Mach Learn (2010) 81: 69–83
DOI 10.1007/s10994-010-5199-2

Mining adversarial patterns via regularized loss minimization

Wei Liu · Sanjay Chawla

Received: 30 April 2010 / Accepted: 20 June 2010 / Published online: 22 July 2010

Abstract Traditional classification methods assume that the training and the test data arise from the same underlying distribution. However, in several adversarial settings, the test set is deliberately constructed in order to increase the error rates of the classifier. A prominent example is spam email where words are transformed to get around word based features embedded in a spam filter.

In this paper we model the interaction between a data miner and an adversary as a Stackelberg game with convex loss functions. We solve for the Nash equilibrium which is a pair of strategies (classifier weights, data transformations) from which there is no incentive for either the data miner or the adversary to deviate. Experiments on synthetic and real data demonstrate that the Nash equilibrium solution leads to solutions which are more robust to subsequent manipulation of data and also provide interesting insights about both the data miner and the adversary.

Keywords Stackelberg game · Nash equilibrium · Loss minimization

1 Introduction

Conventional supervised learning algorithms build classification models by learning relationships between the independent (features) and dependent (class) variables from a given input data. Typically, the underlying, though often unstated, assumption is that the relationship between the features and the class remain unchanged over time. However, in many real world applications, such as email spam detection systems, there often exist adversaries who are continuously modifying the underlying relationships in order to avoid detection by the

Editors: José L. Balcázar, Francesco Bonchi, Aristides Gionis, and Michèle Sebag.

W. Liu (✉) · S. Chawla
School of Information Technologies, University of Sydney, Sydney, Australia
e-mail: wei.liu@sydney.edu.au

S. Chawla
e-mail: sanjay.chawla@sydney.edu.au

classifier. Therefore, in order to minimize the effects of the adversaries, data miners should not only learn from the data in the past, but also learn from potential data manipulations that adversaries are likely to make in future. As a result, the problem of "adversarial data mining" has attracted significant interest in the machine learning and data mining community (Dalvi and Domingos 2004; Lowd and Meek 2005; Globerson and Roweis 2006; Globerson et al. 2008; Kołcz and Teo 2009; Kantarcioglu et al. 2009; Liu and Chawla 2009).

Dalvi et al. (2004) modeled adversarial scenarios under the assumption that both the adversary and the data miner have perfect information of each other. In their formulation, the adversary is fully aware of the parameter settings of the classifier, and uses the classifier's decision boundary to undermine the classifier. The data miner in turn periodically retrains the classifier based on the adversary's modifications. This "perfect knowledge" is unrealistic in practice. In Lowd and Meek (2005) the perfect knowledge assumption is relaxed by assuming that the adversary has the ability to issue a polynomial number of membership queries to the classifier in the form of data instances which in turn will report their labels. They refer to their approach as Adversarial Classifier Reverse Engineering (ACRE). The ACRE learning quantifies the "hardness" of attacking a (linear) classifier system, but it is unclear how the data miner should respond to the adversary's ACRE learning attacks, and what state the adversary and the data miner will proceed to eventually.

Globerson et al. (2006, 2008) use deletions of features at test time to approximate the strategies of adversaries. Taking their experiment on handwritten digit classification as an example, their algorithm deletes pixels of "heavy weights" from an image of digit "9", so that it can be confused with a normal image of "7". However, a disadvantage of this feature deletion algorithm is that it fails to simulate scenarios where the adversary is more interested in adding features or generally in linear transformation of features. Using the same example, it is not possible to make a "7" look like a "9" by deleting pixels. Furthermore, the work in Globerson and Roweis (2006) and Globerson et al. (2008) are both "one-shot" games (i.e. the game is played only once) and do not give any considerations to the data miners' responses.

Recently, Kantarcioglu et al. (2009) and Liu et al. (2009) have proposed approaches that model the competing behavior between the adversary and the data miner as a sequential Stackelberg game. They use simulated annealing and genetic algorithm respectively to search for a Nash equilibrium as the final state of play. While Kantarcioglu et al. (2009) assume the two players know each other's payoff function, Liu and Chawla (2009) relaxe this assumption and only the adversary's payoff is required in achieving the equilibrium. But a common problem for Kantarcioglu et al. (2009) and Liu and Chawla (2009) is that the strategies of the adversary are *stochastically* sampled (e.g., Monte Carlo integration in Kantarcioglu et al. 2009) and then among the samples the best fit is selected (e.g., genetic algorithm in Liu and Chawla 2009). This *stochastic optimization* process is not realistic for rational adversaries in practice, since rational adversaries rarely make "random" moves, but instead always try to optimize their payoff at each step of play. Another common limitation of Kantarcioglu et al. (2009) and Liu and Chawla (2009) is that they both assume that data was generated from a normal distributions which can potentially restrict their applicability.

In this paper, we propose an approach based on a Stackelberg game model. However, the method we propose to search Nash equilibrium is fundamentally different from those based on stochastic optimization. More specifically, *the contributions* of this paper are as follows:

1. We model the interactions between the adversary and the data miner as a two-player sequential Stackelberg game, where for each player we design a regularized loss function as the payoff;

2. The game is cast as a *maxmin* optimization problem whose solution is the most rational strategy at each play;
3. We propose an algorithm which efficiently solves the maxmin optimization problem without making distributional assumptions on data features;
4. We perform comprehensive empirical evaluations on spam email and handwritten digit data sets which testify the superiority of our approach.

The rest of the papers is as follows. In Sect. 2 we introduce basic game theory concepts and formulate the maxmin problem in general. We define the players' payoff functions from the perspective of a classification problem and present a sequential Stackelberg algorithm to solve the maxmin problem in Sect. 3. Experiments on synthetic and real data sets are reported in Sect. 4. We conclude in Sect. 5 with a summary and directions for future research.

2 Sequential Stackelberg games

In a Stackelberg game, two players are distinguished as a leader (L) and a follower (F), and it is the leader who makes the first move. In our case the adversary is the leader and the data miner is the follower, since it is always the adversary who proactively attacks her[1] opponent. We call an "attack" from the adversary and "defence" from the data miner as plays/moves of the game.

Each player is associated with a set of strategies, U and V for L and F respectively, where a strategy means a choice of moves available to each player. In this paper, strategy spaces U and V are finite dimensional vector spaces. The outcome from a certain combination of strategies of a player is determined by that player's payoff function, J_L and J_F. Rational players aim to maximize their corresponding payoff functions using their strategy sets. So given an observation v the best strategy of L is

$$u^* = \arg\max_{u \in U} J_L(u, v) \tag{1}$$

Similarly, if L's previous move is u, the reaction of F is

$$v^* = \arg\max_{v \in V} J_F(u, v) \tag{2}$$

In the context of supervised learning, a classified data set is conventionally represented by: true positives (*tp*), true negatives (*tn*), false positives (*fp*) and false negatives (*fn*), where $tp + fn$ and $tn + fp$ always have constant sums (i.e. the actual number of positives and negatives respectively). While the data miner's payoff is maximized via accurate classification (i.e. $tp + tn$), the adversary's profit relies on the failure of the classifier (i.e. $n - tp - tn$, where n is the total number of instances). This situation suggests the application of "constant-sum" game: in a "constant-sum" game, two players divide up a fixed amount of profit gains, so that one player's winnings are the other's losses (Dixit and Skeath 1999). This notion of a "constant-sum" forms the basis of deriving each players' payoff functions in Sect. 3.

As each player seeks to achieve as high a payoff as possible in each of their moves, they will arrive in a state of Nash equilibrium when their rational strategies interact: the state

[1] For ease of interpretations, in this paper we call the data miner a male (i.e. "he/his") player, and the adversary a female (i.e. "she/her") player.

of *Nash equilibrium* means that simultaneously each player is using the strategy that is the best response to the strategies of the other player, so that no player can benefit from changing his/her strategy unilaterally (Fudenberg and Tirole 1991). Thus the problem reduces to efficiently determining the state of the Nash equilibrium.

2.1 Formulation of the maxmin problem

In the formulation of our sequential Stackelberg game, a Nash equilibrium is the strategy pair (u^*, v^*) that simultaneously solves the optimization problems in (1) and (2). Because this Stackelberg game is also a "constant-sum" game, we have $J_F = \phi - J_L$, where ϕ is a constant number standing for the total amount of profits in the game. Then (2) can be rewritten as:

$$\begin{aligned} v^* &= \arg\max_{v \in V} \phi - J_L(u, v) \\ &= \arg\max_{v \in V} -J_L(u, v) \\ &= \arg\min_{v \in V} J_L(u, v) \end{aligned} \tag{3}$$

where we ignore the constant number ϕ, and transform the equation in to a minimization problem which removes the negative sign. By combining (3) with (1), we obtain the following maxmin problem:

$$\text{Maxmin:} \quad (u^*, v^*) = \arg\max_{u \in U} J_L\Big(u, \arg\min_{v \in V} J_L(u, v)\Big) \tag{4}$$

The solution to the maxmin problem maximizes the leader's profit under the worst possible move of her opponent. In the next section, we derive the payoff functions J_L and J_F in classification problems and design an algorithm to solve the maxmin problem in (4).

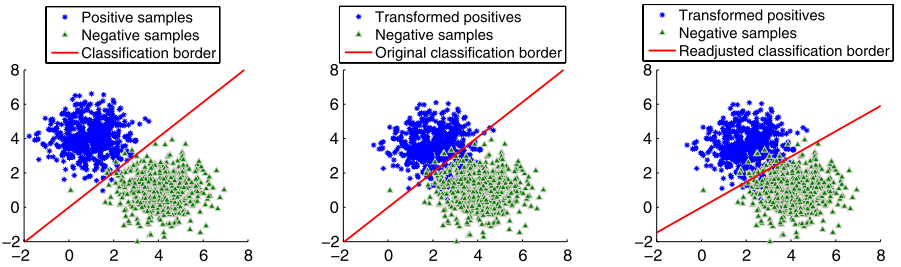
3 The Stackelberg model in adversarial classification

Given a labeled training data $(\mathbf{x}_i, y_i)$ $(i = 1, \ldots, n)$, where $\mathbf{x}_i \in \mathbb{R}^d$ are feature vectors, d is the number of features and $y_i \in \{-1, 1\}$ are binary class labels, many machine learning algorithms build classifiers by minimizing a regularized loss function:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \lambda \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{n} Loss(y_i, \mathbf{w}^T \mathbf{x}_i) \tag{5}$$

where $\mathbf{w}$ is the weight vector including the bias b (and hence $\mathbf{x}$ by default has an added feature $x_{d+1} \equiv 1$), λ and C are positive parameters to balanced the two terms in (5), $\mathbf{w}^T \mathbf{x}_i$ represents the Euclidean dot product of $\mathbf{w}$ and $\mathbf{x}_i$, and $Loss(y_i, \mathbf{w}^T \mathbf{x}_i)$ is the loss function that a data miner minimizes. The regularization term $\mathbf{w}^T \mathbf{w}$ is added so that the classifier has good generalization properties (Lin et al. 2008).

Different settings of loss functions yield different types of classifiers, such as $\ln(1 + e^{-y\mathbf{w}^T\mathbf{x}})$ in logistic regression (Collins et al. 2002) and $\max(0, 1 - y\mathbf{w}^T\mathbf{x})$ as the binary hinge loss in linear support vector machines (SVMs) (Keerthi and DeCoste 2006). One important requirement for valid loss functions is that they must be convex and subdifferentiable, but not necessarily differentiable, as in the hinge loss function.



(a) Initial model state before the game is played.

(b) The adversary moves positive data towards negative samples.

(c) The data miner reacts to the adversary's movement by reconstructing the classifier.

Fig. 1 (Color online) An example of the states of the Stackelberg game model in a classification scenario. The *line* in the middle of the data clouds represents the classification boundary. Based on the initial state of the game (**a**), the adversary moves positive instances towards negative samples and produces more false detections (**b**); the data miner then reacts by shifting the classification boundary (**c**)

3.1 Payoff functions in classification problem

Recall that a player's objective in each of his plays is to ultimately maximize his payoff. Since data miners seek to minimize their loss functions in supervised learning, their payoffs are closely related to their specific loss functions. To this end, we define the payoff function of the data miner as the negative (additive inverse) of their loss function:

$$J_F(\mathbf{w}) = -Loss(y, \mathbf{w}^T\mathbf{x}) \tag{6}$$

Then from (5) we know that the data miner's optimal strategy $\mathbf{w}^*$ on original training data is:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \lambda\mathbf{w}^T\mathbf{w} + C(-J_F) \tag{7}$$

The data miner's optimal strategy on original data constitutes the *initial state* of our game theoretical model when there are no moves made by the adversary. Figure 1a shows an example of such initial state: without malicious modifications on positive samples (blue asterisks), the (solid red) boundary line learned from (7) separates the two classes of data samples and defines the optimal initial classification boundary. Settings of the data samples and loss function used in Fig. 1 are introduced in Sect. 4.1.

Although data miners are able to obtain optimal feature weights from solving (7), these initial feature weights become ineffective when adversaries change their input feature vectors. We assume the adversaries modify feature vectors by introducing a transformation vector α, so that a feature vector $\mathbf{x}_i$ in training is shifted to $\mathbf{x}_i + \alpha$ during the test phase. From the analysis of (3), we obtain the payoff function of the adversary:

$$J_L(\alpha) = -J_F(\mathbf{w}) = Loss(y, \mathbf{w}^T(\mathbf{x} + \alpha)) \tag{8}$$

Now, the further the original positive instances are transformed the higher the cost the adversary has to pay, and when positive instances are transformed to the same as negatives the adversary pays the highest cost, since such positive instances bring no profit to the adversary even if they are undetected by the classifier. Therefore at the same time of maximizing

her payoff, a rational adversary also attempts to minimize the step size of transformations. So we propose that the adversary's movement is determined by the following optimization problem:

$$\begin{aligned}\alpha^* &= \arg\max_{\alpha} -\lambda' \alpha^T \alpha + C' J_L(\alpha) \\ &= \arg\max_{\alpha} -\lambda' \alpha^T \alpha + C' \sum_{i=1}^{n} Loss(y_i, \mathbf{w}^T(\mathbf{x}_i + \alpha)) \end{aligned} \tag{9}$$

where λ' and C' are constant numbers different from λ and C in (5). The *maximization* problem in the function of (9) is *concave* since it is a polynomial of degree 2 with respect to α and the second derivative of this function to α is always negative. The best move for the adversary based on the positive samples of Fig. 1a is show in Fig. 1b: the adversary transforms positive samples towards the direction of negatives; this direction and the length of transformation are acquired from the solution of (9). As in a sequential game, after observing the transformed data, the data miner relearns from his observation of the adversary's movement and rebuilds the classifier as shown in Fig. 1c.

3.2 Solving for the Nash equilibrium

By combining (7) with (9) we obtain the following maxmin optimization problem for the Stackelberg game of adversarial classification:

$$\text{Maxmin:} \quad \max_{\alpha} \min_{\mathbf{w}} -\lambda' \alpha^T \alpha + \lambda \mathbf{w}^T \mathbf{w} + C' \sum_{i=1}^{n} Loss(y_i, \mathbf{w}^T(\mathbf{x}_i + \alpha)) \tag{10}$$

The target function in this maxmini problem is concave with respect to α and convex with respective to $\mathbf{w}$. As explained in Sect. 2.1 by the pair of strategy (u^*, v^*), the strategy pair (α^*, $\mathbf{w}^*$) that solves the maxmin problem of (10) is the optima for the adversary in inflicting maximally high costs on the data miner. It is possible to directly solve (10) under the assumption that each player of the game has the control of future moves of the other player, and hence is able to calculate his/her optimal equilibrium strategy by a "one-shot" game (Globerson and Roweis 2006; Globerson et al. 2008). However, here we relax this unrealistic assumption and use a repeated Stackelberg model, so that each player's next move is based only on the *observation* of his/her opponent's last play. Algorithm 1 lists our method of solving (10). In the interpretations of the algorithm, a line starting with "//" is the comment for its adjacent next line.

The initial state of the Stackelberg model is built via Line 4 of the algorithm, which is the same as the situation of Fig. 1a. After that the adversary iteratively attacks the classifier by her best strategy of transforming the original training data (Line 7), followed by the data miner's reactions that rebuild classifiers though his regularized loss function (Line 9). Note that the readjustments of classifiers are based on the data miner's observations (i.e. $\mathbf{x}_i + \alpha_{tmp}$) of the adversary's modifications on the training data, and the adversary's strategy of play is fully determined by herself. This observation-based algorithm makes the game theoretical model more realistic than a"one-shot" game.

The game is repeated until the adversary's payoff does not increase (Line 23) or the maximum number of iterations is reached (Line 26), and we only consider moves of the adversary are valid when her payoff has nontrivial increases (Line 13 to 20). In addition, the

Algorithm 1 Solving Stackelberg equilibrium

Input: Training data $\mathbf{x}$ and labels y; maximum number of iterations *MaxIter*; adversarial payoff improvement threshold ϵ; loss function of the classifier $Loss(y, \mathbf{w}^T\mathbf{x})$
Output: Stackelberg equilibrium strategy pair $(\alpha^*, \mathbf{w}^*)$

1: Initiate the highest adversarial payoff $HighestPayoff \leftarrow 0$;
2: $Iter \leftarrow 0$;
3: // Build the initial classifier from training data:
4: $\mathbf{w}_{tmp} \leftarrow \arg\min_{\mathbf{w}} \lambda \mathbf{w}^T\mathbf{w} + C \sum_{i=1}^{n} Loss(y_i, \mathbf{w}^T\mathbf{x}_i)$
5: **repeat**
6: // The adversary moves as the *leader of the game*:
7: $\alpha_{tmp} \leftarrow \arg\max_{\alpha} -\lambda' \alpha^T\alpha + C' \sum_{i=1}^{n} Loss(y_i, \mathbf{w}_{tmp}^T(\mathbf{x}_i + \alpha))$
8: // The data miner reacts as the *follower of the game*:
9: $\mathbf{w}_{tmp} \leftarrow \arg\min_{\mathbf{w}} \lambda \mathbf{w}^T\mathbf{w} + C \sum_{i=1}^{n} Loss(y_i, \mathbf{w}^T(\mathbf{x}_i + \alpha_{tmp}))$
10: // Update the adversary's payoff:
11: $J_L \leftarrow \sum_{i=1}^{n} Loss(y_i, \mathbf{w}_{tmp}^T(\mathbf{x}_i + \alpha_{tmp}))$
12: **if** $J_L > HighestPayoff$ **then**
13: **if** $J_L - HighestPayoff > \epsilon$ **then**
14: $HighestPayoff \leftarrow J_L$;
15: $\alpha^* \leftarrow \alpha^* + \alpha_{tmp}$;
16: $\mathbf{w}^* \leftarrow \mathbf{w}_{tmp}$;
17: **else**
18: // Exit current iteration:
19: continue;
20: **end if**
21: **else**
22: // Terminate the loop:
23: break;
24: **end if**
25: $Iter \leftarrow Iter + 1$;
26: **until** $Iter = MaxIter$.
27: Return $(\alpha^*, \mathbf{w}^*)$;

final movement of the adversary is the sum of each previous valid play, since the modification of transformed positives should be calculated from the very original positive instances (Line 15). In Sect. 4, we provide experimental results that suggest Algorithm 1 converges quickly in practice.

3.3 Convex optimization

In this section, we describe how we solve the convex optimization problem in (5) and (9) (Line 4, 7 and 9 of Algorithm 1) via *trust region* methods—a powerful yet simple technique for solving convex optimization problems (Moré and Sorensen 1983; Steihaug 1983; Byrd et al. 1988). The following unconstrained minimization problem is an abstraction of (5) and (9):

$$z^* = \arg\min_{z} f(z) \tag{11}$$

where z are vectors. For solving (5), one can define function f as:

$$f(z) = \lambda z^T z + C \sum_{i=1}^{n} Loss(y_i, z^T \mathbf{x}_i)$$

and for solving (9) function f can be defined as:

$$f(z) = \lambda' z^T z - C' \sum_{i=1}^{n} Loss(y_i, \mathbf{w}^T(\mathbf{x}_i + z))$$

Note that both of the above f functions are convex with respect to z. Suppose we are at the point z_0 of function f, and we want to move to another point with a lower value of f. The main idea of trust region method is to approximate f with a simpler function q, which mirrors the behavior of function f in a neighborhood Ω around the point z_0. This neighborhood is the so-called *trust region* (Moré and Sorensen 1983). Then instead of minimizing f on the unconstrained range as in (11), the trust region method minimizes q in the constrained neighborhood Ω:

$$\begin{aligned} s^* &= \arg\min_s q(s) \\ &\text{subject to} \quad s \in \Omega \end{aligned} \tag{12}$$

and the next point is determined as $z_0 + s^*$ if it has a lower f value. The approximation function q by convention is defined though the second order Taylor expansion of f at z_0, and the neighborhood Ω is usually a spherical or ellipsoidal in shape (Byrd et al. 1988). So the problem in (12) is reduced to:

$$\begin{aligned} s^* &= \arg\min_s \frac{1}{2} s^T H s + s^T g \\ &\text{subject to} \quad \|Ds\| \leq \bigtriangledown \end{aligned} \tag{13}$$

where g and H are the gradient and the Hessian matrix of f, D is a diagonal scaling matrix, and $\bigtriangledown$ is a positive number. The problem in (13) is also known as the *trust region subproblem* (Steihaug 1983). While there many ways to avoid the expensive computation on H, we reuse the straightforward subspace approximation (Branch et al. 2000), which restricts the problem in (13) to a two-dimensional subspace S. In this subspace, the first dimension s_1 is in the direction of the gradient g, and the second dimension s_2 is an approximated Newton direction (i.e. the solution to $H \cdot s_2 = -g$). Within the subspace S, (13) becomes easy and efficient to solve since it's always in a two-dimensional space.

3.4 Feature selection through Nash equilibrium

The weights learned from (5) have been used to suggest the significance of features in training data (Hastie et al. 2001). For example, in a typical logistic regression, the optimal vector of weights are obtained from:

$$\min_{\mathbf{w}} \lambda \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{n} \ln(1 + e^{-y\mathbf{w}^T \mathbf{x}}) \tag{14}$$

Since each feature weight (aka. "regression coefficient") describes to what extent that feature contributes to the class of interest, only the features whose weights have large absolute values are considered important. However, since data is transformed by the adversary, weights learned from original data turn unreliable as time proceeds. Thus when facing an adversary with transformation α on test set, the following optimization problem is more robust to learn the logistic regression coefficient:

$$\begin{aligned} &\min_{\mathbf{w}} \lambda \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{n} \ln(1 + e^{-y\mathbf{w}^T(\mathbf{x}+\alpha)}) \\ &\text{subject to} \quad \alpha = \arg\max J_L(\alpha) \end{aligned} \tag{15}$$

Note that (15) is equivalent to (10) with Lagrange multiplier λ'. And this is why we believe the weights obtained from Nash equilibrium ($\mathbf{w}^*$ returned by Algorithm 1) are better indicators for feature selection. The differences of feature weights learned from (14) and (15) are shown in Sect. 4.3.

4 Experiments and analysis

In this section we report on the outcome of three experiments to test the efficacy of the Stackelberg model introduced above. The experiments are carried out on both synthetic and real data sets.[2] *We shall specifically focus on the performance of classifiers trained on data sets obtained from the Nash equilibrium.* In all the experiments we have set both the Lagrange multipliers λ and λ' to $\frac{1}{2}$, and C and C' to $\frac{1}{n}$. The convergence threshold ϵ is set to 1e–4.

4.1 Rational behavior on synthetic data

We first report on an experiment carried out to ensure that the classifier and the adversary do behave in a *rational* manner under the Stackelberg model. For this experiment we have used SVM's binary hinge loss function and so the maxmin problem in (10) is:

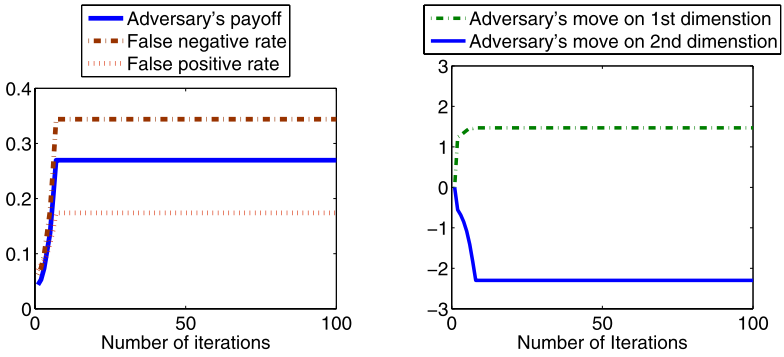
$$\text{Maxmin:} \quad \max_{\alpha} \min_{\mathbf{w}} -\lambda' \alpha^T \alpha + \lambda \mathbf{w}^T \mathbf{w} + C' \sum_{i=1}^{n} \max(0, 1 - y_i \mathbf{w}^T(\mathbf{x}_i + \alpha))$$

We generated a two-dimensional data set using a normal distribution. The positive and the negative class data was generated from a normal distribution with mean $[\mu_1^p, \mu_2^p] = [1, 4]$ and $[\mu_1^n, \mu_2^n] = [4, 1]$ respectively and a common standard deviation I (the identity matrix). We would expect a rational adversary to transform the data so that the positive class elements are displaced towards the negative class and prevent the two classes overlap. Thus we expect that the transformation in the first dimension to be in the range $(0, 3)$ and second dimension in $(-3, 0)$. The data miner on the other hand is re-classifying at each step and moving the separating hyperplane in order to minimize the false positive and false negative rates. Note that the normal distribution has been used only to generate the data and not for the purpose of solving for the Nash equilibrium.

[2] All source code and data sets used in our experiments can be obtained from http://www.cs.usyd.edu.au/~weiliu/.

Table 1 Iterations of Stackelberg model on synthetic data. $\mathbf{w}_0$ in the data miner's reaction is the coefficient of the intercept. The manipulations α_1 and α_2 made by the adversary are always in the range of $(0, 3)$ and $(-3, 0)$ respectively

Iterations	Adversary's move		Modified positives		Data Miner's reaction			Adversary's payoff
	α_1	α_2	μ_1^p	μ_2^p	$\mathbf{w}_0$	$\mathbf{w}_1$	$\mathbf{w}_2$	
0 (initial)	0	0	1	4	0	−0.2962	−0.2955	0
1	1.1729	−0.5356	2.1729	3.4644	0	−0.2710	0.3715	0.0493
2	1.2219	−0.6505	2.2219	3.3495	0	−0.2700	0.3814	0.0508
3	1.2825	−0.8065	2.2825	3.1934	0	−0.2682	0.3943	0.0750
4	1.3533	−1.0221	2.3533	2.9778	0	−0.2646	0.4104	0.0852
5	1.4251	−1.3221	2.4251	2.6778	0	−0.2569	0.4278	0.1142
6	1.4595	−1.7262	2.4595	2.2737	0	−0.2412	0.4356	0.1608
7	1.4096	−2.2063	2.4096	1.7936	0.0347	−0.2265	0.4142	0.2294
8 (final)	1.4143	−2.4282	2.4143	1.5857	0.1322	−0.2287	0.3744	0.2696



(a) The adversary always takes a strategy of play that moves positives closer to negatives, which produces higher false negative rate.

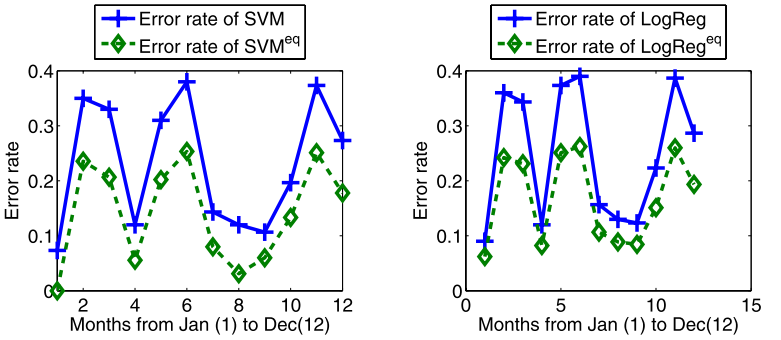
(b) The adversary's movement are always in the range of (0, 3) and (-3 0) in the two data dimensions respectively.

Fig. 2 Detailed information of the iterations of Stackelberg model on synthetic data. The global maxima of adversary's payoff is observed at the 8th iteration

Figure 1a shows the initial data distribution of the two classes. The results of each iteration of the algorithm are shown in Table 1. We can make the following observations: (1) the algorithm converges at the eight iteration with the adversary's payoff increasing from 0 to 0.2696; (2) The transformations made by the adversary (α_1 and α_2) are always in the range $(0, 3)$ and $(-3, 0)$ for each of the two dimensions and we can conclude that the adversary has acted in a rational manner; (3) Finally, as shown in Fig. 2, the adversary's payoff never increases after the eighth observation and the false negative rate is always higher than the false positive rate as the adversary always moves positive instances towards the negative instances. This results in more false detections than false alarms.

4.2 Email spam filtering

We have evaluated the Stackelberg model on a data set consisting of evolving email spam. The objective was to compare the performance of classifiers built on a *normal* training data



(a) Error rate of SVM and SVM^{eq}. (b) Error rate of LogReg and LogReg^{eq}

Fig. 3 Classification performance comparisons of models learned from original data (SVM and LogReg) and equilibrium data (SVM^{eq} and LogReg^{eq})

and on a training data set obtained at equilibrium after the application of the Stackelberg algorithm.

The real data set consists of fifteen months of email obtained from an anonymous individual's mailbox (Delany et al. 2005). The data has 166 thousand unique features and the feature types are words, characters and structure formats. The first three months of data was used for training and the remaining twelve months for testing. We further split the test data into twelve bins—one for each month. Since at any given time spam can be received from diverse sources and spammers have different goals, the intrinsic nature of spam evolves over time. A feature ranking process using information gain was carried out and the top twenty features were selected to build the classifier (Witten and Frank 2002). In the comparisons of classification performance between original data and the data obtained from equilibrium, we use both linear SVMs and logistic regression which is in the form of:

$$\text{Maxmini:} \quad \max_{\alpha} \min_{\mathbf{w}} -\lambda' \alpha^T \alpha + \lambda \mathbf{w}^T \mathbf{w} + C' \sum_{i=1}^{n} \ln(1 + e^{-y_i \mathbf{w}^T (\mathbf{x}_i + \alpha)})$$

The convergence of the maxmin problem above is similar to that of Fig. 2, and the game reaches the Nash equilibrium at the 19th iteration. Here we focus on the performance of the classifiers learned separately from the original data ($\mathbf{x}$) and the equilibrium data ($\mathbf{x}+\alpha^*$). We call the linear SVM and logistic regression model trained on original data as "SVM" and "LogReg"; and those trained on equilibrium data as "SVM^{eq}" and "LogReg^{eq}". Figure 3 shows the error rate of the four classification models on the test set split into twelve months. It can be seen that the test error rates of the classifiers are lower for models based on training on the equilibrium data (SVM^{eq}/LogReg^{eq}) than on the original data (SVM/LogReg). Especially in month 1 and 8 of Fig. 3a, SVM^{eq} has almost zero error rate while SVM has an error rate around 10%. To determine whether the differences between the two sets of classifiers are significant we performed the Friedman test (Demšar 2006) with 95% confidence, under the hypothesis that the error rates from the two types of classifiers are not significantly different. This hypothesis is rejected if the p-value of test is lower than 0.05. Table 2 shows the statistics of the test.

From the low p-values of the Friedman test we can conclude that the error rates from equilibrium classifiers are significantly lower than those of the normal classifiers. The relative success of the equilibrium classifiers vis-a-vis the normal classifiers is due to the a pri-

Table 2 Friedman tests on the improvements of equilibrium classifiers over normal classifiers. A p-value lower than 0.05 rejects with 95% confidence the hypothesis that the sequences of error rates in comparison are not significantly different

Classifiers	Error rate on each month's emails												p-value
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	
SVM	0.073	0.355	0.335	0.122	0.311	0.389	0.143	0.122	0.107	0.197	0.373	0.273	0.0044
SVM^{eq}	0.00	0.236	0.207	0.056	0.202	0.253	0.089	0.031	0.067	0.133	0.251	0.178	
LogReg	0.091	0.364	0.343	0.126	0.373	0.395	0.157	0.133,	0.123,	0.223,	0.387	0.287	0.0044
LogReg^{eq}	0.062	0.242	0.231	0.082	0.251	0.262	0.107	0.089	0.084	0.151	0.267	0.193	

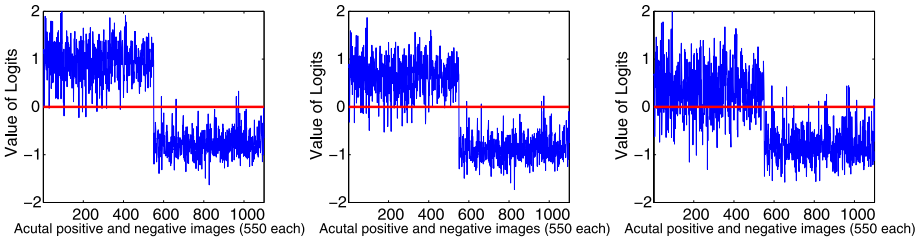
ori prediction of the adversary's possible movements under the Stackelberg model. Since the data miner already takes into account the adversary's future behavior, the knowledge learned from the rational adversarial movements places the equilibrium classifier in an advantage compared with a normal classifier.

4.3 Handwritten digit recognition

In this section we examine the influence of equilibrium feature weights on the problem of feature selection. We use the classic US Postal Service (USPS) dataset which was created for distinguishing handwritten digits on envelopes (Hastie et al. 2001). This dataset consists of gray-scale images of digit "0" through "9" where each image consists of $16 \times 16 = 256$ pixels or features in the classification problem. We assume that the data was generated in an i.i.d. manner and the objective of the data miner is to separate the digits while that of the adversary is to transform an image so that one digit can be confused with another. We use logistic regression in this experiment to examine feature weights as explained in Sect. 3.4.

Each digit has 2200 images, and we divide them equally into training and test set. All combinations of pairs of digits from "0" to "9" are tested and we select the ones whose false positive rates are higher than 0.02 in the initial game state. These are $(2, 6)$, $(2, 8)$, $(3, 8)$, $(4, 1)$, $(5, 8)$, $(7, 9)$. We then apply the Stackelberg algorithm on these six pairs. In this experiment the first digit of a pair is the class of interest for the adversary (i.e. the positive class).

For all the six pairs the Nash equilibrium is reached within 50 iterations of play. Here we ignore the details of reaching the equilibria, since these patterns are similar to what we have analyzed previously. To demonstrate the reliability of feature weights learned from the original data, we check the *logits* (i.e. the z in a logistic function $\frac{1}{1+e^{-z}}$) on predicting test sets with different adversarial transformations. Figure 4 shows an example on the classification of "2" against "6". Note that an instance is classified as negative if its logit is below zero. As we can see from Fig. 4a, in the initial state of the game, most of the actual positives have their logits around the value of 1 (the positive class label) and very few of them are below zero. But as soon as the adversary starts to move (Fig. 4b), the performance of the initial feature weights drops discernibly. When the adversary plays her equilibrium strategy, there are substantially more false detections (Fig. 4c) and some of the logits are even below -1 (the negative class label). This is a situation where equilibrium weights should be applied. We choose several of the undetected transformed positive images and show them in Fig. 5 in comparison with their original images. While the transformations from Fig. 5a to 5b and 5c to 5d is carried out by adding values to pixels and of Fig. 5e to 5f by subtracting values of pixels, the modification on Fig. 5g contains both additions and subtractions for

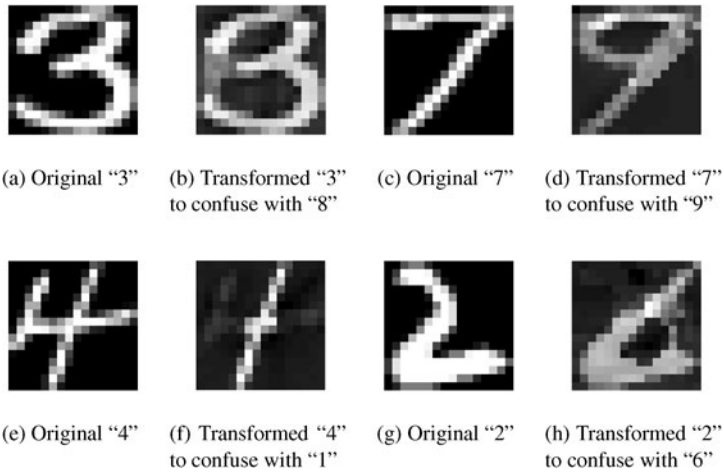


(a) Logits of being positives in initial state

(b) Logits of being positives after the adversary's first move.

(c) Logits of being positives after the adversary's equilibrium move.

Fig. 4 Comparisons of logits targeting on positive class on differently transformed test sets from the experiments of handwritten digit pair (2, 6). Among the 1100 test images, the first 550 are actual positives and the rest 550 are actual negatives. A positive instance is undetected if its logit is below zero



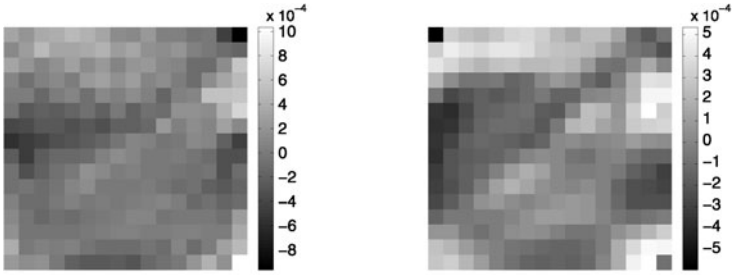
(a) Original "3" (b) Transformed "3" to confuse with "8" (c) Original "7" (d) Transformed "7" to confuse with "9"

(e) Original "4" (f) Transformed "4" to confuse with "1" (g) Original "2" (h) Transformed "2" to confuse with "6"

Fig. 5 Examples of transformed images from Stackelberg Nash equilibrium. To confuse with the digit in negative class, the adversary adds values of pixels on (**a**) and (**c**), removes values from pixels of (**e**), and does both additions and deletions on (**g**)

manipulation. The difference of initial weight $\mathbf{w}^0$ and equilibrium weight $\mathbf{w}^*$ results from training processes where (for example) the former is learned from Fig. 5a, 5c, 5e, and 5g, and the latter from Fig. 5b, 5d, 5f, and 5h. To gain a better understanding of the differences, we discard the intercept weight, and plot the weights of the 16×16 pixels in a graph, as shown in Fig. 6. Due to space limitation, here we only present the weights from the classification of "2" vs. "6" (the weights from other digit pairs have similar observations).

The gray-scale color-bars in Fig. 6 shows the weights of features contributing to the two classes: the pure white pixels contribute the most to the digit "2", the pure black ones contribute most to the digit "6" and the ones whose degrees of color are between pure white and pure black (i.e. weights are close to zero) are features that are less important. The robustness of $\mathbf{w}^*$ can be seen from the difference of Fig. 6a and 6b. For example, $\mathbf{w}^0$ (Fig. 6a) considers the pixel on the very upper right corner as a significant indicator of "6" (as it is pure black), which results in the misclassification of Fig. 5h. However, $\mathbf{w}^*$ (Fig. 6b) treats that pixel as an unimportant feature, and instead gives heavy negative weight to the pixel on the upper



(a) Feature weights from $\mathbf{w}^0$ (b) Feature weights from $\mathbf{w}^*$

Fig. 6 Comparison of feature weights obtained from initial game state $\mathbf{w}^0$ and Nash equilibrium $\mathbf{w}^*$ in the classification of digit pair (2, 6). The color-bars besides the figure shows the relationship between the colors and feature weights. The pure white pixels contribute the most to the digit "2", the pure black ones contribute the most to "6"

left corner which can not be manipulated to confuse with "6". Similar patterns can also be observed from other pixels. These types of characteristics of $\mathbf{w}^*$ make equilibrium weights more reliable in the selection of features in an adversarial setting.

5 Conclusions

In this paper we have studied the classification problem in the presence of adversaries. In this scenario data miners produce classification models and adversaries transform the data to deceive the classifier. We have modeled the interaction of a data miner and an adversary using a sequential non-cooperative Stackelberg model, where the two players compete within a constant-sum game. The payoff function of the adversary is derived from the data miner's regularized loss function, which is formulated into a maxmin optimization problem. We also show that the solution of the maxmin problem is the equilibrium strategy pair where the game achieves highest false negative rate and lowest transformation cost simultaneously.

We have demonstrated that classifiers trained on equilibrium data significantly outperform normal classifiers against adversary's data manipulations. This is due to the fact that the data miner learns not only from the original data, but also from the predictions of the adversary's possible future rational movements. This leads to more robust classification boundaries at test time. We have also illustrated that feature weights obtained from Nash equilibrium are more reliable for the selection of features which are more robust to adversarial manipulation. It is important to identify the features that a rational adversary will most likely modify to deceive the classifier, and this information can be obtained from the equilibrium weight vector.

In future we plan to investigate the use of coalition games to model scenarios in which multiple adversaries exist and collaborate against one data miner.

Acknowledgements The first author of this paper acknowledges the financial support of the Capital Market CRC. This research is partially funded by Australia Research Council Discovery Grant (DP0881537). The authors are also grateful to Lile Li and members of the Pattern Mining Group (University of Sydney) for proofreading the paper.

References

Branch, M., Coleman, T., & Li, Y. (2000). A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM Journal on Scientific Computing*, *21*(1), 1–23.

Byrd, R., Schnabel, R., & Shultz, G. (1988). Approximate solution of the trust region problem by minimization over two-dimensional subspaces. *Mathematical Programming*, *40*(1), 247–263.

Collins, M., Schapire, R., & Singer, Y. (2002). Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, *48*(1), 253–285.

Dalvi, N., Domingos, P. Mausam, Sanghai, S., & Verma, D. (2004). Adversarial classification. In *KDD'04: proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 99–108). New York: ACM.

Delany, S. J., Cunningham, P., Tsymbal, A., & Coyle, L. (2005). A case-based technique for tracking concept drift in spam filtering. *Knowledge-Based Systems*, *18*(4–5), 187–195.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, *7*, 30.

Dixit, A., & Skeath, S. (1999). *Games of strategy*. New York: Norton.

Fudenberg, D., & Tirole, J. (1991). *Game theory* (1st ed.). Cambridge: MIT Press.

Globerson, A., & Roweis, S. (2006). Nightmare at test time: robust learning by feature deletion. In *Proceedings of the 23rd international conference on machine learning* (pp. 353–360). New York: ACM.

Globerson, A., Teo, C. H., Smola, A., & Roweis, S. (2008). An adversarial view of covariate shift and a minimax approach. In *Dataset shift in machine learning*. Cambridge: MIT Press.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). The elements of statistical learning.

Kantarcioglu, M., Xi, B., & Clifton, C. (2009). *Classifier evaluation and attribute selection against active adversaries* (Technical report). Department of Statistics, Purdue University.

Keerthi, S., & DeCoste, D. (2006). A modified finite Newton method for fast solution of large scale linear SVMs. *Journal of Machine Learning Research*, *6*(1), 341.

Kołcz, A., & Teo, C. (2009). Feature weighting for improved classifier robustness. In *CEAS'09: sixth conference on email and anti-spam*.

Lin, C., Weng, R., & Keerthi, S. (2008). Trust region Newton method for logistic regression. *The Journal of Machine Learning Research*, *9*, 627–650.

Liu, W., & Chawla, S. (2009). A game theoretical model for adversarial learning. In *Proceedings of the 2009 IEEE international conference on data mining workshops* (pp. 25–30). Los Alamitos: IEEE Comput. Soc.

Lowd, D., & Meek, C. (2005). Adversarial learning. In *KDD'05: proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining* (pp. 641–647). New York: ACM.

Moré, J., & Sorensen, D. (1983). Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, *4*, 553.

Steihaug, T. (1983). The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, *20*(3), 626–637.

Witten, I., & Frank, E. (2002). Data mining: practical machine learning tools and techniques with Java implementations. *ACM SIGMOD Record*, *31*(1), 76–77.