Adversarial Machine Learning

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFIMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

Master of Engineering

IN

Computer Science and Engineering

BY

Sweta Sharma

Computer Science and Automation

Indian Institute of Science

Bangalore – 560 012 (INDIA)

June, 2017

Declaration of Originality

I, **Sweta Sharma**, with SR No. **04-04-00-10-41-15-1-12392** hereby declare that the material presented in the thesis titled

Adversarial Machine Learning

represents original work carried out by me in the **Deparment of Computer Science and Automation** at **Indian Institute of Science** during the years **2016-2017**. With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: Student Signature

In my capacity as supervisor of the above-mentioned work, I certify that the above statements are true to the best of my knowledge, and I have carried out due diligence to ensure the originality of the report.

Advisor Name: Chiranjib Bhattacharyya Advisor Signature

DEDICATED TO

My Family and Friends

Acknowledgements

I would like to express my sincere gratitude to my project advisor, Prof. Chiranjib Bhattacharyya for giving me an opportunity to work on this project. I am thankful to him for his valuable guidance and moral support. I feel lucky to be able to work under his supervision. I also sincerely thank my lab mates for constant motivation and support to put all the ideas into reality. I am overwhelmed to acknowledge their humbleness. I would also like to thank the Department of Computer Science and Automation for providing excellent study environment. The learning experience has been really wonderful here. Finally I would like to thank all IISc staff, my family and friends for helping me at critical junctures and making this project possible.

Abstract

Most machine learning models are designed assuming that both the training data and test data are coming from the same, static distribution. However, in some applications where there is presence of an adversary, the training or test data could be subject to strategic manipulations which the conventional machine learning models cannot handle. Infact, even a small manipulation in data can cause the machine learning model to breakdown. We study the problem of training-set attacks by an adversary on machine learning models. We show that it is possible for an adversary to manipulate data in a way that the resulting model is favourable to him. We have also done experiments to show that such attacks by an adversary can be counteract by using Robust Optimization techniques.

Contents

Acknowledgements **i**

Abstract **ii**

Contents **iii**

List of Figures **v**

List of Tables **vi**

1 Introduction **1**

1.1 Problem Definition and Motivation 1

1.2 Related Work 2

1.3 Our Contributions 3

1.4 RoadMap 3

2 Training-Set Attacks **4**

2.1 Assumptions 4

2.2 Training-Set attacks using KKT conditions 5

2.3 Training-Set attacks against SVM 6

2.4 Training-Set attacks against Linear Regression 8

2.5 Training-Set attacks against K-means Clustering 9

3 Algorithm for Robust Learning **10**

3.1 Idea behind the algorithm 10

3.2 Spectral norm of gradients 12

3.2.1 Calculation of S for SVM 12

3.2.2 Calculation of S for Linear Regression 12

3.2.3 Calculation of S for K-means clustering 13

4 Experiments **14**
4.1 SVM Poisoning attack on Real Dataset 14
4.1.1 Dataset 14
4.1.2 Target vector computation 15
4.1.3 Results 15
4.2 SVM poisoning attack on Synthetic Dataset 17
4.2.1 Dataset 17
4.2.2 Target vector computation 17
4.2.3 Results 17
4.2.3.1 Effort v/s Fraction of data poisoned 19
4.2.3.2 Risk v/s Fraction of data poisoned 20
4.2.3.3 Scatter plot of norm of difference of Poisoned data points and Unpoisoned data points 22

5 Conclusion and Future Work **24**

Bibliography **25**

List of Figures

4.1 Effort versus number of iterations when the adversary wishes to corrupt 70% of the training data 15
4.2 Risk versus number of iterations when the adversary wishes to corrupt 70% of the training data 16
4.3 Norm of the difference of the original data and corrupt data when the adversary wishes to corrupt 70% of the training data 16
4.4 Effort versus Number of iterations when we poison 0% data. 19
4.6 Effort versus Number of iterations when we poison 90% data. 19
4.5 Effort versus Number of iterations when we poison 10% data. 20
4.7 Risk versus Number of iterations when we poison 0% data. 20
4.8 Risk versus Number of iterations when we poison 10% data. 21
4.9 Risk versus Number of iterations when we poison 90% data. 21
4.10 Scatter plot of norm of difference of Poisoned data points and Unpoisoned data points when trying to poison 0% of the training data . . . 22
4.11 Scatter plot of norm of difference of Poisoned data points and Unpoisoned data points when trying to poison 10% of the training data . . 22
4.12 Scatter plot of norm of difference of Poisoned data points and Unpoisoned data points when trying to poison 90% of the training data . . 23

List of Tables

4.1 Features in the wine quality dataset 14
4.2 The fraction of data we wish to corrupt versus fraction of data that actually gets corrupted when trying to get the learned model to get close to the target model. 17
4.3 Fraction of data we want to corrupt versus the norm of difference of model learned on poisoned training data and unpoisoned training data. 17
4.4 Fraction of data we wish to corrupt versus the accuracy achieved when using w_i for prediction. 17
4.5 Fraction of data we wish to corrupt versus the hinge loss when using w_i for prediction. 18
4.6 Fraction of data we wish to corrupt versus the accuracy achieved when using weighted average of $w_i's$ for prediction. 18
4.7 Fraction of data we wish to corrupt versus the hinge loss when using weighted average of $w_i's$ for prediction. 18

Chapter 1

Introduction

Adversarial Machine Learning is a research field that lies at the intersection of Machine Learning and Computer Security. Many Machine Learning Algorithms are currently being used in the domain of Computer Security such as for Malware Detection, Spam Filtering etc. Apart from these, they also find applications in systems such as Robots and Self-driving cars etc. Such systems may have the presence of an adversary who can attempt to compromise the security of such systems. Adversarial Machine Learning thus comes into the picture here and helps in the safe adoption of Machine Learning Algorithms where there could be the presence of an adversary.

1.1 Problem Definition and Motivation

Machine Learning Algorithms were originally designed under the assumption that both the training and test data come from the same, static distribution. However, in some applications a malicious adversary could carefully manipulate the data and exploit the vulnerabilities of the Learning Algorithm. Some systems are equipped with learning capability and can be hacked via the data that they receive from the environment. This is popularly known as *Poisoning Attacks* against the Learning Algorithm. Traditional Machine Learning Algorithms are not robust to such *Poisoning Attacks*. In order to prevent attacks against Machine Learning Algorithms, it is important to formulate and understand the attack strategies against such algorithms. We have thus studied *Poisoning Attacks* on Machine Learning Algorithms as well as we have tried to see if Robust Optimization techniques could be used to counteract such attacks.

Our Problem Definition can be thus divided into two subparts :

- **Identification of Optimal Training-Set attacks against Machine Learning Algorithms** - We assume that the adversary has a target model in mind and he wants the learned model to be close to this target model. We use a bilevel optimization framework to identify the minimal manipulation to the training data that could bring the learned model as close as possible to the attacker's target model.

- **Learning from poisoned training data using Robust Optimization technique** - Once the training data has been poisoned successfully we would like to learn a model using this poisoned data using Robust Optimization techniques.

1.2 Related Work

There are papers[2, 14] that talk about the various types of attacks against Machine Learning Algorithms as well as possible defense strategies. The different types of attacks are *Poisoning Attacks*, *Evasion Attacks* and *Reverse Engineering attacks*. *Poisoning Attacks* usually happens during the training phase, where an adversary poisons the training data strategically. *Evasion Attacks* typically happens during test time where an adversary carefully crafts the data so as to evade the classifier resulting in a misclassification. *Reverse Engineering attacks* is another type of attack against Machine Learning Algorithm where an adversary with th help of carefully selected test data tries to determine the Machine Learning Algorithm that is deployed. In [8, 19, 21] various label flipping attack strategies against SVM have been discussed. In [9] the author has presented a formal framework for poisoning the features of the training data in a greedy manner. [10, 11], [20] and [13] discusses poisoning attacks against Clustering, LASSO and Deep Neural Networks respectively. [1] discusses reverse engineering attack strategy against SVM as well has presented a novel randomization technique to prevent such attacks. In [16] a bilevel optimization framework for poisoning attacks has been presented.
There are several papers on robust learning under noisy data[5, 6, 17, 4, 3, 10, 13]. There are papers that deal with learning under missing data assumption[7, 17]. [15] discusses the estimation of mean and covariance of data under adversarial contamination. [18] discusses how to retrieve true ranking of items from crowdsourced data which could possibly be contaminated by a group of adversaries. [12] discusses how to learn distributions as well as model parameters under adversarial contamination.

They discuss a more general setting which could be directly used for learning purpose. We have used their algorithm for learning from poisoned data.

1.3 Our Contributions

Our main contributions are :

- **Simulation of Poisoning attacks** - We have tried to simulate poisoning attacks against SVM[16]. For this purpose we have used Synthetic as well as Real Dataset[1]. We could show that it is possible for an adversary to manipulate the training data in a way that the resulting model is closer to the target model.

- **Robust Learning** - We have used Robust Optimization Techniques[12] to learn from the poisoned training data. We have shown that Robust Optimization helps us to achieve decent accuracy when we learn SVM using poisoned training data.

1.4 RoadMap

In the rest of this document we shall first discuss the poisoning attack formulation in Chapter 2. Further Chapter 3 discusses the algorithm used for Robust Learning in the adversarial setting. Chapter 4 discusses the experimental results and finally we conclude with some future work that can be explored.

[1]http://www3.dsi.uminho.pt/pcortez/wine/

Chapter 2

Training-Set Attacks

2.1 Assumptions

- The attack model assumes that the adversary has full knowledge of the machine learning algorithm and has access to the entire training data.
- Attacker is restricted to manipulate only a fraction of the training data.

The attacker wants minimum training-set manipulation to attack the learned model. The machine learning optimization problem is :

$$\begin{aligned} \hat{\theta}_D \in \text{argmin}_{\theta \in \Theta} \quad & O_L(D, \theta) \\ \text{such that } \; g_i(\theta) \leq 0, \quad & i = 1...m \\ h_i(\theta) = 0, \quad & i = 1...p \end{aligned}$$

where D is the training data. In machine learning algorithm D is an iid sample from the underlying distribution. $O_L(D, \theta)$ is the learner's objective. The g and h functions are potentially nonlinear, together with hypothesis space Θ they determine the feasible region. $\hat{\theta}_D$ is the learned model.
Let the original training data be D_0. The attacker can manipulate the training data in such a way that the learned model is beneficial to him. He can do so by say, adding malicious data points to D_0 or by manipulating the features or labels or both in the training data. The resulting dataset D is such that the learned model $\hat{\theta}_D$ is beneficial to the attacker, example, evading spam filters for certain emails etc. The attacker thus has a risk function $R_A(\hat{\theta}_D)$ to minimize which is defined as $R_A(\hat{\theta}_D) = ||\hat{\theta}_D - \theta^*||$ The attacker could also be restricted by certain feasible manipulations or the amount

of manipulation he can make to the training data D_0. The former can be encoded as a search space over $\mathbb{D}$ from which the attacker chooses D, i.e. $D \in \mathbb{D}$, where $\mathbb{D} = \{D : |D_{\Delta} D_0| \leq \beta\}$. The latter can be encoded as an effort function $E_A(D, D_0) = ||X - X_0||_F$. Let us define $O_A(D, \hat{\theta}_D) = R_A(\hat{\theta}_D) + E_A(D, D_0)$, thus the training-set attack problem becomes :

$$\begin{aligned} \min_{D \in \mathbb{D}, \hat{\theta}_D} \quad & O_A(D, \hat{\theta}_D) \\ \text{such that} \quad \hat{\theta}_D \in \operatorname{argmin}_{\theta \in \Theta} \quad & O_L(D, \theta) \\ \text{s.t.} \quad g(\theta) \leq 0, \quad & h(\theta) = 0. \end{aligned}$$

The machine learning equation appears as the constraint of the above optimisation problem. The optimisation over D is called the upper level problem and the optimisation over θ is called the lower level problem.

2.2 Training-Set attacks using KKT conditions

Bilevel optimisation problems are NP hard in general. Such problems can be solved efficiently under following conditions :

1. The attack space $\mathbb{D}$ should be differentiable.
2. The objective O_L should be convex and regular.

If the above conditions are satisfied, the bilevel problem can be replaced by a single level optimisation problem using KKT conditions of the lower level problem. The attacker's optimisation problem becomes :

$$\begin{aligned} \min_{D \in \mathbb{D}, \hat{\theta}_D} \quad & O_A(D, \theta) \\ \text{s. t. } \partial_\theta(O_L(D, \theta) + \lambda^T g(\theta) + \mu^T h(\theta)) &= 0 \\ \lambda_i g_i(\theta) = 0, \quad i &= 1...m \\ g(\theta) \leq 0, h(\theta) = 0, \lambda &\geq 0 \end{aligned}$$

In the above optimisation problem, λ_i's and μ_i's are KKT multipliers for the lower level constraints g and h respectively. The single level optimisation problem can be solved use projected gradient descent method. In any iteration t, the data D is updated as follows :

$$D^{(t+1)} = \mathrm{Prj}_{\mathbb{D}}\Big(D^{(t)} + \alpha_t \nabla_D O_A(D, \theta^{(t)})\Big|_{D=D^{(t)}}\Big)$$

where α_t is the step size. In the second step of iteration t, we fix $D^{(t+1)}$ and solve for $\theta^{(t+1)}$, $\mu^{(t+1)}$ and $\lambda^{(t+1)}$.

The gradient of the upper-level problem is computed by the chain rule

$$\nabla_D O_A(D,\theta) = \nabla_\theta O_A(D,\theta)\frac{\partial \theta}{\partial D}$$

$\nabla_\theta O_A(D,\theta)$ is easy to compute. It is often difficult to compute $\frac{\partial \theta}{\partial D}$. The equalities in the KKT conditions define a function $\mathbf{f} : \mathbb{R}^{n+d+m+p}$ -> $\mathbb{R}^{d+m+p}$. A dataset D and its learned model θ, λ, μ will satisfy $\mathbf{f}(D,\theta,\lambda,\mu) = 0$. According to the implicit function theorem, if $\mathbf{f}$ is continuously differentiable w.r.t its parameters and the Jacobian matrix $\left[\frac{\partial f}{\partial \theta}\middle|\frac{\partial f}{\partial \lambda}\middle|\frac{\partial f}{\partial \mu}\right]$ is of full rank, then $\mathbf{f}$ induces a unique function $(\theta,\lambda,\mu) = i(D)$ in an open neighbourhood. Furthermore,

$$\frac{\partial i}{\partial D} = -\left[\frac{\partial f}{\partial \theta}\middle|\frac{\partial f}{\partial \lambda}\middle|\frac{\partial f}{\partial \mu}\right]^{-1}\left(\frac{\partial f}{\partial D}\right)$$

The $\frac{\partial \theta}{\partial D}$ is the first d rows of $\frac{\partial i}{\partial D}$.

2.3 Training-Set attacks against SVM

Let $D_0 = (X_0, y_0)$ be the original training dataset. We assume that the attacker is only allowed to change the features. Therefore the attacker's search space $\mathbb{D} = \{(X, y_0) | X \in \mathbb{R}^{d\times n}\}$. The SVM parameters are $\hat{w}_D$ and $\hat{b}_D$ which is learnt by solving the following lower level optimisation problem :

$$\begin{aligned} O_L(D,w,b,\xi) &= \frac{1}{2}||w||_2^2 + C\sum_i \xi_i \\ g_i &= 1 - \xi_i - y_i(x_i^T w + b) \\ g_{i+n} &= -\xi_i \end{aligned}$$

for $i = 1...n$, where ξ_i is the hinge loss and C the regularization parameter. The attacker's risk function is :

$$R_A(\hat{w}_D) = \frac{1}{2}||\hat{w}_D - w^*||_2^2$$

where w^* is the attacker's target parameter. The attacker's effort function is :

$$E_A(D, D_0) = \frac{\lambda}{2}||X - X_0||_F^2$$

The SVM KKT conditions can be reduced to :

$$w_j - \alpha_i \sum_i \mathbb{I}_1(1 - y_i(x_i^T w + b) \geq 0) y_i x_{ij} = 0, \quad j = 1...d$$

where $\mathbb{I}_1(z) = 1$ is z is true otherwise 0, $\alpha_i \in [0, C]$. The bilevel optimisation for SVM thus becomes :

$$\begin{aligned} &\min_{D \in \mathbb{D}, w} \quad \frac{1}{2}||w - w^*||_2^2 + \frac{\lambda}{2}||X - X_0||_F^2 \\ \text{s.t.} \quad & w_j - \alpha_i \sum_i \mathbb{I}_1(1 - y_i(x_i^T w + b) \geq 0) y_i x_{ij} = 0, \quad j = 1...d \end{aligned}$$

We apply gradient descent to solve the above optimisation problem. The gradient is :

$$\begin{aligned} &\nabla_X = \nabla_w R_A(w)\Big|_{\hat{w}(X)} \frac{\partial \hat{w}(X)}{\partial X} + \nabla_X E_A(D, D_0) \\ &\text{with} \\ &\nabla_w R_A(w) = w - w^* \\ &\nabla_{x_{ij}} E_A(D, D_0) = \lambda(X_{ij} - X_{0,ij}) \end{aligned}$$

To compute $\frac{\partial \hat{w}}{\partial X}$ we use the implicit function theorem. The Jacobian matrix of $\frac{\partial f}{\partial w}$ is the identity matrix. The Jacobian matrix $\frac{\partial f}{\partial X}$ at row $j\prime$ and column ij is :

$$\left[\frac{\partial f}{\partial X}\right]_{j\prime, ij} = -\alpha_i y_i \mathbb{I}_1(j\prime = j) \mathbb{I}_1(1 - y_i x_i^T w \geq 0)$$

The element at row $j\prime$ and column ij in $\frac{\partial \hat{w}}{\partial X}$ is :

$$\left[\frac{\partial \hat{w}(X)}{\partial X}\right]_{j\prime, ij} = \alpha_i y_i \mathbb{I}_1(j\prime = j) \mathbb{I}_1(1 - y_i x_i^T w \geq 0)$$

Thus an instance x will have an effect on w only if it is a support vector. When the features are discrete then we need to take sub-gradients in which case the updates remain the same except that we have to take projection on the feasible space for X.

2.4 Training-Set attacks against Linear Regression

We consider ordinary least squares regression. Let $D = (X, y)$ be the training data. The maximum likelihood estimate is given by :

$$O_L(D, w) = ||y - Xw||^2$$

The above problem is convex and unconstrained and the KKT condition is the ordinary least square solution :

$$\hat{w}_D - (X^T X)^{-1} X^T y = 0$$

The attacker's optimisation problem thus becomes :

$$\begin{aligned} \min_{D \in \mathbb{D}, w} \quad & \frac{1}{2}||w - w^*||_2^2 + \frac{\lambda}{2}||X - X_0||_F^2 \\ \text{s.t.} \quad & \hat{w}_D - (X^T X)^{-1} X^T y = 0 \end{aligned}$$

where $E_A(D, D_0) = \frac{\lambda}{2}||X - X_0||_F^2$ and $R_A(\hat{w}_D) = \frac{1}{2}||w - w^*||_2^2$
We apply gradient descent to solve the above optimisation problem. The gradient is :

$$\begin{aligned} &\nabla_X = \nabla_w R_A(w)\Big|_{\hat{w}(X)} \frac{\partial \hat{w}(X)}{\partial X} + \nabla_X E_A(D, D_0) \\ &\text{with} \\ &\nabla_w R_A(w) = w - w^* \\ &\nabla_{x_{ij}} E_A(D, D_0) = \lambda(X_{ij} - X_{0,ij}) \end{aligned}$$

The matrix $\frac{\partial \hat{w}_i(X)}{\partial X}$ is :

$$\begin{aligned} \frac{\partial \hat{w}_i(X)}{\partial X} &= \frac{\partial [e_i^T (X^T X)^{-1} (X^T y)]}{\partial X} \\ &= \frac{\partial Tr[(X^T X)^{-1} (X^T y e_i^T)]}{\partial X} \\ &= -X (X^T X)^{-1} (X^T y e_i^T + e_i y^T X)(X^T X)^{-1} + \\ &(y e_i^T (X^T X)^{-1}) \end{aligned}$$

2.5 Training-Set attacks against K-means Clustering

Let $D = (X, y)$ be the training dataset. The K-means optimisation problem is as follows :

$$O_L = \min_{\mu_1 \dots \mu_k} \sum_{j=1}^{k} \sum_{i=1}^{n} \frac{1}{2} ||X_i - \mu_j||^2 \mathbb{I}(X_i, \mu_j) \tag{2.1}$$

where $\mathbb{I}(X_i, \mu_j) = 1$ is $X_i \in$ cluster centered at μ_j and 0 otherwise and μ_j's are the cluster centres. Let the target means for the attacker be μ_j^*, $j = 1 \dots k$. The bilevel optimisation problem becomes :

$$\min_{D, \mu_1 \dots \mu_k} \quad \sum_{j=1..k} \frac{1}{2} ||\mu_j - \mu_j^*||_2^2 + \frac{\lambda}{2} ||X - X_0||_F^2$$

$$\text{such that} \quad \mu_1 \dots \mu_k \in \text{argmin} \quad O_L$$

We substitute μ_j by $X f_j$ where $(f_j)_i = (1/n_j)\mathbb{I}(X_i, \mu_j)$ where n_j are the number of data points belonging to the cluster centred at μ_j. We apply gradient descent to solve the above optimisation problem. The gradient is :

$$\nabla_X = \sum_{j=1}^{k} X f_j * f_j + \nabla_X E_A(D, D_0)$$

with

$$\nabla_{x_{ij}} E_A(D, D_0) = \lambda (X_{ij} - X_{0,ij})$$

Chapter 3

Algorithm for Robust Learning

General Setting : Given convex functions f_1, f_2 ... f_n: $\mathcal{H}$ -> $\mathbb{R}$ where $\mathcal{H}$ is a convex parameter space. A subset $I_{good} \subseteq [n]$ of size αn, $f_i \sim p^*$, and the remaining f_i are chosen by the adversary. Let $\overline{f}$ denote the mean of f under p^*, i.e. $\overline{f} = \mathbb{E}_{f \sim p^*}\ [f(w)]$ for $w \in \mathcal{H}$, the goal is to find a parameter $\hat{w}$ such that $\overline{f}(\hat{w})$ - $\overline{f}(w^*)$ is small, where w^* is the minimiser of $\overline{f}$. Let r denote the l_2 radius of $\mathcal{H}$, i.e. $r = max_{w \in \mathcal{H}} ||w||_2$.

- This setting could be used for mean estimation corresponding to $f_i(w) = ||w - x_i||_2^2$, linear regression corresponding to $f_i(w) = (y_i - < w, x_i >)^2$, logistic regression corresponding to $f_i(w) = = log(1 + exp(-y_i < w, x_i >))$ and SVM parameter estimation corresponding to $f_i = max(0, 1 - y_i w^T x_i)$.

- We could also think of good data coming from one distribution and adversarial data coming from some other distribution. This setting can therefore be used for robustly learning mixture of distributions.

Algorithm 1 is used to robustly learning the parameter $\hat{w}$. The algorithm outputs a parameter w_i for each of the functions f_i with the guarantee that atleast one of the values will be close to the true minimiser w^*. The parameter $\hat{w}$ could be approximated by the weighted average of the parameters w_i where the weights c are output by **Algorithm 2**.

3.1 Idea behind the algorithm

The algorithm is an SDP along with an along removal step. At high level, the algorithm works by assigning a w_i parameter to each of the f_i's and minimize $\sum_{i=1}^{n} f_i(w_i)$ subject to regularization which ensures that all the w_i's lie close to each other, informally within an ellipse. This ensures that whenever an adversary affect the shape

of the ellipse more than a small amount, they are necessarily outliers and can be identified and removed.
The outlier removal step assumes that if a function f_i is drawn from the true distribution p^* then there would be many other functions $f_j, j \neq i$, that are similar to f_i. This is achieved by the optimisation problem given in **Algorithm 2**. The optimisation roughly finds out a parameter $\widetilde{w}_i$ that minimises f_i, where $\widetilde{w}_i$ is the average of atleast $\frac{\alpha n}{2}$ distinct parameters $\hat{w}_j$. Then, the weight c_i of the *ith* data point is downweight based on the value of $f_i(\widetilde{w}_i) - f_i(\hat{w}_i)$. The weight c_i is multiplied by $1 - \eta(f_i(\widetilde{w}_i) - f_i(\hat{w}_i))$ for some appropriate η.

Algorithm 1: Algorithm for fitting p^*

```
Input : $f_1, ..., f_n$
Initialise : $c \leftarrow [1, ..., 1] \in \mathbb{R}^n$
Set $\lambda \leftarrow \frac{\sqrt{8\alpha n S}}{r}$
while true do
    Let $\hat{w}_{1:n}, \hat{Y}$ be the solution to

        $\min_{w_1,...,w_n,Y} \sum_{i=1}^{n} c_i f_i(w_i) + \lambda tr(Y)$
        subject to $w_i w_i^T \leq Y$ for all i=1,...,n.

    If tr$(\hat{Y}) \leq \frac{6r^2}{\alpha}$ then
        return $\hat{w}_{1:n}, Y, c$
    else
        $c \leftarrow$ UPDATEWEIGHTS$(c, \hat{w}_{1:n}, \hat{Y})$
    end if
end while
```

Algorithm 2: Algorithm for updating weights c to downweight outliers.

```
Procedure UPDATEWEIGHTS$(c, \hat{w}_{1:n}, \hat{Y})$
  for $i = 1, ..., n$ do
    Let $\widetilde{w}_i$ be the solution to
      $\min_{\widetilde{w}_i, a_{i1}, .., a_{in}} f_i(\widetilde{w}_i)$
      subject to $\widetilde{w}_i = \sum_{j=1}^{n} a_{ij} \hat{w}_j$, $0 \leq a_{ij} \leq \frac{2}{\alpha n}$, $\sum_{j=1}^{n} aij = 1$.
    Let $z_i \leftarrow f_i(\widetilde{w}_i) - f_i(\hat{w}_i)$
    end for
    $z_{max} \leftarrow \max\{z_i | c_i \neq 0 \}$
    $c_i^{'} \leftarrow c_i . \frac{z_{max} - z_i}{z_{max}}$ for $i$=1,..,n
    return $c^{'}$
end procedure
```

3.2 Spectral norm of gradients

$$S = \max_{w \in \mathcal{H}} \frac{1}{\sqrt{|\mathcal{I}_{good}|}} \left|\left| [\nabla f_i(w) - \nabla \bar{f}(w)]_{i \in \mathcal{I}_{good}} \right|\right|_{op}$$

If we form the matrix of gradients $[\nabla f_{i1}(w) ... \nabla f_{i\alpha n}(w)]$, where $\{i_1, .., i_{\alpha n}\} = \mathcal{I}_{good}$, then S measures the difference between this matrix and its expectation in operator norm, maximised over all $w \in \mathcal{H}$.

3.2.1 Calculation of S for SVM

For SVM we define f_i to be the hinge loss, i.e. $f_i = max(0, 1 - y_i w^T x_i)$. Therefore, the gradient of f_i is :

$$\begin{aligned} \nabla f_i &= -y_i x_i \quad \text{if}(y_i w^T x_i < 1), \\ &= 0 \quad \text{otherwise} \end{aligned}$$

Therefore,

$$S = \frac{1}{\sqrt{|\mathcal{I}_{good}|}} \left|\left| [-y_i(x_i - \mu_i)]_{i \in \mathcal{I}_{good}} \right|\right|_{\sigma_{max}}$$

where μ_i is the expectation of x_i and σ_{max} is the largest singular value of the matrix so formed.

3.2.2 Calculation of S for Linear Regression

For Linear Regression we define f_i as :

$$f_i = \frac{1}{2}(y_i - w^T x_i)^2$$

Therefore, the gradient of f_i is :

$$\nabla f_i = -(y_i - w^T x_i) x_i$$

The Expectation of ∇f_i is :

$$\begin{aligned}\bar{\nabla} f_i &= -y_i \mu_i + E[x_i^T w x_i] \\ &= -y_i \mu_i + E[x_i x_i^T w] \\ &= -y_i \mu_i + \Sigma_i w\end{aligned}$$

Therefore, S is :

$$S = \max_{w \in \mathcal{H}} \frac{1}{\sqrt{|\mathcal{I}_{good}|}} \left|\left| [-(y_i - w^T x_i) x_i + y_i \mu_i - \Sigma_i w]_{i \in \mathcal{I}_{good}} \right|\right|_{\sigma_{max}}$$

3.2.3 Calculation of S for K-means clustering

For k-means clustering we define f_i as :

$$f_i = \frac{1}{2} ||x_i - w||^2$$

where w is the cluster center to which x_i belongs. Therefore, the gradient of f_i is :

$$\nabla f_i = -(x - w)$$

The Expectation of ∇f_i is :

$$\bar{\nabla} f_i = -(\mu_i - w)$$

Therefore, S is :

$$S = \frac{1}{\sqrt{|\mathcal{I}_{good}|}} \left|\left| [\mu_i - x_i]_{i \in \mathcal{I}_{good}} \right|\right|_{\sigma_{max}}$$

Chapter 4

Experiments

4.1 SVM Poisoning attack on Real Dataset

4.1.1 Dataset

We perform experiments on the wine quality dataset[1]. The dataset has n=1599 points and d=11 features. Each feature is normalized to zero mean and unit standard deviation. The wine quality number is threshold at 5 to produce a binary label y. We demonstrate attacks using Linear SVM. The regularisation parameter C is set 1.

No.	feature
1	fixed acidity
2	volatile acidity
3	citric acid
4	residual sugar
5	chlorides
6	free sulfur dioxide
7	density
8	pH
9	sulphates
10	alcohol
11	quality

Table 4.1: Features in the wine quality dataset

[1]http://www3.dsi.uminho.pt/pcortez/wine/

4.1.2 Target vector computation

For demonstration purpose, we assume that the attacker's goal is to make it seem that the feature "alcohol" is correlated with wine quality. We therefore generate the target vector w^* by creating another set of labels $y\prime$ by thresholding $x_{i,alcohol}$ at 0. We then let the attacker risk function be with this vector w^*. We set λ to 0.1. The step-length α_t is set to $0.5/t$. The results of the experiment is shown in Figure 4.1,4.2 and 4.3.

4.1.3 Results

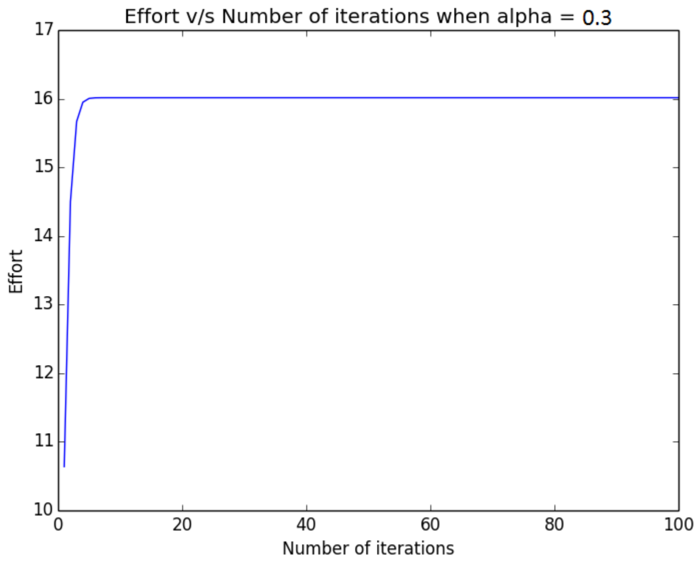

Figure 4.1: Effort versus number of iterations when the adversary wishes to corrupt 70% of the training data

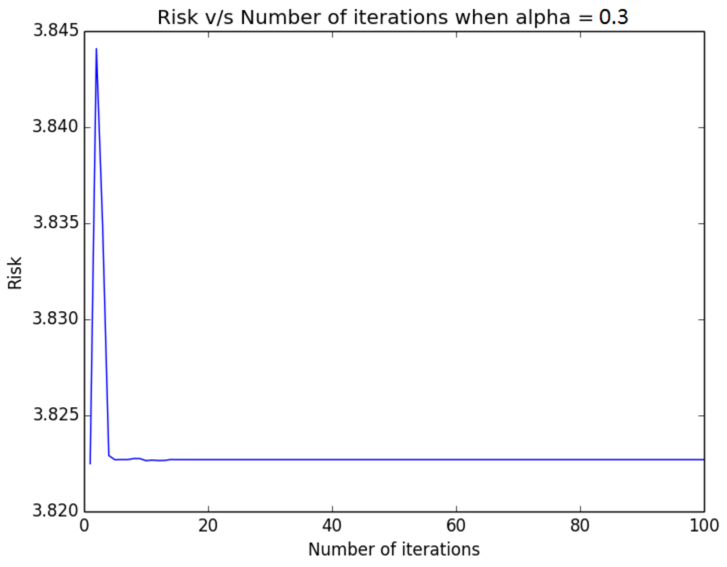

Figure 4.2: Risk versus number of iterations when the adversary wishes to corrupt 70% of the training data

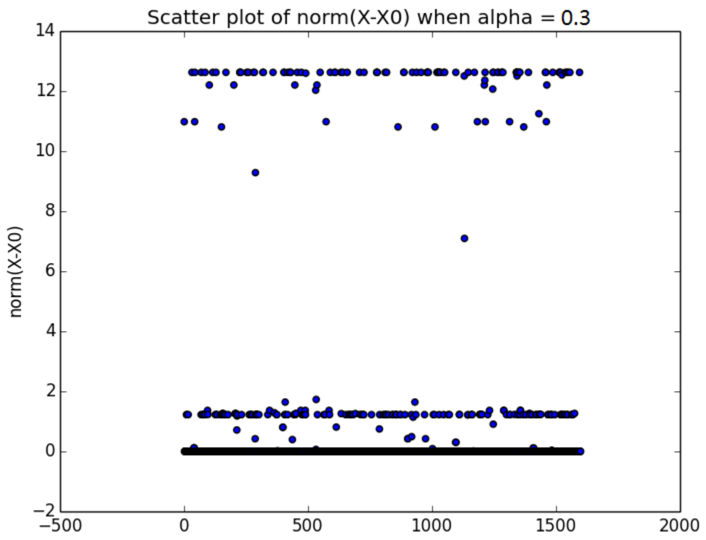

Figure 4.3: Norm of the difference of the original data and corrupt data when the adversary wishes to corrupt 70% of the training data

4.2 SVM poisoning attack on Synthetic Dataset

4.2.1 Dataset

We performed training set poisoning on Synthetic Dataset with n=50 data points and d=10 features. The features were generated from spherical gaussian having random mean and covariance.

4.2.2 Target vector computation

Attacker's target vector w^* was obtained by learning Linear SVM on dataset where another set of labels $y^{'}$ was generated by randomly flipping 50% of the labels. After poisoning, we trained Robust SVM on the poisoned data.

4.2.3 Results

fracPoison	fracPoisonActual
0.0	0.0
0.1	0.020000000000000018
0.9	0.14

Table 4.2: The fraction of data we wish to corrupt versus fraction of data that actually gets corrupted when trying to get the learned model to get close to the target model.

fracPoison	|| wNotRobust-wTrue ||
0.0	0.0
0.1	0.00177302
0.9	0.00417633

Table 4.3: Fraction of data we want to corrupt versus the norm of difference of model learned on poisoned training data and unpoisoned training data.

fracPoison	accuracyS
0.0	100.0
0.1	100.0
0.9	100.0

Table 4.4: Fraction of data we wish to corrupt versus the accuracy achieved when using w_i for prediction.

fracPoison	lossS
0.0	0.0
0.1	0.0
0.9	0.0

Table 4.5: Fraction of data we wish to corrupt versus the hinge loss when using w_i for prediction.

fracPoison	accuracySavg
0.0	57.99999999999999
0.1	57.99999999999999
0.9	57.99999999999999

Table 4.6: Fraction of data we wish to corrupt versus the accuracy achieved when using weighted average of $w_i's$ for prediction.

fracPoison	lossSavg
0.0	48.60681347
0.1	48.60761007
0.9	48.61049927

Table 4.7: Fraction of data we wish to corrupt versus the hinge loss when using weighted average of $w_i's$ for prediction.

- Accuracy on unpoisoned training data : 64.0 %
- Total Hinge Loss on unpoisoned training data : 56.56618687
- ||wTarget-wTrue|| : 2.5365158433

4.2.3.1 Effort v/s Fraction of data poisoned

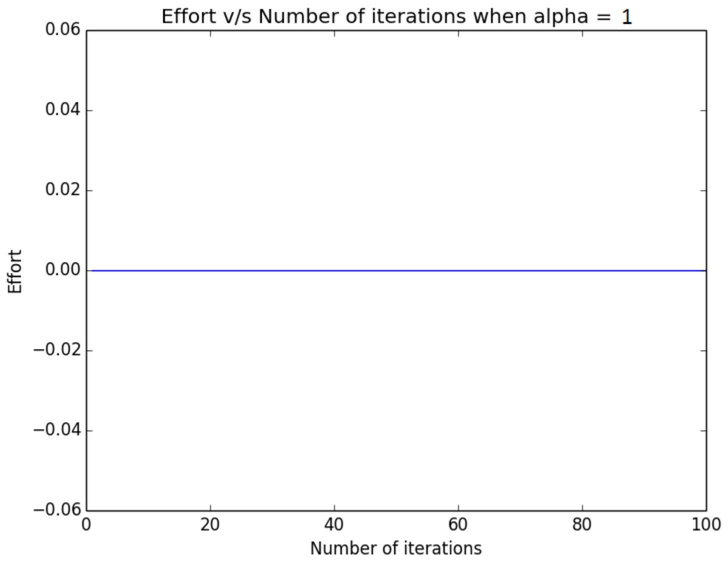

Figure 4.4: Effort versus Number of iterations when we poison 0% data.

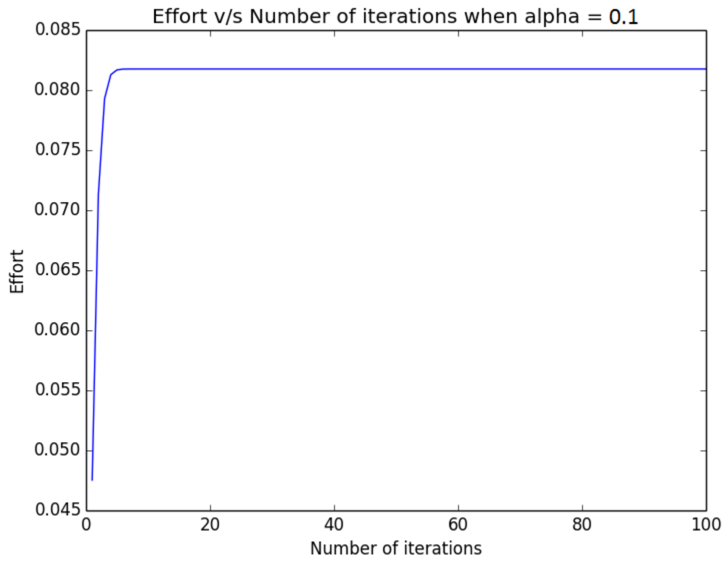

Figure 4.6: Effort versus Number of iterations when we poison 90% data.

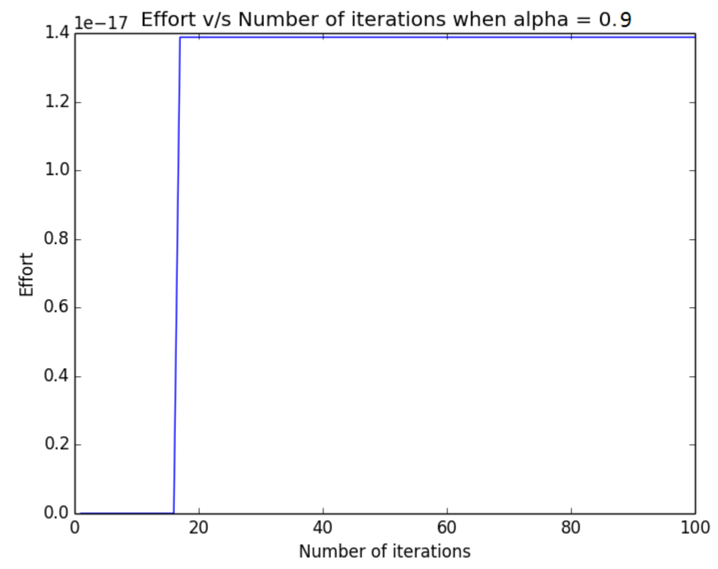

Figure 4.5: Effort versus Number of iterations when we poison 10% data.

4.2.3.2 Risk v/s Fraction of data poisoned

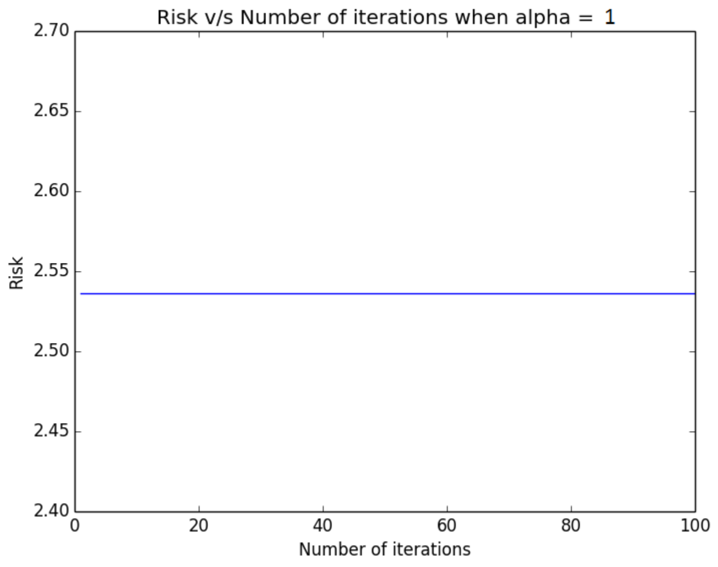

Figure 4.7: Risk versus Number of iterations when we poison 0% data.

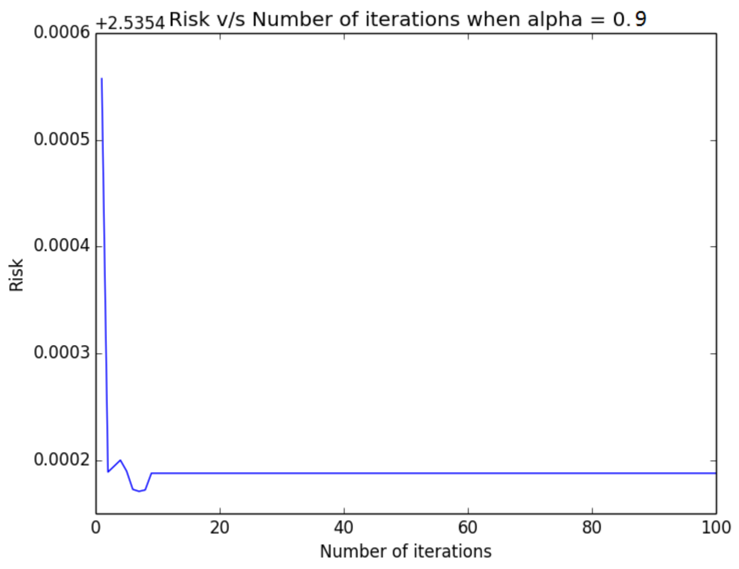

Figure 4.8: Risk versus Number of iterations when we poison 10% data.

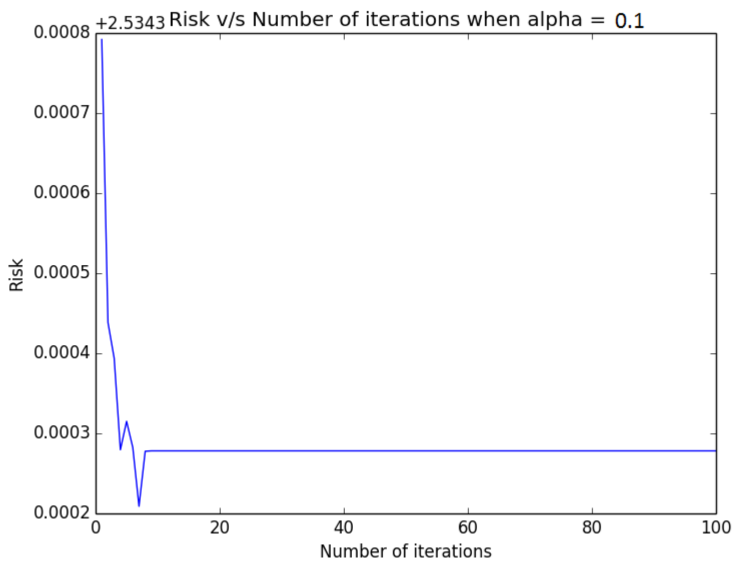

Figure 4.9: Risk versus Number of iterations when we poison 90% data.

4.2.3.3 Scatter plot of norm of difference of Poisoned data points and Unpoisoned data points

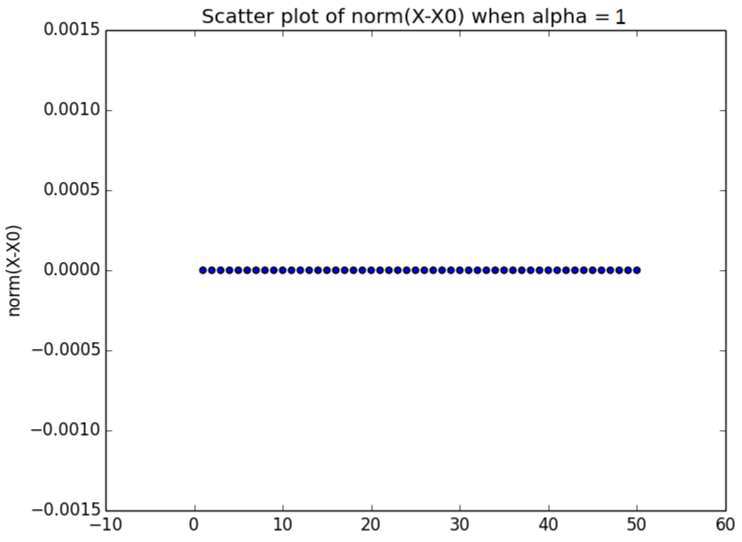

Figure 4.10: Scatter plot of norm of difference of Poisoned data points and Unpoisoned data points when trying to poison 0% of the training data

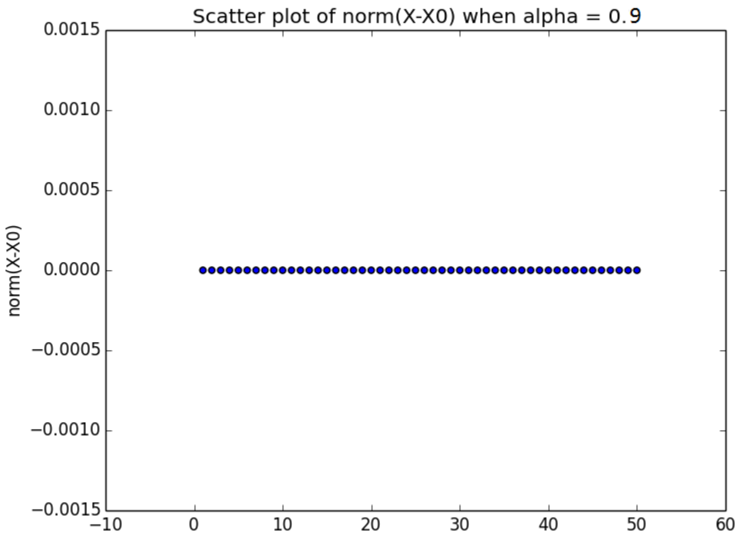

Figure 4.11: Scatter plot of norm of difference of Poisoned data points and Unpoisoned data points when trying to poison 10% of the training data

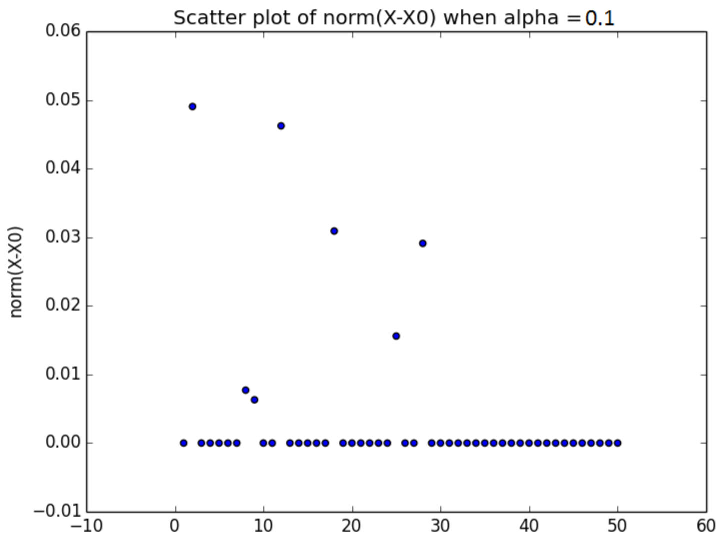

Figure 4.12: Scatter plot of norm of difference of Poisoned data points and Unpoisoned data points when trying to poison 90% of the training data

Chapter 5

Conclusion and Future Work

We have performed experiments to demonstrate that it is possible for an adversary to manipulate the training data strategically to get the learned model closer to a certain target model. In order to robustly learn a model it is thus necessary to understand and study the poisoning mechanism. We have also shown that we can robustly learn models with certain accuracy. We have studied the mechanism for SVM.
As a part of Future Work, we would like to perform similar studies on other machine learning algorithms such as Regression and Clustering. Also, the robust learning algorithm is an SDP and is not scalable, so we would also like to come up with an algorithm which would help us learn in an adversarial setting on a large data.

Bibliography

[1] Ibrahim M Alabdulmohsin, Xin Gao, and Xiangliang Zhang. Adding robustness to support vector machines against adversarial reverse engineering. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 231–240. ACM, 2014. 2

[2] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 16–25. ACM, 2006. 2

[3] Aharon Ben-Tal, Sahely Bhadra, Chiranjib Bhattacharyya, and Arkadi Nemirovski. Efficient methods for robust classification under uncertainty in kernel matrices. *Journal of Machine Learning Research*, 13(Oct):2923–2954, 2012. 2

[4] Sahely Bhadra, Sourangshu Bhattacharya, Chiranjib Bhattacharyya, and Aharon Ben-Tal. Robust formulations for handling uncertainty in kernel matrices. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 71–78, 2010. 2

[5] Chiranjib Bhattacharyya. Robust classification of noisy data using second order cone programming approach. In *Intelligent Sensing and Information Processing, 2004. Proceedings of International Conference on*, pages 433–438. IEEE, 2004. 2

[6] Chiranjib Bhattacharyya, LR Grate, Michael I Jordan, L El Ghaoui, and I Saira Mian. Robust sparse hyperplane classifiers: application to uncertain molecular profiling data. *Journal of Computational Biology*, 11(6):1073–1089, 2004. 2

[7] Chiranjib Bhattacharyya, Pannagadatta K Shivaswamy, and Alex J Smola. A second order cone programming formulation for classifying missing data. In *Advances in neural information processing systems*, pages 153–160, 2005. 2

[8] Battista Biggio, Blaine Nelson, and Pavel Laskov. Support vector machines under adversarial label noise. In *Asian Conference on Machine Learning*, pages 97–112, 2011. 2

[9] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012. 2

[10] Battista Biggio, Ignazio Pillai, Samuel Rota Bulò, Davide Ariu, Marcello Pelillo, and Fabio Roli. Is data clustering in adversarial settings secure? In *Proceedings of the 2013 ACM workshop on Artificial intelligence and security*, pages 87–98. ACM, 2013. 2

[11] Battista Biggio, Konrad Rieck, Davide Ariu, Christian Wressnegger, Igino Corona, Giorgio Giacinto, and Fabio Roli. Poisoning behavioral malware clustering. In *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*, pages 27–36. ACM, 2014. 2

[12] Moses Charikar, Jacob Steinhardt, and Gregory Valiant. Learning from untrusted data. *CoRR*, abs/1611.02315, 2016. URL http://arxiv.org/abs/1611.02315. 2, 3

[13] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Analysis of classifiers' robustness to adversarial perturbations. *arXiv preprint arXiv:1502.02590*, 2015. 2

[14] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016. 2

[15] Kevin A. Lai, Anup B. Rao, and Santosh Vempala. Agnostic estimation of mean and covariance. *CoRR*, abs/1604.06968, 2016. URL http://arxiv.org/abs/1604.06968. 2

[16] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *AAAI*, pages 2871–2877, 2015. 2, 3

[17] Pannagadatta K Shivaswamy, Chiranjib Bhattacharyya, and Alexander J Smola. Second order cone programming approaches for handling missing and uncertain data. *Journal of Machine Learning Research*, 7(Jul):1283–1314, 2006. 2

[18] Jacob Steinhardt, Gregory Valiant, and Moses Charikar. Avoiding imposters and delinquents: Adversarial crowdsourcing and peer prediction. In *Advances in Neural Information Processing Systems*, pages 4439–4447, 2016. 2

[19] Han Xiao, Huang Xiao, and Claudia Eckert. Adversarial label flips attack on support vector machines. In *Proceedings of the 20th European Conference on Artificial Intelligence*, pages 870–875. IOS Press, 2012. 2

[20] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. Is feature selection secure against training data poisoning? In *International Conference on Machine Learning*, pages 1689–1698, 2015. 2

[21] Huang Xiao, Battista Biggio, Blaine Nelson, Han Xiao, Claudia Eckert, and Fabio Roli. Support vector machines under adversarial label contamination. *Neurocomputing*, 160:53–62, 2015. 2