Sparse Models for Adversarial Learning

Submitted for Blind Review

ABSTRACT

As the use of prediction methods becomes more widespread in applications the chances of adversarial manipulation becomes more likely. The canonical example is email and web spam where there is a constant tussle. We propose classification models which are robust against data manipulation by adversaries. For example, spammers are constantly manipulating data to breach spam filters by either reverse engineering the feature set used in the classifier or carrying out randomized attacks against the classifier.

Unlike previous game theoretic models, we define the adversary's behaviour based on the assumption that at a given time a rational adversary can only manipulate a limited number of features. We name this behaviour of the adversary as 'sparse feature attack'. Further we show that classifiers should use a sparse feature weight to fight against an adversary's sparse feature attack. More realistically, we model the interaction between a classifier and an adversary as a *repeated game*. Experiments on benchmark data sets show that the classifier learnt from the game outperforms a corresponding standard model. To the best of our knowledge, this paper is the first attempt to use sparse modelling techniques for adversarial learning.

Categories and Subject Descriptors

H.2.8 [**Data Mining**]

General Terms

Adversarial classification

Keywords

Game theory, ℓ_1 regularizer, Laplace priors.

1. INTRODUCTION

The classification problem is perhaps the most intensively studied problem in machine learning. A fundamental assumption underlying most classification problems is that the training and the test data are generated from the same underlying probability distribution. This assumption is crucial for proving theoretical accuracy bounds regarding the performance of the classifier. However there are at least two scenarios where the assumption does not hold in practice.

- Concept Drift: In many scenarios, data naturally involves. For example, suppose a credit card scoring model was built during "good" economic times. Then it is natural to expect that the performance of the classifier is likely to deteriorate during a recession.
- Adversarial Reaction: In some situations there is a natural adversarial reaction to the classification. For example, spam filters (which are classifiers) routinely have to be updated as an "adversary" tries to circumvent the classifier by modifying spam templates.

The focus of this paper is on adversarial learning, to be specific, classification by taking the existence of the adversary into account. Dalvi et al. [4] have modelled the interaction between a data miner and an adversary as a game between two cost sensitive players. The authors made an assumption that both adversary and data miner have full information of each other. This perfect information model is unrealistic in a web setting. Lowd et al. [10] relaxed the perfect information assumption and devised an approach known as adversarial classifier reverse engineering (ACRE) to study the possible attacks the adversary may carry out. While this framework can help a learner identify its vulnerability, no classifier was proposed that was robust against manipulation.

Globerson et al. [7] formalized the interaction between the two players as a minimax game, in which both players know the strategy space of each other. They made the assumption that the effect of the adversary will be deletions of features at application time. This feature deletion assumption, however, fails to capture the scenarios where the adversary is capable of arbitrarily changing the features.

Liu et al. [9] formulated the interaction between a data miner and an adversary as a zero-sum sequential Stackelberg game, where the adversary is the leader and the data miner is the follower. However, the zero-sum game is unrealistic when the adversary is not entirely antagonistic towards the classifier. For example, in the case of spam email, a classifier's loss is not necessarily the spammer's gain. We denote this model as SSG (sequential Stackelberg game).

Brückner et al. [3] also modelled the adversarial learning scenario as a Stackelberg game between two players. However, the leader role is played by the data miner and the authors assume the payoff of the two players while in conflict, are not entirely antagonistic. Unlike Liu et al. [9], they made the assumption that the adversary can manipulate both positive and negative instances. However, the final formulation of the game is an example of bi-level optimization problem. Thus, the solution obtained is not guaranteed

to attain a global optimal. We denote this model as SG (Stackelberg game). Moreover, both Liu et al. [9] and Brückner et al. [3] made the unstated (but unrealistic) assumption that the adversary has the ability to change all the features i.e., 'dense feature attack'.

Recently, Zhou et al. [16] devised an model based on support vector machines that can tackle two kinds of attacks an adversary may carry out. However, the effectiveness are verified on artificially manipulated test data instead of real world adversarially involved data.

Xu et al. [15] find that solving lasso is equivalent to solving a robust regression problem. This robust property of lasso itself exibits the merits of sparse modelling technique in the presence of potential adversaries.

In this paper, we make the following *contributions*:

- We derive a new model which formulates the interactions between data miner and the adversary as a *repeated game*.
- We propose algorithm and regularized loss functions so that the game is casted into two convex optimization problems.
- We investigate the use and robustness of the ℓ_0 and ℓ_1 regularizer (both for the data miner and the adversary) to create sparse models.
- We conduct experiments on real email spam data sets which testified the superiority of the sparse models.

The outline of this paper is as follows. Section 2 introduce the problem. We begin with an introduction of the formulation of a repeated game and preliminary in Section 3. In Section 4, we elaborate on the repeated game and the approach to solve the game. Finally, in Section 5 we conduct all the experiments together with analysis. Section 6 is conclusions and future work.

2. PROBLEM DEFINITION

Given a sample of data $(\mathbf{x}_i, y_i)_{i=1}^n$ the standard classification problem is defined as

$$\mathbf{w}^* = \arg\min_w \sum_{i=1}^{n} \ell(y_i, \mathbf{w^t x_i}) + \lambda_w \|w\|_p$$

Now, we bring in an adversary, who controls a vector $\mathbf{a}$ with which it modfies the data $\mathbf{x}$. However, it is important to note that the adversary changes the spam data (which is $y = 1$). In order to formalize the problem we separate the data into positive and negative part. Assume the positive data is indexed as $(\mathbf{x}_i, 1)_{i=1}^{npos}$ and the negative data is indexed as $(\mathbf{x}_i, -1)_{i=npos+1}^{n}$

The classifier still aims to find the optimal $\mathbf{w}$ give by $\mathbf{w}^*$

$$\arg\min_w \sum_{i=1}^{npos} \ell(1, \mathbf{w^t}(\mathbf{x_i} + \mathbf{a}^*)) + \sum_{i=npos+1}^{n} \ell(-1, \mathbf{w^t x_i}) + \lambda_w \|w\|_p$$

subject to the constraint that $\mathbf{a}^*$ is given by

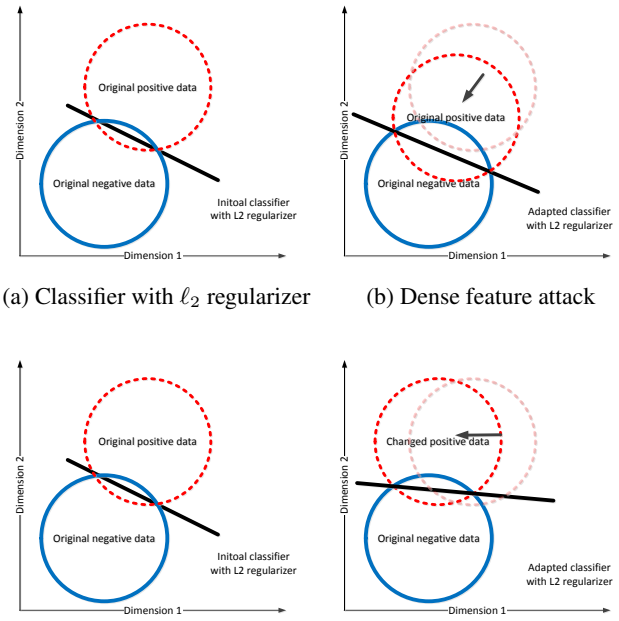
$$\arg\min_a \sum_{i=1}^{npos} \ell(-1, \mathbf{w^t}(\mathbf{x_i} + \mathbf{a})) + \lambda_a \|a\|_p$$

$\ell(y_i, \mathbf{w^t x_i})$ can be any of the three loss functions given in Table 1;

A number of existing literature tried to solve a similar simultaneous game and investigated varies loss functions. However, there are at least four defects that prevail in all these models. One problem is that they consider both players will conduct dense strategy i.e., all the values in $\mathbf{w}$ and $\mathbf{a}$ are non-zeros which is practically inappropriate for real world players. The second one is that the parameter $\lambda_\mathbf{a}$ is mostly arbitrarily decided. thirdly, real players in adversarial environment rarely play a simultaneous game. Last but not least, the resulting problem is 'NP' hard to solve and thus only local optimal is obtained under certain relaxations.

	$\ell(y_i, \mathbf{w^t x_i})$
Square	$\frac{1}{2}\|y_i - \mathbf{w^t x_i}\|_2$
Logistic	$\log(1 + \exp(-y_i \mathbf{w^t x_i}))$
Hinge	$(1 - y_i \mathbf{w}^T \mathbf{w^t x_i})_+$

Table 1: Commonly used loss functions for the two players.



(a) Classifier with ℓ_2 regularizer (b) Dense feature attack

(c) Classifier with ℓ_2 regularizer (d) Sparse feature attack

Figure 1: The figures describe adversarial classification problems in two dimensional space. Circle represents a group of data belonging to the the same category, the straight lines represent the classification boundary. Figures 1a and 1b reflect the adversary is applying dense feature attack. In this case, the classifier moves the boundary backwards in parallel and suffers large classification loss. Figures 1c and 1d shows the adversary is applying sparse feature attack. Noticeably, positive data can still be distinctive to negative data and at the same time, breach the classification boundary. Classifier adapt itself by changing the slop of the boundary and does not suffer as much.

Let we first illustrate the rational behind our assumption that an adversary should apply sparse feature attack. Consider the two scenarios shown in Figure 1 which demonstrate the classical three-step life-cycle for two classification games in two dimensional space: (**1**) The data miner (as the spam filter) uses an acquired labelled data set to build a classifier. Figure 1a and Figure 1c depict classification boundary with ℓ_2 regularizer. (**2**) An adversary (as spammer), over time, inspects the classification boundary and deliberately transforms the positive data (i.e., spam email) to cross the decision boundary. In Figure 1b dense feature attack can be con-

sidered as the worst case assumption that the spammer can manipulate the whole feature spaces and carry out the maximum damage, which is practically impossible. In other worlds, adversary suffer great cost by changing all the features and losing its advertising utility because of becoming more like non-spam emails. However, in Figure 1d sparse feature attack implies the adversary can only transform positive data either vertically or horizontally. This is reasonable since in the case of spam email filtering, spammer can only manipulate a limited numbers of features. More importantly, by applying sparse feature attack, the direct objective of the spammer is to circumvent the classifier instead of making spam more like non-spams. (**3**) Data miner responds by rebuilding the classifier. We should notice under dense feature attack, classifier moves its boundary backwards in parallel to adapt the new situation and thus suffers great loss in classification accuracy. However, under sparse feature attack, the classifier adapt itself by changing the slop of the boundary and keeps the classification accuracy almost status quo. In a true adversarial environment with high dimensionality, the three step game given here can be played repeatedly in time. Thus, the game between the two players can be naturally simulated as a *repeated game*. We claim when both players are utilizing sparse models, a more robust adapted classifier can be achieved through the game.

3. PRELIMINARY

In this section we first propose the concept of a repeated game, then we introduce the two main regularization technique in the case of logistic loss. finally we report on the connection between lasso and robust regression.

3.1 Repeated game

The simultaneous game proposed in the literature has limited practical use as it is impossible to decide how much an adversary will affect the data distribution. Also, the resulting problem is NP hard. Therefore, the theoretical Nash Equilibrium not necessarily leads to a better feature weights $\mathbf{w}$ for the data miner [2]. To model the problem in a more practical way, we should relax the assumptions of the simultaneous game.

The model of a *repeated game* in the case of adversarial classification can be described as below:

Data miner chooses strategy $\mathbf{w}_0$ based on the observed samples drawn from the sample space by minimizing its loss function:
$\mathbf{w}_0 = arg\min_{\mathbf{w}} \sum_{i=1}^{n} \ell(y_i, \mathbf{w^t x_i}) + \lambda_{\mathbf{w}} \|\mathbf{w}\|_p$.
In the following steps, $i = 1, ... + \infty$;

1. Adversary chooses strategy $\mathbf{a}_i$, which is the manipulation vector, with the knowledge of data miner's strategy
   $\mathbf{w}_{i-1}$, i.e., $\mathbf{a}_i = arg\min_{\mathbf{a}} \sum_{i=1}^{npos} \ell(-1, \mathbf{w^t}(\mathbf{x_i} + \mathbf{a})) + \lambda_{\mathbf{a}} \|\mathbf{a}\|_p$.
   The manipulation is then applied to the sample space and $\sum_{i=1}^{npos} \mathbf{x}_i^* = \sum_{i}^{npos} \mathbf{x}_i + \mathbf{a}$.
2. Data miner chooses strategy $\mathbf{w}_i$ based on the samples drawn from the manipulated sample space.
   $\mathbf{w}_i = arg\min_{\mathbf{w}} \sum_{i=1}^{n} \ell(y_i, \mathbf{w^t x_i^*}) + \lambda_{\mathbf{w}} \|\mathbf{w}\|_p$.

In real world, the game can keep playing repeatedly. Thus, we also developed a mechanism to stop the game at a proper time which will be elaborated in Section 4.1.

The goal of the data miner is to determine a decision boundary based on continuously manipulated training data in each step. For the adversary, the goal is to determine a manipulating vector based on a given budget in each step. What we are interested is an adapted classifier, i.e., feature weights $\mathbf{w}_i$, which has the potential of being more robust on future adversarially influenced data set. For example, if we play the game " Rock, Paper, Scissor " in a repeated fashion, the player who plays last always wins.

With properly defined loss functions, adversary's behaviour in real world can be captured and thus a more robust feature weights can be obtained through this simulated game. The challenge here is to find the properly defined regularized loss functions for both data miner and adversary.

3.2 Gaussian prior

Here we illustrate the ℓ_2 regularizer from the perspective of the Bayesian inference with Logistic loss function. Logistic loss function is defined as:

$$L(\mathbf{w}) = \sum_{i}^{n} log(1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i})$$

Where $y_i \in \{-1, 1\}$ is the binary class label, $\mathbf{x}_i (i = 1, ..., n)$ is the feature vectors and $\mathbf{w} \in \mathbb{R}^{d+1}$ is the feature weights, d is the number of features. ℓ_2 norm was originally added to prevent overfitting. The mathematical explanation was given later. In the view of Bayesian inference, the logistic loss function is the negative log of likelihood $P(\mathbf{x}|\mathbf{w})$. By adding the ℓ_2-norm, we are effectively assuming each elements w_j of $\mathbf{w}$ is generated from a Gaussian distribution with hyper-parameter $\mathcal{N}(0, \tau_j)$ [6]

$$p(w_j \mid \tau_j) = \frac{1}{\sqrt{2\pi\tau_j}} \exp(\frac{-w_j^2}{2\tau_j}), \;\; j = 1, ..., d. \qquad (1)$$

Now we minimize the negative log of likelihood with a prior distribution, that is $-\log P(\mathbf{x}|\mathbf{w})P(\mathbf{w})$. This is essentially the maximum a posterior estimation. The effect of the prior $P(\mathbf{w})$ is reflected as penalizing large absolute values of w_j. The maximum a posterior estimation in this case can be written as:

$$\prod_{i}^{n} \frac{1}{1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i}} \prod_{j}^{d+1} \frac{1}{\sqrt{2\pi\tau_j}} \exp(\frac{-w_j^2}{2\tau_j}),$$

By assume τ_j is a constant τ for all j, it is known [6] that:

$$L(\mathbf{w}) = \sum_{i}^{n} log(1 + e^{-y_i \mathbf{w}^t \mathbf{x}_i}) + \sum_{j}^{d+1} \frac{w_j^2}{2\tau} + (d+1)(\ln\sqrt{\tau} + \frac{\ln 2\pi}{2})$$

The last part of the above equation is a constant which can be dropped and resulting in the final equation:

$$L(\mathbf{w}) = \sum_{i}^{n} log(1 + e^{-y_i \mathbf{w}^t \mathbf{x}_i}) + \frac{1}{2\tau} \|\mathbf{w}\|_2^2$$

This regularizer will penalize and constrain the squared sum of vector $\mathbf{w}$, however, it does not favour w_j being exactly zero.

In many applications, a feature vector with a lot zeros in it (i.e., a sparse solution) is preferable in terms of both computation and memory efficiency. To achieve this, we have to assume another prior distribution over w_j.

3.3 Laplace prior

Similar to ℓ_2 norm, for ℓ_1 norm we first assume each w_j comes from a Gaussian distribution with hyper-parameter $\mathcal{N}(0, \tau_j)$. However, similarity ends here, we further assume each τ_j is generated from an exponential distribution with parameter γ_j

$$p(\tau_{j|\gamma_j}) = \frac{\gamma_j}{2} \exp(-\frac{\gamma_j}{2}\tau_j), \quad \gamma_j > 0.$$

Combining Equation (1) and the above equation we get Laplace distribution:

$$p(w_j|\gamma_j) = \frac{\sqrt{\gamma_j}}{2}\exp(-\sqrt{\gamma_j}|w_j|),$$

Effectively, this means we assume w_j follow a Laplace distribution. Again we assume each γ_j equals to γ and the posterior in this case will be:

$$\prod_i^n \frac{1}{1+e^{-y_i\mathbf{w}^T\mathbf{x}_i}} \prod_j^{d+1} \frac{\sqrt{\gamma}}{2}\exp(-\sqrt{\gamma}|w_j|),$$

The negative log of the above formulation will be:

$$L(\mathbf{w}) = \sum_i^n log(1+e^{-y_i\mathbf{w}^T\mathbf{x}_i}) + \sum_j^{d+1}\sqrt{\gamma}|w_j| + (d+1)(\ln 2 + \ln\sqrt{\gamma})$$

Again, we omit the last part and get the final equation:

$$L(\mathbf{w}) = \sum_i^n log(1+e^{-y_i\mathbf{w}^T\mathbf{x}_i}) + \sqrt{\gamma}\|\mathbf{w}\|_1$$

This ℓ_1 regularized logistic loss optimization problem can be solved by transform its lower bound optimization problem into linear programming [14].

3.4 Lasso and robust regression

A learning algorithm is robust if the model it generates is resistant to bounded perturbations in the data. Robust learning algorithms is an active area of research and the robust linear regression problem is defined as

$$\min_{w\in R^d}\{\max_{|z|\le\lambda}\|y-(x+z)w\|_2\} \quad (2)$$

The key insight about robust regression as defined in [15] can be derived from considering the one-dimensional case. For example,

$$\max_{|z|\le\lambda}|y-(x+z)w| \quad \le |y-xw|+c|w|$$

Now consider a specific $z^* = -\lambda sgn(w)sgn(y-xw)$. Clearly $|z^*|\le\lambda$. Furthemore,

$$\begin{aligned}\max_{|z|\le\lambda}|y-(x+z)w| &\ge |y-(x+z^*)w| \\ &= |y-xw|+|\lambda sgn(w)sgn(y-xw)w| \\ &= |y-xw|+\lambda|w|\end{aligned}$$

Thus

$$\max_{|z|\le\lambda}|y-(x+z)w| = |(y-xw)|+\lambda|w|$$

This generalizes to

$$\min_{w\in R^d}\{\max_{|z|\le\lambda}\|y-(\mathbf{x}+\mathbf{z})\mathbf{w}\|_2\} = \min_{w\in R^d}\|y-\mathbf{x}\mathbf{w}\|_2+\lambda\|\mathbf{w}\|_1$$

On the other way around, from the work by Huan et al. [15] we know that solving the lasso problem is equivalent as solving the problem

$$\min_{w\in\mathbb{R}^d}\{\max_{\triangle\mathbf{x}\in\mu}\|y-(\mathbf{x}+\triangle\mathbf{x})\mathbf{w}\|_2\},$$

where $\triangle\mathbf{x}\in\mu$ is the worst case disturbance of noise and μ has

$$\mu \triangleq \{(\delta_1,...,\delta_d)|\|\delta_i\|_2\le\lambda_i, i=1,...,d\}.$$

This means solving a ℓ_1 regularized least square problem is equivalent to solving a worst case linear square problem with noise $\triangle\mathbf{x}\in\mu$. This intrinsic property of ℓ_1 regularization indicates that it will outperform ℓ_2 regularization when the data set is noisy. More importantly, this robust regression equivalence provides us a way for setting a reasonable budget for adversary.

4. ADVERSARIAL CLASSIFICATION AS A REPEATED GAME

In this section, we first elaborate on the game with logistic loss function and present the formal algorithm. Then we report on the intuition behind the assumption that for both players, sparse strategy should be applied.

4.1 Repeated game with logistic loss

We present the repeated game model here in detail. The data miner first builds the initial classifier on the original training data $D = \{\mathbf{x}_i, y_i\}_{i=1}^n$ using standard logistic loss function: $\log(1+e^{-y_i\mathbf{w}^T\mathbf{x}_i})$. Here we arrange the training data so that the first p data points are positive data (spam email), and the rest $n-p$ data are negative data (ham email). The original classifier is obtained through the following minimization problem:

Initial classifier

$$\mathbf{w} = \arg\min_{\mathbf{w}}\frac{1}{n}\sum_{i=1}^n log(1+e^{-y_i\mathbf{w}^T\mathbf{x}_i})+\lambda_{\mathbf{w}}\|\mathbf{w}\|_p \quad (3)$$

Regularizer $\|\mathbf{w}\|_p$ can be ℓ_1 or ℓ_2 norm. $\lambda_{\mathbf{w}}\ge 0$ is the regularization parameter. After the initial classifier is established, in the near future the feature weights $\mathbf{w}$ determined by the data miner can be discovered by the adversary. The adversary will then attempt to change the positive data distribution with a vector $\mathbf{a}$. The initial classifier built on the original training data will be obsolete. Loss function for adversary is defined as: $\log(1+e^{\mathbf{w}^T(\mathbf{x}_i+\mathbf{a})})$ [3]. This logistic loss function measures the loss of a data point being classified as negative. Adversarial square loss and hinge loss can be derived with the same logic. For hinge loss of adversarial it measure the loss of the true positive points being classified as a negative. Here we assume adversary can only manipulate positive data which is based on the intuition that spam emails are more like non-spam emails in the future, while, non-spam emails rarely change. A rather desirable property of this loss function is its convexity, which is a necessary condition for us to apply the efficient convex solver. The manipulation vector $\mathbf{a}$ of adversary is decided by minimizing the loss of positive data being classified as negative data:

Adversary Attack

$$\mathbf{a} = \arg\min_{\mathbf{a}}\frac{1}{npos}\sum_{i=1}^{npos} log(1+e^{\mathbf{w}^T(\mathbf{x}_i+\mathbf{a})})+\lambda_{\mathbf{a}}\|\mathbf{a}\|_p \quad (4)$$

$npos$ is the number of positive data, $\lambda_{\mathbf{a}}$ is the parameter which controls the scale of a_j . The adversarial transformation is made by adding positive data points with adversarial change $\mathbf{a}$. In the next step, data miner defends itself by retraining the classifier based on the transformed training data $D^* = \{(\mathbf{x}_1^* = \mathbf{x}_1+\mathbf{a}, y_1), ..., (\mathbf{x}_{npos}^* = \mathbf{x}_{npos}+\mathbf{a}, y_{npos}), (\mathbf{x}_{npos+1}, y_{npos+1}), ..., (\mathbf{x}_n, y_n)\}$. Notice that the data here is neither the future data nor the original training data. It is the simulated future data generated by the adversary based on the original training data D. The following equation describes the play of the data miner:

Data Miner defend

$$\mathbf{w} = \arg\min_{\mathbf{w}}\frac{1}{n}\sum_{i=1}^n log(1+e^{-y_i\mathbf{w}^T\mathbf{x}_i^*})+\lambda_{\mathbf{w}}\|\mathbf{w}\|_p \quad (5)$$

The three steps played by the two players correspond with the two examples depicted in Figure 1. Then adversary may attack again

Algorithm 1 Repeated Game

Input: Training data $D = \{\mathbf{x}_i, y_i\}_{i=1}^n$, upper bound of the maximum cost $MC = \frac{\lambda_{\mathbf{w}}^* d}{\sqrt{n}}$, Norm $\|\mathbf{w}\|_p$, $\|\mathbf{a}\|_p$
Output: all **w** generated

// Build the initial classifier using original training data:
$\mathbf{w} = \arg\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^n log(1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i}) + \lambda_{\mathbf{w}} \|\mathbf{w}\|_p$
$Cost \leftarrow 0, \mathbf{a}_{Sum} \leftarrow 0, D^* = D$
while $Cost <= MC$ **do**
  // First step: Adversary attack
  $\mathbf{a} = \arg\min_{\mathbf{a}} \frac{1}{npos} \sum_{i=1}^{npos} log(1 + e^{\mathbf{w}^T(\mathbf{x}_i + \mathbf{a})}) + \lambda_{\mathbf{a}} \|\mathbf{a}\|_p$
  for positive data : $\sum_{i=1}^{npos} \mathbf{x} = \sum_{1}^{npos} \mathbf{x} + \mathbf{a}_{Sum}$
  // Second step: Data miner defend
  $\mathbf{w} = \arg\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^n log(1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i}) + \lambda_{\mathbf{w}} \|\mathbf{w}\|_p$
  //Calculate accumulated cost
  $Cost+ = \|\mathbf{a}\|_1$
end while
return **w** generated

and the game goes to infinite. The pseudo-code of the repeated interaction is described in Algorithm 1. We let the game terminate when the accumulated change applied by adversary reaches a predefined threshold MC. The value of MC in this algorithm is defined as upper bound of the real $Cost$ which is the accumulated ℓ_1 norm of the manipulating vectors $\mathbf{a}_i$. Formally present it as:

$$MC \geq Cost = \sum_{i=1}^{r} \|\mathbf{a}_i\|_1$$

where r is the number of repetitions of the game. From Section 3.4 we know that by adding ℓ_1 regularizer, we are practically assuming there is an adversary that is adding nose to both positive and negative data to maximize the loss of the classifier with respect to any classification boundary. Parameter $\lambda_{\mathbf{w}}$ controls how much the noise the adversary can apply. One can observe that the problem is solved at $\|z^*\|_2 = \lambda_{\mathbf{w}}$. If the parameter $\lambda_{\mathbf{w}}^*$ is obtained by ten-fold cross validation, then it practically decides the amount of noise that is necessary to counter the over-fitting phenomenon. While for our repeated game model, adversary is also trying indirectly increase the loss of the classifier, however not as aggressive as the one in the robust regression problem since we only assume that an adversary can only manipulate positive data. Thus we expect the the budget for our adversary should be at least as the same as the amount of nose that is applied by the adversary in robust regression problem.

Formally, we first assume the noise of one particular feature is the same to all the data points, thus we have

$$\begin{aligned} z\sqrt{n} &= \lambda_{\mathbf{w}}^* \\ z &= \frac{\lambda_{\mathbf{w}}^*}{\sqrt{n}} \end{aligned}$$

Then we assume the noise to each feature is the same i.e., $\lambda = \lambda_i$

$$\begin{aligned} MC &= \|\mathbf{z}\|_1 \\ &= \frac{\lambda_{\mathbf{w}}^* d}{\sqrt{n}} \\ &\geq Cost \end{aligned}$$

Notice that the $\lambda_{\mathbf{w}}^*$ here is always the value tuned with Lasso.

4.2 Evaluation of regularizer

As we illustrated in Section 1, both the two players should apply ℓ_0 norm. However, as the resulting optimization problem will be NP hard to solve, we should apply ℓ_1 norm as an approximation. We then compare the performance of the model resulting from ℓ_1 or ℓ_2 regularizer.

It is obvious that we have four variants by combining the two players' loss function with different regularizers. $\|\mathbf{w}\|_p$ and $\|\mathbf{a}\|_p$ can be ℓ_1 or ℓ_2-norm. In the case of $\|\mathbf{w}\|_p = \|\mathbf{w}\|_2$ and $\|\mathbf{a}\|_p = \|\mathbf{a}\|_2$, we denote this model as $Game - L2_d L2_a$. Similarly we have denote the other three model as $Game - L2_d L1_a$, $Game - L1_d L2_a$ and $Game - L1_d L1_a$. We also denote initial classifier with ℓ_2 and ℓ_1 regularizer as Initial-L2-classifier and Initial-L1-classifier respectively. Experiments of the repeated game are reported in Section 5.4;

Here we report on the behavioural difference between each player with different regularizers. Specifically, we justify why the behaviour of the two players resulted from ℓ_1 regularizer should be the most reasonable combination.

4.3 Why sparse feature attack for adversary?

We begin by reveal the inappropriate assumption that adversary will simply minimize a loss function with or with out an ℓ_2 regularizer. Since this assumption indicates that adversary will apply dense feature attack as in [9, 3], it implies every single feature of the sample space will be tampered. This becomes impractical, for example, in the case of spam email; formidably costs will be brought to the spammer for changing various details of the spam template. Also, this dense feature attack behaviour also indicate that the spam template will be more resemble a non-spam email. The consequence of this fact is that the advertising utility of the spam email will decrease dramatically. Common practice tells us that a rational spammer can only concentrate on changing a limited number of features of the spam template in a short period. Thus a sparse feature attack is more realistic as we assert. By assuming the adversary will apply sparse feature attack, practically it indicates that the spam-template will be engineered in the aim of cross the classification, and in the meantime, be distinctive to non-spam emails. This sparse feature attack can be exactly modeled by combining its loss function with ℓ_0 regularizer and solved by using ℓ_1 regularizer, which also gives sparse formulations of the manipulation vector **a**. In a simple case, as depicted in Figure 1d, adversary can only move the positive data leftward.

As one can speculate, if we let the game keep playing with no ending, it will reach a state where the positive and negative data almost overlap. However we can imagine the classifier learned from such a game will have the most undesirable performance. Still we conduct this experiment to examine under how much cost will the adversary achieve such a 'Equilibrium'. We expect the adversary with sparse feature attack achieve the 'Equilibrium' with less cost as the one with dense feature attack. The results are reported in section 5.2

4.4 Why ℓ_1 regularizer for data miner?

As reported in section 3.4, the lasso problem is equivalent to a robust regression problem. While regularizer has always been considered as a method to penalize the weight value to achieve a better generalization property, Here we find that ℓ_1 regularizer can not only be as a technique to prevent overfilling, but also make a more robust classification boundary. This itself indicate that classifier learnt with an ℓ_1 regularizer is more robust in the presence of certain data manipulations. Therefore, in the case of an adversarial environment, a normal classifier with $\ell 1$ regularizer is preferred.

Existing literatures [12, 1] have indicated that in a classical classification environment, when ℓ_1 regularizer is applied, there will always be a trade-off between sparsity and accuracy of the classifier acquired. However, depending on the data set itself, the overall

Algorithm 2 Robustness evaluation under sparse feature attack

Input: Original data set $\{\mathbf{x}_i, y_i\}_{i=1}^n$, Feature weights $\mathbf{w} \in \mathbb{R}^{d+1}$, Number of features to be changed $\{c \in \mathbb{N} | (0 < c \leq d)\}$. Attack strength $\{\delta \in \mathbb{R} | (0 < \delta < 1)\}$
Output: Changed data set $\{\mathbf{x}_i^*, y_i\}_{i=1}^n$

// Randomly select c features of the data and index in vector $\mathbf{a} = \{0 < a_k \leq d\}_{k=1}^c$
for $i = 1; i \leq p; i++$ **do**
  $\mathbf{x}_i^* \leftarrow \mathbf{x}_i, k \leftarrow 1$
  while $k <= c$ **do**
    if $w_{a_k} > 0$ **then**
      $x_{a_k}^* = x_{a_k}^*(1-\delta)$
    else
      if $w_k == 0$ **then**
        $do\ nothing$
      else
        if $w_k < 0$ **then**
          $x_{a_k}^* = x_{a_k}^*(1+\delta)$
        end if
      end if
    end if
    $k = k + 1$
  end while
end for
evaluate $\mathbf{w}$ on changed data set $\{\mathbf{x}_i^*, y_i\}_{i=1}^n$

performance of an ℓ_1 regularized classifier can sometimes beat an ℓ_2 regularized classifier [11] in terms of both bias and variance. The discussion of that is beyond the scope of this research. Here, we compare the performance of the two classifier under the condition that the distribution of the test data is altered by an adversary.

To investigate the influence of an adversary, we start by looking at how the loss function can be influenced by the adversary:

$$L(\mathbf{w}) = \sum_i^n log(1 + e^{-y_i < \mathbf{w}^T, \mathbf{x}_i + \mathbf{a} >}) + \lambda_{\mathbf{w}} \|\mathbf{w}\|_p$$
$$= \sum_i^n log(1 + e^{-y_i < \mathbf{w}^T, \mathbf{x}_i > + < \mathbf{w}^T, \mathbf{a} >}) + \lambda_{\mathbf{w}} \|\mathbf{w}\|_p$$

As we can see from this equation, the adversarial influence is generated in only the factor $< \mathbf{w}^T, \mathbf{a} >$. This is the dot product of the feature weight vector and manipulation vector, which can be described in detail as: $w_1 a_1 + w_2 a_2 + ... + w_d a_d$. One should notice that when more than one of the two vectors are dense vectors, $< \mathbf{w}^T, \mathbf{a} >$ will always be non-zero. In other worlds, only if both the two vectors are sparse vectors, $< \mathbf{w}^T, \mathbf{a} >$ will have a high probability of being zero. When this factor is zero, the influence of adversary also disappears. Thus, we can conclude that when adversary is with sparse feature attack, data miner should apply ℓ_1 regularization. This analysis can be easily generalized into much higher dimensions. We can also conclude here that the adversarial influence of the sparse classifier has a negative correlation with its sparsity. To prove the effectiveness of the above analysis, we conduct experiments to see if classifier with ℓ_1 regularizer have better results under the sparse feature attack. We generate a data set which is adversarially transformed by an adversary who can only manipulate a limited number of features. We then test the two initial classifiers with different regularizers on the transformed data set. The detailed procedure is described in Algorithm 2. Results are reported in Section 5.3

5. EXPERIMENTS

The benchmark data set and feature selection technique are reported in Section 5.1. Section 5.2 illustrates the rationality of an adversary conduct sparse feature attack. In Section 5.3, we conduct experiments to show the superiority of initial classifier with ℓ_1 regularizer. Results of our four game-theoretic models are reported in Section 5.4. We further verify the effectiveness of the assumption of the adversary in Section 5.5. For logistic loss we use the efficient solver BMRM [14], while for square and hinge loss we use CVX [8]. Data set and the code can be downloaded here [1].

5.1 Data set

To test the validity of our game model, we conduct experiments on real world data set. The data set is a collection of chronologically arranged $128,117$ emails from publicly available *mailing lists* augmented by spam emails from Bruce Guenter's spam trap of the same time period ($01/04/1999 - 31/05/2006$). This data set has also been used in previous adversarial learning research [3]. The original data set we obtained is the inverted table of all the words, symbols etc. of the original spam emails. We conducted feature selection by applying kernel-PCA map [13, 2] which is defined as:

$$\phi PCA : \mathbf{x} \mapsto \Lambda^{\frac{1}{2}^+} V^T [k(\mathbf{x}_1, \mathbf{x}), ..., k(\mathbf{x}_n, \mathbf{x})]^T. \quad (6)$$

Here V is the column matrix of eigenvectors of kernel matrix K, K is defined as dot product of data points $k(\mathbf{x}, \mathbf{x}) = \mathbf{x}^T\mathbf{x}$. We use the first 2000 instances to formulate a 2000×2000 kernel matrix. Λ is the diagonal eigenvalue matrix of K such that $K = V\Lambda V^T$, and $\Lambda^{\frac{1}{2}^+}$ represents the pseudo-inverse of $\Lambda^{\frac{1}{2}}$.

Traditionally, kernel PCA map is used for transform a data set from lower dimension into a higher dimensional space. However, in our case we shrink the feature dimensions, from $266,378$ to 50 to be precise. The selected 50 features are informative enough for training a regular logistic classifier with desirable F-measure score (0.967) on training data. High efficiency and information preserving quality can be achieved with dot product as the kernel.

The first 2000 instance are used as the training data, the rest are test data, we train our model with unbalanced instead of balanced data. This is crucial to accurately reflect the distribution of test data.

One more spam email data sets are also used to compare the robustness of the learning algorithm in Section 5.3;

- 'Spambase' is also a spam email data set[5]. Spam e-mails came from their postmaster and individuals who had filed spam and non-spam e-mails from work and personal e-mails. Most of the features (48 out of 57) are frequency of key words.

5.2 Why sparse feature attack for the adversary?

For this experiment, we simply play the two games $Game - L1_d L2_a$ and $Game - L1_d L1_a$ by setting the MC to infinity and play as many repetition as possible. We then measure and compare the $Cost$ of the two models. Results are averaged value of 30 runs. Figure 2 shows the accumulated cost of the adversary to the repetition of the game. We can observe clearly that for the sparse model $Game - L1_d L1_a$, adversary with sparse feature attack reaches a stable state with significantly less cost. While adversary with dense feature attack in model $Game - L1_d L2_a$ continues to change the data even after the positive and negative data almost overlapped.

5.3 Why ℓ_1 regularizer for data miner?

Here we conduct experiments to compare the performance of an initial classifier with ℓ_1 and ℓ_2 regularizer respectively under few

[1] `https://sites.google.com/site/adversariallearning/code`

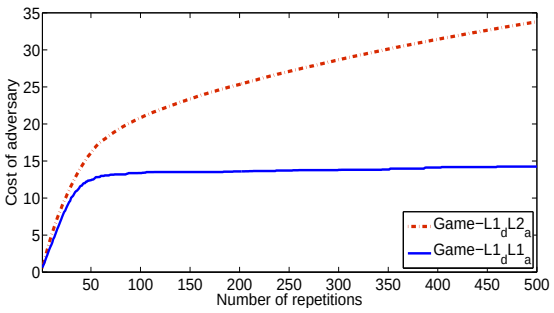
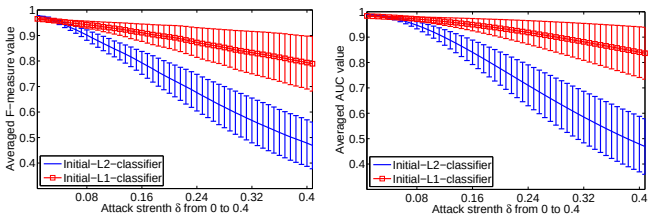


Figure 2: Adversary with sparse feature attack reaches stable state faster and is associated with lower cost. Results are averaged value of 30 runs.

feature attack as proposed in Section 4.4. We generate the tempered training data as described in Algorithm 2. The attack is conducted as follow: We randomly select twenty percent of the number of features to be changed, i.e. we set $c = 10\% \times d$. The attack strength δ is set from 0 to 40% with step size 0.2%.



(a) F-measure comparison. (b) AUC value comparison .

Figure 3: Initial classifier with ℓ_1 regularizer is more robust in both F-measure and AUC value compared that with ℓ_2 regularizer.

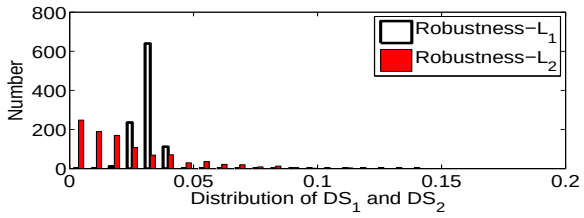
In terms of F-measure and AUC value, as depicted in Figure 3, without adversarial effect, classifier with ℓ_1 regularizer appears initially undesirable in both F-measure and AUC value. However, as the strength of the attack increases, accuracy of classifier with ℓ_2 regularizer decreases rapidly and always outperformed by classifier with ℓ_1 regularizer. This shows that in the presence of an adversary who can only manipulate a limited number of features, ℓ_1 regularizer should be applied by the classifier to achieve higher accuracy.

Here we also test the robustness of the learning algorithm with the existence of noise. We again begin with a data set X. On each separate run i we add bounded noise ΔX_i. Let w_1 be the model on X using the ℓ_1 regularizer and w_2 using the ℓ_2 regularizer. For each i we compute the AUC for $X + \Delta X_i$ for each of the regularizers. Thus $f_1^{\Delta}(i) = AUC(X + \Delta X_i | w_1)$ and likewise for $f_2(i)$. Let $f_2^{\Delta}(i) = AUC(X + \Delta X_i | w_2)$, i.e., the accuracy of the model trained on the data set $X + \Delta X_i$. Like in the case of stability, we form the sets

$$DR_1 = \{f_1^{\Delta}(i) - f_1(i) \mid i = 1 \ldots 1000\}$$

$$DR_2 = \{f_2^{\Delta}(i) - f_2(i) \mid i = 1 \ldots 1000\}$$

The results of the distribution of both DR_1 and DR_2 are shown in Figure 4 and clearly show that the data points of ℓ_1 are more tightly concentrated compared to ℓ_2. This again confirms that when the data is noisy, then ℓ_1 leads to more robust classifiers compared to ℓ_2.



(a) SpamBase.

Figure 4: Robustness: The figure indicates that distribution results of ℓ_1 regularizer has lower standard deviation and thus more robust.

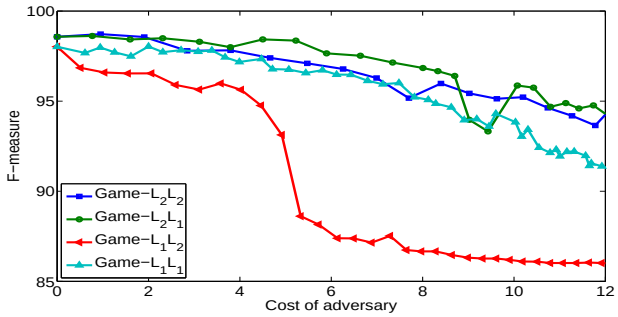


Figure 5: F-measure results of all game-theoretic models decreases on the original training data as the adversary's attack strength increases.

5.4 Experiments of the repeated game

Parameter tuning of our models are explained in Section 5.4.1. In the following two sub sections, we split our four new models into two groups: one is data miner with ℓ_2 regularizer (Initial-L2-classifier, $Game - L2_dL2_a$ and $Game - L2_dL1_a$), and the other group is data miner with ℓ_1 regularizer (Initial-L1-classifier, $Game - L1_dL2_a$ and $Game - L1_dL1_a$). We compare and present the best sparse game-theoretic model: $Game - L1_dL1_a$ in Section 5.4.4.

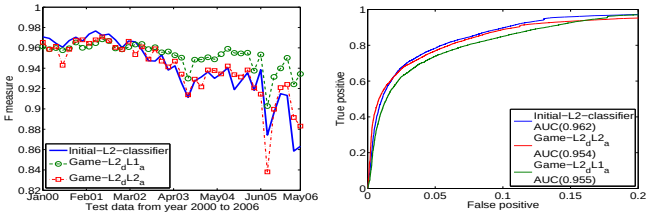
5.4.1 Parameter tuning

We take F-measure as the metric for tuning the parameters. $\lambda_{\mathbf{w}}$ of the the two initial classifiers are tuned with ten-fold cross validation. Based on our assumption that spammer will change their template incrementally with small steps, we set parameter $\lambda_{\mathbf{a}}$ for the adversary as the same proper value. We run our algorithm with no upper bound of the $Cost$ and stops with sufficiently enough repetitions. For each repetition of the game, we will get a pair of vectors for each of the players: feature weight $\mathbf{w}$ and manipulating vector $\mathbf{a}$. We then select the adapted classifier when $Cost$ reaches MC. As one can think of, the adapted classifier will probably have inferior performance on original training data compared with the initial classifier. In a way, we have a balance between the performance of the adapted classifier on the near future data and far future data. This is because as we assume training and test data have different distribution characteristic, you end up with a balance between performance on near future test data (which has similar distribution with original training data) and performance on far future test data. Here we study whether the threshold MC give us a proper balance. We run the game with sufficient repetitions and plot the performance of the adapted classifier on original training data in the accumulated costs of adversary. From Figure 5 we can notice that F-measure of all the four models degrade as the adversarial change

increases. Game mode $Game - L1_dL2_a$ have a sudden drop when adversary cost reaches 5. This reflect that ℓ_1 regularized classifier can be vulnerable to an adversary which conduct dense feature attack. We look for the cost when it is equal to MC and find that it is near 4. This seems to be a very good balance since after this point performance of the model $Game - L1_dL2_a$ dropped sharply. We then compare the performance of the four adapted classifiers on the future test data in the following sections.

5.4.2 *Initial-L2-classifier,* $Game-L2_dL2_a$ *and* $Game-L2_dL1_a$

There are two adapted classifiers obtained from the two models when data miner is with ℓ_2 regularizer: $Game - L2_dL2_a$ and $Game - L2_dL1_a$. We examine and compare the two classifiers together with initial classifier with ℓ_2 regularizer in terms of F-measure, Roc-curve and AUC-value. F-measure results of the three



(a) F-measure on mailinglist. (b) Roc-curve on mailinglist.

Figure 6: $Game - L2_dL1_a$ model outperforms initial classifier with ℓ_2 in terms of F-measure, while $Game - L2_dL2_a$ and $Game - L2_dL1_a$ outperformed by initial classifier in terms of AUC-value.

models are illustrated in Figure 6a. Notice that both the two adapted classifiers sacrificed performance on near future test data (from Jan00 to Mar02), which verified that there is a trade-off between performance on near and far future test data. It also indicates the classifier of model $Game - L2_dL1_a$ has a better performance on the real future data than the initial classifier, while classifier of $Game - L2_dL2_a$ shows no improvements. This result complies with our example given in Figure 1.

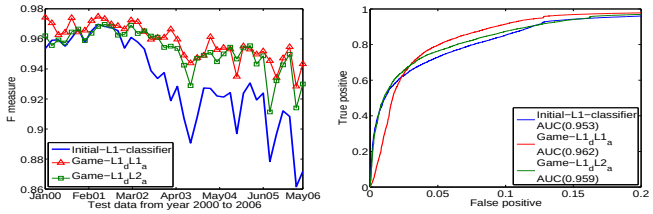
For ROC-curve, we only show the portion when False positive is from 0 to 0.2 to better examine the difference. We noticed that in Figure 6b, both the two adapted classifiers are outperformed by the initial classifier in terms of AUC-value. The undesirable results in AUC-value can be explained as the two new classifiers scarified more on near future test data than the robustness gained on far future test data.

5.4.3 *Initial-L1-classifier,* $Game-L1_dL2_a$ *and* $Game-L1_dL1_a$

The other two variant models are $Game-L1_dL2_a$ and $Game-L1_dL1_a$. Again we examine the two adapted classifiers together with initial classifier with ℓ_1 regularizer.

From Figure 7a, surprisingly we can observe that F-measure results of the two classifiers outperform the initial classifier on almost the whole time span. In other words, the two new adapted classifiers are both robust to near and far future test data. Noticeably, classifier from sparse model $Game - L1_dL1_a$ has the best performance.

Unlike $Game - L2_dL1_a$ and $Game - L2_dL2_a$, as indicated in Figure 7b, performance of AUC-value comparisons also indicate that classifiers of the two new models outperform initial classifier with ℓ_1 regularizer.



(a) F-measure on mailinglist. (b) Roc-curve on mailinglist.

Figure 7: $Game-L1_dL2_a$ and $Game-L1_dL1_a$ both outperform the initial classifier with ℓ_1 regularizer in terms of both F-measure and AUC-value. More importantly, sparse model $Game-L1_dL1_a$ achieves the best performance.

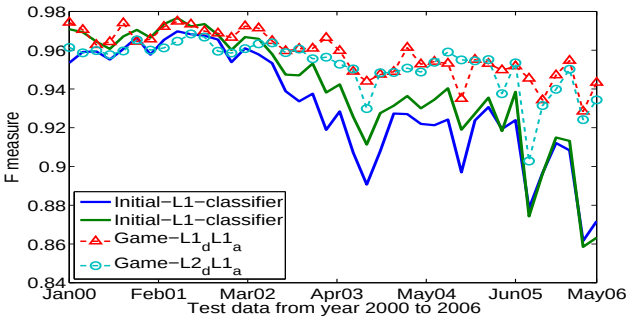
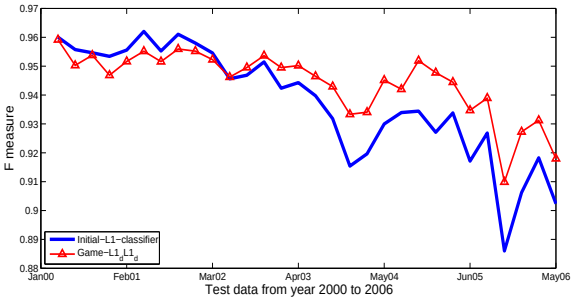


Figure 8: Loss function as Logistic Loss: $Game - L1_dL1_a$ has the best F-measure results on far future test data and better results on the whole test data overall.

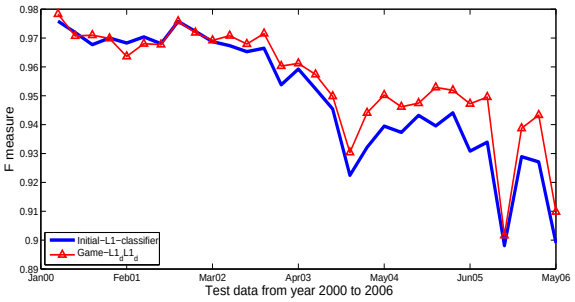
5.4.4 $Game - L2_dL1_a$ *vs.* $Game - L1_dL1_a$

Here we first compare the two initial classifiers with ℓ_1 and ℓ_2 regularizer respectively. From Figure 8 we can observe that initial classifier with ℓ_2 regularizer is better in F-measure. This is also reflected in terms of AUC-value as can be observed in Figure 7 and Figure 6. By fully utilize the information of the data set, a classifier with ℓ_2 regularizer can always determine a more precise decision boundary. However, with the existence of an adversary, this merit of ℓ_2 regularizer may disappear in the future. Noticeably, in the far future, data miner with ℓ_1 regularizer can compete or even outperform that with ℓ_2 regularizer. We further compare the two adapted classifiers of the two sparse model $Game - L2_dL1_a$ and $Game - L1_dL1_a$. In Figure 8 we notice that both the two adapted classifiers have better F-measure score compared to their corresponding initial classifier. Noticeably, $Game - L1_dL1_a$ outperforms $Game - L2_dL1_a$ on most portions of test data. To be specific, $Game - L1_dL1_a$ has a better performance on both near and far future test data compared with $Game - L2_dL1_a$. AUC-value shown in Figure 6b and Figure 7b also indicates that $Game-L1_dL1_a$ achieves the best performance. Based on the above result comparisons, we can make the conclusion that $Game-L1_dL1_a$ is the best setting for our game-theoretic model. In other worlds, by assuming adversary is conducting sparse feature attack, classifier is with ℓ_1 regularizer, the game theoretic model produces the most robust classifier.

We then apply the same procedure using square and hinge loss functions. One should notice that for adversary the two loss functions are quite similar. The main difference between the two is that hinge loss function for adversary represent the the loss of correctly classified positive data being classified as negative data. While for



(a) Loss function as Square Loss.



(b) Loss function as Hinge Loss.

Figure 9: Sparse model $Game - L1_dL1_a$ has the best F-measure results on far future data, and for hinge loss, from Figure b, we can notice that the performance is also competitive on near future data.

square loss, the explanation is the same as logistic loss. We find that compared with the initial classifier, classifier learnt from model $Game-L1_dL1_a$ has significant improved robustness to data in the future. From Figure 9, we can see that in both cases, Sparse model $Game - L1_dL1_a$ has better F-measure on data that is after about $Mar02$. Noticeably, for hinge loss, performance of the classifier learnt from the game model on the near future data is almost the same as the corresponding initial classifier.

5.5 Adversary as random concept shift

Preceding experiments conducted on spam email data set validated the effectiveness of our assumption on the adversarial behaviour (sparse feature attack). However, one may argue that training and test data are intrinsically different without any assumptions. In other words, the two sets of data are different despite they come from the same distribution. As we mentioned before, there may be unpredictable random concept drift in the future. Thus, the question is, can we obtain a robust adapted classifier by taking a random concept change into consideration?

To answer the above question, we can simply simulate the adversary's attack in Algorithm 1 as randomly changing all the training data. Thus, for each repetition in Algorithm 1, the original training data is transformed from $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ to $D^* = \{(\mathbf{x}_i + \mathbf{a}_i, y_i)\}_{i=1}^n$, where $\mathbf{a}_i$ is a vector with elements generated from a normal distribution $\mathcal{N}(0, 0.1)$. We use classifier with ℓ_1 regularizer to compare the effectiveness between the initial classifier and the classifier learned from the game model. We run the model until the $Cost$ is nearly equal to MC and take averaged F-measure value of ten runs of the experiment as the result. We can observe that in Figure 10, F-measure of the adapted classifier trained on randomly changed data shows no improvement on the

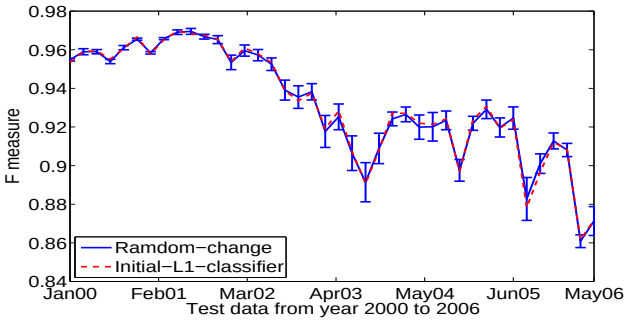


Figure 10: Classifier trained on random changed training data has no improvement compared with initial classifier in F-measure.

test data compared with initial classifier. The failure of this model further verified the effectiveness of our assumption that an adversary will conduct sparse feature attack on positive data.

6. CONCLUSIONS

In many classification environments including spam email filtering and fraud detection, positive data are continuously transformed to deceive the classifier in the future. Traditional machine learning methods built on static training data fail in this scenario. A number of researches have formulated this adversarial scenario into a game played by a data miner and an adversary.

We formulated the interactions between the data miner and the adversary into a *repeated game* (not a Stackelberg game), where adversary and data miner act sequentially. With properly defined loss functions, the game is casted into two convex optimization problems. We provided insights of why sparse strategy instead of dense strategy should be applied by the two players in the *repeated game*. The robustness superiority of the classifier learnt through the sparse model $Game - L1_dL1_a$ is verified through experiments conducted on real spam email data set. To sum up, in adversarial learning, classifier learnt through the game-theoretic model $Game - L1_dL1_a$ achieves the best performance in terms of robustness.

In the future we plan to design a new algorithm for the simultaneous game so that we can solve for a real Nash Equilibrium. Also, we may combine multi-assumptions of the adversary's behaviour and build an ensemble classifier that is more robust in the presence of adversaries.

7. REFERENCES

[1] C. Aliprantis and S. Chakrabarti. *Games and decision making*. Oxford University Press Oxford, 2000.

[2] M. Brückner, C. Kanzow, and T. Scheffer. Static prediction games for adversarial learning problems. 2011.

[3] M. Brückner and T. Scheffer. Stackelberg games for adversarial prediction problems. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 547–555. ACM, 2011.

[4] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108, New York, NY, USA, 2004. ACM.

[5] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[6] A. Genkin, D. Lewis, and D. Madigan. Large-scale bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304, 2007.

[7] A. Globerson and S. Roweis. Nightmare at test time: robust learning by feature deletion. In *Proceedings of the 23rd international conference on Machine learning*, pages 353–360, New York, NY, USA, 2006. ACM.

[8] M. Grant, S. Boyd, and Y. Ye. Cvx: Matlab software for disciplined convex programming, 2008.

[9] W. Liu and S. Chawla. Mining adversarial patterns via regularized loss minimization. *Machine learning*, 81(1):69–83, 2010.

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

[11] G. Pekhimenko. Penalizied logistic regression for classification.

[12] S. Shalev-Shwartz, N. Srebro, and T. Zhang. Trading accuracy for sparsity in optimization problems with sparsity constraints. *SIAM Journal on Optimization*, 20(6), 2010.

[13] A. J. Smola and B. Schölkopf. *Learning with kernels*. Citeseer, 1998.

[14] C. Teo, S. Vishwanthan, A. Smola, and Q. Le. Bundle methods for regularized risk minimization. *Journal of Machine Learning Research*, 11:311–365, 2010.

[15] H. Xu, C. Caramanis, and S. Mannor. Robust regression and lasso. *Information Theory, IEEE Transactions on*, 56(7):3561–3574, 2010.

[16] Y. Zhou, M. Kantarcioglu, B. Thuraisingham, and B. Xi. Adversarial support vector machine learning. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1059–1067. ACM, 2012.