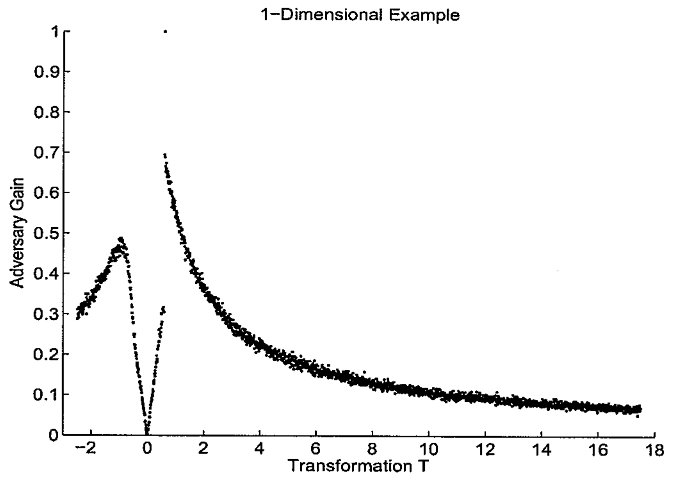

Figure 4.4: Adversary gain with true transformation equal to 0.6

The discontinuity is created by the sudden change of the classification region. At the equilibrium the "bad" class coincides with the "good" class, and under the above setting all the objects will be classified as "good". We then integrate over the entire space to compute $W(\mathbf{T})$. For any other transformation $\mathbf{T}$, the classification region $L(h_{\mathbf{T}}, 1)$ will experience a continuous change with respect to $\mathbf{T}$ and will not be the entire space.

If the two classes won't have the same distribution under any transformation, then we will be able to accurately estimate the equilibrium state.

5 A Heuristic Solution for Bayesian Classifier

Section 3 provides a general solution. It works well in low dimensional space. However when there are a large number of attributes, it needs a large Monte Carlo sample to estimate $W(\mathbf{T})$, and simulated annealing takes a long time to converge. To conquer the computational issues associated with the algorithms in Section 3, we will examine an approximate equilibrium in this section. We focus on the Bayesian classifier, since it has an explicit clean expression for the classification regions. Results are developed under the following assumptions, for two attributes or more in the model.

1. Assume attributes $x_1, x_2, ..., x_q$ are independent.
2. Assume the profit function $g(\mathbf{T}, \mathbf{x}) = \sum_{j=1}^{q} g_j(T_j, x_j)$, where $\forall j, g_j(T_j, x_j) \geq 0$. (i.e., we assume each attribute is individually penalized.)

The region where objects will be classified into the "good" class π_1 is:

$$L(h_{\mathbf{T}}, 1) = \{\mathbf{x}: \ (c(1,2) - c(2,2)) \times p_2 \times f_2^{\mathbf{T}}(\mathbf{x}) \leq (c(2,1) - c(1,1)) \times p_1 \times f_1(\mathbf{x})\}.$$

Under the independence assumption the density of the "good" class π_1 is: $f_1(\mathbf{x}) = \prod_{j=1}^{q} f_{1j}(x_j)$; the density of the "bad" class π_2 is: $f_2^{\mathbf{T}}(\mathbf{x}) = \prod_{j=1}^{q} f_{2j}^{T_j}(x_j)$. Using the independence assumption, $L(h_{\mathbf{T}}, 1)$ can be written as:

$$L(h_{\mathbf{T}}, 1) = \{\mathbf{x}: \prod_{j=1}^{q} \left(\left(\frac{(c(1,2) - c(2,2)) \times p_2}{(c(2,1) - c(1,1)) \times p_1} \right)^{\frac{1}{q}} \times \frac{f_{2j}^{T_j}(x_j)}{f_{1j}(x_j)} \right) \leq 1 \ \} \tag{5.1}$$

Notice that this is also the classification region of the "Naive" Bayes classifier. Let

$$y_j = \left(\frac{(c(1,2) - c(2,2)) \times p_2}{(c(2,1) - c(1,1)) \times p_1} \right)^{\frac{1}{q}} \times \frac{f_{2j}^{T_j}(x_j)}{f_{1j}(x_j)}. \tag{5.2}$$

From now on, we will only consider the case where $c(1,2) - c(2,2) > 0$ and $c(2,1) - c(1,1) > 0$. Then $\forall j, \ y_j > 0$, we have $L(h_{\mathbf{T}}, 1) = \{\mathbf{x} : \prod_1^q y_j \ \leq \ 1\}$. Then the region where objects will be classified into the "bad" class π_2 can be written as $L(h_{\mathbf{T}}, 2) = \{\mathbf{x} : \prod_1^q y_j > 1\}$. Notice $L(h_{\mathbf{T}}, 1) + L(h_{\mathbf{T}}, 2) = \mathcal{R}^q$, where $\mathcal{R}^q$ is the q-dimensional Euclidean space, the whole sample space.

Later our simulation results indicate that even if the independence assumption is not satisfied, the approximate solution will point to the neighborhood of the true equilibrium. We also meet the second assumption by approximating the exact profit function.

5.1 Lower and Upper Bounds of the Classification Regions

In high dimensional space it will take a huge Monte Carlo sample to estimate the adversary gain $W(\mathbf{T})$ and the Bayes error $e(\mathbf{T})$. Here we aim at building lower and upper bounds for $W(\mathbf{T})$ and $e(\mathbf{T})$, which will only involve one-dimensional integrals and the marginal densities of the attributes. The Monte Carlo sample size required in the estimation process will grow linearly as the number of attributes grow. Maximizing the lower bound of $W(\mathbf{T})$ will direct us to the neighboring region of the equilibrium, and minimizing the upper bound of $e(\mathbf{T})$ will help us to select effective attributes. The idea behind the lower and upper bounds in this section is similar to how Riemann integral is defined: Approximate the region. We will not attempt to arrive at the limit of lower/upper bounds. Computing the exact values of $W(\mathbf{T})$ and $e(\mathbf{T})$ will involve too many one-dimensional integrals to be practical.

Equations 5.3 and 5.4 give straightforward lower and upper bounds of $L(h_{\mathbf{T}}, 1)$ and $L(h_{\mathbf{T}}, 2)$:

$$\prod_{j=1}^{q}[y_j \leq 1] \subset L(h_{\mathbf{T}}, 1) \subset \mathcal{R}^q - \prod_{j=1}^{q}[y_j > 1] \tag{5.3}$$

$$\prod_{j=1}^{q}[y_j > 1] \subset L(h_{\mathbf{T}}, 2) \subset \mathcal{R}^q - \prod_{j=1}^{q}[y_j \leq 1] \tag{5.4}$$

When we gradually add hyper-rectangles below the surface of $\{\mathbf{x} : \prod_{j=1}^{q} y_j = 1\}$, and subtract hyper-rectangles above the surface from $\mathcal{R}^q$, we will obtain tighter bounds of the classification regions, and improve the lower and upper bounds of the adversary gain, the Bayes error, and the expected misclassification cost. The formulas are given in Appendix A.

Remark 5.1. *When there are several highly correlated attributes, one of them can be selected as a representative and the rest could be excluded from the model. Otherwise we can break the entire set of q attributes into several groups. The attributes within the same group are correlated, while the groups themselves are not. Using such grouping, we can reduce the high dimensional search into lower dimensions, and the heuristic solution proposed next naturally follows.*

5.2 A Heuristic Solution for the Equilibrium Transformation

Following well established classification theory, the misclassification rate for each class is defined as: $e_1(\mathbf{T}) = \int_{L(h_{\mathbf{T}},2)} f_1(\mathbf{x})d\mathbf{x}$ and $e_2(\mathbf{T}) = \int_{L(h_{\mathbf{T}},1)} f_2^{\mathbf{T}}(\mathbf{x})d\mathbf{x}$. The Bayes error is $e(\mathbf{T}) = p_1 \times e_1(\mathbf{T}) + p_2 \times e_2(\mathbf{T})$. The expected misclassification cost is:

$$C(\mathbf{T}, h_{\mathbf{T}}) = c(1,1)p_1(1 - e_1(\mathbf{T})) + c(2,2)p_2(1 - e_2(\mathbf{T})) + c(2,1)p_1e_1(\mathbf{T}) + c(1,2)p_2e_2(\mathbf{T}).$$

The adversary gain is defined as $W(\mathbf{T}) = \int_{L(h_{\mathbf{T}},1)} g(\mathbf{T}, \mathbf{x}) f_2^{\mathbf{T}}(\mathbf{x})\, d\mathbf{x}$. Based on the lower and upper bounds of the classification regions, we can obtain lower and upper bounds of all the values given above. First we define the following three functions:

$$G_j(a,b) = \int I_{[a<y_j\leq b]}(x_j) \times g_j(T_j, x_j) \times f_{2j}^{T_j}(x_j)\, dx_j, \tag{5.5}$$

$$P_j(a,b) = \int I_{[a<y_j\leq b]}(x_j) \times f_{2j}^{T_j}(x_j)\, dx_j, \tag{5.6}$$

$$Q_j(a,b) = \int I_{[a<y_j\leq b]}(x_j) \times f_{1j}(x_j)\, dx_j, \tag{5.7}$$

where a and b are non-negative real numbers that mark the range of y_j in the indicator function $I_{[a<y_j\leq b]}(x_j)$.

We could have quite tight lower and upper bounds of $e(\mathbf{T})$, $W(\mathbf{T})$, and $C(\mathbf{T}, h_{\mathbf{T}})$ based on Equations .1 and .2. Please refer to Appendix A for the exact formulas. Here we point out that a simple lower bound of the adversary gain $W(\mathbf{T})$ will lead to a heuristic solution for the equilibrium transformation $\mathbf{T}^e$.

Based on Equation 5.3 we have a lower bound of the adversary gain:

$$W^l(\mathbf{T}) = \left(\prod_{j=1}^{q} P_j(0,1)\right) \times \left(\sum_{j=1}^{q} \frac{G_j(0,1)}{P_j(0,1)}\right). \tag{5.8}$$

Although $W^l(\mathbf{T})$ is much smaller than the actual adversary gain $W(\mathbf{T})$, we observed that if a transformation $\mathbf{T}$ generates a large $W(\mathbf{T})$, it will also generate a large $W^l(\mathbf{T})$. The following two-dimensional example compared $W^l(\mathbf{T})$ and $W(\mathbf{T})$, as shown in Figure 5.1. The two functions will reach large values in the same region of the action space. The parameter values for this two-dimensional example are given in Table 5.1. The profit function used is the following:

$$\begin{aligned} g_1(T_1, x_1) &= \max(0.5 - a_1 \times |T_1^{-1}(x_1) - x_1|, 0), \\ g_2(T_2, x_2) &= \max(0.5 - a_2 \times |T_2^{-1}(x_2) - x_2|, 0), \\ g(\mathbf{T}, \mathbf{x}) &= g_1(T_1, x_1) + g_2(T_2, x_2). \end{aligned} \tag{5.9}$$

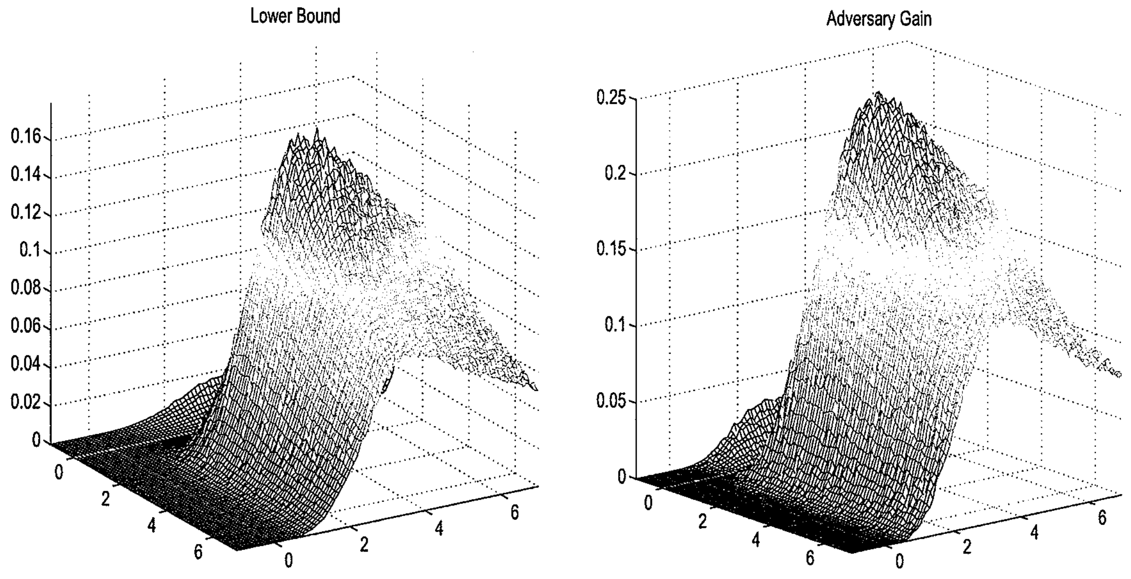

Figure 5.1: Left: a lower bound of the adversary gain $W^l(\mathbf{T})$; Right: the actual adversary gain $W(\mathbf{T})$

A transformation that maximizes the lower bound $W^l(\mathbf{T})$ falls into the neighborhood of the equilibrium transformation $\mathbf{T}^e$. When there are no penalties ($\vec{a} = \vec{0}$), maximizing $W^l(\mathbf{T})$ as specified by Equation 5.8 is equivalent to maximizing the one-dimensional integrals $G_j(0,1)$s or $P_j(0,1)$s. Both will return the same transformation. When there is penalty for transformation, our simulation results suggest maximizing $G_j(0,1)$s will return good results, though it will not be the same transformation that maximizes the lower bound. Let

$$T_j^a = \text{argmax } G_j(0,1) = \text{argmax} \int I_{[0<y_j\leq 1]}(x_j) \times g_j(T_j, x_j) \times f_{2j}^{T_j}(x_j)\, dx_j \tag{5.10}$$

Table 5.1:

a_1	a_2	Cost Matrix c	π_1	p_1	π_2	p_2
0.05	0.1	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	$N\left(\begin{bmatrix} 3.6 \\ 2.5 \end{bmatrix}, \begin{bmatrix} 0.9^2 & 0 \\ 0 & 1.2^2 \end{bmatrix}\right)$	0.5	$N\left(\begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix}, \begin{bmatrix} 0.5^2 & 0 \\ 0 & 0.9^2 \end{bmatrix}\right)$	0.5

Table 5.2:

π_1	π_2
$N\left(\begin{bmatrix} 3.6 \\ 2.5 \end{bmatrix}, \begin{bmatrix} 0.9^2 & 0.6 \times 0.9 \times 1.2 \\ 0.6 \times 0.9 \times 1.2 & 1.2^2 \end{bmatrix}\right)$	$N\left(\begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix}, \begin{bmatrix} 0.5^2 & 0.7 \times 0.5 \times 0.9 \\ 0.7 \times 0.5 \times 0.9 & 0.9^2 \end{bmatrix}\right)$

$\mathbf{T}^a = (T_1^a, T_2^a, ..., T_q^a)$ is an *approximate equilibrium transformation.* Here $G_j(0,1)$ can be considered as the *one dimensional gain* generated by each attribute.

To see whether equation 5.10 works well even if the attributes are moderately correlated, we have done two experiments. The parameter values for Experiment 1 are given in Table 5.1. Parameter values are the same in Experiment 2, except that the attributes are correlated as given in Table 5.2: $\rho_1 = 0.6$ and $\rho_2 = 0.7$.

Transformation $\mathbf{T}$ is a diagonal matrix, diag(T_1, T_2). A transformed object is: $\mathbf{T}(\mathbf{x}) = \mathbf{T}\mathbf{x}$. In both experiments, after transformation $\mathbf{T}$, π_2 will have distribution $N(\mathbf{T}\mu_2, \mathbf{T}\Sigma_2\mathbf{T}')$. Table 5.3 contains the heuristic transformations found by maximizing $G_j(0,1)$, the transformations that maximized the lower bound $W^l(\mathbf{T})$, and the equilibrium transformations. Table 5.3 also has the corresponding adversary gain and Bayes error. The lower bound and the actual adversary gain for Experiment 1 with independent attributes are plotted in Figure 5.1.

The heuristic transformation $\mathbf{T}^a = (3.6, 2.3)$, stays the same for both experiments. When the two attributes are independent, the equilibrium transformation becomes $\mathbf{T}^e = (3.7, 2.1)$. When they are correlated, the equilibrium transformation becomes $\mathbf{T}^e = (3.3, 2.2)$. The second line of Table 5.3 showed slightly different numbers, because while computing the one dimensional functions in the lower bound $W^l(\mathbf{T})$, the simulated sample points are from a distribution with correlated variables versus a distribution with independent ones.

When attributes are correlated, the heuristic solution proposed by Equation 5.10 is a little further away from the true equilibrium transformation $\mathbf{T}^e$. But in both experiments the heuristic solution is in the surrounding area of $\mathbf{T}^e$, and the Bayes errors of the heuristic transformations are fairly close to the equilibrium Bayes errors. The results are even better than directly maximizing $W^l(\mathbf{T})$, because $W^l(\mathbf{T})$ is much smaller than the actual adversary gain. We consistently observed this phenomenon in many other experiments. The distance between the heuristic transformation and the true equilibrium transformation depends on how big the penalty is and how much the attributes are correlated.

The heuristic solution obtained by maximizing one dimensional gain is computationally less expensive compared to the stochastic search algorithms required to find the true equilibrium. It is a well known fact that stochastic search algorithms require lots of steps to converge. Simulated annealing will sample a huge number of transformations before it drops into the neighborhood of the equilibrium transformation. The combination of Monte Carlo integration and simulated annealing will require extensive computational

Table 5.3:

	Independent			*Correlated*		
	Transformation	$W(\mathbf{T})$	$e(\mathbf{T})$	Transformation	$W(\mathbf{T})$	$e(\mathbf{T})$
(argmax $G_j(0,1)$)	$\mathbf{T}^a = (3.6, 2.3)$	0.2465	0.1234	$\mathbf{T}^a = (3.6, 2.3)$	0.2717	0.1360
argmax $W^l(\mathbf{T})$	$\mathbf{T}^a = (4.0, 2.3)$	0.2473	0.1237	$\mathbf{T}^a = (4.0, 2.5)$	0.2546	0.1273
argmax $W(\mathbf{T})$	$\mathbf{T}^e = (3.7, 2.1)$	0.2519	0.1260	$\mathbf{T}^e = (3.3, 2.2)$	0.2877	0.1442

resources as the dimensionality grows. Therefore it is a feasible solution to apply the heuristic method to high dimensional data.

In several two dimensional experiments, we fixed parameter values, and compared the profit function specified by Equation 5.9 with the following one:

$$g(\mathbf{T}, \mathbf{x}) = max(1 - a_1 \times |T_1^{-1}(x_1) - x_1| - a_2 \times |T_2^{-1}(x_2) - x_2|, 0). \quad (5.11)$$

We observed similar equilibrium transformations. The details are omitted here. We believe that the profit function defined by Equation 5.11 is more intuitive: An object can produce up to one unit profit; a transformation being applied onto each attribute will receive linear penalty; and penalties together will be deducted from the maximum profit available. For a single transformation $\mathbf{T}$ and one single "bad" object $\mathbf{x}$, Equation 5.9 and Equation 5.11 might yield quite different results. Equation 5.11 gives a number smaller than Equation 5.9 for large transformations. On the other hand for moderate transformations, a transformed object will produce same amount of profit based on both profit functions. Extreme transformations, where the two functions differ, will not produce much profit in either case. Hence extreme transformations will not likely be the equilibrium. Since we are interested in the classifier's equilibrium behavior, using Equation 5.9 and Equation 5.11 will create similar equilibrium states.

Under our framework Equation 5.9 can be considered as a one-dimensional approximation to the intuitive profit function defined by Equation 5.11. With profit function defined by Equation 5.9 we don't have to employ a full scale stochastic search to discover the equilibrium point in the game.

Remark 5.2. *When there are large penalties and attributes are correlated, we can improve the heuristic solution by using it as the starting point in a stochastic search. In high dimensional space, when we are not able to have enough sample to have an accurate estimate of $W(\mathbf{T})$ by Monte Carlo integration, we can search for a transformation that maximizes an improved lower bound of $W(\mathbf{T})$ as given in Appendix A. A bigger lower bound of $W(\mathbf{T})$ is close to the true function $W(\mathbf{T})$. Hence we will better approximate $\mathbf{T}^e$. At the same time, since it only involves one-dimensional integrals, we will not encounter the sample size problem.*

6 Attribute Selection for Bayesian Classifier

Given the above computationally tractable approach for a Bayesian classifier, we can now look at equilibria for multiple attribute scenarios. In Section 4 we showed a classifier's initial error rate can be much smaller than its equilibrium error rate. Using a simple example, we can show that a classifier's equilibrium performance is not solely determined by the size of the penalty, although the penalty is one important factor. For the sake of example, let the cost matrix $c = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ and $p_1 = p_2 = 0.5$. Among the three attributes as in Table 6.1, we will select the one most effective at the equilibrium. With only one attribute in the model, we define the profit function as in Equation 4.1. Let $k = 1$. The equilibrium transformation can be obtained from Equation 4.2. Table 6.1 shows the Bayesian classifier's equilibrium performance for each of the three attributes in the model.

Without transformation and only based on the initial distributions, a Bayesian classifier using the third attribute X_3 is the most successful. Focusing on penalties, X_1 receives the heaviest penalty. However the second attribute X_2 is the most effective at the equilibrium. This example indicates that an attribute's long term success is not entirely determined by its initial success, or the size of the penalty. All factors interacted with each other and produced X_2 as the winner.

Table 6.1: Three Attributes' Equilibrium Performance

Attribute	π_1	π_2	Penalty	Equilibrium Bayes Error
X_1	$N(1,1)$	$N(3,1)$	$a = 1$	0.16
X_2	$N(1,1)$	$N(3.5,1)$	$a = 0.45$	0.13
X_3	$N(1,1)$	$N(4,1)$	$a = 0$	0.23

With q attributes, there are $2^q - 1$ subsets. We will use two information criteria to evaluate the long-term effectiveness of every subset:

1. The equilibrium Bayes error $e(\mathbf{T}^e)$.

2. The equilibrium expected misclassification cost $C(\mathbf{T}^e, h_{\mathbf{T}^e})$.

Note that when the cost matrix $c = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, the two criteria are equal.

With very powerful computing equipment, one can employ a forward or backward attribute selection algorithm, choosing the next subset with the smallest equilibrium Bayes error or expected misclassification cost. However to search for the equilibrium in every step is very time-consuming. We will examine their performance at the heuristic transformations instead.

In the next simulation we have 5 attributes in the model, and examine all the 31 subsets of these five attributes. We assume the five attributes are independent for both classes. Let $c = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, $p_1 = 0.7$, and $p_2 = 0.3$. When there are d attributes in the model $(1 \leq d \leq q)$, the profit function is given as:

$$g_j(T_j, x_j) = \max(1/d - a_j \times |T_j^{-1}(x_j) - x_j|, 0),$$

$$g(\mathbf{T}, \mathbf{x}) = \sum_{j=1}^{d} g_j(T_j, x_j).$$

Hence for all the different subsets of attributes, a "bad" object will produce one unit profit without transformation when it passes the classifier. The profit function can be considered as one dimensional approximation to the following function:

$$g^*(\mathbf{T},\mathbf{x}) = max(1 - \sum_{j=1}^{d} a_j \times |T_j^{-1}(x_j) - x_j|, 0).$$

The distributions for the "good" and "bad" classes and the penalties for every attribute are given in Table 6.2. As in Section 5.2, transformation $\mathbf{T}$ is a diagonal matrix, diag$(T_1, ..., T_d)$; a transformed object is: $\mathbf{T}(\mathbf{x}) = \mathbf{T}\mathbf{x}$; the transformed "bad" class has distribution $N(\mathbf{T}\mu_2, \mathbf{T}\Sigma_2\mathbf{T}')$. In Table 6.2, we chose the parameters such that the simulations could demonstrate how the heuristic performs in very different scenarios. For example, the parameters are set so that some distributions have large variances while the rest have small variances. Similarly, for some attributes the initial distributions are well separated, while they overlap a lot for some other attributes. In addition, we gradually increase the penalty too.

Table 6.2:

	π_1	π_2	a_j		π_1	π_2	a_j
X_1	$N(1.8, 0.6^2)$	$N(0.5, 1.6^2)$	0.02	X_2	$N(3.2, 1.1^2)$	$N(1, 0.8^2)$	0.04
X_3	$N(3.8, 1.8^2)$	$N(1.5, 2^2)$	0.06	X_4	$N(6, 2.4^2)$	$N(2.5, 1.2^2)$	0.08
X_5	$N(5.5, 0.8^2)$	$N(3.5, 0.4^2)$	0.10				

We computed the heuristic transformations for every subset by maximizing the one dimensional gains, as defined by Equation 5.10. We calculated the real Bayes error of the heuristic transformation $e(\mathbf{T}^a)$ and one upper bound of the Bayes error $e^u(\mathbf{T}^a)$. Here we included only several terms from the formula of the Bayes error upper bound given in Appendix A:

$$e_1^u(\mathbf{T}) = 1 - \prod_{j=1}^{q} Q_j(0,1) - \sum_{j=1}^{q} \left[\sum_{n=1}^{10} \left(Q_j(n, n+1) \times \prod_{k \neq j} Q_k(0, (\frac{1}{n+1})^{\frac{1}{q-1}}) \right) \right]$$

$$e_2^u(\mathbf{T}) = 1 - \prod_{j=1}^{q} P_j(1,\infty) - \sum_{j=1}^{q} \left[\sum_{n=1}^{10} \left(P_j(\frac{1}{n+1}, \frac{1}{n}) \times \prod_{k \neq j} P_k((n+1)^{\frac{1}{q-1}}, +\infty) \right) \right]$$

Then we have $e^u(\mathbf{T}) = p_1 \times e_1^u(\mathbf{T}) + p_2 \times e_2^u(\mathbf{T})$.

Table 6.3 shows the simulation results. It contains the heuristic transformation $\mathbf{T}^a$, the initial adversary gain $W(\mathbf{I})$, the initial Bayes error $e(\mathbf{I})$, the heuristic adversary gain $W(\mathbf{T}^a)$, the heuristic Bayes error $e(\mathbf{T}^a)$, and the upper bound of the heuristic Bayes error $e^u(\mathbf{T}^a)$, for each subset. For one dimensional subset, since the heuristic is the true equilibrium, an upper bound value is not applicable. The initial and the heuristic adversary gains, and the initial and the heuristic Bayes errors are plotted in Figure 6.1.

Figure 6.1 displays the difference between the initial values and the heuristic values. Experiment results indicate that choosing attributes based on the initial Bayes error may not be a good choice. For example, among all the subsets with two attributes, although (X_2, X_5) has the smallest initial Bayes error, it has one of the worst Bayes error at equilibrium (i.e. high $e(\mathbf{T}^a)$ value). Similar phenomenon could be observed for subsets with four and five attributes.

Table 6.3:

Index	Attributes	Heuristic $\mathbf{T}^a$	$W(\mathbf{I})$	$e(\mathbf{I})$	$W(\mathbf{T}^a)$	$e(\mathbf{T}^a)$	$e^u(\mathbf{T}^a)$
1	1	1.02	0.4268	0.1503	0.4301	0.1516	NA
2	2	2.6	0.1907	0.1177	0.6346	0.2368	NA
3	3	1.4	0.5381	0.2187	0.6335	0.2359	NA
4	4	2	0.1953	0.1649	0.7656	0.2986	NA
5	5	1.5	0.0538	0.0482	0.8249	0.3000	NA
6	$1,2$	1.2, 2.5	0.1409	0.0767	0.3138	0.1245	0.2637
7	$1,3$	1, 1.5	0.3088	0.1212	0.3182	0.1255	0.2604
8	$1,4$	1, 2.1	0.2302	0.1080	0.3263	0.1496	0.3124
9	$1,5$	1, 1.5	0.0460	0.0358	0.3390	0.1455	0.3291
10	$2,3$	2.4, 1.4	0.1517	0.0916	0.4529	0.1949	0.3099
11	$2,4$	2.5, 2.2	0.0818	0.0595	0.4622	0.2369	0.3579
12	$2,5$	2.5, 1.5	0.0265	0.0202	0.4973	0.2343	0.3462
13	$3,4$	1.4, 2.2	0.1964	0.1215	0.4590	0.2336	0.3626
14	$3,5$	1.5, 1.5	0.0491	0.0393	0.4945	0.2326	0.3482
15	$4,5$	2.2, 1.5	0.0259	0.0234	0.5468	0.2974	0.3234
16	$1,2,3$	1.2, 2.3, 1.4	0.1109	0.0589	0.2429	0.1067	0.3275
17	$1,2,4$	1, 2.3, 2.2	0.0623	0.0407	0.2386	0.1266	0.3742
18	$1,2,5$	1, 2.4, 1.6	0.0193	0.0150	0.2441	0.1268	0.3913
19	$1,3,4$	1.1, 1.4, 2.2	0.1505	0.0765	0.2380	0.1245	0.3753
20	$1,3,5$	1, 1.3, 1.6	0.0399	0.0279	0.2480	0.1246	0.3989
21	$1,4,5$	1, 2.1, 1.6	0.0215	0.0175	0.2363	0.1482	0.3999
22	$2,3,4$	2.5, 1.4, 2.2	0.0673	0.0464	0.3218	0.1921	0.3983
23	$2,3,5$	2.4, 1.4, 1.6	0.0218	0.0166	0.3383	0.1933	0.4175
24	$2,4,5$	2.5, 2.2, 1.6	0.0124	0.0103	0.3211	0.2345	0.4044
25	$3,4,5$	1.4, 2.2, 1.6	0.0235	0.0191	0.3219	0.2307	0.4087
26	$1,2,3,4$	1, 2.4, 1.2, 1.3	0.0521	0.0329	0.1654	0.0879	0.4141
27	$1,2,3,5$	1.2, 2.2, 1.3, 1.6	0.0170	0.0115	0.1787	0.1041	0.4678
28	$1,2,4,5$	1, 2.2, 1.2, 1.6	0.0091	0.0074	0.1570	0.0977	0.5086
29	$1,3,4,5$	1, 1.2, 1.2, 1.6	0.0187	0.0138	0.1549	0.0953	0.5196
30	$2,3,4,5$	2.5, 1.2, 1.3, 1.6	0.0105	0.0085	0.1925	0.1459	0.5624
31	$1,2,3,4,5$	1, 2.3, 1.3, 1.2, 1.3	0.0082	0.0058	0.0865	0.0608	0.4969

It is computationally infeasible to obtain the equilibrium Bayes error and the equilibrium expected misclassification cost, the best long-term effectiveness measures, for every subset of attributes. The heuristic values are easy to obtain and useful effectiveness measures.

As in the static environment, monitoring more attributes improved the classifier performance under the heuristic transformation in this simulation. X_1 is an interesting attribute. It receives the smallest

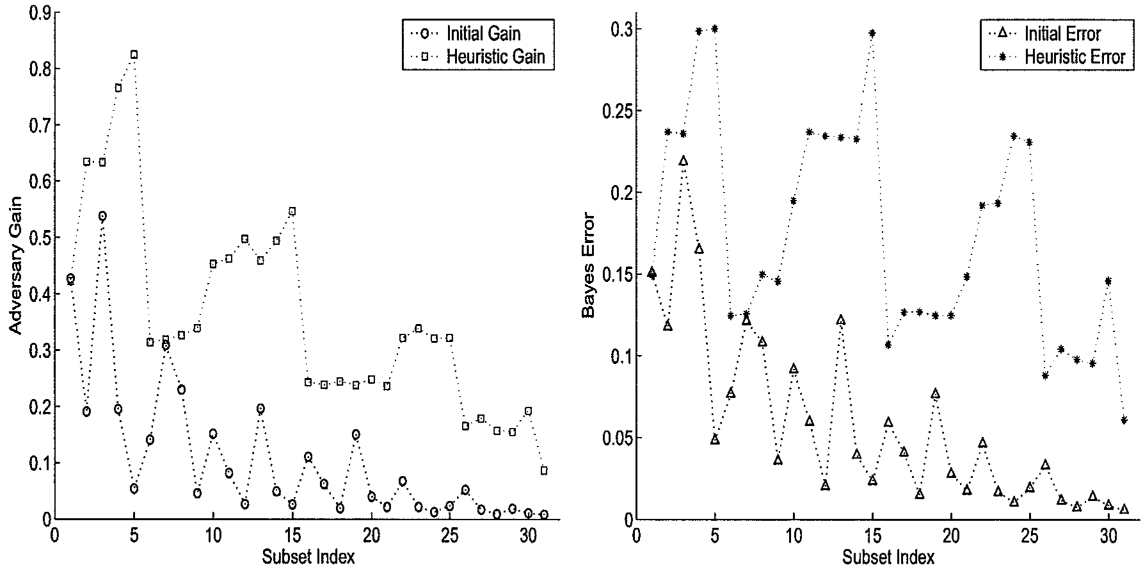

Figure 6.1: Left: the initial and heuristic adversary gain; Right: the initial and heuristic Bayes error. Numbers recorded in Table 6.3.

penalty. Yet transformation on X_1 didn't increase the adversary gain much, due to the large variance of the "bad" class. All the successful subsets contain X_1. Although X_5 receives the heaviest penalty 0.1, it alone is not more effective than the other four combined, $e(T_5^a) = 0.3$. Simply adding more attributes won't necessarily change the equilibrium performance much. For example (X_1, X_4) and (X_1, X_4, X_5) have very similar heuristic Bayes error. We notice using three attributes (X_1, X_2, X_3) is an effective and economical choice.

In higher dimensional space, when the Monte Carlo sample size is too big to be practical for accurately estimating the Bayes error, we can select attributes based on the upper bound of the Bayes error. With only a few terms in the upper bound, we already observed promising results. Among all the subsets of two attributes, (X_1, X_2) and (X_1, X_3) have the smallest Bayes upper bound values; among all the subsets of three attributes, (X_1, X_2, X_3) has the smallest Bayes upper bound value; among all the subsets of four attributes, (X_1, X_2, X_3, X_4) has the smallest Bayes upper bound value. Choosing subsets with small Bayes error upper bounds deliver the same results as minimizing the actual Bayes error, when the subsets contain same number of attributes. Assuming the data miner could only use a fixed number of attributes, the upper bound of the Bayes error is another useful long-term effectiveness measure, and more computationally efficient.

This simulation study confirms that a classifier's initial success has little impact on its long term performance. A set of attributes' long term success or failure depends on many factors: their initial distributions, penalties for transformation, the shape of the profit function, etc. The long-term effectiveness measures defined above consider all the factors under a proper model and recommend the best sets of attributes. If none of the attributes or the combinations of attributes can deliver satisfactory long term performance, the data miner must aggressively change the rules of the game (e.g., identify entirely new ways of authenticating non-spam messages) in order to win.

7 Conclusion

Many classification problems operate in a setting with active adversaries: while one party tries to identify the members of a particular class, the other tries to reduce the effectiveness of the classifier. Although this may seem like a never-ending cycle, it is possible to reach a steady-state where the actions of both parties stabilize. The game has an equilibrium because both parties facing costs: costs associated with misclassification on the one hand, and for defeating the classifier on the other. By incorporating such costs in modeling, we can determine where such an equilibrium could be reached, and whether it is acceptable to the data miner.

Although we proposed simulated annealing to find subgame perfect equilibrium in this paper, other stochastic search techniques could also be used. We considered the stochastic hill climbing technique as an alternative. With stochastic hill climbing, a new point is chosen only if it improves the current result. It can quickly find a good local optimal solution. The heuristic solution can be used as the initial start point in stochastic hill climbing. It will return an improved transformation much faster than simulated annealing.

In this article we had done experiments on a combination of Bayesian classifier, Gaussian mixture distributions, and linear loss of profit for transformations. Our formulation of the problem can accommodate a wide range of distributions, classifiers and profit functions. We emphasize that our formulation applies to many real life scenarios, such as intrusion detection and profiling for homeland security. Wherever there are two parties involved, and one party tries to avoid being detected by modifying its current strategy to mimic the other party, our formulation can be applied to the scenario.

In the past people have been making constant adjustments to their classifiers, while watching the adversary's actions for years. Content based spam filtering is one example. A well written spam email will reach the end users. We all have observed the evolution of spam emails. Lately by checking the IP address of each email and blocking port 25 from home computers, two attributes expensive to modify, people successfully reduced the overall proportion of spam emails in 2005 [15], although the majority of emails are still spam. The tools developed in this paper will help developers to arrive at such a decision without spending a long time on making small and constant adjustment to an eventually ineffective strategy. With our model we only need to observe the adversary for a short time period. Then we will think like the adversary and build an action space, including the observed adversary's actions and the actions they might take in the future. We plan to build a test-bed to predict the long term success or failure of a strategy. Such a test-bed targeting for a certain application will further improve our framework.

In summary, game theory is a valuable tool for understanding the adversarial environments. It gives us an idea about how effective we can expect a classifier to be in the long term. It also provides important insight that enables us to build better classifiers/filters.

Appendix A: Lower and Upper Bounds of the Bayes Error, Adversary Gain and Expected Misclassification Cost

We first present tighter bounds of the classification regions. Write

$$\mathrm{RB}_0 = \prod_{j=1}^{q} \, [y_j \le 1].$$

RB_0 is a subset of $L(h_{\mathrm{T}}, 1) = \{\prod_{j=1}^{q} y_j \le 1\}$ and a hyper-rectangle with all y_js not greater than 1. Write

$$\mathrm{RB}_1 = \bigcup_{j=1}^{q} \left\{ \bigcup_{n=1}^{\infty} \left\{ [n < y_j \le n+1] \times \prod_{k \ne j} \left[y_k \le \left(\frac{1}{n+1}\right)^{\frac{1}{q-1}} \right] \right\}\right\}.$$

Again RB_1 is a subset of $L(h_{\mathrm{T}}, 1)$. RB_1 is also the union of hyper-rectangles, where each one has exactly one y_j great than 1. $\mathrm{RB}_0 \cap \mathrm{RB}_1 = \emptyset$. Write

$$\mathrm{RB}_2 = \bigcup_{j_1=1}^{q} \bigcup_{j_2=j_1+1}^{q} \left\{ \bigcup_{n_1=1}^{\infty} \bigcup_{n_2=1}^{\infty} \left\{ \prod_{\ell=1}^{2} [n_\ell < y_{j_\ell} \le n_\ell + 1] \times \prod_{k \ne j_\ell, \ell=1,2} \left[y_k \le \left(\frac{1}{\prod_{\ell=1}^{2}(n_\ell+1)} \right)^{\frac{1}{q-2}} \right] \right\}\right\}.$$

RB_2 is the union of hyper-rectangles, where each one has exactly two y_js great than 1. We continue to add more hyper-rectangles below the surface of $\{\prod_{j=1}^{q} y_j = 1\}$ in this fashion up to RB_{q-1}. Write

$$\mathrm{RB}_{q-1} =$$
$$\bigcup_{j_1=1}^{q} \bigcup_{j_2=j_1+1}^{q} \cdots \bigcup_{j_{q-1}=j_{q-2}+1}^{q} \left\{ \bigcup_{n_1=1}^{\infty} \cdots \bigcup_{n_{q-1}=1}^{\infty} \left\{ \prod_{\ell=1}^{q-1} [n_\ell < y_{j_\ell} \le n_\ell + 1] \times \left[y_k \le \frac{1}{\prod_{\ell=1}^{q-1}(n_\ell+1)} \right]_{k \ne j_\ell} \right\}\right\}.$$

Notice $\forall i \ne j, \mathrm{RB}_i \cap \mathrm{RB}_j = \emptyset$. And $\bigcup_0^{q-1} \mathrm{RB}_j$ offers a larger lower bound than the one in Equation 5.3. Similarly we can construct an improved upper bound for $L(h_{\mathrm{T}}, 1)$. Write RA_j, $j = 0, ..., q-1$, as the following:

$$\mathrm{RA}_0 = \prod_{j=1}^{q} \, [y_j > 1],$$

$$\mathrm{RA}_1 = \bigcup_{j=1}^{q} \left\{ \bigcup_{n=1}^{\infty} \left\{ \left[\frac{1}{n+1} < y_j \le \frac{1}{n} \right] \times \prod_{k \ne j} \left[y_k > (n+1)^{\frac{1}{q-1}} \right] \right\}\right\},$$

...... up to

$$\mathrm{RA}_{q-1} =$$
$$\bigcup_{j_1=1}^{q} \bigcup_{j_2=j_1+1}^{q} \cdots \bigcup_{j_{q-1}=j_{q-2}+1}^{q} \left\{ \bigcup_{n_1=1}^{\infty} \cdots \bigcup_{n_{q-1}=1}^{\infty} \left\{ \prod_{\ell=1}^{q-1} \left[\frac{1}{n_\ell+1} < y_{j_\ell} \le \frac{1}{n_\ell} \right] \times \left[y_k > \prod_{\ell=1}^{q-1}(n_\ell+1) \right]_{k \ne j_\ell} \right\}\right\}.$$

$\forall i \neq j, \mathrm{RA}_i \cap \mathrm{RA}_j = \emptyset$. Every hyper-rectangle in RA_k has exactly k y_js smaller than 1. We now have improved lower and upper bounds for the classification regions:

$$\bigcup_{0}^{q-1} \mathrm{RB}_j \subset L(h_{\mathbf{T}}, 1) \subset \mathcal{R}^q - \bigcup_{0}^{q-1} \mathrm{RA}_j \quad (.1)$$

$$\bigcup_{0}^{q-1} \mathrm{RA}_j \subset L(h_{\mathbf{T}}, 2) \subset \mathcal{R}^q - \bigcup_{0}^{q-1} \mathrm{RB}_j \quad (.2)$$

Based on the lower and upper bounds of the classification regions in Equation .1 and .2, we can easily have the lower and upper bounds for the Bayes error, the adversary gain, and the expected misclassification cost. First we will show the bounds of $e_1(\mathbf{T})$ and $e_2(\mathbf{T})$.

A lower bound of the classification error rate of the "good" class π_1 is:

$$\begin{aligned} e_1^{\text{lower}}(\mathbf{T}) &= \prod_{j=1}^{q} Q_j(1,\infty) + \sum_{j=1}^{q}\left[\sum_{n=1}^{\infty}\left(Q_j(\frac{1}{n+1},\frac{1}{n}) \times \prod_{k\neq j} Q_k((n+1)^{\frac{1}{q-1}},\infty)\right)\right] \\ &+\sum_{j_1=1}^{q}\sum_{j_2=j_1+1}^{q}\left[\sum_{n_1=1}^{\infty}\sum_{n_2=1}^{\infty}\left(\prod_{\ell=1}^{2} Q_{j_\ell}(\frac{1}{n_\ell+1},\frac{1}{n_\ell}) \times \prod_{k\neq j_1,j_2} Q_k([(n_1+1)(n_2+1)]^{\frac{1}{q-2}},\infty)\right)\right] \\ &+ \;\ldots\ldots \\ &+\sum_{j_1=1}^{q}\sum_{j_2=j_1+1}^{q}\cdots\sum_{j_{q-1}=j_{q-2}+1}^{q}\left[\sum_{n_1=1}^{\infty}\cdots\sum_{n_{q-1}=1}^{\infty}\left(\prod_{\ell=1}^{q-1} Q_{j_\ell}(\frac{1}{n_\ell+1},\frac{1}{n_\ell}) \times Q_{k;k\neq j_\ell}(\prod_{1}^{q-1}(n_\ell+1),\infty)\right)\right] \end{aligned}$$

An upper bound of the classification error rate of the "good" class π_1 is:

$$\begin{aligned} e_1^{\text{upper}}(\mathbf{T}) &= 1 - \prod_{j=1}^{q} Q_j(0,1) - \sum_{j=1}^{q}\left[\sum_{n=1}^{\infty}\left(Q_j(n,n+1) \times \prod_{k\neq j} Q_k(0,(\frac{1}{n+1})^{\frac{1}{q-1}})\right)\right] \\ &-\sum_{j_1=1}^{q}\sum_{j_2=j_1+1}^{q}\left[\sum_{n_1=1}^{\infty}\sum_{n_2=1}^{\infty}\left(\prod_{\ell=1}^{2} Q_{j_\ell}(n_\ell,n_\ell+1) \times \prod_{k\neq j_1,j_2} Q_k(0,[\frac{1}{(n_1+1)(n_2+1)}]^{\frac{1}{q-2}})\right)\right] \\ &- \;\ldots\ldots \\ &-\sum_{j_1=1}^{q}\sum_{j_2=j_1+1}^{q}\cdots\sum_{j_{q-1}=j_{q-2}+1}^{q}\left[\sum_{n_1=1}^{\infty}\cdots\sum_{n_{q-1}=1}^{\infty}\left(\prod_{\ell=1}^{q-1} Q_{j_\ell}(n_\ell,n_\ell+1) \times Q_{k;k\neq j_\ell}(0,\frac{1}{\prod_{1}^{q-1}(n_\ell+1)})\right)\right] \end{aligned}$$

A lower bound of the classification error rate of the "bad" class π_2 is:

$$\begin{aligned} e_2^{\text{lower}}(\mathbf{T}) &= \prod_{j=1}^{q} P_j(0,1) + \sum_{j=1}^{q}\left[\sum_{n=1}^{\infty}\left(P_j(n,n+1) \times \prod_{k\neq j} P_k(0,(\frac{1}{n+1})^{\frac{1}{q-1}})\right)\right] \\ &+\sum_{j_1=1}^{q}\sum_{j_2=j_1+1}^{q}\left[\sum_{n_1=1}^{\infty}\sum_{n_2=1}^{\infty}\left(\prod_{\ell=1}^{2} P_{j_\ell}(n_\ell,n_\ell+1) \times \prod_{k\neq j_1,j_2} P_k(0,[\frac{1}{(n_1+1)(n_2+1)}]^{\frac{1}{q-2}})\right)\right] \\ &+ \;\ldots\ldots \\ &+\sum_{j_1=1}^{q}\sum_{j_2=j_1+1}^{q}\cdots\sum_{j_{q-1}=j_{q-2}+1}^{q}\left[\sum_{n_1=1}^{\infty}\cdots\sum_{n_{q-1}=1}^{\infty}\left(\prod_{\ell=1}^{q-1} P_{j_\ell}(n_\ell,n_\ell+1) \times P_{k;k\neq j_\ell}(0,\frac{1}{\prod_{1}^{q-1}(n_\ell+1)})\right)\right] \end{aligned}$$

A upper bound of the classification error rate of the "bad" class π_2 is:

$$
\begin{aligned}
e_2^{\text{upper}}(\mathbf{T}) \;=\; & 1 - \prod_{j=1}^{q} P_j(1,\infty) - \sum_{j=1}^{q}\left[\sum_{n=1}^{\infty}\left(P_j(\frac{1}{n+1},\frac{1}{n}) \times \prod_{k\neq j} P_k((n+1)^{\frac{1}{q-1}},+\infty)\right)\right] \\
& -\sum_{j_1=1}^{q}\sum_{j_2=j_1+1}^{q}\left[\sum_{n_1=1}^{\infty}\sum_{n_2=1}^{\infty}\left(\prod_{\ell=1}^{2} P_{j_\ell}(\frac{1}{n_\ell+1},\frac{1}{n_\ell}) \times \prod_{k\neq j_1,j_2} P_k([(n_1+1)(n_2+1)]^{\frac{1}{q-2}},\infty)\right)\right] \\
& - \;\;\ldots\ldots \\
& -\sum_{j_1=1}^{q}\sum_{j_2=j_1+1}^{q}\cdots\sum_{j_{q-1}=j_{q-2}+1}^{q}\left[\sum_{n_1=1}^{\infty}\cdots\sum_{n_{q-1}=1}^{\infty}\left(\prod_{\ell=1}^{q-1} P_{j_\ell}(\frac{1}{n_\ell+1},\frac{1}{n_\ell}) \times P_{k;k\neq j_\ell}(\prod_{1}^{q-1}(n_\ell+1),\infty)\right)\right]
\end{aligned}
$$

Then we have the bounds of the Bayes error:

$$
p_1 \times e_1^{\text{lower}}(\mathbf{T}) + p_2 \times e_2^{\text{lower}}(\mathbf{T}) \;<\; e(\mathbf{T}) \;<\; p_1 \times e_1^{\text{upper}}(\mathbf{T}) + p_2 \times e_2^{\text{upper}}(\mathbf{T}) \tag{.3}
$$

We also have the bounds of the expected misclassification cost, assuming $c(1,1) = c(2,2) = 0$:

$$
c(2,1)\times p_1\times e_1^{\text{lower}}(\mathbf{T})+c(1,2)\times p_2\times e_2^{\text{lower}}(\mathbf{T}) \;<\; C(\mathbf{T},h_{\mathbf{T}}) \;<\; c(2,1)\times p_1\times e_1^{\text{upper}}(\mathbf{T})+c(1,2)\times p_2\times e_2^{\text{upper}}(\mathbf{T}) \tag{.4}
$$

A lower bound of the adversary gain is:

$$
\begin{aligned}
W^{\text{lower}}(\mathbf{T}) \;=\; & \left(\prod_{j=1}^{q} P_j(0,1) \times \sum_{j=1}^{q}\frac{G_j(0,1)}{P_j(0,1)}\right) \\
& +\sum_{j=1}^{q}\left[\sum_{n=1}^{q}\left(\prod_{k\neq j} P_k(0,(\frac{1}{n+1})^{\frac{1}{q-1}}) \times P_j(n,n+1) \times (\frac{G_j(n,n+1)}{P_j(n,n+1)} + \sum_{k\neq j}\frac{G_k(0,(\frac{1}{n+1})^{\frac{1}{q-1}})}{P_k(0,(\frac{1}{n+1})^{\frac{1}{q-1}})})\right)\right] \\
& +\sum_{j_1=1}^{q}\sum_{j_2=j_1+1}^{q}\;\{\;\sum_{n_1=1}^{\infty}\sum_{n_2=1}^{\infty}\;[\;\prod_{k\neq j_1,j_2} P_k(0,(\frac{1}{(n_1+1)(n_2+1)})^{\frac{1}{q-2}}) \\
& \times\prod_{\ell=1}^{2} P_{j_\ell}(n_\ell,n_\ell+1) \times \left(\sum_{\ell=1}^{2}\frac{G_{j_\ell}(n_\ell,n_\ell+1)}{P_{j_\ell}(n_\ell,n_\ell+1)} + \sum_{k\neq j_1,j_2}\frac{G_k(0,(\frac{1}{(n_1+1)(n_2+1)})^{\frac{1}{q-2}})}{P_k(0,(\frac{1}{(n_1+1)(n_2+1)})^{\frac{1}{q-2}})}\right)\;]\;\} \\
& + \;\;\ldots\ldots \\
& +\sum_{j_1=1}^{q}\sum_{j_2=j_1+1}^{q}\cdots\sum_{j_{q-1}=j_{q-2}+1}^{q}\;\{\;\sum_{n_1=1}^{\infty}\cdots\sum_{n_{q-1}=1}^{\infty}\;[\;P_{k;k\neq j_\ell}(0,1/\prod_{1}^{q-1}(n_\ell+1)) \\
& \times\prod_{\ell=1}^{q-1} P_{j_\ell}(n_\ell,n_\ell+1) \times \left(\sum_{\ell=1}^{q-1}\frac{G_{j_\ell}(n_\ell,n_\ell+1)}{P_{j_\ell}(n_\ell,n_\ell+1)} + \frac{G_{k;k\neq j_\ell}(0,1/\prod_1^{q-1}(n_\ell+1))}{P_{k;k\neq j_\ell}(0,1/\prod_1^{q-1}(n_\ell+1))}\right)\;]\;\}
\end{aligned}
$$

Finally a upper bound of the adversary gain is:

$$
\begin{aligned}
W^{\text{upper}}(\mathbf{T}) = 1 - & \left(\prod_{j=1}^{q} P_j(1,\infty) \times \sum_{j=1}^{q} \frac{G_j(1,+\infty)}{P_j(1,\infty)} \right) \\
- & \sum_{j=1}^{q} \{ \sum_{n=1}^{q} [\prod_{k\neq j} P_k((n+1)^{\frac{1}{q-1}}, \infty) \times P_j(\frac{1}{n+1}, \frac{1}{n}) \times \left(\frac{G_j(\frac{1}{n+1}, \frac{1}{n})}{P_j(\frac{1}{n+1}, \frac{1}{n})} + \sum_{k\neq j} \frac{G_k((n+1)^{\frac{1}{q-1}}, +\infty)}{P_k((n+1)^{\frac{1}{q-1}}, +\infty)} \right)]\} \\
- & \sum_{j_1=1}^{q} \sum_{j_2=j_1+1}^{q} \{ \sum_{n_1=1}^{\infty} \sum_{n_2=1}^{\infty} [\prod_{k\neq j_1,j_2} P_k([(n_1+1)(n_2+1)]^{\frac{1}{q-2}}, \infty) \\
\times & \prod_{\ell=1}^{2} P_{j_\ell}(\frac{1}{n_\ell+1}, \frac{1}{n_\ell}) \times \left(\sum_{\ell=1}^{2} \frac{G_{j_\ell}(\frac{1}{n_\ell+1}, \frac{1}{n_\ell})}{P_{j_\ell}(\frac{1}{n_\ell+1}, \frac{1}{n_\ell})} + \sum_{k\neq j_1,j_2} \frac{G_k([(n_1+1)(n_2+1)]^{\frac{1}{q-2}}, \infty)}{P_k([(n_1+1)(n_2+1)]^{\frac{1}{q-2}}, \infty)} \right) \ ] \ \} \\
- & \quad \ldots\ldots \\
- & \sum_{j_1=1}^{q} \sum_{j_2=j_1+1}^{q} \cdots \sum_{j_{q-1}=j_{q-2}+1}^{q} \{ \sum_{n_1=1}^{\infty} \cdots \sum_{n_{q-1}=1}^{\infty} [\ P_{k;k\neq j_\ell}(\prod_{1}^{q-1}(n_\ell+1), +\infty) \\
\times & \prod_{\ell=1}^{q-1} P_{j_\ell}(\frac{1}{n_\ell+1}, \frac{1}{n_\ell}) \times \left(\sum_{\ell=1}^{q-1} \frac{G_{j_\ell}(\frac{1}{n_\ell+1}, \frac{1}{n_\ell})}{P_{j_\ell}(\frac{1}{n_\ell+1}, \frac{1}{n_\ell})} + \frac{G_{k;k\neq j_\ell}(\prod_{\ell=1}^{q-1}(n_\ell+1), \infty)}{P_{k;k\neq j_\ell}(\prod_{\ell=1}^{q-1}(n_\ell+1), \infty)} \right) \ ] \ \}
\end{aligned}
$$

References

[1] Tamer Basar and Geert Jan Olsder. *Dynamic Noncooperative Game Theory.* SIAM, 1999.

[2] Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games.* Cambridge University Press, 2006.

[3] Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108, New York, NY, USA, 2004. ACM Press.

[4] Richard Duda, Peter E. Hart, and David Stork. *Pattern Classification.* John Wiley & Sons, 2001.

[5] Tom Fawcett and Foster J. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1(3):291–316, 1997.

[6] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, San Diego, CA, 1990.

[7] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 97–106, San Francisco, CA, 2001. ACM Press.

[8] Dimitrios K. Vassilakis Ion Androutsopoulos, Evangelos F. Magirou. A game theoretic model of spam e-mailing. In *Proceedings of the 2nd Conference on Email and Anti-Spam (CEAS 2005)*, 2005.

[9] Richard P. Lippmann, David J. Fried, Isaac Graf, Joshua W. Haines, Kristopher R. Kendall, David McClung, Dan Weber, Seth E. Webster, Dan Wyschogrod, Robert K. Cunningham, and Marc A. Zissman. Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation. In *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition*, volume 2, 2000.

[10] Daniel Lowd and Christopher Meek. Adversarial learning. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 641–647, New York, NY, USA, 2005. ACM Press.

[11] Matthew V. Mahoney and Philip K. Chan. Learning nonstationary models of normal network traffic for detecting novel attacks. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 376–385, New York, NY, USA, 2002. ACM Press.

[12] Richard D. McKelvey, Andrew M. McLennan, and Theodore L Turocy. Gambit: Software tools for game theory, version 0.2007.01.30, 2007.

[13] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory.* MIT Press, 1999.

[14] Calton Pu and Steve Webb. Observed trends in spam construction techniques: A case study of spam evolution. In *Proceedings of the 3rd Conference on Email and Anti-Spam (CEAS 2006)*, 2006.

[15] Spam accounts for 68% of year's email, 2005.

[16] Thomas Vallee and Tamer Basar. Off-line computation of stackelberg solutions with the genetic algorithm. *Journal Computational Economics*, 13(3):201–209, 1999.