L-BFGS Based Adversarial Input Attacks Against Support Vector Machine

Da Zhong, Yi Sun

May 2017

Abstract

Support Vector Machine have been found to be vulnerable against adversarial input attacks. Imperceptible and specific modifications on a correctly classified image could fool the svm to output a totally different label. In this project, we performed targeted adversarial input attacks based on L-BFGS methods, and successfully mislead the SVM to output our targeted label with minimum distortion.

1 Introduction

Although S have achieved great success in classification tasks and outperform humans in areas such as image classification, they have been discovered to be susceptible to adversarial input attacks [4]. A small and intentionally chosen perturbation added to a correctly classified image can mislead the network to output a totally different label, even though the perturbation is so small and appears imperceptible to humans [4] [6].

An illustration of adversarial input attacks is shown in Figure 1. The first row images are valid inputs with their correct prediction labels shown above, and the second row images are adversarial inputs with their corresponding prediction labels shown below. Apparently, the adversarial inputs have successfully breach the neural network to output all wrong labels. However, the distortions between valid and adversarial inputs are so trivial to be recognized by humans.

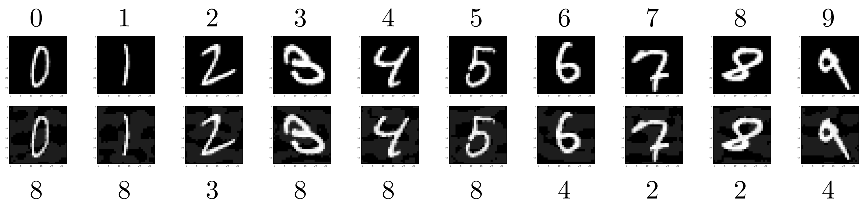

Figure 1: Valid inputs, adversarial inputs and their prediction labels.

There are broadly two types of adversarial input attacks: one is non-targeted attack and the other is targeted attack [7]. Non-targeted attack only cares to fool the model with a wrong label, as long as it's different than the genuine one. While targeted attack specifies the target label and misleads the model to output the specific prediction. Targeted attack is more powerful than non-targeted attack.

Our objective in this project is to perform targeted attacks SVM. In detail, given an image x with a correct classification label $y = \text{classifier}(x)$, where classifier() is the function of the model, we aim to:

- find an image x' whose classification label is y', such that $y' = \text{classifier}(x') \neq y$;
- ensure that $\|x' - x\| \leq \delta$, where δ is an upper bound of the distortion from x to x'.

2 Preliminary and Notation

2.1 Support Vector Machine

Suport Vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. There are not only linear SVM, which is simple, but also complex SVM with kernels. For linear SVM We are given a training dataset of

$$(\overrightarrow{x_1}, y_1), \cdots, (\overrightarrow{x_n}, y_n)$$

where the y_i are either 1 or -1, each indicating the class to which the point$\overrightarrow{x_i}$ belongs. Each $\overrightarrow{x_i}$ is a p-dimensional real vector. We want to find the "maximum-margin hyperplane" that divides the group of points $\overrightarrow{x_i}$ for which $y_i = 1$ from the group of points for which $y_i = -1$, which is defined so that the distance between the hyperplane and the nearest point $\overrightarrow{x_i}$ from either group is maximized.

Any hyperplane can be written as the set of points $\vec{x_i}$ satisfying:

$$\vec{w}\vec{x} - b = 0$$

Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors. where $\vec{w}$ is the (not necessarily normalized) normal vector to the hyperplane. This is much like Hesse normal form, except that $\vec{w}$ is not necessarily a unit vector. The parameter $\frac{b}{\|\vec{w}\|}$ determines the offset of the hyperplane from the origin along the normal vector $\vec{w}$

2.2 Adversary Capabilities

How much information of the neural network that an adversary can possess determines the capability of the adversary. Such information includes network architecture, network parameters, training algorithms, activation functions, etc.

However, it is a realistic and reasonable assumption that the adversary has full access to all the information of a targeted neural network due to the property of transferability [5]. The transferability of a neural network states that an adversarial input of one network can also be misleading to other networks. In other words, it is possible to construct a substitute of the target network with the same classification task, and generate adversarial inputs to the substitute in a white-box manner. And most probably those adversarial inputs are transferable to the black-box target network and can perform adversary attacks.

Although the adversary can be presented in both stages of training and testing, we only consider test time adversary attacks in this project.

2.3 Distance Metric

The distance between valid input and adversarial input is measured by a metric $D(x, x')$, and in most literature [4] [6] [7] $D(x, x')$ takes the form of L_p norms.

$$D(x, x') = \|x - x'\|_p = \left(\sum_{i=1}^{n} |x_i - x'_i|^p \right)^{1/p} \tag{1}$$

Typically the distance is defined by L_0, L_2 or L_∞. In detail,

- L_0 distance measures the number of pixels that have been altered from x to x'.
- L_2 is the standard Euclidean distance (root mean square) between x and x'.
- L_∞ distance measures the maximum amount of perturbation within all pixels from x to x' since $\|x-x'\|_\infty = \max(|x_1-x'_1|, |x_2-x'_2|, \cdots, |x_n-x'_n|)$. L_∞ distance is upper bounded by the maximum value the pixel can take.

3 Problem Formulation

The process of generating adversarial inputs can be formulated into solving an optimization problem:

$$\text{minimize } \|x' - x\|, \ s.t. \ \text{label}(x') = y', x' \in [0, 255]^n \tag{2}$$

Here the constraints ensure that the output is y' and the optimal x' is the adversarial input closest to x. Considering that the inputs are pixels of an image, the range of x' is $[0, 225]^n$.

Since the neural network and label function label() is highly non-linear and non-convex, it is quite difficult to solve the problem directly. However, when an input x' is predicted as y', in most cases the loss function in terms of y' is minimized. In this case, the problem can be approximated with an alternative one shown as follows and solved by box-constrained L-BFGS [9]:

$$\text{minimize } C\|x' - x\| + \ell_{F,y'}(x'), \ s.t. \ x' \in [0, 255]^n \tag{3}$$

A common choice for the loss function $\ell(x)$ is cross-entropy. C is a constraint scalar. By performing line search of different C, adversarial input x' with minimum distortion can be achieved.

4 L-BFGS Based Adversarial Attack

L-BFGS method is developed from Newton's method. Compared with Newton's method, L-BFGS optimization achieves resource-efficient in both computation and storage.

4.1 Newton's Method

To minimize the target function $f(x)$, the second order in Taylor formula of $f(x)$ is computed,

$$\phi(x) = f(x_k) + \nabla f(x_k)(x - x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k) \tag{4}$$

When $f(x)$ reaches its optimum, the Jacobian matrix should be zero,

$$\nabla \phi(x) = 0 \tag{5}$$

which means

$$g_k + H_k(x - x_k) = 0 \tag{6}$$

With an initial search point, it is always possible to find a proper search direction and reach the optimum through an iteration process.

$$x_{k+1} = x_k - a * H_k^{-1} g_k \tag{7}$$

Although Newton's method is simple in theory, it is difficult to apply in practice when the order of x is very high. Because,

- calculation is too complicated for high-dimensional H^{-1}.
- the storage of H^{-1} is too large.

4.2 BFGS Method

In order to reduce calculation, BFGS method approximate H^{-1} using vectors s_k and y_k, where

$$s_k = x_{k+1} - x_k,\ y_k = g_{k+1} - g_k \tag{8}$$

It is easy to prove that

$$y_k \approx D_{k+1} s_k \tag{9}$$

where D is an approximation of H^{-1}. To calculate D, further derive Jacobian matrix ΔD_k

$$\Delta D_k = \frac{s_k s_k^T}{s_k^T y_k} - \frac{D_k y_k y_k^T D_k}{y_k^T D_k y_k} \tag{10}$$

$$D_{k+1} = D_k + \Delta D_k \tag{11}$$

Follow the steps above, it's efficient to calculate D through an iterative process with low cost of calculation complexity.

However, the storage required by BFGS is still large. To store all the parameters, the storage scale is $O(N^2)$.

4.3 L-BFGS Method

To further reduce the storage amount, L-BFGS (limit-memory BFGS) method stores only s_k vectors and y_k vectors instead of D matrix. Another difference is that only the latest m pairs of $[s_k, y_k]$ are stored, $m \ll N$. In this way the storage scale can be reduced to $O(mN)$.

4.4 L-BFGS Based Adversarial Attack

Intuitively, with a relatively small C, it is easier to craft adversarial input x' that fools the network since the distortion in the objective function, Equation 5, plays a minor role. In such cases, however, we could end up with a x' far away from valid input x. To minimize the perturbation a larger C is required in the objective function. As C increases, an upper bound of C will finally be achieved such that no x' would be available to fool the network, since the distortion between x and x' is so strictly small. To bear this idea in mind, we utilize L-BFGS method to minimize our objective function along with bisection search to find the maximal and feasible C. The following algorithm shows in detail the generation of adversarial input with minimum distortion:

```
Require: A small positive value $\epsilon$
Ensure: L-BFGS-B(x, y', C) solves optimization
1:   {Finding initial C}
2:   $C \leftarrow \epsilon$
3:   repeat
4:   $C \leftarrow 2 \times C$
5:   $x',\ y,\ D \leftarrow L - BFGS - B(x,\ y',\ C)$
6:   until $y \neq y'$
7:   {Bisection search}
8:   $C_{low} \leftarrow 0,\ C_{high} \leftarrow C$
9:   repeat
10:    $C_{half} \leftarrow (C_{high} + C_{low})/2$
11:    $x',\ y,\ D' \leftarrow L - BFGS - B(x,\ y,\ C_{half})$
12:    if $y \neq y'$ then
13:    $D \leftarrow D'$
14:    $C_{high} \leftarrow C_{half}$
15:    else
16:    $C_{low} \leftarrow C_{half}$
17:    end if
```

```
18:     until $(C_{high} - C_{low}) < \epsilon$
19:     return $x'$, $D$
```

The final x' obtained is the optimal adversarial input with the minimum perturbation of distance D.

5 Experiment

5.1 Setup

Before the experiment we pre-train the data by extract the central part of each picture where the digit locate. Then we train a SVM model using 5000 data and as the classifier. The factors for the model are shown in Table 1.

Table 1: The Factors for the SVM Model

cache size	200
Input	$28 * 28$
decision$_{function_shape}$	'ovr'
kernel	'rbf'
gamma	0.0273
output	1

5.2 Results

5.2.1 Attack on MNIST dataset

We obtain adversarial inputs of hand written digits as shown in Figure 2. The first row pictures are the valid inputs all with correct label 8. The Second row are the perturbations added to the valid inputs. Adversarial inputs are shown in row 3. All the generated adversarial inputs successfully mislead MNIST classifier and output wrong predictions as 1, 2, 3, 4 that we randomly defined.

5.2.2 Constraint Scalar C

We also empirically analyzed the influence of constraint scalar C on perturbation norm and adversary success rate. The right plot of Figure 4 shows the relation between constraint scalar C and adversary success rate. When C is small enough, it's always available to find an adversarial input to fool the network. However, the perturbation norms with smaller C tend to be large as shown in the left plot. As we increase C to minimize the distortion, adversary success rate decreases as we expected. There is an optimum C that can both lead to successful adversarial input and yield a minimum distortion.

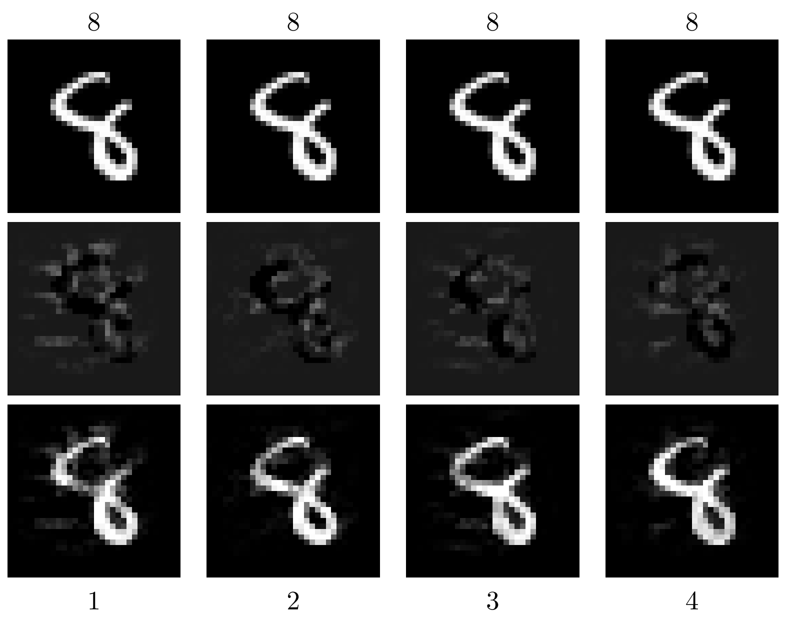

Figure 2: Valid inputs, adversarial inputs and their prediction labels.

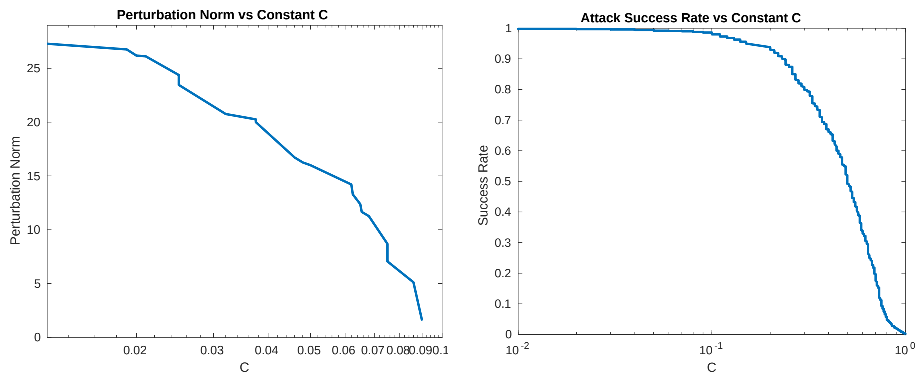

Figure 3: Constraint scalar C with perturbation norm and adversary success rate.

6 Conclusion

In this project, we utilize L-BFGS optimization method to generate adversarial inputs with minimum distortion for SVM and successfully mislead the classifier to output a pre-defined label. The influence of constraint scalar C on perturbation norm and adversary success rate is also analyzed.

7 Reference

[1] LeCun Y, Cortes C, Burges C J C. The MNIST database of handwritten digits[J]. 1998.
[3] Sermanet P, Eigen D, Zhang X, et al. Overfeat: Integrated recognition, localization and detection using convolutional networks[J]. arXiv preprint arXiv:1312.6229, 2013.
[4] Szegedy C, Zaremba W, Sutskever I, et al. Intriguing properties of neural networks[J]. arXiv preprint arXiv:1312.6199, 2013.
[5] Papernot N, McDaniel P, Goodfellow I. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples[J]. arXiv preprint arXiv:1605.07277, 2016.
[6] Goodfellow I J, Shlens J, Szegedy C. Explaining and harnessing adversarial examples[J]. arXiv preprint arXiv:1412.6572, 2014.
[7] Kurakin A, Goodfellow I, Bengio S. Adversarial examples in the physical world[J]. arXiv preprint arXiv:1607.02533, 2016.
[8] Nair V, Hinton G E. Rectified linear units improve restricted boltzmann machines[C] Proceedings of the 27th international conference on machine learning (ICML-10). 2010: 807-814.
[9] Zhu C, Byrd R H, Lu P, et al. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization[J]. ACM Transactions on Mathematical Software (TOMS), 1997, 23(4): 550-560.