

# An Efficient Adversarial Learning Strategy for Constructing Robust Classification Boundaries\*

Wei Liu<sup>†§</sup>, Sanjay Chawla<sup>‡</sup>, James Bailey<sup>§</sup>, Christopher Leckie<sup>§</sup>, and Kotagiri Ramamohanarao<sup>§</sup>

<sup>§</sup> Dept of Computing and Information Systems, The University of Melbourne

<sup>‡</sup> School of Information Technologies, The University of Sydney

**Abstract.** Traditional classification methods assume that the training and the test data arise from the same underlying distribution. However in some adversarial settings, the test set can be deliberately constructed in order to increase the error rates of a classifier. A prominent example is email spam where words are transformed to avoid word-based features embedded in a spam filter. Recent research has modeled interactions between a data miner and an adversary as a sequential Stackelberg game, and solved its Nash equilibrium to build classifiers that are more robust to subsequent manipulations on training data sets. However in this paper we argue that the *iterative* algorithm used in the Stackelberg game, which solves an optimization problem at each step of play, is sufficient but not necessary for achieving Nash equilibria in classification problems. Instead, we propose a method that transforms singular vectors of a training data matrix to simulate manipulations by an adversary, and from that perspective a Nash equilibrium can be obtained by solving a novel optimization problem only *once*. We show that compared with the *iterative* algorithm used in recent literature, our *one-step* game significantly reduces computing time while still being able to produce good Nash equilibria results.

## 1 Introduction

Conventional supervised learning algorithms build classification models by learning relationships between independent variables (features) and dependent variables (classes) from given input data. Typically, the often unstated underlying 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 classifier. Therefore, in order to minimize the effects of the adversaries, data miners should not only learn from data in the past, but also from potential data manipulations

---

\* This research was partially funded by Australia Research Council Discovery Grants (DP110102621 and DP0881537).

<sup>†</sup> Correspondence goes to wei.liu@unimelb.edu.au

that adversaries are likely to make in future. As a result, the problem of “adversarial learning” has attracted significant interest in the machine learning and data mining community [1–7].

Dalvi et al. [1] 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. In [2] 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. Globerson et al. [3] use deletions of features at test time to approximate the strategies of adversaries. 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 more generally applying a linear transformation of the features.

In addition, Kantarcioglu et al. [5] and Liu et al. [6] 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 algorithms respectively to search for a Nash equilibrium as the final state of play. While [5] assumes the two players know each other’s payoff function, [6] relaxes this assumption and only the adversary’s payoff is required in achieving the equilibrium. But a common problem for [5] and [6] is that the strategies of the adversary are *stochastically* sampled (e.g., Monte Carlo integration in [5]) and then among the samples the best fit is selected (e.g., genetic algorithm in [6]). 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.

More recently, Liu et al. [7] formulate the adversarial learning problem into a maxmin problem where they relaxed the assumption of a normal distribution and forced adversarial transformations to be rational. Similarly, Bruckner et al. [8] formulate the maxmin problem by a different ordering of the players’ movements. However, to solve these maxmin problems the authors have to use an *iterative* algorithm that *solves a convex optimization problem in every iteration of their algorithms*, which is computationally very expensive. We refer to these existing maxmin optimization based methods (i.e., [7]) the “iterative methods”.

In this study, we assert that simulations of adversarial transformations and solutions of Nash equilibria do not have to be modeled as an expensive iterative optimization process, and it is possible to discover the equilibrium state significantly quicker and cheaper by solving one optimization problem only. In contrast to the existing “iterative method”, we call the method proposed in this paper the “one-step method”. More specifically, the innovations and contributions we make in this paper are as follows:

1. We model malicious manipulations of an adversary as transformations on singular vectors of the training data matrix, and these transformed singular vectors determine distributions of subsequent malicious training samples.

2. We propose a novel payoff function for the adversary, which leads to an optimization problem whose solution achieves the final game state of Nash equilibrium.
3. We perform comprehensive empirical evaluations on spam email and handwritten digit data sets, and demonstrate that our method is able to produce comparative Nash equilibrium results while using significantly less computation time compared to the previous iterative maxmin approach.

## 2 Game Theory and Nash Equilibrium

In this paper we model the interactions between data miners and adversaries in 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<sup>1</sup> 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

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

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

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

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 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 [9]. Thus the problem reduces to efficiently determining the state of the Nash equilibrium.

### 2.1 The Maxmin Problem

In the formulation of the sequential Stackelberg game proposed in [7], a Nash equilibrium is the strategy pair  $(u^*, v^*)$  that simultaneously solves the optimiza-

---

<sup>1</sup> For ease of interpretation, 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.

tion problems in Eq. 1 and 2. Because this Stackelberg game is also a “constant-sum” game, we have  $J_F = \phi - J_L$ , where  $\phi$  is a constant number standing for the total profit in the game. Then Eq. 2 can be rewritten as:

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

where the constant number  $\phi$  is ignored, and the equation is transformed into a minimization problem which removes the negative sign. By combining Eq. 3 with Eq. 1, the following maxmin problem is obtained:

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

The solution to the maxmin problem maximizes the leader’s profit under the worst possible move of her opponent. To solve this maxmin optimization problem, the authors in [7] simulate two players of the data miner and the adversary iteratively, and solve one optimization problem (either the “minimization” or the “maximization”) at a time. They discover Nash equilibrium when the adversary’s payoff stops increasing through the iterating process. Since this iterative algorithm has to solve an optimization problem at every play of the game, it can be computationally expensive to find the final state of Nash equilibrium (which we show in the experiment section).

### 3 One-step Method for Finding the Nash Equilibrium

We derive our efficient one-step equilibrium searching method by utilizing singular value decomposition (SVD) on the training data. Among many kinds of matrix factorization methods, SVD has the property that it gives bases for both the *row* and the *column space* of a matrix simultaneously. It also “orders” the information contained in a matrix so that it is possible to spot the “principle components” of that matrix. Given a  $m \times n$  matrix  $A$ , it can be factorized such that

$$A = U \Sigma V^T$$

where  $U \in \mathbb{R}^{m \times m}$  and  $V \in \mathbb{R}^{n \times n}$  are orthogonal matrices whose columns are (left and right) singular vectors of  $A$ , and  $\Sigma \in \mathbb{R}^{m \times n}$  is a diagonal matrix whose diagonal entries  $\Sigma_{i,i}$  specify singular values of  $A$ .

In binary-class classification problems, given training data of positive instances  $X^+ \in \mathbb{R}^{m^+ \times n}$  and negative instances  $X^- \in \mathbb{R}^{m^- \times n}$ , where  $m^+$ ,  $m^-$  are the numbers of positive and negative instances, and  $n$  is the number of features, the label of a new instance  $x_{new}$  can be determined by using orthogonal basis vectors from SVD as follows.

We compute the SVD of each class of the instance matrix:

$$X^+ = S^+ \Sigma^+ (V^+)^T; \quad X^- = S^- \Sigma^- (V^-)^T.$$

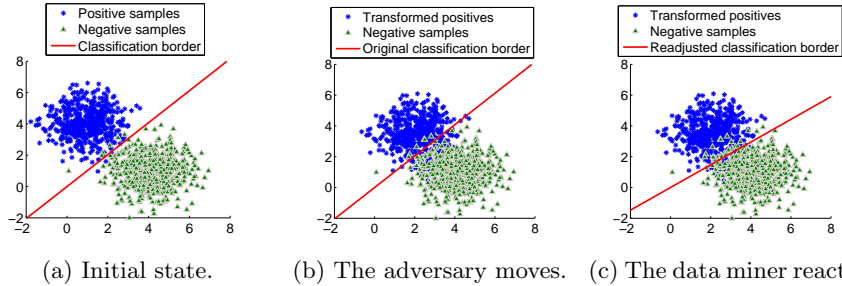


Fig. 1: An example of the interactions between a data miner and an adversary. The line in the middle of the data clouds represents the classification boundary. Based on the initial state of the game (subfig. a), the adversary moves positive instances towards negative samples and produces more false detections (subfig. b); the data miner then reacts by re-constructing (shifting) the classification boundary (subfig. c).

The instances of  $X^+$  and  $X^-$  both span a linear space of  $\mathbb{R}^n$ , so their *right singular vectors* (i.e. column vectors of  $V^+ \in \mathbb{R}^{n \times n}$  and  $V^- \in \mathbb{R}^{n \times n}$ ) form an orthogonal basis of their corresponding type of class.

We characterize each type of class by the first  $k$  singular vectors of that class. If the new class-unknown instance  $x$  can be better represented in the basis of singular vectors of one class (e.g., the positive class) than in those of the opposite class, then  $x$  is more likely to belong to the former class (i.e., positive class). Then the classification process is equivalent to choosing the smaller residual vector generated from the representations of the two classes:

$$\text{Label of } x_{new} = \arg \min_{c \in \{+, -\}} \left\| \sum_{i=1}^k \left( v_i^c \times ((v_i^c)^T \times x_{new}) \right) - x_{new} \right\| \quad (5)$$

where  $v_i$  is the  $i$ th column vector in  $V^c$ ,  $c \in \{+, -\}$ ,  $\|\cdot\|$  is the Euclidean norm, and we assume  $x_{new}$  and  $v_i^c$  are column vectors. This classification strategy forms the fundamental basis of many practical classifiers (e.g., [10,11]). Now we proceed to analyze how manipulations from adversaries can degrade the performance of this SVD classification model.

### 3.1 Formulation of Adversarial Manipulations

The data miner’s classification strategy in Eq. 5 (i.e., on the original data) constitutes the **initial state** of our game theoretical model when there are no moves made by the adversary. Fig. 1a shows an example of such an initial state: without malicious modifications on positive samples (blue asterisks), the (solid red) boundary line learned from Eq. 5 separates the two classes of data samples and defines the optimal initial classification boundary.

Although data miners are able to obtain correct singular vectors from solving Eq. 5, these initial singular vectors become ineffective when adversaries change

the distribution of their input feature vectors. We assume the adversaries modify feature vectors by introducing a transformation vector  $\alpha$ , so that positive instances “ $X^+$ ” in the training phase are shifted to “ $\alpha + X^+$ ” during the test phase. Moreover, in order to decrease the data miner’s classification accuracy, a rational adversary will transfer positive instances<sup>2</sup> in a way that makes its distribution similar to that of the negative class, as shown in Fig. 1b. By denoting

$$\alpha + X^+ = S^\alpha \Sigma^\alpha (V^\alpha)^T$$

as the SVD of the positive instances transformed by an adversary, and  $v_i^\alpha \in V^\alpha$  as the transformed positive singular vectors, the payoff function of an adversary can be stated as

$$J(\alpha) = \sum_{i=1}^k \|v_i^\alpha - v_i^-\|, \quad (6)$$

whose aim is to minimize the difference between the singular vectors of the negative instances and those of the (transformed) positive instances.

However, 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. For example, when the pattern of words in spams is modified such that it is the same to legitimate emails, these spams might not be detected by a spam filter but they also bring no profit at all to the spammer. 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 optimal movement  $\alpha^*$  is determined by the following optimization problem:

$$\begin{aligned} \alpha^* &= \arg \min_{\alpha} J(\alpha) + \lambda \|\alpha\|^2 \\ &= \arg \min_{\alpha} \sum_{i=1}^k \|v_i^\alpha - v_i^-\| + \lambda \|\alpha\|^2 \end{aligned} \quad (7)$$

Eq. 7 is our overall objective function. It reflects that a rational adversary wants to not only minimize the distance between distributions of negative instances and transformed positive instances (i.e., the first term of Eq. 7), but also minimize the transformation itself (i.e., the second term of Eq. 7). In contrast to the iterative method that solves Eq. 4, we aim to find the final equilibrium state of the players by solving one optimization problem, and hence we call the minimization problem in Eq. 7 the “one-step” method.

### 3.2 Solving the Minimization Problem

In this section, we describe how we solve the minimization problem in Eq. 7 via *trust region* methods – a powerful yet simple technique for solving convex

<sup>2</sup> In this paper, we assume it is the positive class which is of value to an adversary. For example, in spam filtering domain, we assume spam emails belong to the positive class, and legitimate emails belong to the negative class.

optimization problems [12]. The following unconstrained minimization problem is an abstraction of Eq. 7:

$$\alpha^* = \arg \min_{\alpha} f(\alpha) \quad (8)$$

Suppose we are at the point  $\alpha_0$  of function  $f$ , and we want to move to another point with a lower value of  $f$ . The main idea of the trust region method is to approximate  $f$  with a simpler function  $q$ , which mirrors the behavior of function  $f$  in a neighborhood  $\Omega$  around the point  $\alpha_0$ . This neighborhood is the so-called *trust region* [13]. Then instead of minimizing  $f$  on the unconstrained range as in Eq. 8, the trust region method minimizes  $q$  in the constrained neighborhood  $\Omega$ :

$$s^* = \arg \min_{s \in \Omega} q(s) \quad (9)$$

and the next point is determined as  $\alpha_0 + s^*$  if it has a lower  $f$  value. The approximation function  $q$  by convention is defined through the second order Taylor expansion of  $f$  at  $\alpha_0$ , and the neighborhood  $\Omega$  is usually a spherical or ellipsoidal in shape [12]. So the problem in Eq. 9 is reduced to:

$$\begin{aligned} s^* = \arg \min_s \quad & \frac{1}{2} s^T H s + s^T g \\ \text{subject to} \quad & \|Ds\| \leq \nabla \end{aligned} \quad (10)$$

where  $g$  and  $H$  are the gradient and the Hessian matrix of  $f$ ,  $D$  is a diagonal scaling matrix, and  $\nabla$  is a positive number. The problem in Eq. 10 is also known as the *trust region sub-problem*. While there are many ways to avoid the expensive computation on  $H$ , we reuse the straightforward subspace approximation [14], which restricts the problem in Eq. 10 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  $S$ , Eq. 10 becomes easy and efficient to solve since it is always in a two-dimensional space.

## 4 Experiments and Analysis

We focus on comparing the difference between the equilibrium states generated from our *one-step* model, and the ones from the *iterative* process proposed in [7], in terms of their efficiency and the accuracy of the classifiers at equilibrium. Our experiments are carried out on a real email spam data set and a handwritten digit data set. To balance the two terms in our objective function (Eq. 7), we set  $\lambda$  to the number of singular vectors (i.e.,  $k$ ) used in each experiment.

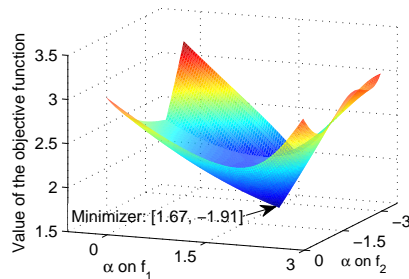
### 4.1 Rational Behavior on Synthetic Data

We first carry out experiments on synthetic data, and examine whether the data miner and the adversary do behave in a *rational* manner under our one-step game model. We generated positive and negative class data from multivariate normal distributions with mean  $[\mu_1^p, \mu_2^p] = [4, 1]$  and  $[\mu_1^n, \mu_2^n] = [1, 4]$  respectively, and a

common standard deviation  $I$  (the identity matrix). Note that the multivariate normal distribution has been used only to generate the data and not for the purpose of solving for the Nash equilibrium. We expect a rational adversary to transform the data so that the positive class elements are displaced towards the negative class, and at the same time to try to prevent the two classes from completely overlapping. Thus we expect that the transformation in the first dimension ( $f_1$ ) to be in the range of  $(-3, 0)$  and second dimension ( $f_2$ ) in the range  $(0, 3)$ .

We put the synthetic data into our objective function with  $k = 2$  (the data has only two features/dimensions), and obtain equilibrium transformation  $\alpha^* = [1.67, -1.91]$ . In Fig. 2 we show the value of the objective function with respect to different settings of the transformation  $\alpha$  on  $f_1$  and  $f_2$ . As we can visually inspect from the figure, the obtained equilibrium transformation  $\alpha^*$  is an effective minimizer of the objective function. This visualization of the objective function also confirms our expectations that a rational adversary would transform the first dimension by a value in the range  $(-3, 0)$  and second dimension in  $(0, 3)$ .

Fig. 2: Values of the adversary’s objective function (Eq. 7), with respect to different setting of transformations ( $\alpha$ ) on the two-dimension synthetic data. The arrow in the figure points to a minimizer of the objective function, generated by our one-step method.



## 4.2 Email Spam Filtering

The objective of this experiment is to compare the performance of classifiers built on a normal training data set and on a training data set obtained at equilibrium after the application of an adversary’s final optimal manipulation.

The spam data set consists of fifteen months of emails obtained from an anonymous individual’s mailbox [15]. 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. The data has 166,000 unique features. A feature ranking process using information gain was carried out and we selected the top 20 features to build the classifier.

To test the influence of  $k$  in terms of classification, we control  $k$  and vary it from 1 through 20, in testing all the twelve months’ test data with 5-fold cross validation. With spam emails belonging to the positive class, the true positive rate (TPR), true negative rate (TNR) and overall accuracy is illustrated in



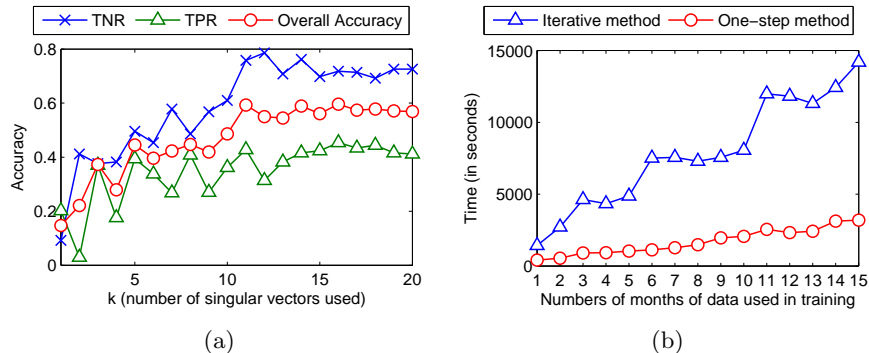


Fig. 3: Subfig.(a): Accuracy of SVD classifications on all test data with varying  $k$ . Subfig.(b): Comparisons on time (in seconds) between iterative processes and our one-step processes in finding Nash equilibria. To compare these two methods, in (b) we used varying amounts of training data indicated on the  $x$ -axis. We find that the one-step process substantially reduces computation time in searching for Nash equilibria, especially when the total amount of data is large.

Table 1: Comparisons on classification accuracy on spam data between iterative processes and our one-step processes, by using the data at Nash equilibria. Although the one-step method needs much less time compared against the iterative method, it is still capable of producing comparable equilibrium classification results.

Months	M1	M2	M3	Months	M1	M2	M3	Months	M1	M2	M3
Jan	.7686	.9179	.9195	May	.6803	.7655	.7439	Sep	.7648	.8277	.8187
Feb	.6256	.7258	.6817	Jun	.9249	.9671	.9882	Oct	.5169	.6046	.6490
Mar	.9071	1	.9991	Jul	.9384	1	1	Nov	.5915	.6597	.6946
Apr	.9150	.9892	.9534	Aug	.5527	.6032	.6795	Dec	.7950	.9323	.9146
Friedman test (M1 vs. M2): $5 \times 10^{-4}$											
Friedman test (M1 vs. M3): $8 \times 10^{-5}$											
Friedman test (M2 vs. M3): 0.7630											

Figure 3a. The overall accuracy stops increasing at around 16, which means the training data can be well represented by the first 16 singular vectors. We choose 16 as the value of  $k$  in our following experiments on spam emails.

We first test the difference on time used in achieving Nash equilibrium between the previous iterative method and our one-step method. Since there are fifteen months of data available to us, we test the total time used in training one month, two months, ..., until all fifteen months. The total running time is shown in Figure 3b, where we can see that the one-step process has substantial savings in computation time on searching for Nash equilibrium, especially when the size of data set is large.

We then compared the one-step method with the iterative method on classification accuracy, where we used the first three months of data for training and the other twelve for test. We perform Friedman tests on the classification accuracy across all data sets, where  $p$ -values that are lower than 0.05 reject the

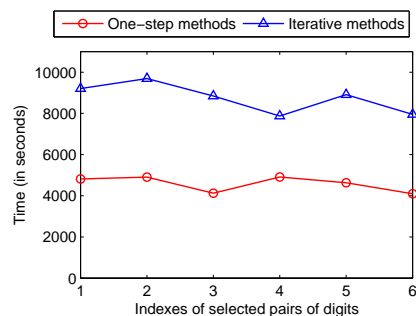
hypothesis with 95% confidence that the classifiers in the comparison are not statistically different. The Friedman test is reported as the most appropriate method for validating multiple classifiers among multiple data sets [16]. In Table 1, “M1” uses the original data (i.e.,  $X^+$  and  $X^-$ ), “M2” uses the equilibrium data generated by the iterative method (i.e.,  $X^+ + \alpha^*$  and  $X^-$ , where  $\alpha^*$  is from the iterative method [7]), and “M3” uses the equilibrium data generated by our one-step method (i.e.,  $X^+ + \alpha^*$  and  $X^-$ , where  $\alpha^*$  is from solving Eq. 7). As shown by the Friedman tests in Table 1, although one-step processes save a great amount of time in finding the Nash equilibrium, it can still generate comparable classification results compared against the iterative method (i.e., not significantly different under the Friedman test).

### 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) data set which was created for distinguishing handwritten digits on envelopes [17]. This data set 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 independent and identically distributed, 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.

Each digit has 2200 images, and we divide them equally into training and test sets. 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. The selected pairs are (2,6), (2,8), (3,8), (4,1), (5,8), (7,9), where we use the first digit of a pair as the class of interest for the adversary (i.e., the positive instances that the adversary manipulates).

Fig. 4: Comparisons on time (in seconds) between iterative processes and our one-step processes in finding Nash equilibria. Numbers on the  $x$ -axis represent indexes of selected digits pairs in the order of (2,6), (2,8), (3,8), (4,1), (5,8), (7,9). Similar to experiments on spam emails, here we also can see that the one-step process used much less computing time compared against the iterative method.



We first compare the total amount of time (in seconds) used for finding Nash equilibrium of these digit pairs, as shown in Figure 4. As expected, we can see from the figure that the one-step process used much less computation time compared to the iterative method in each of the six pairs of digits. More importantly,

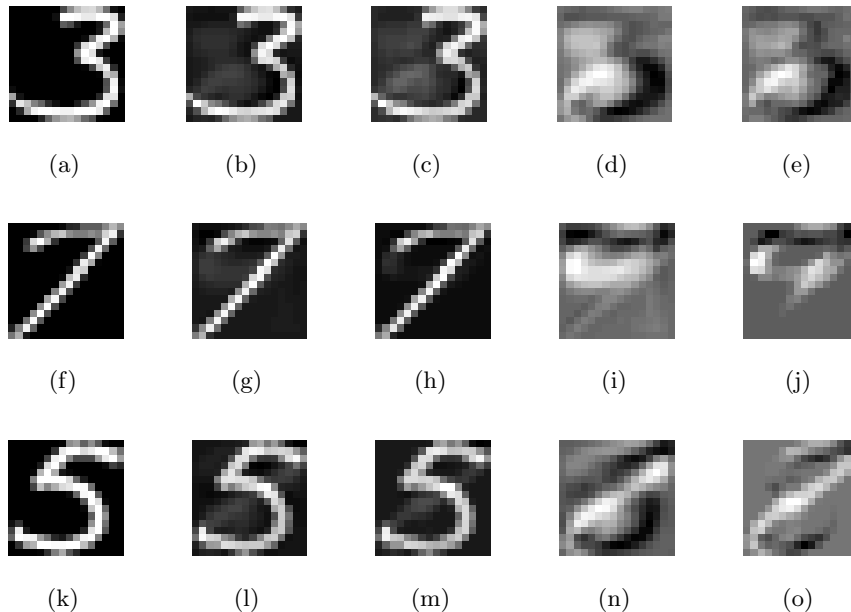


Fig. 5: Some example comparisons of transformed images in digit pair “3” vs. “8”, “7” vs. “9”, and “5” vs. “8”, with the first digit being positive class. Subfig. (a), (f) and (k) show the original digit “3”, “7” and “5”; (b), (g) and (l) show these digits transformed by the iterative method; (c), (h) and (m) are these digits transformed by our one-step method; (d), (i) and (n) are the equilibrium transformations generated by the iterative method; (e), (j) and (o) are the equilibrium transformations generated by our one-step method. Although our one-step method require much less computational time, it can still produce comparable equilibria results compared with the iterative method.

although the one-step method requires much less time, Nash equilibrium results obtained from the one-step method can still reasonably approximate adversarial manipulations on positive training data. As shown by some examples of our digit pairs in Figure 5, the adversary simulated by our one-step method increases or decreases values on some specific pixels on the positive images (i.e., digits “3”, “7” and “5”), so that they look more like negative images (i.e., digits “8”, “9” and “8”) after the equilibrium play. Moreover, the close similarities of the final equilibrium adversarial transformations between the interactive method and the one-step method further confirm that the one-step model can perform comparably to iterative models in terms of finding correct Nash equilibria.

## 5 Conclusions and future research

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 one-step game theoretical model, where the adversary aims to minimize both the difference between distributions of positive and negative classes and the adversarial movement itself. The solution to the minimization problem unveils the state of Nash equilibrium in the interactions between the data miner and the adversary. We have also demonstrated that our one-shot game significantly reduces computation time compared with iterative process used in the previous literature, while it can still generate comparable results in searching for Nash equilibria.

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

## References

1. Dalvi, N., Domingos, P., Mausam, Sanghai, S., Verma, D.: Adversarial classification. In: Proc of KDD 2004. (2004) 99–108
2. Lowd, D., Meeck, C.: Adversarial learning. In: KDD 2005. (2005) 641–647
3. Globerson, A., Roweis, S.: Nightmare at test time: robust learning by feature deletion. In: Proc of ICML 2006. (2006) 353–360
4. Kolcz, A., Teo, C.: Feature weighting for improved classifier robustness. In: CEAS’09: Sixth Conference on Email and Anti-Spam. (2009)
5. Kantarcioglu, M., Xi, B., Clifton, C.: Classifier evaluation and attribute selection against active adversaries. *Data Min. Knowl. Discov.* **22**(1) (2011) 291–335
6. Liu, W., Chawla, S.: A game theoretical model for adversarial learning. In: Proceedings of the 2009 IEEE International Conference on Data Mining Workshops. (2009) 25–30
7. Liu, W., Chawla, S.: Mining Adversarial Patterns via Regularized Loss Minimization. *Machine Learning* **81**(1) (2010) 69–83
8. Brückner, M., Scheffer, T.: Stackelberg games for adversarial prediction problems. In: Proc of KDD 2011. 547–555
9. Fudenberg, D., Tirole, J.: *Game Theory*. 1st edn. The MIT Press (1991)
10. Fortuna, J., Capson, D.: Improved support vector classification using PCA and ICA feature space modification. *Pattern Recognition* **37**(6) (2004) 1117–1129
11. Selvan, S., Ramakrishnan, S.: SVD-based modeling for image texture classification using wavelet transformation. *IEEE Transactions on Image Processing* **16**(11) (2007) 2688–2696
12. Byrd, R., Schnabel, R., Shultz, G.: Approximate solution of the trust region problem by minimization over two-dimensional subspaces. *Mathematical programming* **40**(1) (1988) 247–263
13. Moré, J., Sorensen, D.: Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing* **4** (1983) 553
14. Branch, M., Coleman, T., Li, Y.: A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM Journal on Scientific Computing* **21**(1) (2000) 1–23
15. Delany, S.J., Cunningham, P., Tsymbal, A., Coyle, L.: Tracking concept drift in spam filtering. *Knowledge-Based Systems* **18**(4–5) (2005) 187–195
16. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* **7** (2006) 1–30
17. Hastie, T., Tibshirani, R., Friedman, J.: *The elements of statistical learning*. (2001)