Data Min Knowl Disc (2011) 22:291–335
DOI 10.1007/s10618-010-0197-3

Classifier evaluation and attribute selection against active adversaries

Murat Kantarcıoğlu · Bowei Xi · Chris Clifton

Received: 16 June 2008 / Accepted: 17 July 2010 / Published online: 12 August 2010

Abstract Many data mining applications, such as spam filtering and intrusion detection, are faced with active adversaries. In all these applications, the future data sets and the training data set are no longer from the same population, due to the transformations employed by the adversaries. Hence a main assumption for the existing classification techniques no longer holds and initially successful classifiers 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 are not detected. We develop a game theoretic framework where equilibrium behavior of adversarial classification applications can be analyzed, and provide solutions for finding an 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.

Responsible editor: Johannes Fürnkranz.

M. Kantarcıoğlu (✉)
Computer Science Department, University of Texas at Dallas, Richardson, TX, USA
e-mail: muratk@utdallas.edu

B. Xi
Department of Statistics, Purdue University, West Lafayette, IN, USA
e-mail: xbw@stat.purdue.edu

C. Clifton
Department of Computer Science, Purdue University, West Lafayette, IN, USA
e-mail: clifton@cs.purdue.edu

A case study on online lending data demonstrates how to apply the proposed game theoretic framework to a real application.

Keywords Adversarial classification · Game theory · Attribute selection · Simulated annealing

1 Introduction

Many data mining applications, both current and proposed, are faced with active adversaries. 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 (Lippmann et al. 2000). These problems pose a significant new challenge not addressed in previous research: The behavior of a class controlled by the adversary may adapt to avoid detection. Traditionally a classifier is constructed from a training data set, and future data sets come from the same population as the training data set. A classifier constructed by the data miner in such a static environment cannot maintain its optimal performance for long, when faced with an active adversary.

One 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 Dalvi et al. (2004). The problem is that this becomes a never-ending game between the classifier and the adversary. Another approach is to minimize the worst case error through a zero-sum game (Lanckriet et al. 2003; El Ghaoui et al. 2003).

Our approach is **not** to develop a learning strategy for the classifier to stay ahead of the adversary. Instead, we propose an Adversarial Classification Stackelberg Game, a two-player game, to model the sequential moves of the adversary and the data miner. Each player follows their own interest in the proposed game theoretic framework: The adversary tries to maximize its return from the false negative items (those that get through the classifier), and the data miner tries to minimize the misclassification cost.

We then predict the end state of the game—an equilibrium state. When considering the whole strategy space of all the possible transformations and the penalties for transformation, an equilibrium state offers insight into the error rate to be expected from a classifier in the long run. Equilibrium information also offers an alternative to the minimax approach which could be too pessimistic in some cases.

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 equilibrium strategies on the subsets of attributes. Such information is used to select the most effective attributes to build a classifier. When none of the subset's equilibrium performance is satisfactory, the data miner will have to change the rules of the game. For example, consider new input attributes or increase the penalties for existing ones.

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 are effective. But besides blocking some good traffic, a trade-off 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 Sect. 7 we show experimentally that an attribute that is effective at the beginning may not be effective in the long-term.

Our framework is general enough to be applicable in a variety of adversarial classification scenarios and can accommodate different classification techniques. 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).[1]

Another example is botnet detection where several detection algorithms have been proposed in the literature, each monitoring different sets of attributes (Stinson and Mitchell 2008). The equilibrium performance of different defensive algorithms can help the data miner to determine their effectiveness. Furthermore, the data miner can apply the proposed framework to select the most effective attributes from each defensive approach. In return, these attributes can be combined to build a more robust defensive algorithm.

Finally, we apply the proposed framework to model online lending data in Sect. 9. In this case, the adversary's equilibrium transformation can help the lender to identify high risk applications and determine how often additional information needs to be verified to increase the penalty for the adversary.

The paper is organized as follows: In Sect. 2 we present a game theoretic model. In Sect. 3 we propose a stochastic search method to solve for an equilibrium. We demonstrate that penalty costs can affect the equilibrium Bayes error in interesting ways for Gaussian distribution in Sect. 4. We examine the impact of extreme classification rules (to pass all or to block all objects) on an equilibrium, using Gaussian and Bernoulli random variables respectively, in Sect. 5. In Sect. 6 we provide a computationally efficient heuristic solution for Bayesian classifier, which allows us to handle high dimensional data. Section 7 presents a simulation study, where we evaluate the equilibrium performance of multiple combinations of Gaussian attributes, demonstrating the effect of different combinations of distributions and penalties without transformation and in equilibrium. Section 8 presents another simulation study with Bernoulli random variables. Section 9 presents a case study of modeling online lending data. We conclude with a discussion of future work. First, we discuss related work below.

[1] Please refer to Pu and Webb (2006) for an extensive study of adversarial behavior evolution in spam filtering.

1.1 Related work

Learning in the presence of an adaptive adversary is an issue in many different applications. Problems ranging from intrusion detection (Mahoney and Chan 2002) to fraud detection (Fawcett and Provost 1997) need to be able to cope with adaptive malicious adversaries. As discussed in Dalvi et al. (2004), the challenges created by the malicious adversaries are quite different from those in concept drift (Hulten et al. 2001), because the concept is maliciously changed based on the actions of the classifier. There have been applications of game theory to spam filtering. In Androutsopoulos et al. (2005), 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 Lowd and Meek (2005a), the adversary tries to reverse engineer the classifier to learn the parameters. In Dalvi et al. (2004), the authors applied game theory to produce a Naïve Bayes classifier that could automatically adapt to the adversary's expected actions. While recognizing the importance of an equilibrium state, they simplified the situation by assuming the adversary bases its strategy on the initial Naïve Bayes classifier rather than their proposed adaptive strategy.

In Lowd and Meek (2005b), the authors studied the impact of attaching words that serve as strong indicator of regular user emails. The study shows that by adding 30-150 good words, spammer can significantly increase its success rate. In this paper, we propose a game theoretic model where such conclusions could be reached automatically. In addition, robust classification techniques that use minimax criterion have been proposed in El Ghaoui et al. (2003) and Lanckriet et al. (2003). Compared to those works, we assume that the adversary can modify the entire bad class distribution to avoid being detected. Also our model allows the data miner and the adversary to have different utility functions. There has been other work on how to construct robust classifiers for various tasks. For example, in Globerson and Roweis (2006), the authors constructed robust classifiers in domains such as document classification by not over-weighting any single attribute. Similarly, in Teo et al. (2008), the authors applied domain specific knowledge of invariance transformations to construct a robust supervised learning algorithm. Compared to that work, we incorporate the cost for the adversaries into our model as well as the cost for data miner.

In online learning such as Cesa-Bianchi and Lugosi (2006), 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 works do not deal with situations where an adversary changes the distribution of the underlying concept.

Overall, we take a very different approach from the existing work. By directly investigating an equilibrium state of the game, at which point all parties stick to their current strategies, we aim at providing a guideline for building 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. Before we discuss the model in details, we provide a motivating example that explains the basic aspects of our model.

2.1 Motivating example

Consider the case where a classifier is built to detect whether a computer is compromised or not by malware that sends spam e-mails. This malware detection system could be built based on various system wide parameters. For example, looking at the number of e-mails sent by a compromised computer could be one useful measure to detect such malware.

In such a scenario, a simple classifier that raises an alarm if the number of e-mails sent by a computer exceeds a threshold value could be initially very successful. Now an attacker that designs a malware can reduce the number of spam e-mails sent to avoid detection. The attacker can possibly reduce the spam e-mails sent per day to near zero and avoid being detected, but this will not be profitable for the attacker. Afterward, depending on the number of spam e-mails sent per day, the data miner set a threshold value. To keep the number of false positives at a reasonable level, the threshold value chosen to be the classification rule cannot be too small. In such a game the attacker and the data miner may reach an equilibrium point where both parties have no incentive to change their strategies. The attacker would set the spam e-mails sent per day to a number that is most profitable: Increasing the number causes the computer to be detected, and reducing the number is less profitable and not necessary. When the attacker receives maximum payoff and does not change its strategy, the data miner does not need to re-set the threshold value: If the threshold used by the classifier were further lowered to detect the spamming machine, too many legitimate machines would be misidentified as sources of spam. The important question is whether the classifier equilibrium performance is satisfactory for the data miner. If not, the data miner would need a new classification approach, such as including additional attributes (e.g. system call sequences executed by the programs) to build a classifier and improve its equilibrium performance.

Below, we discuss how an example given above could be modeled in our framework to understand the equilibrium performance for a given classifier and the set of attributes used for building such a classifier.

2.2 Adversarial classification stackelberg game

The adversarial classification scenario is formulated as a two class problem, where class one (π_g) is the "good" class and class two (π_b) is the "bad" class. Assume q attributes are measured from an object coming from either class. We denote the vector of attributes by $\mathbf{x} = (x_1, x_2, \ldots, x_q)'$. Furthermore, we assume that the attributes of an object $\mathbf{x}$ follow different distributions for different classes. Let $f_i(\mathbf{x})$ be the probability density function of class π_i, $i = g$ or b. The overall population is formed by combining the two classes. Let p_i denote the proportion of class π_i in the overall population; $p_g + p_b = 1$. 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 $f(\mathbf{x}) = p_g f_g(\mathbf{x}) + p_b f_b(\mathbf{x})$.

We assume that the adversary can control the distribution of the "bad" class π_b. 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_b. Hence $f_b(\mathbf{x})$ is transformed into $f_b^{\mathbf{T}}(\mathbf{x})$. Each such transformation comes with a cost; the transformed object is less likely to benefit the adversary, although more likely to pass the classifier. For example, a spammer could send a legitimate journal call for papers; while this would be hard to detect as spam, it would not result in sales of the spammer's product. When a "bad" object from π_b is misclassified as a "good" object into π_g, it generates profit for the adversary. A transformed object from $f_b^{\mathbf{T}}(\mathbf{x})$ generates less profit than the original one. In all of the simulation studies, we assume that the values of p_g and p_b are not affected by transformation, meaning that adversary transforms the distribution of π_b, but in a short time period does not significantly increase or decrease the number of "bad" objects. However, for a Bayesian classifier p_b and p_g are just parameters that define the classification regions. They can be transformed by the adversary and be adjusted in Bayesian classifier to optimize the classification rule by data miner. 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 chooses a transformation $\mathbf{T}$ from the set of all feasible transformations $\mathcal{S}$, the strategy space.
2. After observing the transformation $\mathbf{T}$, data miner creates a classification rule h.

Consider the case where data miner wants to minimize its misclassification cost. Given transformation $\mathbf{T}$ and the associated $f_b^{\mathbf{T}}(\mathbf{x})$, the data miner responds with a classification rule $h(\mathbf{x})$. Let $L(h, i)$ be the region where the objects are classified as π_i based on $h(\mathbf{x})$ for $i = g$ or b. 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 only profits from the "bad" objects that are classified as "good". Also note that transformation may change the adversary's profit of an object that successfully passes detection. Define $g(\mathbf{T}, \mathbf{x})$ as the profit function for a "bad" object $\mathbf{x}$ being classified as a "good" one, after transformation $\mathbf{T}$ being applied. Define the adversary's payoff function of a transformation $\mathbf{T}$ given h as the following:

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

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 (Leader) chooses an action a_b first and the second player (Follower), after observing the action of the leader, 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 follower responds 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 equilibria 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. Furthermore, follower 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 utilities can be estimated in some application areas. For the case study in Sect. 9, the penalties for transformation can be measured by the amount of money the adversary spends to improve its profile. For botnet detection the penalties for transformation can be the reduction in the amount of traffic generated by the adversary. In addition, different penalties can be used to run what-if analysis. For example, in loan applications, to prevent an adversary transforming the home ownership attribute (falsely claiming to own a home), we can put verification requirements in place. Such verification will make it costly to transform the home ownership attribute in a loan application. Our model can be re-run with different penalties to predict the effect of instituting such a verification process.

An interesting special case of Stackelberg game is the zero-sum game, when the two utility functions have the following relationship: $u_b(\mathbf{T}, h) = -u_g(\mathbf{T}, h)$ (Basar and Olsder 1999). In that case, the Stackelberg solution concept for adversarial classification corresponds to the minimax solution studied in depth by many authors (Lanckriet et al. 2003; El Ghaoui et al. 2003). A general Stackelberg solution for the adversarial classification game automatically handles the minimax solution concept. [2]

The game theoretic framework we propose is different than the well known strategic games, such as non-cooperative strategic games. In a strategic game, each player is not informed about the other player's plan of action. Players take "simultaneous" actions. The famous Nash equilibrium concept (Osborne and Rubinstein 1999) captures the steady state of such a game. In strategic games, a player cannot change its chosen action after it learns the other player's action. Consequently if one player chooses the equilibrium strategy while the other does not, the result can be bad for both of them.

Compared with strategic games with "simultaneous" actions, we choose the Stackelberg game to emphasize the sequential actions of the two players. We assume that the data miner monitors a certain set of attributes through a classifier and the adversary is aware of the existence of the classifier before the game starts. The data miner empirically sets the parameters of the classifier in its initial state. This initial action does not need to be directly modeled by the proposed Stackelberg Game framework. When the game starts, first the adversary transforms the attributes being monitored by the classifier to increase its payoff. In the second step, after observing the transformation employed by the adversary, the data miner optimizes the parameter values of the classifier. The proposed Stackelberg game mimics the parameter tuning action, because the data miner enjoys the freedom to adjust the classifier parameters after it observes a significant change in the data. Although the adversary is the leader in the game, the data miner chooses a certain set of attributes and builds a classifier

[2] Of course, more efficient methods for minimax solution concept could be found using some of the existing techniques such as the ones given in Lanckriet et al. (2003). Here we focus on the general framework.

before the game starts. The data miner sets the tone for the game and therefore has the advantage over the adversary.

We define the **Adversarial Classification Stackelberg Game** $G = (N, H, P, u_b, u_g)$: $N = \{\text{adversary, data miner}\}$. 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 a 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.

In our framework we need the distribution of the "bad" class to understand the transformation being employed and to assess penalty for the adversary. Depending on the type of the classifier being used, the knowledge of the distributions may or may not be necessary to obtain the optimal classification rule employed by data miner. The classifier can be re-trained using a data set containing the transformed bad instances that are collected or simulated.

However, we assume that data miner sticks to one type of classifier while the adversary can choose from all the transformations in the strategy space. For example, if data miner chooses a Bayesian classifier, it adjusts the Bayesian classifier with new weights in order to defeat the adversary's transformations. The data miner will not 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 in a model (or even retraining the model) is much less expensive than switching to a new model.

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

Examining the classifier's performance at equilibrium, where the adversary maximizes its gain given the classifier being optimized to defeat its action, is a sensible choice in a Stackelberg game. For a fixed transformation, it is true that the problem reduces to a regular learning problem. On the other hand, given a certain set of attributes monitored through a classifier, every possible transformation and the optimized classification rule to defeat this transformation generates a corresponding error rate. The main issue is to know which one of the potential transformations are more likely to be adopted in practice and what to do about it if such transformation occurs. One approach is to select a set of attributes that minimizes the worst case error rate under all possible transformations. For some application areas such as online lending in Sect. 9, the worst case scenario is unlikely to happen because of the heavy penalties for transformation. Equilibrium information offers an alternative to the minimax approach. When considering the whole strategy space of all the possible transformations and the penalties for transformation, an equilibrium state offers insight into the error rate to be expected from a classifier in the long run. As suggested by our paper, such insights are used to guide attribute selection.

In this game, we assume that the adversary acts by first applying a transformation $\mathbf{T}$. After observing $\mathbf{T}$ being applied to the "bad" class $\left(f_b^{\mathbf{T}}(\mathbf{x})\right)$, 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 $L\,(h_{\mathbf{T}}, g)$ be the region where the objects are classified as π_g given $h_{\mathbf{T}}$. Define the adversary gain of applying transformation $\mathbf{T}$ as:

$$W(\mathbf{T}) = u_b\,(\mathbf{T}, h_{\mathbf{T}}) = \int\limits_{L(h_{\mathbf{T}},g)} g(\mathbf{T}, \mathbf{x}) f_b^{\mathbf{T}}(\mathbf{x}) d\mathbf{x} = E_{f_b^{\mathbf{T}}}\left(I_{\{L(h_{\mathbf{T}},g)\}}(\mathbf{x}) g(\mathbf{T}, \mathbf{x})\right). \tag{2.1}$$

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

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

Game theory (Osborne and Rubinstein 1999) establishes that the solution of the above maximization problem is a subgame perfect equilibrium. Furthermore if the strategy space $\mathcal{S}$ is compact and $W(\mathbf{T})$ is continuous, the maximization problem has a solution.

The above formulation can accommodate any well-defined set of transformations $\mathcal{S}$, any appropriate distributions with densities $f_g(\mathbf{x})$ and $f_b(\mathbf{x})$, 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 an equilibrium in a Stackelberg game is equivalent to solving an optimization problem. We present a general solution based on stochastic search in Sect. 3, and a heuristic solution based on an approximation of the classification region for minimal cost Bayesian classifier in Sect. 6, for high dimensional tasks.

3 Solving for the equilibrium

To search for a subgame perfect equilibrium, the underlying problem is converted to an optimization problem similar to the one defined by Eq. 2.2 (Basar and Olsder 1999). Although there are computational game theory tools to find subgame perfect equilibria for *finite* games (McKelvey et al. 2007), searching for subgame perfect equilibria is a hard problem in general (Basar and Olsder 1999). Therefore, optimization techniques such as genetic algorithms, have been applied to search for subgame perfect equilibria (Vallee and Basar 1999). 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 integration region $L\,(h_{\mathbf{T}}, g)$ for the adversary gain $W(\mathbf{T})$ is a function of transformation $\mathbf{T}$, finding an analytical solution to the maximization problem is challenging. In addition, even calculating the integration analytically for a specific transformation is not possible for high dimensional data. We have to evaluate $W(\mathbf{T})$ numerically. Because of such 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 a global optimal solution is highly desirable. In the rest of this section, Monte Carlo integration method is introduced to compute $W(\mathbf{T})$ and simulated annealing algorithm is implemented to solve for a subgame perfect equilibrium.

3.1 Monte carlo integration

Monte Carlo integration technique converts an 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

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

$\int g(\mathbf{x})dx$ 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 $\left\{\mathbf{x}^i\right\}$ from $f(\mathbf{x})$ and calculate $\mu_m = 1/m \times \sum_1^m \left(g\left(\mathbf{x}^i\right)/f\left(\mathbf{x}^i\right)\right)$. 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:

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

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

Algorithm 3.1 Monte Carlo Integration

```
{Evaluating W(T) for a given transformation T}
Generate m samples {x^i} from f_b^T(x)
sum = 0
for i = 1 to m do
  if x^i ∈ L(h_T, g) then
    sum = sum + g(T, 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 from physics (Duda et al. 2001). 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 temperature decreases, the system will choose a low energy state with higher probability. Similarly, when temperature is high, our simulated annealing algorithm will accept nearly all new points. As 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 (Duda et al. 2001). The current temperature is decreased by multiplying it by a reduction rate 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: $TempMin, TempMax, ReductionRate \in (0, 1), SampleSize$
Select random $\mathbf{T}$ and evaluate $W(\mathbf{T})$
Let $\mathbf{T}_c$ be the starting transformation with value $evalc = W(\mathbf{T}_c)$
Let $\mathbf{T}_g$ be the best transformation seen in the search with value $evalg = W(\mathbf{T}_g)$
$\mathbf{T}_g = \mathbf{T}_c, evalg = evalc$
$TempCurrent = TempMax$
while $TempCurrent \geq TempMin$ **do**
  for $i = 1$ *to* $SampleSize$ **do**
    Randomly select $\mathbf{T}_n$ in neighborhood of $\mathbf{T}_c$
    Let $evaln = W(\mathbf{T}_n)$ for $\mathbf{T}_n$
    if $evaln > evalc$ **then**
      $\mathbf{T}_c = \mathbf{T}_n, evalc = evaln$
      if $evalg < evaln$ **then**
        $\mathbf{T}_g = \mathbf{T}_n, evalg = evaln$
      end if
    else if $rand(0, 1) \leq exp\left\{\frac{evaln - evalc}{TempCurrent}\right\}$ **then**
      $\mathbf{T}_c = \mathbf{T}_n, evalc = evaln$
    end if
  end for
  $TempCurrent \times = ReductionRate$
end while

There are several issues about searching for a subgame perfect equilibrium. First of all, we need a sample of transformed objects, for re-training the classifier and Monte

Carlo integration. We can solve this problem by either simulating a data set from the transformed population, or generating transformed samples from a real life data set. Next we need to re-train the classifier and construct the classification region $L(h_{\mathbf{T}}, g)$ 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 by using bootstrap samples or simulated data without the knowledge of the densities. Therefore, our formulation could work with any type of classifier.

Given a proper cooling schedule, the simulated annealing algorithm has the ability to approach an equilibrium transformation for the adversary. Afterward we can obtain the data miner's corresponding equilibrium classification rule. We have done experiments to test simulated annealing algorithm's effectiveness. The results showed that simulated annealing algorithm performed up to expectation. Details are omitted from the paper due to limited space. None of the simulations in Sect. 4 uses simulated annealing, for such a search algorithm demands excessive computational time. Later in Sects. 6 and 7 and 8, we focus on Bayesian classifier and provide a simple heuristic solution. The heuristic solution avoids estimating the multivariate density function, which needs a large sample in high dimensional space. For detailed discussion please refer to Sect. 6.2.4.

3.2.1 Multiple equilibria and "Clean" bad objects

To apply our game theoretic model to a real problem, the earliest available "bad" objects can be considered as the "clean" untransformed bad objects. We could then compare "bad" objects obtained in a latter time with the "clean" bad objects to assess the extent of transformation and penalty. For example, spam emails collected in the mid-1990s, could be used as the "clean" bad objects.

Another important issue for the implementation of our game theoretic framework is how to deal with multiple equilibria. In a Stackelberg game the payoff for the leader is unique (Basar and Olsder 1999), but the optimal strategy for the leader may not be unique. Given the leader's action, the follower chooses the best available strategy. In our case, the optimal classification rule is unique given the leader's action. When there are multiple equilibria in an Adversarial Classification Stackelberg game, multiple transformations will generate identical maximum adversary gains. They may or may not introduce identical equilibrium Bayes errors or equilibrium misclassification costs. Ideally we should investigate all the equilibria, and use either the average or the maximum of the equilibrium Bayes errors, and the average or the maximum of the equilibrium misclassification costs for attribute selection.

Mitra et al. (1986) has done a thorough theoretical study of simulated annealing convergence condition and convergence speed on NP complete problems. As long as the target function is bounded (it can take arbitrary shape inside the bounds), and temperature decreases at a proper rate (a proper cooling schedule), simulated annealing algorithm forms a strongly ergodic time inhomogeneous Markov Chain and is able to converge to a global optimum.

In our model, our adversary gain is indeed bounded, $k_1 \leq W(\mathbf{T}) \leq k_2$. However, proofs in Mitra et al. (1986) consider a finite set of discrete variables. In our case, for continuous transformations, we can cover the strategy space of transformations with very fine mesh grids. Then we could obtain a good approximation of a global optimum. An approximate solution can provide important information about the adversary's long term plan of action, and can often provide quite accurate estimates of an equilibrium Bayes error and the associated misclassification cost, as shown in Sects. 5.1, 6.2.1, and 6.2.2.

4 Equilibrium performance

We have conducted simulations to examine various equilibrium strategies. The results show that there exist interesting equilibria for this problem. The simulation results demonstrate the effect of various penalties and cost matrices. For example, the penalty imposed on a given attribute needs to be larger than a threshold value to force the adversary to stop. Yet after a certain point, further increasing the penalty will not have a significant impact. It remains a challenge to discover the penalty threshold values for a given attribute, as well as its 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 be valuable for classification purposes. 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 involve only one attribute. The results are obtained from exhaustive search on fine grids in the strategy space. By using just one attribute in the model, we are able to obtain very accurate estimates of equilibrium transformations without a time consuming stochastic search. Gaussian distributions and minimal cost Bayesian classifier are used in the experiments. Gaussian distributions have a 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 offers important insight into equilibrium strategies. Next we 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:

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

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 an original "bad" object 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 declines linearly according to the extent of

the transformation. Here a is the reduction rate. The minimum profit is 0. A "bad" object will not generate negative profit. This definition of the profit is based on the following intuition: The more the original distribution changes, the higher the penalty 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 an equilibrium is established. Further assume that each class $\pi_i, i = g$ or b, follows a Gaussian distribution. $f_i(x)$ is the density function of $N\left(\mu_i, \sigma_i^2\right)$.

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 instead of the entire real line, because equilibrium transformation will not be too far away from the identity transformation when there is a penalty. Under transformation T, $f_b^T(x)$ becomes the density of $N\left(T \times \mu_b, T^2 \times \sigma_b^2\right)$, which is the new distribution for the "bad" class π_b.

The expected cost of misclassification can be written as in Fukunaga (1990):

$$C(T,h) = \int_{L(h,g)} \left(c(g,g)p_g f_g(x) + c(g,b)p_b f_b^T(x)\right) dx + \int_{L(h,b)} \left(c(b,g)p_g f_g(x) + c(b,b)p_b f_b^T(x)\right) dx,$$

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

$$h_T(x) = \begin{cases} \pi_g & (c(g,b) - c(b,b))p_b f_b^T(x) \le (c(b,g) - c(g,g))p_g f_g(x) \\ \pi_b & \text{otherwise} \end{cases}$$

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

We can re-write the subgame perfect equilibrium transformation using the above specifications as follows:

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

4.2 Adversary gain and bayes error

In the experiments we examine the adversary gain $W(\mathbf{T})$ as a function of $\mathbf{T}$. Furthermore we demonstrate how an equilibrium responds to increasing penalties and various misclassification costs.

4.2.1 *Effect of penalty and misclassification cost*

The general set-up of the experiments is in Sect. 4.1. The parameter values are in Table 1. There are three experiments. In each experiment we fix a cost matrix c and the initial distributions of the two classes. We then gradually increase 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: $c(b, g)/c(g, b) = 1$. In the second experiment, misclassifying, a "good" object costs twice as much: $c(b, g)/c(g, b) = 2$. Then in the third experiment, misclassifying a "good" object costs ten times as much: $c(b, g)/c(g, b) = 10$. From the three experiments, we are able to observe the effect of misclassification cost and penalty for transformation on equilibrium strategies.

When $T = 1, \pi_b$ is 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 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 achieves very high success rate as shown in both Fig. 1 and Table 1. When increasing the cost of misclassifying a "good" object slightly, the error rate moves from 0.0022 to 0.0041, and the adversary gain moves from 0.0045 to 0.0113.

However when the adversary starts to transform the "bad" objects, even the optimal classification rule against the transformation fails to block a significant proportion of

Table 1 Parameter values for three experiments

	k	Cost c	π_g	p_g	π_b	p_b	$W(1)$	$e(1)$
Experiment 1	1	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	$N(5, 0.64)$	0.65	$N(1, 0.36)$	0.35	0.0045	0.0022
Experiment 2	1	$\begin{bmatrix} 0 & 1 \\ 2 & 0 \end{bmatrix}$	$N(5, 0.64)$	0.65	$N(1, 0.36)$	0.35	0.0059	0.0027
Experiment 3	1	$\begin{bmatrix} 0 & 1 \\ 10 & 0 \end{bmatrix}$	$N(5, 0.64)$	0.65	$N(1, 0.36)$	0.35	0.0113	0.0041

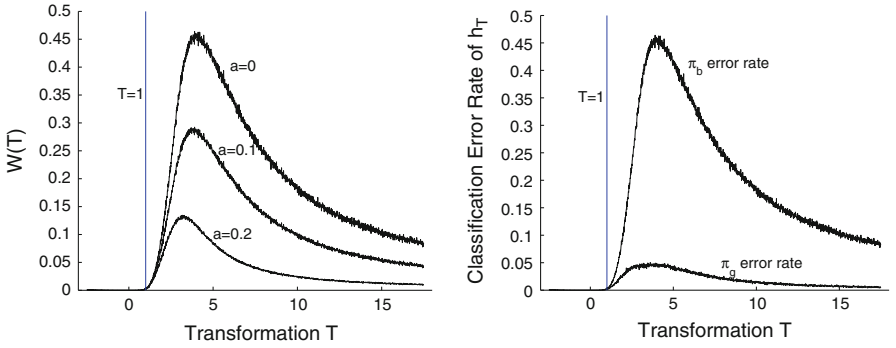

Fig. 1 *Left:* the adversary gains for increasing penalty; *Right:* the Bayes error. Both for Experiment 1

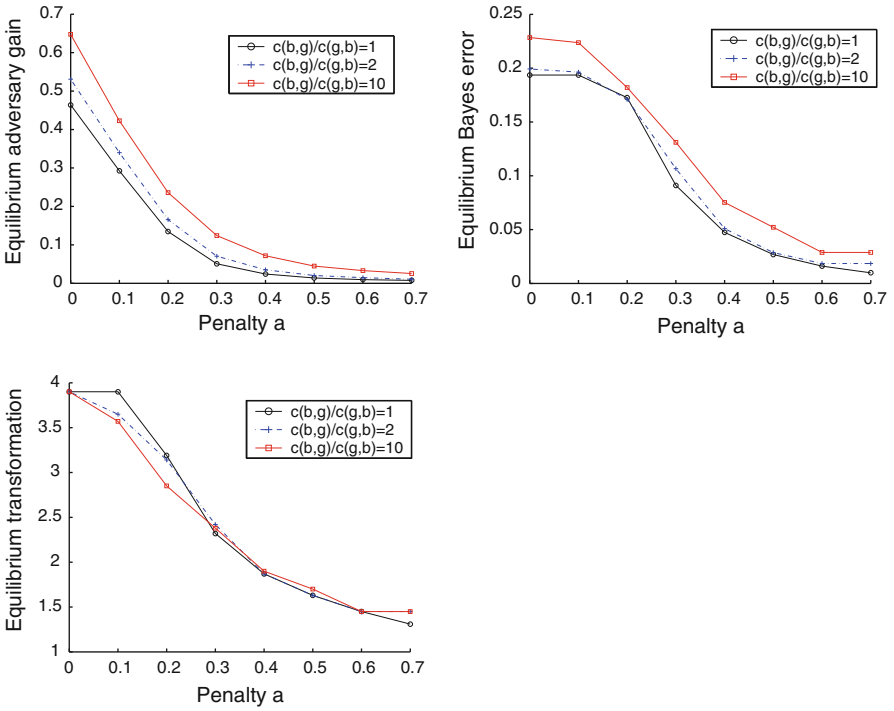

Fig. 2 *Top left:* the equilibrium adversary gains; *Top right:* the equilibrium Bayes errors; *Bottom left:* the equilibrium transformations. Penalty a from 0 to 0.7 by 0.1

the transformed "bad" objects. It also makes a few mistakes with the "good" objects. Since we set $k = 1$, we have $W(T) = e_b(T)$ when $a = 0$. Also notice that the Bayes error (the overall misclassification error rate) as a function of T is not affected by the penalty a. Increasing the penalty a reduces the adversary gains for all the transformations because a "bad" object generates less profit, as shown in the left panel of Fig. 1.

Figure 2 illustrates the effect of penalty a more clearly. As a increases from 0 to 0.7, the equilibrium transformation is pushed toward 1 (i.e., the identity transformation). Both the equilibrium adversary gain and the equilibrium Bayes error drop down to near 0. The equilibrium Bayes error moves closer to the initial Bayes error as the equilibrium transformation moves toward the identity transformation.

The left panel of Fig. 2 shows that $a = 0.2$ is an elbow point of the equilibrium adversary gain $W(T^e)$. For a heavy penalty $a \geq 0.2$, $W(T^e)$ is close to 0, and the equilibrium Bayes error $e(T^e)$ starts declining sharply. Misclassification costs also affect the equilibrium performance. As the cost of misclassifying a "good" object $c(b, g)$ increases, the equilibrium Bayes error and $W(T^e)$ both become larger.

To more carefully examine the effect of penalties and cost matrices on the equilibrium Bayes error and the equilibrium transformation, we gradually increase a from 0 to 0.2 (a moderate penalty) in 0.01 increments. In Fig. 3, $W(T^e)$ decreases as a increases for all three cost matrices. The equilibrium transformation T^e starts moving

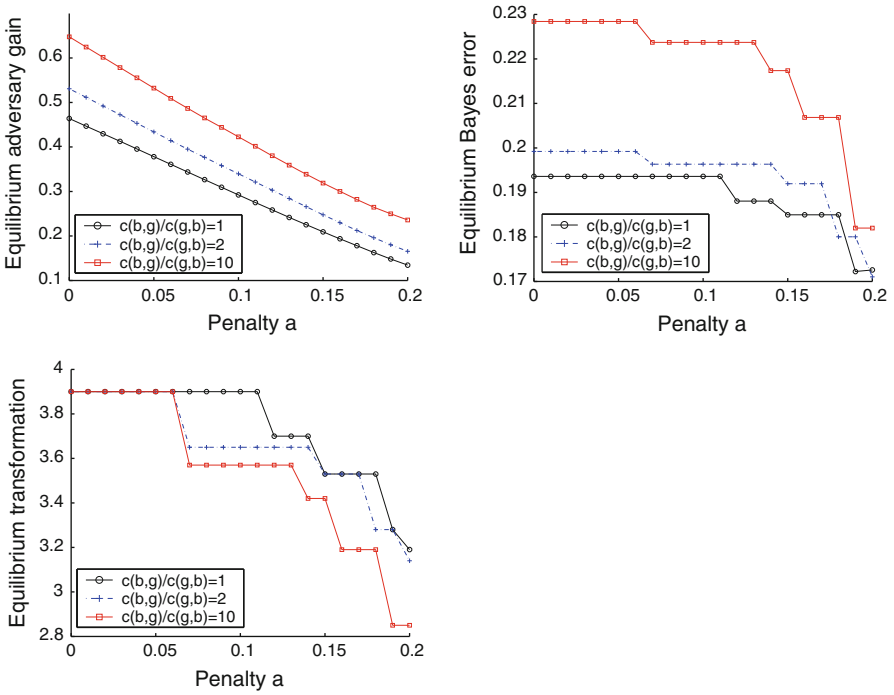

Fig. 3 *Top left:* the equilibrium adversary gains; *Top right:* the equilibrium Bayes errors; *Bottom left:* the equilibrium transformations. Penalty a from 0 to 0.2 by 0.01

notably toward 1 only when a exceeds certain threshold values. The same phenomenon occurs for the equilibrium Bayes error $e\,(T^e)$. When the cost of misclassifying a "good" object $c(b, g)$ is equal to $c(g, b)$, it required a heavier penalty for T^e to significantly move toward identity transformation and for $e\,(T^e)$ to drop.

Above results indicate that 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 and the performance of a classifier depends heavily on how the adversary will transform the "bad" class. 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 predict 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 are stated as follows: (1) Classifier's 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 the adversary's equilibrium transformation is close to the identity transformation, then the data miner wins the battle. However when the cost of misclassifying "good" objects is high, the classifier will

pass more objects from both classes. In this case the adversary does not have to transform much to maximize its gain. This is not a favorable scenario for the data miner. Figure 3 shows that adversary's equilibrium transformation might be close to the identity transformation, yet the adversary manages to have more "bad" objects pass the classifier and receives big profit from each "bad" object. Therefore in Sect. 6, we only use the 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.

5 Extreme classification rules

A minimal cost Bayesian classifier under certain conditions may create one of the two extreme classification rules: Either pass all the objects, or block all the objects. Next we study two one-dimensional examples, using Gaussian and Bernoulli random variables respectively. We examine the impact of such extreme classification rules on equilibrium strategies. The two examples show that an approximate solution provides information about the adversary's potential action. It reveals how attributes will be transformed to maximize the adversary gain and the magnitude of such transformations.

5.1 One discontinuous point

Here we present a special one-dimensional Gaussian example. The experiment setting follows the one in Sect. 4.1. The profit function is the same as Eq. 4.1. Unlike Sect. 4, now there exists a transformation $\mathbf{T}^0$ that makes two classes indistinguishable (i.e., $f_b^{\mathbf{T}^0}(\mathbf{x}) = f_g(\mathbf{x})$). Then a Bayesian classifier, depending on the misclassification costs and the population proportions p_g and p_b, will either pass all objects or block all objects. In this example π_g has distribution $N(0.2160, 0.3168)$, and π_b has distribution $N(0.36, 0.88)$. $T^0 = 0.6$ will transform $f_b(x)$ into $f_g(x)$. We set $p_g = p_b = 0.5, k = 1, c(g, g) = c(b, b) = 0, c(b, g)/c(g, b) = 1$, and penalty $a = 0$. We observe that the adversary gain $W(T)$ is discontinuous at $T^0 = 0.6$.

In Fig. 4, the value of the adversary gain $W(T)$ is plotted against the transformation T. When there is no penalty for transformation ($a = 0$), and misclassification cost of "good" objects is the same as the misclassification cost of "bad" objects, the classification rule is to pass all objects given transformation $T^0 = 0.6$. The error rate in the "bad" class π_b is 100%. The adversary gain $W(T)$ reaches a maximum value 1. $T^0 = 0.6$ and the classification rule to pass all objects form the equilibrium.

$W(T)$ is continuous everywhere else except for the equilibrium transformation $T^0 = 0.6$. The discontinuity is created by the sudden change of the classification region. At T^0 the "bad" class coincides with the "good" class, and the Bayesian classifier decides to pass all objects. We then integrate over the entire space to compute $W(T)$. In this case, a stochastic search algorithm is unlikely to return the exact transformation T^0 and the corresponding classification rule. We will be able to discover

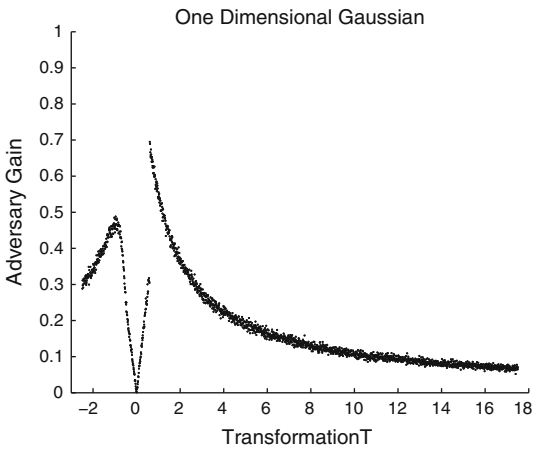

Fig. 4 Equilibrium transformation equal to 0.6

an approximate solution $T = 0.6 + \varepsilon$, where $\varepsilon > 0$. The associated Bayes error $e(T)$ and the adversary gain $W(T)$ are much smaller than the equilibrium ones. However the approximate solution informs us toward which direction the attribute will be transformed.

5.2 One-dimensional Bernoulli

In spam filtering, many attributes are binary and can be modeled as Bernoulli random variables. For example, the popular spam filtering software SpamAsssassin works by "scoring" every e-mail message based on a range of tests designed to decide if that email is spam or not (e.g., checking if the body contains any forbidden words). Each test can be considered as a Bernoulli variable with two values, "pass" or "fail". The sum of the individual test scores is an email's overall spam score. Emails with spam scores over a certain user-controlled threshold value are classified as spam. In addition to these tests, SpamAssassin uses a modified Naïve Bayes algorithm. Although SpamAssassin is not purely Naïve Bayes, the following experiments attempt to mimic a component of the spam filter. Our results could be useful for tools such as Spam Assassin to evaluate different attributes.

Based on the above intuition, we first consider one Bernoulli variable X. We use the following probabilities. In the "good" class π_g let

$$X = \begin{cases} 1 & \text{with probability} \quad r_g \\ 0 & \text{with probability} \quad 1 - r_g \end{cases}$$

In the "bad" class π_b the initial probabilities before transformation are

$$T^{-1}(X) = \begin{cases} 1 & \text{with probability} \quad r_b \\ 0 & \text{with probability} \quad 1 - r_b \end{cases}$$

Many attributes are selected to build a classifier if they are highly likely to appear in the "bad" class and rarely appear in the "good" class or vice-versa. If an attribute is a strong indicator of spam emails, such as containing a blacklisted IP address, we assume $X = 1$ means it appears in the email. On the other hand an attribute (a good word) that serves as a strong indicator of legitimate emails will have X set to 1 if an email does not contain that good word. Then a Bernoulli attribute will always have r_g close to 0 and r_b close to 1. A transformation adopted by the adversary will reduce the difference between r_b and r_g.

In real life, spam emails have become the majority. Hence the size of the "bad" class p_b is much greater than the size of the "good" class p_g, $p_b >> p_g$.

We model the transformation of a Bernoulli variable X as a probability T: With a probability $T (0 \leq T \leq 1)$, a "1" in the "bad" class will be switched to a "0". When $T = 0$, the "bad" objects are not transformed. It has the same effect as what identity transformation does for Gaussian distribution. Then the transformed "bad" class has the following probability function:

$$X = \begin{cases} 1 & \text{with probability} \quad r_b \times (1 - T) \\ 0 & \text{with probability} \quad 1 - r_b + r_b \times T \end{cases}$$

A transformation employed by the adversary will be penalized. Here we introduce a profit function defined by Eqs. 5.1 and 5.2 for Bernoulli variable X. It assumes the following structure: If a "bad" object is being successfully transformed ($T^{-1}(X) = 1$ and $X = 0$), the transformation will be most heavily penalized. If the transformation is not successful ($T^{-1}(X) = 1$ and $X = 1$), it will be lightly penalized. If originally a "bad" object is not on the blacklist, then it will not be transformed nor penalized $\left(T^{-1}(X) = X = 0\right)$. For example, obfuscating a word always reduces readability of a spam email. Obfuscating a word to the degree that a filter cannot recognize it heavily reduces readability and consequently heavily reduces response rate to an email spam. We have $k > 1$ and $a < 1$ for Eqs. 5.1 and 5.2.

$$g(T, X) = \begin{cases} k^{a(T)} & \text{if} \quad T^{-1}(X) = 1 \quad \text{and} \quad X = 0 \\ k^{\frac{1+a(T)}{2}} & \text{if} \quad T^{-1}(X) = 1 \quad \text{and} \quad X = 1, \\ k & \text{if} \quad T^{-1}(X) = X = 0 \end{cases} \tag{5.1}$$

where penalty $a(T)$ is a function of T:

$$a(T) = \begin{cases} a \times (1 - T) & \text{if} \quad a \geq 0 \\ a \times T & \text{if} \quad a < 0 \end{cases} \tag{5.2}$$

The region $L(h_T, g)$ where objects are classified as "good" becomes:

$$\frac{(c(g, b) - c(b, b)) \times p_b}{(c(b, g) - c(g, g)) \times p_g} \times \frac{(r_b(1 - T))^X (1 - r_b + r_b T)^{1-X}}{(r_g)^X (1 - r_g)^{1-X}} \leq 1.$$

A Bernoulli variable X can take only two values (0 or 1) in both classes. The two extreme classification rules happen quite often, as shown in Sect. 5.2.1.

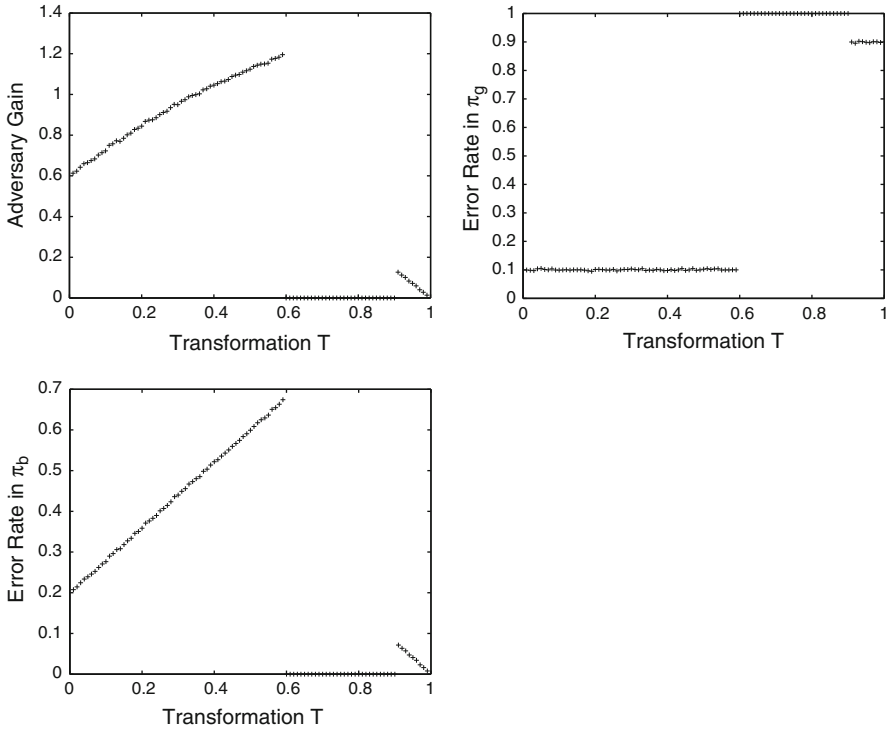

Fig. 5 *Top left:* adversary gains; *Top right:* "good" class error rates; *Bottom left:* "bad" class error rates. $c(b, g)/c(g, b) = 3$

5.2.1 Experiment one

Here we apply profit function defined by Eqs. 5.1 and 5.2 with $a = 0.5$ and $k = 3$. We set the initial probabilities in each class as $r_b = 0.8$ and $r_g = 0.1$. We have the population proportions as $p_b = 0.8$ and $p_g = 0.2$. The transformation T is from 0.01 to 0.99 by an increment of 0.01. We assume correct classification cost is 0: $c(g, g) = c(b, b) = 0$. The misclassification cost of "good" object usually is much bigger. We examine two sets of misclassification costs: (1) $c(b, g)/c(g, b) = 3$; (2) $c(b, g)/c(g, b) = 6$.

Figure 5 shows the adversary gain and the error rates in two classes, given the misclassification cost of "good" objects is three times that of "bad" objects, $c(b, g)/c(g, b) = 3$. Straightforward calculation shows when $\frac{19}{32} < T < \frac{29}{32}$, the classifier blocks all objects and cause 100% error in the "good" class π_g. As T increases from 0, more "bad" objects disappear from the blacklist and the adversary gain increases. The equilibrium transformation is $T = \frac{19}{32}$, right before a slightly heavier transformation triggers the block-all classification rule. $r_b(1 - T) = 0.3250$. At the equilibrium, whenever $X = 1$, it is classified into π_b. All 0s are classified into π_g. When $T \geq \frac{29}{32}$, it is the opposite: all 1s are classified into "good" class and all 0s into "bad" class.

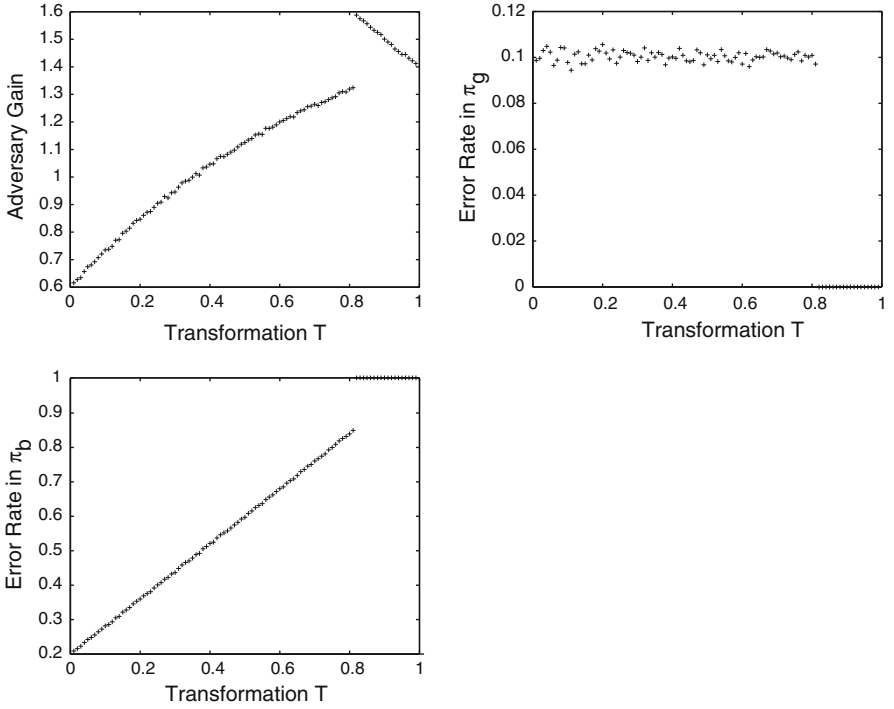

Fig. 6 *Top left:* adversary gains; *Top right:* "good" class error rates; *Bottom left:* "bad" class error rates. $c(b, g)/c(g, b) = 6$

Figure 6 shows the adversary gain and the error rates in two classes, given the misclassification cost of "good" objects is six times that of "bad" objects, i.e., $c(b, g)/c(g, b) = 6$. As the misclassification cost of "good" objects becomes bigger, the classifier tends to pass more objects. When $T \geq \frac{13}{16}$, the optimal classification rule is to pass all objects, producing 100% error in the "bad" class π_b. The equilibrium transformation is $T = \frac{13}{16}$. It is the smallest transformation to force the Bayesian classifier to pass all objects. $r_b(1 - T)$ (0.15) is very close to r_g (0.1). The equilibrium classification rule is to pass all. Further increase in T (i.e., $T > \frac{13}{16}$) receives heavier penalty, therefore reduces the adversary gain.

5.2.2 Experiment two

Next we fix $c(b, g)/c(g, b) = 3, c(b, b) = c(g, g) = 0, p_g = 0.2, p_b = 0.8$, $r_g = 0.1$, and $r_b = 0.8$. We examine the impact of penalty on equilibrium strategies using profit function defined by Eqs. 5.1 and 5.2. We gradually reduce penalty a in Eq. 5.2 from 0.9 to -4 by 0.1. Only when $a \leq -1.5$, the equilibrium transformation becomes significantly smaller than $\frac{19}{32}$ as shown in Fig. 7. For Bernoulli random variables, only really heavy penalty will discourage the adversary from transforming the objects under its control. Meanwhile in order to reduce equilibrium Bayes error, data

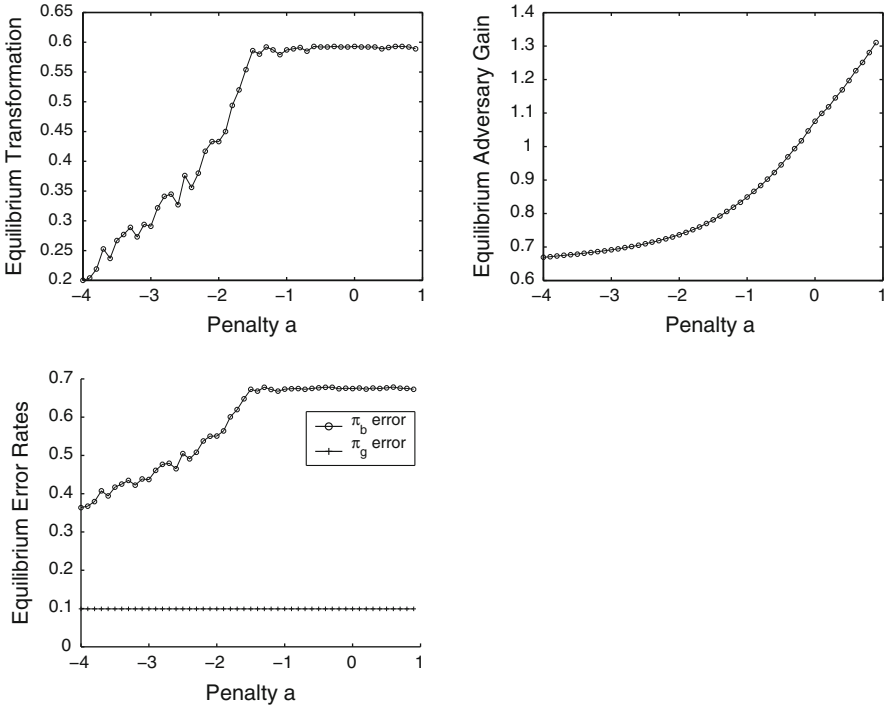

Fig. 7 *Top left:* Equilibrium transformations; *Top right:* Equilibrium adversary gains; *Bottom left:* Equilibrium Bayes errors

miner needs some tolerance for making a wrong decision with a "good" object. This is different from Gaussian case, where even very small penalty can discourage the adversary. We report a simulation study with multiple Bernoulli variables in Sect. 8.

6 A heuristic solution for Bayesian classifier

Section 3 provides a general solution that works well in lower dimensional space. In high dimensional space a large sample is needed to estimate the adversary gain $W(\mathbf{T})$ using Monte Carlo integration. Furthermore, simulated annealing algorithm evaluates large number of transformations before it converges. To conquer the computational issues associated with the algorithms in Sect. 3, we propose an approximate solution in this section. We focus on the Bayesian classifier, because it has an explicit clean expression for the classification regions. For two attributes or more in the model, results are developed under the following assumptions.

1. Attributes $x_1, x_2, \ldots, x_q$ are independent.
2. The profit function $g(\mathbf{T}, \mathbf{x}) = \sum_{j=1}^{q} g_j\left(T_j, x_j\right)$, where $\forall j, g_j\left(T_j, x_j\right) \geq 0$. Here we assume each attribute is penalized individually.

When the attributes are correlated in data, the effect of the above conditions is similar to Naïve Bayes classifier. The classification rule is determined by the product

of the marginal densities instead of the joint density. Naïve Bayes classifier has good performance even if attributes are correlated. The simulation results in this section show that the heuristic solution developed under the above condition also has good performance when attributes are correlated.

The region where objects are classified into the "good" class π_g is:

$$L\left(h_{\mathbf{T}}, g\right) = \Big\{\mathbf{x} : (c(g,b) - c(b,b)) \times p_b \times f_b^{\mathbf{T}}(\mathbf{x}) \leq (c(b,g) - c(g,g)) \times p_g \times f_g(\mathbf{x})\Big\}.$$

Under the independence assumption the density of the "good" class π_g is

$$f_g(\mathbf{x}) = \prod_{j=1}^{q} f_{g,j}\left(x_j\right),$$

and the density of the "bad" class π_b is

$$f_b^{\mathbf{T}}(\mathbf{x}) = \prod_{j=1}^{q} f_{b,j}^{T_j}\left(x_j\right).$$

$L\left(h_{\mathbf{T}}, g\right)$ can be written as:

$$L\left(h_{\mathbf{T}}, g\right) = \left\{\mathbf{x} : \prod_{j=1}^{q}\left(\left(\frac{(c(g,b) - c(b,b)) \times p_b}{(c(b,g) - c(g,g)) \times p_g}\right)^{\frac{1}{q}} \times \frac{f_{b,j}^{T_j}\left(x_j\right)}{f_{g,j}\left(x_j\right)}\right) \leq 1\right\} \tag{6.1}$$

Notice that this is the classification region of the Naïve Bayes classifier regardless of whether the attributes are correlated or not. Let

$$y_j = \left(\frac{(c(g,b) - c(b,b)) \times p_b}{(c(b,g) - c(g,g)) \times p_g}\right)^{\frac{1}{q}} \times \frac{f_{b,j}^{T_j}\left(x_j\right)}{f_{g,j}\left(x_j\right)}. \tag{6.2}$$

From now on we only consider the case where

$$c(g,b) - c(b,b) > 0,$$

and

$$c(b,g) - c(g,g) > 0.$$

Then $\forall j,\ y_j > 0$. We have $L\left(h_{\mathbf{T}}, g\right) = \left\{\mathbf{x} : \prod_1^q y_j \leq 1\right\}$. The region where objects are classified into the "bad" class π_b can be written as $L\left(h_{\mathbf{T}}, b\right) = \left\{\mathbf{x} : \prod_1^q y_j > 1\right\}$. Notice $L\left(h_{\mathbf{T}}, g\right) + L\left(h_{\mathbf{T}}, b\right) = \mathcal{R}^q$, where $\mathcal{R}^q$ is the q-dimensional Euclidean space

(i.e., the whole sample space). Our simulation results shown in the following sections indicate that even if the independence assumption is not satisfied, the approximate solution points to the neighborhood of a true equilibrium. We meet the second assumption by approximating the exact profit function.

6.1 Lower and upper bounds of the classification regions

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

The following are simple lower and upper bounds of $L(h_{\mathbf{T}}, g)$ and $L(h_{\mathbf{T}}, b)$:

$$\prod_{j=1}^{q} [y_j \le 1] \subset L(h_{\mathbf{T}}, g) \subset \mathcal{R}^q - \prod_{j=1}^{q} [y_j > 1],$$
$$\prod_{j=1}^{q} [y_j > 1] \subset L(h_{\mathbf{T}}, b) \subset \mathcal{R}^q - \prod_{j=1}^{q} [y_j \le 1].$$

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

Remark 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 perform several lower dimensional searches instead of a high dimensional search. The heuristic solution proposed next naturally follows.

6.2 A heuristic solution for the equilibrium transformation

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

$$G_j(c,d) = \int I_{[c<y_j\le d]}\left(x_j\right) \times g_j\left(T_j, x_j\right) \times f_{b,j}^{T_j}\left(x_j\right)\, dx_j, \tag{6.3}$$

$$P_j(c,d) = \int I_{[c<y_j\le d]}\left(x_j\right) \times f_{b,j}^{T_j}\left(x_j\right)\, dx_j, \tag{6.4}$$

$$Q_j(c,d) = \int I_{[c<y_j\le d]}\left(x_j\right) \times f_{g,j}\left(x_j\right)\, dx_j, \tag{6.5}$$

where c and d are non-negative real numbers that mark the range of y_j in the indicator function $I_{[c<y_j\le d]}\left(x_j\right)$.

We could have quite tight lower and upper bounds of $W(\mathbf{T})$ based on Eqs. 2.1 and 2.2 Please refer to Appendix A for the formulas. By maximizing a good lower bound of the adversary gain $W(\mathbf{T})$, we obtain a heuristic solution to approximate an equilibrium transformation. However, the lower bound $W^{\text{lower}}(\mathbf{T})$ given in Appendix A involves an infinite number of one dimensional integrals, $G_j(c,d)$s, $P_j(c,d)$s, and $Q_j(c,d)$s. In practice we can only include finite number of terms from the $W^{\text{lower}}(\mathbf{T})$ formula given in Appendix A. In our experiments we observe that maximizing the one-dimensional integral $G_j(0,1)$ in the first term of the lower bound $W^{\text{lower}}(\mathbf{T})$ formula, $\left(\prod_{j=1}^{q} P_j(0,1)\right) \times \left(\sum_{j=1}^{q} \frac{G_j(0,1)}{P_j(0,1)}\right)$, leads to a transformation close to equilibrium transformation $\mathbf{T}^e$. Define

$$T_j^a = \text{argmax}_{T_j}\, G_j(0,1) = \text{argmax} \int I_{[0<y_j\le 1]}\left(x_j\right) \times g_j\left(T_j, x_j\right) \times f_{b,j}^{T_j}\left(x_j\right) dx_j. \tag{6.6}$$

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

The heuristic transformation $\mathbf{T}^a$ is not equal to the exact equilibrium transformation. Sections 6.2.1 and 6.2.2 contain bivariate Gaussian examples that demonstrate how well the heuristic solution approximates the exact equilibrium transformation. The results show that heuristic transformation provides important information about the adversary's long term plan of actions.

6.2.1 Experiment one

In this experiment we use the following profit functions:

$$\begin{aligned} g_1(T_1, x_1) &= \max\left(0.5 - a_1 \times \left|T_1^{-1}(x_1) - x_1\right|, 0\right), \\ g_2(T_2, x_2) &= \max\left(0.5 - a_2 \times \left|T_2^{-1}(x_2) - x_2\right|, 0\right), \\ g(\mathbf{T}, \mathbf{x}) &= g_1(T_1, x_1) + g_2(T_2, x_2). \end{aligned} \tag{6.7}$$

Transformation $\mathbf{T}$ is a diagonal matrix, $\text{diag}(T_1, T_2)$. $\mathbf{T}(\mathbf{x}) = \mathbf{T}\mathbf{x}$ is a transformed object. In both Sections, after transformation $\mathbf{T}$, π_b has distribution $N\left(\mathbf{T}\mu_b, \mathbf{T}\Sigma_b\mathbf{T}'\right)$.

Table 2 Initial distributions

	π_g
Independent	$N\left(\begin{bmatrix}3.6\\2.5\end{bmatrix}, \begin{bmatrix}0.9^2 & 0\\0 & 1.2^2\end{bmatrix}\right)$
Correlated	$N\left(\begin{bmatrix}3.6\\2.5\end{bmatrix}, \begin{bmatrix}0.9^2 & 0.6\times 0.9\times 1.2\\0.6\times 0.9\times 1.2 & 1.2^2\end{bmatrix}\right)$
	π_b
Independent	$N\left(\begin{bmatrix}0.8\\0.6\end{bmatrix}, \begin{bmatrix}0.5^2 & 0\\0 & 0.9^2\end{bmatrix}\right)$
Correlated	$N\left(\begin{bmatrix}0.8\\0.6\end{bmatrix}, \begin{bmatrix}0.5^2 & 0.7\times 0.5\times 0.9\\0.7\times 0.5\times 0.9 & 0.9^2\end{bmatrix}\right)$

Table 3 Exact and heuristic equilibrium performance

	Independent			*Correlated*		
	Transformation	$W(\mathbf{T})$	$e(\mathbf{T})$	Transformation	$W(\mathbf{T})$	$e(\mathbf{T})$
$\left(\text{argmax}\, G_j(0,1)\right)$	$\mathbf{T}^a = (3.6, 2.3)$	0.2465	0.1234	$\mathbf{T}^a = (3.6, 2.3)$	0.2717	0.1360
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

We set $a_1 = 0.05, a_2 = 0.1$, and $p_g = p_b = 0.5$. The data miner's cost matrix is $\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$.

In Table 2 the two attributes are first set to be independent. Next we let the attributes to be correlated, $\rho_g = 0.6$ and $\rho_b = 0.7$, and set the rest of the parameter values be the same. Table 3 contains the heuristic transformations obtained by maximizing $G_j(0, 1)$ and the equilibrium transformations. Table 3 also has the corresponding adversary gains and Bayes errors. $\mathbf{T}^a = (3.6, 2.3)$, the heuristic transformation, stays the same for both. When the two attributes are independent, the exact equilibrium transformation is $\mathbf{T}^e = (3.7, 2.1)$. When they become correlated, the exact equilibrium transformation is $\mathbf{T}^e = (3.3, 2.2)$. When attributes are correlated, the heuristic solution proposed by Eq. 6.6 is a little further away from the true equilibrium transformation $\mathbf{T}^e$. In both experiments the heuristic solution is in the surrounding area of $\mathbf{T}^e$. The Bayes errors of the heuristic transformations are fairly close to the exact equilibrium Bayes errors.

6.2.2 Experiment two

In this experiment we use the same profit functions given above and set $a_1 = 0.03, a_2 = 0.07$, and $p_g = p_b = 0.5$. The data miner's cost matrix again is $\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$. The means and variances for the bivariate Gaussian distributions are given in Table 4. The Heuristic transformation is $\mathbf{T}^a = (4.1, 0.3)$. The corresponding adversary gain and Bayes error are $W(\mathbf{T}^a) = 0.0673$ and $e(\mathbf{T}^a) = 0.0336$. The exact equilibrium

Table 4 Initial distributions

π_g	π_b
$N\left(\begin{bmatrix}4.8\\1\end{bmatrix}, \begin{bmatrix}1.8^2 & 0\\0 & 0.6^2\end{bmatrix}\right)$	$N\left(\begin{bmatrix}1.2\\5\end{bmatrix}, \begin{bmatrix}0.5^2 & 0\\0 & 0.8^2\end{bmatrix}\right)$

transformation is $\mathbf{T}^e = (7, 0.3)$. The exact equilibrium adversary gain and Bayes error are $W(\mathbf{T}^e) = 0.1124$ and $e(\mathbf{T}^e) = 0.0562$.

In this experiment, the heuristic solution does not perform as well as in the first experiment. However, when we ignore the transformation penalty for the adversary (set $a_1 = a_2 = 0$) and examine the worst case scenario, the worst case Bayes error is 0.1113. It is much larger than the equilibrium Bayes error 0.0562. The maximum Bayes error is produced by the transformation $\mathbf{T} = (0.3, 3.8)$. The worst case scenario suggests that X_2 will be heavily transformed. When considering the transformation penalty, the heuristic solution correctly predicts that X_1 will be heavily transformed. When the transformation penalty for the adversary can be obtained or estimated, we have information about both players in the game. Such information enables us to make more accurate prediction about the adversary's long term plan of action.

6.2.3 Alternative profit function

In several two dimensional experiments, we fixed parameter values, and compare the profit function specified by Eq. 6.7 with the following one:

$$g(\mathbf{T}, \mathbf{x}) = max\left(1 - a_1 \times \left|T_1^{-1}(x_1) - x_1\right| - a_2 \times \left|T_2^{-1}(x_2) - x_2\right|, 0\right). \quad (6.8)$$

We observe similar equilibrium transformations. The details are omitted here. We believe that the profit function defined by Eq. 6.8 is more intuitive: An object can produce up to one unit profit; transformation of each attribute receives linear penalty; and penalties together are deducted from the maximum profit available. For a single transformation $\mathbf{T}$ and one single "bad" object $\mathbf{x}$, Eqs. 6.7 and 6.8 might yield quite different results. Equation 6.8 gives a number smaller than Eq. 6.7 for large transformations. On the other hand, under moderate transformations, a transformed object produces the same amount of profit based on both profit functions. Extreme transformations, where the two functions differ, do not produce much profit in either case. It is not likely for an extreme transformation to be an equilibrium. Since we are interested in the classifier's equilibrium behavior, using Eqs. 6.7 and 6.8 create similar equilibrium states.

Under our framework Eq. 6.7 can be considered as a one-dimensional approximation to the more intuitive profit function defined by Eq. 6.8. With profit function defined by Eq. 6.7 we do not have to employ a full scale stochastic search to discover an equilibrium state in the game.

6.2.4 Complexity reduction

The general solution for equilibrium strategies proposed in Sect. 3 utilizes Monte Carlo integration and simulated annealing algorithm. It can accommodate different types of classifiers. When we focus on the minimal cost Bayesian classifier, the heuristic solution proposed in this section is computationally much less demanding.

The heuristic solution is obtained by maximizing one dimensional gains $G_j(0, 1)$s, which are one-dimensional integrals. We only need to estimate the one-dimension marginal distributions of the attributes in the transformed bad class. The heuristic solution no longer requires the knowledge of the high dimensional joint distribution of the attributes in the transformed bad class.

Evaluating the one-dimensional gains for one transformation $\mathbf{T}$ at a time is still achieved by Monte Carlo integration. However, the sample size required for an accurate estimate of the one-dimensional gains is much smaller than the sample size needed to estimate the high dimensional adversary gain $W(\mathbf{T})$. [3]

Maximizing $G_j(0, 1)$s can often be achieved by a near exhaustive search over the one-dimensional strategy space of transformations. Hence the heuristic solution does not suffer from slow convergence as what simulated annealing experiences in high dimensional space. The simulation studies reported in Sects. 7 and 8 use the heuristic solution and do not employ a full scale stochastic search. Even if maximizing the one-dimensional gains needs a stochastic search, convergence happens much faster than searching a very high dimensional space.

We can improve the heuristic solution by using it as the starting point in a high dimensional stochastic search. In high dimensional space, when we do not have enough samples for an accurate estimate of $W(\mathbf{T})$ by Monte Carlo integration, we can search for a transformation that maximizes a tight lower bound of $W(\mathbf{T})$ as given in Appendix A. A tight lower bound of $W(\mathbf{T})$ is close to the true function $W(\mathbf{T})$. This approach allows us to find a good approximation for $\mathbf{T}^e$. At the same time, because the lower bound of $W(\mathbf{T})$ only involves one-dimensional integrals, we do not encounter the sample size problem in Monte Carlo integration and avoid estimating the high dimensional joint distribution of the transformed bad class.

7 Attribute selection for Bayesian classifier and gaussian mixture

Using the above computationally tractable approach, we examine the equilibria for multiple attribute scenarios. In Sect. 4 we show that a classifier's initial error rate can be much smaller than its equilibrium error rate. Here a simple example shows that a classifier's equilibrium performance is not solely determined by the size of penalty, although penalty is one important factor. For the sake of example, let the cost matrix $c = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ and $p_g = p_b = 0.5$. Among the three attributes as in Table 5, we select the one most effective at equilibrium. With only one attribute in the model, we define

[3] Robert and Casella (2004) discusses how to computationally estimate the sample size required for Monte Carlo integration to converge.

Table 5 Three attributes' equilibrium performance

Attribute	π_g	π_b	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

the profit function as in Eq. 4.1. Let $k = 1$. Equilibrium transformations are obtained by Eq. 4.2. Table 5 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 penalty. All factors interact with each other and produce X_2 as the winner.

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

1. Equilibrium Bayes error e ($\mathbf{T}^e$).
2. 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 enough computation power, one can employ a forward or backward attribute selection algorithm, choosing the next subset with the smallest exact equilibrium Bayes error or expected misclassification cost. However, to search for an equilibrium in every step is very time-consuming. Instead we examine the performance of different subsets based on the heuristic solution.

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

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

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

Hence for all the different subsets of attributes, a "bad" object produces one unit profit when it passes the classifier without transformation. The profit function can be

Table 6 Initial distributions for five attributes

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

considered as one-dimensional approximation to the following function:

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

The distributions for the "good" and "bad" classes and the penalties for transforming each attribute are given in Table 6. As in Sect. 6.2, transformation $\mathbf{T}$ is a diagonal matrix, $\text{diag}(T_1, \ldots, T_d)$. A transformed object is: $\mathbf{T}(\mathbf{x}) = \mathbf{T}\mathbf{x}$. The transformed "bad" class has distribution $N\left(\mathbf{T}\mu_b, \mathbf{T}\Sigma_b\mathbf{T}'\right)$. In Table 6 we set the parameter values such that the simulations can demonstrate how the heuristic performs in very different scenarios. The parameters are set so that some distributions have large variances while the rest have small variances. 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.

We obtain the heuristic transformations for every subset by maximizing the one-dimensional gains, defined by Eq. 6.6. We calculate the Bayes error of the heuristic transformation $e\left(\mathbf{T}^a\right)$.

Table 7 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 $W\left(\mathbf{T}^a\right)$, and the heuristic Bayes error $e\left(\mathbf{T}^a\right)$ for each subset. Experiment results indicate that choosing attributes based on the initial Bayes error is not a good choice. For example, among all the subsets with two attributes, (X_4, X_5) has the smallest initial Bayes error and one of the worst Bayes error at equilibrium (i.e., high $e\left(\mathbf{T}^a\right)$ value). Similar phenomenon is observed for subsets with more attributes.

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

As in a static environment, monitoring more attributes improves the classifier performance under the heuristic transformation in this simulation. X_1 is an interesting attribute. It receives the smallest penalty. Yet transformation on X_1 does not 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, $e\left(T_5^a\right) = 0.3$. X_5 has a small variance in the "bad" class. For Gaussian attributes, variance appears to be a more important factor than penalty. Simply adding more attributes does not necessarily improve the equilibrium performance much. For example (X_1, X_5) and (X_1, X_4, X_5) have very

Table 7 Heuristic equilibrium performance for 31 subsets

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

similar heuristic Bayes error. We notice that using three attributes (X_1, X_2, X_3) is an effective and economical choice.

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 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, data

miner must aggressively change the rules of the game (e.g., identify entirely new ways of authenticating malicious objects) in order to win.

8 Attribute selection for Bayesian classifier and Bernoulli variables

The following simulation study mimics the spam filtering environment. Because a single binary attribute in a spam email can be modified without changing the rest, we simplify the matter and make the Bernoulli attributes themselves independent. We simulate 25 independent Bernoulli random variables, using the probability functions and the transformation defined in Sect. 5.2. In the "good" class π_g let

$$X_j = \begin{cases} 1 & \text{with probability} \quad r_{gj} \\ 0 & \text{with probability} \quad 1 - r_{gj} \end{cases}$$

In the "bad" class π_b the initial probabilities prior to transformation are

$$T_j^{-1}\left(X_j\right) = \begin{cases} 1 & \text{with probability} \quad r_{bj} \\ 0 & \text{with probability} \quad 1 - r_{bj} \end{cases}$$

Transformation is a probability T_j for π_b:

$$X_j = \begin{cases} 1 & \text{with probability} \quad r_{bj} \times \left(1 - T_j\right) \\ 0 & \text{with probability} \quad 1 - r_{bj} + r_{bj} \times T_j \end{cases}$$

We modify the profit function defined by Eqs. 5.1 and 5.2. In the profit function each Bernoulli attribute can have different initial profit and penalty.

$$g_j\left(T_j, X_j\right) = \begin{cases} k_j^{a_j\left(T_j\right)} & \text{if} \quad T_j^{-1}\left(X_j\right) = 1 \quad \text{and} \quad X_j = 0 \\ k_j^{\frac{1+a_j\left(T_j\right)}{2}} & \text{if} \quad T_j^{-1}\left(X_j\right) = 1 \quad \text{and} \quad X_j = 1 \\ k_j & \text{if} \quad T_j^{-1}\left(X_j\right) = X_j = 0 \end{cases}$$

Later in the simulation we let the initial profit for every attribute equal to 3, i.e., $k_j = 3, \forall j$, and allow them to have different penalty a_j.

$$a_j\left(T_j\right) = \begin{cases} a_j \times \left(1 - T_j\right) & \text{if} \quad a_j \geq 0 \\ a_j \times T_j & \text{if} \quad a_j < 0 \end{cases}$$

The total profit for a subset with d attributes is defined as:

$$g(\mathbf{T}, X_1, \ldots, X_d) = \sum_1^d g_j\left(T_j, X_j\right).$$

The Naïve Bayes classifier using d attributes has the following classification region $L\left(h_{\mathbf{T}}, g\right)$ where objects are classified as "good":

$$\left\{\mathbf{X}: \left(\frac{(c(g,b)-c(b,b))p_b}{(c(b,g)-c(g,g))p_g}\right) \times \left(\prod_1^d \frac{\left(r_{bj}\left(1-T_j\right)\right)^{X_j}\left(1-r_{bj}+r_{bj}T_j\right)^{1-X_j}}{\left(r_{gj}\right)^{X_j}\left(1-r_{gj}\right)^{1-X_j}}\right) \leq 1\right\}.$$

When we examine the above classification region carefully, we discover an interesting phenomenon. For every subset there exists a transformation that makes the two classes indistinguishable. Then based on the parameter values of the Naïve Bayes classifier, it either passes all objects or blocks all objects: If $\forall j$, $r_{bj}\left(1-T_j\right) = r_{gj}$, then for all possible outcomes produced by Bernoulli attributes, we have

$$\prod_1^d \frac{\left(r_{bj}\left(1-T_j\right)\right)^{X_j}\left(1-r_{bj}+r_{bj}T_j\right)^{1-X_j}}{\left(r_{gj}\right)^{X_j}\left(1-r_{gj}\right)^{1-X_j}} = 1.$$

For transformation $\mathbf{T}^* = \left(T_j\right)$ such that $\forall j$, $T_j = 1 - \frac{r_{gj}}{r_{bj}}$, the classification rule is to either pass all objects or block all objects, regardless of how many attributes are used to construct the Naïve Bayes classifier. The only factor that determines which extreme classification rule is triggered by $\mathbf{T}^*$ is the constant term $\frac{(c(g,b)-c(b,b))\times p_b}{(c(b,g)-c(g,g))\times p_g}$.

When $\frac{(c(g,b)-c(b,b))\times p_b}{(c(b,g)-c(g,g))\times p_g} \leq 1$, for every subset there exists a transformation which forces the classifier to pass all objects. Under such an extreme classification rule, the Bayes error rate is p_b. Even we consider the impact of penalty for transformation, we expect most subsets to have equilibrium Bayes error p_b regardless of the number of attributes. Depending on the attributes in a subset and their initial probabilities r_{gj} and r_{bj}, some subsets experience heavier transformation to trigger the pass-all classification rule. That does not affect the equilibrium Bayes error. All subsets are equally bad in the long run. There is no attribute selection issue.

When $\frac{(c(g,b)-c(b,b))\times p_b}{(c(b,g)-c(g,g))\times p_g} > 1$, for every subset there exists a transformation which forces the classifier to block all objects. Again depending on the attributes and their initial probabilities r_{gj} and r_{bj}, some subsets experience heavier transformation to trigger the block-all classification rule. From the simulation results in Sect. 5.2, we expect an equilibrium to surface before the block-all classification rule is triggered. The exact location of equilibrium depends on penalties. To prevent equilibrium Bayes error becoming p_b for all $2^q - 1$ subsets of attributes, the data miner has to increase its tolerance for misclassifying a "good" objects, reducing the value of $c(b, g)$. Next we choose a set of parameters such that $\frac{(c(g,b)-c(b,b))\times p_b}{(c(b,g)-c(g,g))\times p_g} > 1$.

8.1 Experiment

We have $c(b, g)/c(g, b) = 3, c(g, g) = c(b, b) = 0$, and $p_b = 0.8$ and $p_g = 0.2$. The "bad" class π_b dominates the overall population and misclassifying a "good" object costs more. With these parameters, there is no transformation to trigger the pass-all classification rule. There is at least one transformation for every subset to

Table 8 Five levels for the parameters

r_b	0.75	0.80	0.85	0.90	0.99
r_g	0.01	0.05	0.10	0.15	0.20
a	−4.0	−3.0	−2.0	−1.0	1.0

Table 9 Initial parameter values for 25 Bernoulli attributes

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9
r_b	0.75	0.80	0.85	0.90	0.99	0.75	0.80	0.85	0.90
r_g	0.01	0.01	0.01	0.01	0.01	0.05	0.05	0.05	0.05
a	−4.0	−3.0	−2.0	−1.0	1.0	−3.0	−2.0	−1.0	1.0
	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{16}	X_{17}	
r_b	0.99	0.75	0.80	0.85	0.90	0.99	0.75	0.80	
r_g	0.05	0.10	0.10	0.10	0.10	0.10	0.15	0.15	
a	−4.0	−2.0	−1.0	1.0	−4.0	−3.0	−1.0	1.0	
	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	
r_b	0.85	0.90	0.99	0.75	0.80	0.85	0.90	0.99	
r_g	0.15	0.15	0.15	0.20	0.20	0.20	0.20	0.20	
a	−4.0	−3.0	−2.0	1.0	−4.0	−3.0	−2.0	−1.0	

force the Naïve Bayes classifier to block all objects. Each of the key parameters has 5 levels, as specified by Table 8. They form the 25 attribute parameters as specified by Table 9.

Given 25 attributes, there are $2^{25} - 1 = 33554431$ possible subsets that can be used for classification. We focus on subsets of 3 attributes, 2300 subsets, and examine how well the heuristic solution performs with Bernoulli variables. In this setup the maximum profit generated by a "bad" object is 9 if it passes the classifier without transformation. The heuristic transformations are obtained by Eq. 6.6.

Out of all 2300 subsets containing 3 attributes, the best 20 subsets have heuristic Bayes errors ranging from 0.0509 to 0.0829. The best 50 subsets have heuristic Bayes errors ranging from 0.0509 to 0.0958. On the other hand, the worst heuristic Bayes error can reach 0.5068.

The best 10 subsets with the heuristic transformations $\mathbf{T}^a$ and the heuristic Bayes errors $e(\mathbf{T}^a)$ are listed in Table 10. The initial probabilities for the best subsets provide good separation. These subsets need transformations $T \gg 0.70$ to trigger the block-all classification rule. Large penalties force the adversary to stop well before the transformations enter the block-all region. They produce small Bayes errors at equilibrium.

The worst 10 subsets with the heuristic transformations $\mathbf{T}^a$ and the heuristic Bayes errors $e(\mathbf{T}^a)$ are listed in Table 11. The initial probabilities for the worst subsets also provide good separation. Unfortunately good initial separation allows much space for transformation. The heuristic transformations for the worst subsets are very close to

Table 10 The ten best subsets

	1	2	3	4
Xs	1, 2, 10	1, 3, 10	1, 6, 10	2, 6, 10
$\mathbf{T}^a$	(0.21, 0.30, 0.25)	(0.21, 0.45, 0.25)	(0.21, 0.27, 0.25)	(0.30, 0.27, 0.25)
$e\left(\mathbf{T}^a\right)$	0.0509	0.0582	0.0592	0.0620
	5	6	7	
Xs	2, 3, 10	2, 10, 14	1, 7, 10	
$\mathbf{T}^a$	(0.30, 0.45, 0.25)	(0.30, 0.25, 0.29)	(0.21, 0.44, 0.25)	
$e\left(\mathbf{T}^a\right)$	0.0624	0.0631	0.0678	
	8	9	10	
Xs	3, 10, 14	2, 7, 10	3, 6, 10	
$\mathbf{T}^a$	(0.45, 0.25, 0.29)	(0.30, 0.44, 0.25)	(0.45, 0.27, 0.25)	
$e\left(\mathbf{T}^a\right)$	0.0697	0.0707	0.0714	

Table 11 The ten worst subsets

	2291	2292	2293	2294
Xs	8, 9, 16	5, 9, 17	8, 9, 12	5, 9, 16
$\mathbf{T}^a$	(0.80, 0.84, 0.69)	(0.85, 0.84, 0.71)	(0.80, 0.84, 0.74)	(0.85, 0.84, 0.69)
$e\left(\mathbf{T}^a\right)$	0.4842	0.4861	0.4871	0.4882
	2295	2296	2297	
Xs	8, 9, 13	4, 8, 9	5, 9, 12	
$\mathbf{T}^a$	(0.80, 0.84, 0.76)	(0.81, 0.80, 0.84)	(0.85, 0.84, 0.74)	
$e\left(\mathbf{T}^a\right)$	0.4893	0.4920	0.4953	
	2298	2299	2300	
Xs	5, 9, 13	4, 5, 9	5, 8, 9	
$\mathbf{T}^a$	(0.85, 0.84, 0.76)	(0.81, 0.85, 0.84)	(0.85, 0.80, 0.84)	
$e\left(\mathbf{T}^a\right)$	0.4956	0.4977	0.5068	

trigger the block-all classification rule. When penalties are not large enough to discourage the adversary, the adversary chooses to transform the attributes heavily but not to trigger the block-all rule.

Using the heuristic equilibrium transformation and the heuristic Bayes error, forward/backward selection algorithms can be employed to search for a subset of attributes with good performance. Instead of performing a stochastic search at every step in forward/backward selection algorithms, we evaluate one heuristic transformation. The total number of heuristic transformations involved in forward/backward selection algorithms is $O\left(q^2\right)$. It is quite affordable.

9 Case study

For our experiments, we use a data set that was retrieved in February 2010 from an online lending company's website (www.lendingclub.com). The company offers qualified applicants small loans ranging from $1000 to $25,000. Since people can lie about various aspects of their application to get loans, this case can be considered as an adversarial learning scenario. Furthermore, according to the company's prospectus, it initially only reliably verifies information obtained from the applicants' credit reports. Later on, the company may ask applicants to provide employment information at random but there is no guarantee such information can be received. Some of the information such as home ownership is never verified. This verification process indicates that from an adversarial classification point of view, different attributes receive different penalties for transformation. Unverified attributes receive zero transformation penalty for the adversary.

Based on an attribute called "Monthly payment status", we are able to identify loans that the borrowers fail to make payments on time, and loans that receive regular monthly payments or have been fully paid off. For our experiments, we do not use the text and date attributes, such as loan description, loan title, screen name of the applicant, application date etc. We use the following attributes from the data set.

1. X_1—Amount requested
2. X_2—Loan purpose
3. X_3—Debt-to-income ratio
4. X_4—Home ownership (any, none, rent, own, mortgage)
5. X_5—Monthly income
6. X_6—FICO range
7. X_7—Open credit lines
8. X_8—Total credit lines
9. X_9—Revolving credit balance
10. X_{10}—Revolving line utilization
11. X_{11}—Inquiries in the last 6 months
12. X_{12}—Accounts now delinquent
13. X_{13}—Delinquent amount
14. X_{14}—Delinquencies in last 2 years
15. X_{15}—Months since last delinquency

The instances in the data set originally are classified into the following categories: (1) "Removed"; (2) "Loan is being issued"; (3) "Late (31–120 days)"; (4) "Late (16–30 days)"; (5) "Issued"; (6) "In review"; (7) "In grace period"; (8) "In funding"; (9) "Fully paid"; (10) "Expired"; (11) "Default"; (12) "Current"; (13) "Charged Off"; (14) "New".

We perform data cleaning and pre-processing to create good and bad classes from the original data set. First we remove all the instances that are labeled as "New", "Loan is being issued", "Issued", "In review", "In funding", "In grace period", "Late (16–30 days)", and "Expired". We remove instances that belong to those categories to ensure that we have unambiguous information about whether the applicants make regular

payments or not. In the original data set the instances from those categories together are less than 10% of the total.

The loans that have been fully paid off and the ones that receive regular payments form the "good" class. The loans that have been charged off or experience late payments for longer length of time (i.e., being late for more than 30 days) form part of the "bad" class.

A large number of the instances in the original data set belong to the "Removed" category, where the loan applications are removed due to variety of reasons: Applicants refuse to provide employment information; or applicants fail to fill up requisite forms etc. It is not clear whether we can directly consider the "Removed" category as part of the "bad" class, since some of the removed instances actually can be from the "good" class but are removed due to technical reasons. On the other hand, deleting this entire category from the data set would significantly reduce the number of "bad" class instances. Instead of deleting the whole category, we modify some attributes of the instances from the "Removed" category to form the remaining portion of the "bad" class. Basically for each instance in "Removed" category, we perform the following modifications.

1. The FICO range (X_6) is decreased by three intervals: If the original FICO range is "780+" for an instance, it is replaced by "713–679".
2. Debt-to-income ratio (X_3) and revolving line utilization (X_{10}) are multiplied by 3.
3. Monthly income (X_5) is reduced to 1/3 of its original value.

After omitting the instances with missing values and performing the above preprocessing steps, we are left with 5850 instances. In the experiment we discretize continuous numeric attributes using ten equi-width buckets: A discretized attribute has 10 interval values where 10% of the overall population ("good" and "bad" combined) falls into each interval. Then we have the initial data set prior to transformation for the experiment.

We set $\pi_g = \pi_b = 0.5$ since the proportions of the two classes are roughly equal. We set $c(g, b) = c(b, g)$ and $c(g, g) = c(b, b) = 0$. The company charges interest rate as high as 21%. The interest earned by the company from a good loan over three year period equals to a significant portion of the principal. Counting the partial payments received from a "bad" loan and the lost interest from a "good" loan, we assume that the cost of rejecting a loan application that the applicant can make regular payments equals to the cost of offering a bad loan that will not be fully paid back. We use Eq. 6.6 to search for the heuristic equilibrium transformation. Under the above setting, the classification region in Eq. 6.6 is equivalent to that of a one dimensional Bayesian classifier. Furthermore, Eq. 6.6 can be re-written as a simple optimization problem for discrete attributes.

Let $p_i = Pr[X = w_i|\pi_b]$ and $q_i = Pr[X = w_i|\pi_g]$ for an attribute with m different discrete values (e.g., ordered intervals in the experiment). Let f_{ij} be the proportion of attribute X that is transformed from interval w_i to interval w_j by the adversary. Let d_{ij} be the penalty of transforming attribute X from w_i to w_j by the adversary. Let y_i equal to $Pr[T(X) = w_i|\pi_b]$. It is the probability *after the adversary performs transformation*. In one dimensional case, we know that an instance will be classified as

"good" if after transformation y_i is less than q_i. We assume that the adversary earns an expected profit u_i when a transformed "bad" object is classified as "good". Using the above notations, we can rewrite our one dimensional heuristic method as the following optimization problem.

$$\max \left[-\left(\sum_{i=1}^{m} \sum_{j=1}^{m} f_{ij} \times d_{ij} \right) + \left(\sum_{i=1}^{m} I_{\{y_i < q_i\}} \times y_i \times u_i \right) \right]$$

subject to

$$f_{ij} \geq 0 \quad i = 1, \ldots, m, \ j = 1, \ldots, m \tag{9.1}$$

$$\sum_{j=1}^{m} f_{ij} = p_i \quad i = 1, \ldots, m \tag{9.2}$$

$$\sum_{i=1}^{m} f_{ij} = y_j \quad j = 1, \ldots, m \tag{9.3}$$

$$\sum_{j=1}^{m} y_j = 1 \tag{9.4}$$

The above optimization formulation states that we need to maximize the objective function that considers both the profit and transformation penalties under various constraints.

Using the above heuristic one dimensional estimates, we conduct an experiment. The above optimization problem is solved using Matlab's optimization toolbox for each dimension. Depending on the number of constraints, running one dimensional heuristic requires three to five minutes on our server with two Intel(R) Xeon(R) quad-core 3.16 GHz cpus. After the heuristic solution is obtained for each dimension, we modify the bad instances using the transformation returned by our one dimensional heuristic. Later on, a Naïve Bayes classifier is built on the transformed data set using Weka toolbox. Each trial is repeated 10 times using 10 fold cross validation. Below we report the average classification accuracy for different scenarios.

From the initial data set, we obtain classification accuracy of 89.95%. Next using our heuristic technique, we explore the worst case scenario where the adversary can transform the bad class without any penalty, $d_{i,j} = 0 \forall i, j$. In all of the one dimensional optimizations, we let all u_i equal to the average loan amount requested \$8978.8. In the worst case scenario, the heuristic transformation returns an equilibrium performance accuracy of 61.52%. Finally, we examine the case where there is a significant penalty for transforming FICO range, revolving credit balance, revolving line utilization, accounts now delinquent, and delinquent amount. We assume that improving each attribute by one level, such as increasing FICO range from "679–713" to "714–740" or reducing revolving line utilization by one level, would cost \$500. Under this penalty, we find a heuristic equilibrium transformation that returns 72.28% accuracy. We would like to stress that we run additional experiments with different penalties,

Table 12 Classification accuracy

Experiment type	Accuracy
Initial data set prior to transformation	89.95%
Worst case scenario	61.25%
Penalized transformation scenario	72.28%
Attribute selection scenario	70.92%

\$600, \$300, and \$400, and still obtain results close to 72.28%. These results indicate that our heuristic solution is not sensitive to mild variations in penalty values.

In addition, we explore the feasibility of using our one dimensional heuristic for attribute selection. Basically, we rank all attributes based on their one dimensional classification accuracy after transformation. Using this rank, we choose the top half of the attributes and build a Naïve Bayes model on those attributes. As shown in Table 12, just using half of the available attributes, we can reach 70.92% accuracy. This indicates that our one dimentional heuristic could be used for efficient attribute selection.

The above results indicate that the online lending company does not have to prepare for the worst case scenario that may force the company to increase the interest rate for every loan and lose potential good borrowers. Due to heavy penalty, the adversary can not afford to transform the attributes to the extreme. The penalties for transformation can be estimated more accurately if a financial model is available. Furthermore, in the heuristic transformation, the probabilities f_{ij} of moving from one level to another, can help the lender to assess the risk of an application.

10 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 are 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 propose simulated annealing to find subgame perfect equilibrium in this paper, other stochastic search techniques can also be used. We consider 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 have done a case study, and experiments on a combination of Bayesian classifier, Gaussian mixture and Bernoulli distributions, and different types of penalty for transformations. 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 detection by modifying its current strategy to mimic the other party, our formulation can be applied to the scenario.

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.

Acknowledgements We thank the reviewers and the editors for their helpful comments that improved the presentation and the content of the article. This work was partially supported by Air Force Office of Scientific Research MURI Grant FA9550 08-1-0265, National Institutes of Health Grant 1R01LM009989, National Science Foundation Grants Career-0845803, DMS-0904548 and CNS-0964350.

Appendix A: Lower and upper bounds of the adversary gain

We first present tight bounds of the classification regions. Write

$$\mathrm{RB}_0 = \prod_{j=1}^{q} \left[y_j \leq 1 \right].$$

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

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

Again RB_1 is a subset of $L\left(h_{\mathbf{T}}, g\right)$. 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

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

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 $\left\{ \prod_{j=1}^{q} y_j = 1 \right\}$ in this fashion up to RB_{q-1}. Write

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

Notice $\forall i \neq j, \mathrm{RB}_i \cap \mathrm{RB}_j = \emptyset$. And $\bigcup_0^{q-1} \mathrm{RB}_j$ offers a tighter lower bound than the one defined by Eq. 6.3. Similarly we construct an improved upper bound for $L(h_{\mathbf{T}}, g)$. Write RA_j, $j = 0, \ldots, q-1$, as follows:

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

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

$$\cdots\cdots,$$

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

$\forall i \neq j, \mathrm{RA}_i \cap \mathrm{RA}_j = \emptyset$. 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:

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

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

Based on the lower and upper bounds of the classification regions in Eqs. A.1 and A.2, we have the lower and upper bounds of the adversary gain. A lower bound of the adversary gain is:

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

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

An upper bound of the adversary gain is:

$$
\begin{aligned}
&W^{\text{upper}}(\mathbf{T})\\
&=1-\left(\prod_{j=1}^{q}P_j(1,\infty)\times\sum_{j=1}^{q}\frac{G_j(1,+\infty)}{P_j(1,\infty)}\right)\\
&-\sum_{j=1}^{q}\left\{\sum_{n=1}^{q}\left[\prod_{k\neq j}P_k\left((n+1)^{\frac{1}{q-1}},\infty\right)\times P_j\left(\frac{1}{n+1},\frac{1}{n}\right)\right.\right.\\
&\left.\left.\times\left(\frac{G_j\left(\frac{1}{n+1},\frac{1}{n}\right)}{P_j\left(\frac{1}{n+1},\frac{1}{n}\right)}+\sum_{k\neq j}\frac{G_k\left((n+1)^{\frac{1}{q-1}},+\infty\right)}{P_k\left((n+1)^{\frac{1}{q-1}},+\infty\right)}\right)\right]\right\}
\end{aligned}
$$

$$
\begin{aligned}
&- \sum_{j_1=1}^{q} \sum_{j_2=j_1+1}^{q} \left\{ \sum_{n_1=1}^{\infty} \sum_{n_2=1}^{\infty} \left[\prod_{k \neq j_1, j_2} P_k \left([(n_1+1)(n_2+1)]^{\frac{1}{q-2}}, \infty \right) \right. \right. \\
&\times \prod_{\ell=1}^{2} P_{j_\ell} \left(\frac{1}{n_\ell + 1}, \frac{1}{n_\ell} \right) \\
&\left. \left. \times \left(\sum_{\ell=1}^{2} \frac{G_{j_\ell} \left(\frac{1}{n_\ell+1}, \frac{1}{n_\ell} \right)}{P_{j_\ell} \left(\frac{1}{n_\ell+1}, \frac{1}{n_\ell} \right)} + \sum_{k \neq j_1, j_2} \frac{G_k \left([(n_1+1)(n_2+1)]^{\frac{1}{q-2}}, \infty \right)}{P_k ([(n_1+1)(n_2+1)]^{\frac{1}{q-2}}, \infty)} \right) \right] \right\} \\
&- \ldots\ldots \\
&- \sum_{j_1=1}^{q} \sum_{j_2=j_1+1}^{q} \cdots \sum_{j_{q-1}=j_{q-2}+1}^{q} \left\{ \sum_{n_1=1}^{\infty} \cdots \sum_{n_{q-1}=1}^{\infty} \left[P_{k;k \neq j_\ell} \left(\prod_{1}^{q-1} (n_\ell + 1), +\infty \right) \right. \right. \\
&\times \prod_{\ell=1}^{q-1} P_{j_\ell} \left(\frac{1}{n_\ell + 1}, \frac{1}{n_\ell} \right) \\
&\left. \left. \times \left(\sum_{\ell=1}^{q-1} \frac{G_{j_\ell} \left(\frac{1}{n_\ell+1}, \frac{1}{n_\ell} \right)}{P_{j_\ell} \left(\frac{1}{n_\ell+1}, \frac{1}{n_\ell} \right)} + \frac{G_{k;k \neq j_\ell} \left(\prod_{\ell=1}^{q-1} (n_\ell + 1), \infty \right)}{P_{k;k \neq j_\ell} \left(\prod_{\ell=1}^{q-1} (n_\ell + 1), \infty \right)} \right) \right] \right\}
\end{aligned}
$$

References

Androutsopoulos I, Magirou EF, Vassilakis DK (2005) A game theoretic model of spam e-mailing. In: Proceedings of the 2nd conference on email and anti-spam, Palo Alto, CA, July 21–July 22, pp 1–8

Basar T, Olsder GJ (1999) Dynamic noncooperative game theory, 2nd edn. Society for Industrial and Applied Mathematics, Philadelphia, PA

Cesa-Bianchi N, Lugosi G (2006) Prediction, learning, and games. Cambridge University Press, Cambridge, United Kingdom

Dalvi N, Domingos P, Mausam, Sanghai S, Verma D (2004) Adversarial classification. In: Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining, New York, NY, August 22–August 25, pp 99–108

Duda R, Hart PE, Stork DG (2001) Pattern classification, 2nd edn. Wiley, Malden, MA

El Ghaoui L, Lanckriet GRG, Natsoulis G (2003) Robust classification with interval data. Tech. Rep. UCB/CSD-03-1279, EECS Department, University of California, Berkeley

Fawcett T, Provost FJ (1997) Adaptive fraud detection. Data Min Knowl Discov 1(3):291–316

Fukunaga K (1990) Introduction to statistical pattern recognition. Academic Press, San Diego, CA

Globerson A, Roweis S (2006) Nightmare at test time: robust learning by feature deletion. In: Proceedings of the 23rd international conference on machine learning, Pittsburgh, PA, June 25–June 29, pp 353–360

Hulten G, Spencer L, Domingos P (2001) Mining time-changing data streams. In: Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining, San Francisco, CA, August 26–August 29, pp 97–106

Lanckriet GR, Ghaoui LE, Bhattacharyya C, Jordan MI (2003) A robust minimax approach to classification. J Mach Learn Res 3:555–582

Lippmann RP, Fried DJ, Graf I, Haines JW, Kendall KR, McClung D, Weber D, Webster SE, Wyschogrod D, Cunningham RK, Zissman MA (2000) Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation. In: Proceedings of the 2000 DARPA information survivability conference and exposition, Hilton Head, South Carolina, January 25–January 27, pp 12–26

Lowd D, Meek C (2005a) Adversarial learning. In: Proceeding of the eleventh ACM SIGKDD international conference on knowledge discovery in data mining, Chicago, IL, August 21–August 24, pp 641–647

Lowd D, Meek C (2005b) Good word attacks on statistical spam filters. In: Proceedings of the 2nd conference on email and anti-spam, Palo Alto, CA, July 21–July 22, pp 1–8

Mahoney MV, Chan PK (2002) Learning nonstationary models of normal network traffic for detecting novel attacks. In: Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining, Edmonton, Alberta, Canada, July 23–July 26, pp 376–385

McKelvey RD, McLennan AM, Turocy TL (2007) Gambit: software tools for game theory, version 0.2007.01.30. http://econweb.tamu.edu/gambit

Mitra D, Romeo F, Sangiovanni-Vincentelli A (1986) Convergence and finite-time behavior of simulated annealing. Adv Appl Probab 18(3):747–771

Osborne MJ, Rubinstein A (1999) A course in game theory, 1st edn. MIT Press, Cambridge, MA

Pu C, Webb S (2006) Observed trends in spam construction techniques: A case study of spam evolution. In: Proceedings of the 3rd conference on email and anti-spam, Mountain View, California, July 27–July 28, pp 1–9

Robert CP, Casella G (2004) Monte carlo statistical methods, 2nd edn. Springer, New York, NY

Stinson E, Mitchell JC (2008) Towards systematic evaluation of the evadability of bot/botnet detection methods. In: Proceedings of the 2nd conference on USENIX workshop on offensive technologies, San Jose, CA, July 28–August 1, pp 1–9

Teo CH, Globerson A, Roweis S, Smola A (2008) Convex learning with invariances. In: Platt J, Koller D, Singer Y, Roweis S (eds) Advances in neural information processing systems, 20. MIT Press, Cambridge, MA pp 1489–1496

Vallee T, Basar T (1999) Off-line computation of stackelberg solutions with the genetic algorithm. J Comput Econ 13(3):201–209