2012 IEEE 12th International Conference on Data Mining

Sparse Bayesian Adversarial Learning Using Relevance Vector Machine Ensembles

Yan Zhou, Murat Kantarcioglu, and Bhavani Thuraisingham
Department of Computer Science
University of Texas at Dallas
Richardson, TX 75080
{yan.zhou2, muratk, bhavani.thuraisingham}@utdallas.edu

***Abstract*—Data mining tasks are made more complicated when adversaries attack by modifying malicious data to evade detection. The main challenge lies in finding a robust learning model that is insensitive to unpredictable malicious data distribution. In this paper, we present a sparse relevance vector machine ensemble for adversarial learning. The novelty of our work is the use of individualized kernel parameters to model potential adversarial attacks during model training. We allow the kernel parameters to drift in the direction that minimizes the likelihood of the positive data. This step is interleaved with learning the weights and the weight priors of a relevance vector machine. Our empirical results demonstrate that an ensemble of such relevance vector machine models is more robust to adversarial attacks.**

***Keywords*-adversarial learning; spare Bayesian learning; relevance vector machine; kernel parameter learning.**

I. Introduction

Existing research in adversarial learning varies in the types of constraints considered in the problem definition. The assumption of unconstrained adversaries is impractical since arbitrary modification to data and its class membership can result in a worst-case error rate of 100% [1], [2]. Therefore, the majority of the recent research focuses on constrained adversaries. Under the constrained-adversary assumption, major research results can be further divided between game-theoretic solutions and non-game theoretic solutions. For practitioners, the difficulty lies in choosing the most appropriate method for problems at hand. Solutions developed in the game-theoretic framework almost always assume a rational game. In addition, each player is assumed to have a certain amount of knowledge about the opponent. Similarly, non-game theoretic methods often make assumptions on the opponent's knowledge, the distribution of corrupted data, and available computing resources. In practice, adversaries are seldom optimal and the knowledge and the resources they possess are hard to assess.

For classification problems, the common assumption is that data are independently and identically distributed. This assumption is easily violated when there is an active adversary who modifies data to influence the prediction. When data is constantly modified in an unpredictable way, training data would never be sufficient to induce an accurate classifier. On the positive side, at training time we can explore the feature space and find the most effective direction for the adversary to move data in the feature space to influence the classifier. Once we find such a direction, we can improve the classifier by countering these potential moves. The learning model we choose to implement this strategy is the *relevance vector machine*.

Similar to the support vector machine method, the relevance vector machine (RVM) [3] is a sparse linearly parameterized model. It is built on a Bayesian framework of the sparse model. Unlike the support vector machine in which a penalty term is introduced to avoid over-fitting the model parameters, the relevance vector machine model introduces a prior over the weights in the form of a set of hyperparameters, one associated independently with each weight. Very large values of the hyperparameters (corresponding to zero-weights) imply irrelevant inputs. Training data points associated with the remaining non-zero weights are referred to as *relevance vectors*. The relevance vector machine typically use much fewer kernel functions compared to the SVM.

In this paper, we propose a sparse relevance vector machine ensemble for adversarial learning. The basic idea of this approach is to learn an individual kernel parameter η_i for each dimension d_i in the input space. The parameters are iteratively estimated from the data along with the weights and the hyperparameters associated with the weights. The kernel parameters are updated in each iteration so that the likelihood of the positive (malicious) data points are minimized. This essentially models adversarial attack as if the adversary were granted access to the internal states of the learning algorithm. Instead of using fixed kernel parameters, we search for kernel parameters that simulate worst-case attacks while the learning algorithm is updating the weights and the weight priors of a relevance vector machine. We learn M such models and combine them to form the final hypothesis. Our main contributions are:

- extending the sparse Bayesian relevance vector machine model to counter adversarial attacks;
- developing a kernel parameter fitting technique to model adversarial attacks within the RVM framework.

The use of individualized kernel parameters has been shown beneficial to kernel-based learning [3]; however, this is the first time it is applied to adversarial learning.

The rest of the paper is organized as follows. Section II presents the related work in adversarial learning. Section III

1550-4786 2012
DOI 10.1109/ICDM.2012.58

discusses the relevance vector machine model. Section IV presents the gradient-based method for modeling adversarial attacks and Section V presents experimental results on both artificial and real data sets. Section VI concludes our work and discusses future directions.

II. Related Work

There are several theoretical conclusions regarding bounds on malicious noise rate and learning accuracy. Kearns and Li [1] prove theoretical upper bounds on tolerable malicious error rates in the sample. They assume the adversary can generate malicious errors with an unknown and unpredictable nature. Bshouty et al. [4] introduce a variant of the PAC learning model for learning in the presence of nasty noise. They prove that when the sample noise rate is η no learning algorithm can learn non-trivial concept classes with accuracy less than 2η. Auer and Cesa-Bianchi [2] present an online learning model for learning with corrupted data. They extend the "Closure Algorithm" and prove a worst case mistake bound for learning an arbitrary intersection-closed concept class. Lowed and Meek [5] present an algorithm that finds the instance with minimal adversarial cost. They refer to this algorithm as an adversarial classifier reverse engineering (ACRE) algorithm.

Adversarial learning problems have been extensively studied in the framework of game theory. Dalvi et al. [6] consider the learning problem as a game between two cost-sensitive opponents. The adversary always plays optimal strategies. Given a cost function, their algorithm predicts the class that maximizes the conditional utility. Kantarcioglu et al. [7] present a simulated annealing and genetic algorithm to search for a Nash equilibrium for choosing an optimal set of attributes. They assume the two players know each other's payoff function. Similar work with improvement on how Nash strategies are played has also been proposed [8], [9]. Brückner and Scheffer [10] present an optimal game by assuming the adversaries always behave rationally. Their algorithm does not require a unique equilibrium.

In a slightly different research avenue, robust learning techniques have been proposed for handling classification-time noise [11]–[14]. Globerson and Roweis [15] consider classification-time feature deletion. They present an SVM algorithm that constructs an optimal classifier under a predefined constraint. El Ghaoui et al [16] present a minimax strategy for training data bounded by hyper-rectangles. They minimize the worst-case loss over data in given intervals.

Zhou et al. [17] present two attack models for which optimal learning strategies are derived. They formulate a convex optimization problem in which the constraint is defined over the sample space based on the proposed attack models. They demonstrate that their adversarial SVM model is more resilient to adversarial attacks compared to the standard SVM and one-class SVM models.

III. Relevance Vector Machine

Given a set of N training data (x_i, y_i), $i = 1, \ldots, N$, where $x_i \in \mathbb{R}^d$ are d-dimensional data points, and $y_i \in \{0, 1\}$ are the labels, a function $h(x)$ over the input space is inferred in the following linearly weighted form:

$$h(x; w) = w^T \phi(x)$$

where $\phi(x)$ represents basis functions, and

$$\phi(x) = [1, K(x, x_1), K(x, x_2), \ldots, K(x, x_N)]$$

where $K(x, x_i)$ is a kernel function. For a binary classification problem, the posterior probability of the class membership given x as the input is:

$$p(y|w) = \prod_{i=1}^{N} g(h(x_i; w))^{y_i} [1 - g(h(x_i; w))]^{1-y_i} \qquad (1)$$

where $g(t)$ is the sigmoid function $g(t) = 1/(1 + e^{-t})$ applied to t. Since there are as many weight parameters as the training examples, the model would suffer over-fitting. To avoid over-fitting, a zero-mean Gaussian prior is defined over the weights, with each prior controlled by its own hyperparameter α. Therefore,

$$p(w|\alpha) = \prod_{0}^{N} \mathcal{N}(w_i|0, \alpha_i^{-1})$$

where α_i is the hyperparameter of w_i. The assignment of an individual hyperparameter to each weight gives an equivalent regularization penalty term. When a hyperparameter has a very large value (often approaching infinity), the regularizing effect becomes so large that the corresponding weight parameter rapidly converges to zero and thus the corresponding basis function can be pruned. When a hyperparameter has a small value, the prior has very little effect on the weight parameter it moderates, and therefore the corresponding basis function $\phi_k(x)$ is a relevant feature, and the example x_k it centers is selected as a relevant vector.

Since the weight posterior $p(w|y, \alpha)$ and the marginal likelihood $p(y|\alpha)$ cannot be computed analytically, Tipping [3] adopted a Laplacian approximation method to iteratively estimate the posterior covariance for a Gaussian approximation to $p(w|y, \alpha)$ centered at the mean $\overline{w}$. The mean and the covariance are then used to optimize the hyperparameter α. Details can be found in his work [3].

Within the Bayesian framework of the relevance vector machine, we introduce another set of parameters that directly weigh the difference between two vectors in each dimension in the input space. In the next section, we discuss the use of individual kernel parameters to model adversarial attacks.

IV. Kernel Parameter Fitting

The RVM training process iteratively updates the weight vector w and the hyperparameter vector α. Imagine in each iteration the adversary has an opportunity to modify the training data, particularly the positive (malicious) training data, so that it could cross the decision boundary inferred in the current iteration. What would be the best strategy for the adversary to modify the data? If the adversary has the freedom to move each data point in his own favor, he would follow the directions that increase the likelihood of misclassifying a positive instance the greatest. Before we discuss the technique to search for this direction, we first discuss the kernel and its input scale parameters.

A. Kernel Parameter Vector

Consider the RBF kernel

$$K(x_i, x_j) = \exp(-\eta \cdot ||x_i - x_j||^2)$$

where $\eta = (\eta_1, \ldots, \eta_d)$ is a vector of d parameters, and η_k is its k^{th} parameter preceding the squared distance $(x_{ik} - x_{jk})^2$ in the k^{th} input dimension. Normally, there is only one kernel parameter and its value is typically determined through cross-validations. We use individual kernel parameters so that we can model adversarial data modification in each dimension. For example, when the adversary modifies the k^{th} dimension such that $x_{ik} \approx x_{jk}$, the same effect can be achieved by having $\eta_k \approx 0$. Therefore, by adjusting the kernel parameter of the k^{th} dimension of the input, we could model adversarial attacks in both the input space and the feature space. We can then update the weight parameter and the corresponding hyperparameters to counter the attacks.

B. Attacks Minimizing the Log-Likelihood

Assuming the adversary is only interested in disguising positive data [1], during RVM training we search for a kernel parameter vector η that renders the most effective attacks on positive training instances. With a given w and α, we update η in the direction that decrease $\mathcal{L}_+$, the log-likelihood of the posterior distribution $p(y|w, \alpha)$ given in Equation (1) for all positive instances. Taking the logarithm of both sides of Equation (1), we have:

$$log(p(t|w)) = \sum_{i=1}^{N} [y_i log(\sigma_i) + (1 - y_i)(1 - log(\sigma_i))] \quad (2)$$

where $\sigma_i = g(h(x_i; w))$ is the output of the sigmoid function. Let $\mathcal{L} = log(p(t|w)) = \mathcal{L}_+ + \mathcal{L}_-$, where

$$\mathcal{L}_+ = \sum_{i=1}^{N} y_i log(\sigma_i) \text{ and } \mathcal{L}_- = \sum_{i=1}^{N} (1 - y_i)(1 - log(\sigma_i)).$$

[1]This is a reasonable assumption since it is typically harder for adversaries to influence negative (legitimate) data.

The gradient of $\mathcal{L}$ given in (2) with respect to the η_k is:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \eta_k} &= \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{\partial \mathcal{L}}{\partial K_{ij}} \frac{\partial K_{ij}}{\partial \eta_k} \\ &= \sum_{i=1}^{N} \sum_{j=1}^{N} (\frac{\partial \mathcal{L}_+}{\partial K_{ij}} + \frac{\partial \mathcal{L}_-}{\partial K_{ij}}) \frac{\partial K_{ij}}{\partial \eta_k} \end{aligned}$$

where K_{ij} is the kernel function K applied to the i^{th} and j^{th} input x_i and x_j. To model attacks on the positive instances, we negate $\frac{\partial \mathcal{L}_+}{\partial K_{ij}}$, and use the following for a gradient-based local optimization over η:

$$\mathcal{G} = \sum_{i=1}^{N} \sum_{j=1}^{N} (-\frac{\partial \mathcal{L}_+}{\partial K_{ij}} + \frac{\partial \mathcal{L}_-}{\partial K_{ij}}) \frac{\partial K_{ij}}{\partial \eta_k}. \quad (3)$$

Working out each term, we have:

$$\begin{aligned} \frac{\partial \mathcal{L}_+}{\partial K_{ij}} &= y_i \cdot \frac{1}{\sigma_i} \cdot \frac{\partial \sigma_i}{\partial h} \cdot \frac{\partial h}{\partial K_{ij}} \\ &= y_i \cdot (1 - \sigma_i) \cdot w_j \end{aligned}$$

$$\begin{aligned} \frac{\partial \mathcal{L}_-}{\partial K_{ij}} &= (1 - y_i) \cdot \frac{-1}{1 - \sigma_i} \cdot \frac{\partial \sigma_i}{\partial h} \cdot \frac{\partial h}{\partial K_{ij}} \\ &= -(1 - y_i) \cdot \sigma_i \cdot w_j \end{aligned}$$

$$\frac{\partial K_{ij}}{\partial \eta_k} = -K_{ij} \cdot (x_{ik} - x_{jk})^2$$

Therefore,

$$\mathcal{G} = \sum_{i=1}^{N} \sum_{j=1}^{N} -(y_i - \sigma_i) \cdot w_j \cdot K_{ij} \cdot (x_{ik} - x_{jk})^2$$

which will be the basis for updating η in each iteration of training a relevance vector machine.

C. Training Issues

During training, we need to iteratively update three parameters: the weight w, the hyperparameter α, and the kernel parameter vector η. One way to update these parameters is to optimize over w/α and η simultaneously. This allows us to find the optimal solution, but will be computationally intensive. Another way to do it is to interleave the update of w/α and η. For each (w, α), we search a few steps for $\Delta\eta$ based on the gradient of the log-likelihood, and update η by adding $\tau \cdot \Delta\eta$ to it, where $\tau > 0$ is the momentum. This approach may not lead to the desired solution, but is more efficient. In this paper, we follow the local update approach. We train multiple adversarial RVM models, and select M models that satisfy $u_+ < \rho \cdot u_-$, where u_+ denotes the uncertainty of predicting a positive training instance, and u_- the uncertainty of predicting a negative training instance, and $\rho \in (0, 1]$ is a positive constant. The smaller the ratio $\rho = \frac{u_+}{u_-}$, the farther away the decision boundary from the positive training examples. The uncertainties u_+ and u_-

are estimated on the positive and negative training data as follows:

$$u_+ = \frac{\sum_{i=1}^{N} y_i \cdot (1 - h(x_i; w))}{\sum_{i=1}^{N} y_i} \quad (4)$$

$$u_- = \frac{\sum_{i=1}^{N} (1 - y_i) \cdot h(x_i; w)}{\sum_{i=1}^{N} (1 - y_i)} \quad (5)$$

where $\sum_{i=1}^{N} y_i$ is the total number of positive training instances, and $\sum_{i=1}^{N}(1 - y_i)$ is the number of negative training instances. We select M such models and combine them to make the final predications through majority vote.

D. Adversarial RVM Learning Algorithm

We now present the algorithm of adversarial RVM learning, namely *AD-RVM*. Given a set of training data, we first initialize the value of α and use it to update the weight vector w; next, we update the kernel parameter vector η with the given w and α; finally we update the hyperparameter α. The process iterates for a pre-defined maximum number of rounds. We train multiple such classifiers and select M classifiers that satisfy $\rho \cdot u_+ < u_-$ to form an ensemble. The detailed algorithm is given in Figure 1.

Input: L, T—training and test data
w, α—weight and hyperparameter
η—kernel parameter
M—number of classifiers in the ensemble
τ—momentum of updating η in each iteration
I_w—number of iterations updating w and α
I_η—number of inner cycles updating η
Output: ensemble $\{h_1, h_2, \ldots, h_M\}$

Initialize α
repeat
  for $i = 1$ **to** I_w **do**
    Update w with current α values
    for $i = 1$ **to** I_η **do**
      Compute $\Delta\eta$ using Equation (3)
      $\eta = \eta + \tau\Delta\eta$ *// end for*
    Update feature space with new kernel parameters
    Solve for α with w and η *// end for*
  Compute u_+ and u_- using equations (4) and (5)
  if $\rho \cdot u_+ < u_-$ **then**
    add $h(w, \alpha, \eta)$ to the ensemble
until M RVMs are added to the ensemble
return $\{h_1, h_2, \ldots, h_M\}$

Figure 1. The *AD-RVM* algorithm.

V. Experimental Results

We used one artificial data set and two real data sets in our experiments. We model the attacks at classification time by moving positive test instances closer to randomly selected negative instances plus local random noise. Attacks on the test data are designed to challenge all the learning models at increasingly more difficult levels. The difficulty is controlled using the attack factor f_{attack}. More specifically,

$$x_{ij}^+ = x_{ij}^+ + f_{attack} \cdot (x_{ij}^- - x_{ij}^+) + \epsilon \quad (6)$$

where ϵ is local random noise. Notice $f_{attack} = 1$ models the worst case attacks where a positive data point is arbitrarily close to a negative one within the range of the random local noise. Figure 2 illustrates the attack with $f_{attack} = 0.3$.

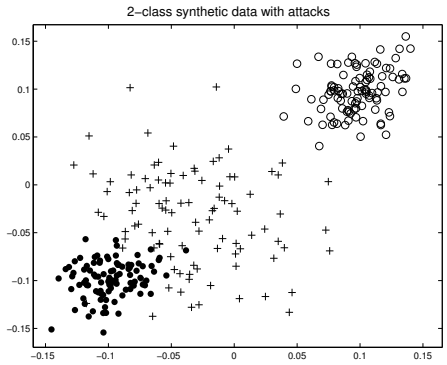


Figure 2. Attack Factor = 0.3. White dots: negative data, black dots: positive data before attacks, black "+": positive data after attacks.

We compare four learning models: AD-RVM, RVM, SVM, and One-class SVM on the three data sets. All results reported are averaged over 10 random runs.

A. Experiments on the Artificial Dataset

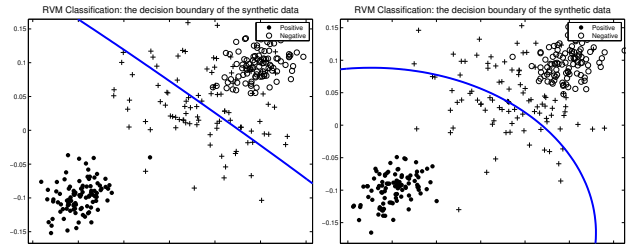
The synthetic data set is generated from a bivariate normal distribution with specified means $(-5.0, -6.0)$ and $(5.0, 6.0)$, and covariance matrix $\begin{pmatrix} 1.0 & 0.5 \\ 0.5 & 2.0 \end{pmatrix}$. We first consider the cases where the training data is clean and only the test data has been corrupted. Attacks on the test data are created using Equation (6). Table I shows the classification performance on this set of data in terms of error rates. The ρ value which controls the uncertainty of predicting positive and negative data is chosen as 0.67. We discuss the impact of ρ later.

Table I
Classification errors of AD-RVM, RVM, SVM, and 1-class SVM on synthetic data. Best results are bolded.

	f_{attack}				
	0.1	0.3	0.5	0.7	0.9
AD-RVM	**0.0025**	**0.0175**	**0.1435**	**0.3205**	**0.4500**
RVM	0.0100	0.0810	0.2542	0.4305	0.5000
SVM	0.0105	0.0705	0.2500	0.4355	0.4910
1-class SVM	0.0059	0.1310	0.5000	0.5000	0.5000

From Table I, we can see that adversarial RVM algorithm has much lower error rates compared to its non-adversarial

self, SVM, and one-class SVM. The improvement is attributed to its adjustment to the decision boundary to counter adversarial attacks. The adjustment includes shifting and curving toward the negative data points as shown in Figure 3.



(a) Decision Boundary Shifting (b) Decision Boundary Curving

Figure 3. Adjustment to decision boundary to take into account potential adversarial attacks. Solid lines in the plots illustrate the decision boundary.

In forming the ensemble, we use the positive and negative uncertainty ratio to determine which hypothesis is included in the ensemble. The higher the ratio, the more confident the classifier is when predicting an instance as a positive example. Note that this is necessary because we took the non-simultaneous updating approach in searching for the kernel parameters. This approach is less reliable than the more expensive simultaneous updating technique. However, we gain speed and, with little additional effort, accuracy as well. We now illustrate the impact of the ρ value used in the uncertainty test. Smaller ρ values imply more bias against the negative prediction, while larger values has less bias. Figure 4 illustrates the impact of the ρ values on the classification errors using AD-RVM. The y-axis shows the improvement of error rates of AD-RVM compared to RVM without adversarial treatment. As can be observed, the lower the ρ value, the more improvement achieved. However, if the ρ value is set too small, eventually false positives will start increasing and defeating the purpose of adversarial learning. The lower bound of the ρ value needs to be carefully set, especially when there are dense positive and negative data points distributed near the decision boundary.

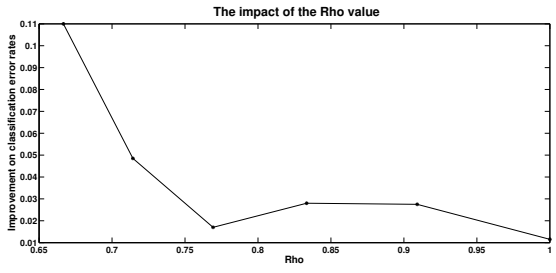
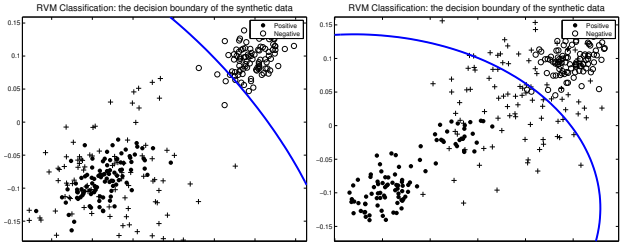


Figure 4. The impact of the ρ (Rho) values. Smaller ρ values has higher inherent bias against negative class membership.

We also tested cases where the training data is not clean. Again, AD-RVM is clearly more robust against adversarial attacks although its superiority becomes less marked. Due to space limitations, we do not report the results here.

Figure 5 illustrates the decision boundary learned from corrupted training datasets. Figure 5(a) shows how the decision boundary can be forced to move drastically closer to the negative side with $f_{attack} = 0.1$ on the test data. The ρ value is set to 0.67. A slightly better decision boundary is learned with $\rho = 0.76$ as shown in Figure 5(b). This experiment reminds us of the importance of setting the lower bound of the ρ value. When the ρ value is too small, false positives become more likely in the output. In practice, we suggest the ρ value not be greater than 0.5. However, in any domain where adversarial attacks could be very aggressive, a smaller ρ value is preferred.



(a) $\rho = 0.67$ (b) $\rho = 0.76$

Figure 5. Adjustment to decision boundary to take into account potential adversarial attacks. Sold lines in the plots illustrate the decision boundary.

Like any other kernel-based approaches, it is important to select the initial η value. Small η values may cause the learning model over-sensitive to the training samples. Large η vales may lead to serious over-fitting. On the negative side, this may require cross-validation, adding additional computational cost. On the positive side, inappropriate initial η values often quickly lead to deterioration to non-positive definite feature vector, which causes the solver to terminate prematurely. If this happens, it often means a larger η value is required.

B. Experiments on Real Datasets

The two real datasets used in our experiments are: *spam base* from the UCI data repository [2], and *web spam* from the LibSVM website [3]. Both datasets are collected from applications typically running in an adversarial environment.

In the *spam base* data set, there are 4601 e-mail examples in total, among which 39.4% is spam. Each example is represented with 57 attributes and one class label. In our experiment, the data set was divided into two equal halves, one for training and the other for testing. 5% of the training data is randomly sampled to build the four learning models in each run. The results are averaged over 10 random runs. Table II shows the classification error rates of the four learning algorithms. The one-class SVM output the best

[2]http://archive.ics.uci.edu/ml/

[3]http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/

results when $f_{attack} \leq 0.5$ while our adversarial RVM algorithm ranked second and had a better performance than RVM and SVM. Our AD-RVM outperformed all the other algorithms when $f_{attack} > 0.5$.

Table II
CLASSIFICATION ERRORS OF AD-RVM, RVM, SVM, AND 1-CLASS SVM ON THE SPAMBASE DATASET. BEST RESULTS ARE BOLDED.

	f_{attack}				
	0.1	0.3	0.5	0.7	0.9
AD-RVM	0.3158	0.3270	0.3368	**0.3438**	**0.3809**
RVM	0.3875	0.3753	0.3775	0.3899	0.3902
SVM	0.3641	0.3751	0.3793	0.4149	0.4079
1-class SVM	**0.3127**	**0.3147**	**0.3248**	0.3555	0.4011

The *web spam* data set is the uni-gram version from the LibSVM website. There are 254 features in each example. We further reduce the number of features to 50. The total number of instances is 350,000. We used one-half for training and the other half for testing. We randomly selected 2% of the samples in the training set to build the learning models. The results are averaged over 10 random runs and are shown in Table III. As can be observed, adversarial-RVM is clearly superior to the other three models.

Table III
CLASSIFICATION ERRORS OF AD-RVM, RVM, SVM, AND 1-CLASS SVM ON THE WEBSPAM DATASET. BEST RESULTS ARE BOLDED.

	f_{attack}				
	0.1	0.3	0.5	0.7	0.9
AD-RVM	0.2426	**0.2926**	**0.3373**	**0.4945**	**0.5866**
RVM	**0.2355**	0.3169	0.4541	0.5560	0.5876
SVM	0.2725	0.4725	0.5604	0.6061	0.6061
One-class SVM	0.3155	0.5625	0.5945	0.6009	0.5997

VI. CONCLUSIONS AND FUTURE WORK

We present a sparse Bayesian adversarial learning model. The algorithm sets individual kernel parameters to model adversarial attacks in the feature space by minimizing the log-likelihood of the positive instances in the training set. The learning models trained under this setup are more robust against attacks including the very aggressive ones. The open problem is to discover an efficient approach to simultaneously update the kernel and the learning parameters. The solution would help find the optimal learning model against the worst case attacks in the sample space.

ACKNOWLEDGMENT

This work was partially supported by The Air Force Office of Scientific Research MURI-Grant FA-9550-08-1-0265 and Grant FA9550-12-1-0082, National Institutes of Health Grant 1R01LM009989, National Science Foundation (NSF) Grant Career-CNS-0845803, NSF Grants CNS-0964350, CNS-1016343, CNS-1111529, and CNS-1228198, Army Research Office Grant 58345-CS.

REFERENCES

[1] M. Kearns and M. Li, "Learning in the presence of malicious errors," *SIAM J. of Computing*, vol. 22, pp. 807–837, 1993.

[2] P. Auer and N. Cesa-Bianchi, "On-line learning with malicious noise and the closure algorithm," *Annals of Mathematics and Artificial Intelligence*, vol. 23, no. 1-2, pp. 83–99, 1998.

[3] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *J. Mach. Learn. Res.*, vol. 1, pp. 211–244, Sep. 2001.

[4] N. H. Bshouty, N. Eiron, and E. Kushilevitz, "Pac learning with nasty noise," *Theoretical Computer Science*, vol. 288, p. 2002, 1999.

[5] D. Lowd and C. Meek, "Adversarial learning," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 2005, pp. 641–647.

[6] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma, "Adversarial classification," in *Proceedings of the tenth ACM SIGKDD*. ACM, 2004, pp. 99–108.

[7] M. Kantarcioglu, B. Xi, and C. Clifton, "Classifier evaluation and attribute selection against active adversaries," *Data Min. Knowl. Discov.*, vol. 22, pp. 291–335, January 2011.

[8] M. Bruckner and T. Scheffer, "Nash equilibria of static prediction games," in *Advances in Neural Information Processing Systems*. MIT Press, 2009.

[9] W. Liu and S. Chawla, "Mining adversarial patterns via regularized loss minimization," *Mach. Learn.*, vol. 81, pp. 69–83, October 2010.

[10] M. Bruckner and T. Scheffer, "Stackelberg games for adversarial prediction problems," in *Proceedings of the 17th ACM SIGKDD*. ACM, 2011.

[11] G. R. G. Lanckriet, L. E. Ghaoui, C. Bhattacharyya, and J. M. I., "A robust minimax approach to classification," *Journal of Machine Learning Research*, vol. 3, pp. 555–582, 2002.

[12] C. H. Teo, A. Globerson, S. T. Roweis, and A. J. Smola, "Convex learning with invariances," in *Advances in Neural Information Processing Systems*, 2007.

[13] O. Dekel and O. Shamir, "Learning to classify with missing and corrupted features," in *Proceedings of the International Conference on Machine Learning*. ACM, 2008, pp. 216–223.

[14] O. Dekel, O. Shamir, and L. Xiao, "Learning to classify with missing and corrupted features," *Machine Learning*, vol. 81(2), pp. 149–178, 2010.

[15] A. Globerson and S. Roweis, "Nightmare at test time: robust learning by feature deletion," in *Proceedings of the 23rd ICML*. ACM, 2006, pp. 353–360.

[16] L. El Ghaoui, G. R. G. Lanckriet, and G. Natsoulis, "Robust classification with interval data," EECS Department, UC Berkeley, Tech. Rep. UCB/CSD-03-1279, Oct 2003.

[17] Y. Zhou, M. Kantarcioglu, B. Thuraisingham, and B. Xi, "Adversarial support vector machine learning," in *Proceedings of the 18th ACM SIGKDD*. ACM, 2012, pp. 1059–1067.