Security and Privacy in Machine Learning

Homework 3

Javad Hezareh
98101074

Spring 2023

Contents

1 Adversarial Training of Linear Classifiers

(a) Yes, even in this simple linear classifier it is possible to have adversarial examples. If we add a small perturbation like $\boldsymbol{\delta}$ to the input, then the model output will be:

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\delta} \implies h(\tilde{\boldsymbol{x}}) = \boldsymbol{W}^T(\boldsymbol{x}+\boldsymbol{\delta}) + b \implies h(\tilde{\boldsymbol{x}}) - h(\boldsymbol{x}) = \boldsymbol{W}^T\boldsymbol{\delta}$$

Therefore the output change depends on $\boldsymbol{\delta}$ and $\boldsymbol{W}$. Let $\boldsymbol{\delta} = \epsilon \, \text{sign}(\boldsymbol{W})$, then we know $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$ and we have:

$$h(\tilde{\boldsymbol{x}}) - h(\boldsymbol{x}) = \epsilon \boldsymbol{W}^T \text{sign}(\boldsymbol{W}) = \epsilon \|\boldsymbol{W}\|_1$$

As we do not have any bound on the value of $\|\boldsymbol{W}\|_1$, then we can have significant change in the output by small perturbation of input if this value be very large.

(b) We know the original form of optimization in adversarial training is

$$\min_{\boldsymbol{W},b} \mathbb{E}\left[\max_{\boldsymbol{\delta}\in\Delta} \mathcal{L}\left(h(\boldsymbol{x}+\boldsymbol{\delta}), y\right)\right]$$

where $\Delta = \{\boldsymbol{\delta} \in \mathbb{R}^n | \|\boldsymbol{\delta}\|_\infty \leq \epsilon\}$. Now if want to solve the inner maximization problem using FGSM attack, then we have:

$$\begin{aligned}
\boldsymbol{\delta}_{\text{FGSM}} &= \epsilon \, \text{sign}\left(\nabla_{\boldsymbol{x}} \mathcal{L}\left(h(\boldsymbol{x}), y\right)\right) \\
&= \epsilon \, \text{sign}\left(\nabla_{\boldsymbol{x}} \log\left(1 + e^{-yh(\boldsymbol{x})}\right)\right) \\
&= \epsilon \, \text{sign}\left(\frac{-ye^{-yh(\boldsymbol{x})}}{1+e^{-yh(\boldsymbol{x})}} \nabla_{\boldsymbol{x}} h(\boldsymbol{x})\right) \\
&= \epsilon \, \text{sign}\left(\frac{-ye^{-yh(\boldsymbol{x})}}{1+e^{-yh(\boldsymbol{x})}} \boldsymbol{W}\right) \\
&= \epsilon \, \text{sign}\left(\alpha \boldsymbol{W}\right) \qquad \alpha = \frac{-ye^{-yh(\boldsymbol{x})}}{1+e^{-yh(\boldsymbol{x})}}
\end{aligned}$$

Applying this value in the loss function we have:

$$\begin{aligned}
\mathcal{L}(h(\boldsymbol{x}+\boldsymbol{\delta}_{\text{FGSM}}), y) &\approx \mathcal{L}(h(\boldsymbol{x}), y) + (\delta_{\text{FGSM}})^T \nabla_{\boldsymbol{x}} \mathcal{L} \\
&\approx \mathcal{L}(h(\boldsymbol{x}), y) + \epsilon \; \alpha \boldsymbol{W}^T \text{sign}(\alpha \boldsymbol{W}) = \mathcal{L}(h(\boldsymbol{x}), y) + \epsilon \|\alpha \boldsymbol{W}\|_1
\end{aligned}$$

Therefore the adversarial training objective for this problem will be:

$$\min_{\boldsymbol{W},b} \mathbb{E}\left[\mathcal{L}(h(\boldsymbol{x}), y) + \epsilon \|\alpha \boldsymbol{W}\|_1\right]$$

(c) If we use l_1 regularizer, then the training objective will be:

$$\min_{\boldsymbol{W},b} \mathbb{E}\left[\mathcal{L}(h(\boldsymbol{x}), y) + \gamma \|\boldsymbol{W}\|_1\right]$$

As we can see this objective is very similar to the one we have in adversarial training except that α multiplier. As we saw in part (a), the reason we have adversarial examples is the l_1-norm of weight matrix hence using l_1 regulizer or a version of that such as what we have in the objective of adversarial training will lead to a model which is robust to adversarial examples.

2 Targeted Attack

(a) We can perform targeted attack for different classes and then choose our target class to be the class for which we have the smallest size of perturbation or the largest size. Former method leads to the best case scenario and the later one leads to the worst case scenario.

(b) Let's say our target class is t, then we want to build an adversarial input on which our model performs well. Therefore we want to minimize the loss function on this target class:

$$x_{adv} = \arg\min_{x'} \mathcal{L}(h_w(x'), t)$$

3 Various Attacks

To build adversarial examples using PGD we need to perform following steps:

Algorithm 1 PGD attack

```
$x :=$ input, $k :=$ number of steps, $\alpha :=$ step size, $\epsilon :=$ attack radius

$x^0 = U(x - \epsilon, x + \epsilon)$
**for** $t = 1 \to k$ **do**
    $\delta = \alpha \operatorname{sign}(\nabla_x \mathcal{L}(h(x^t), y))$
    $x^t = Proj_{[x-\epsilon, x+\epsilon]}(x^{t-1} + \delta)$
    $x^t = Clip_{[0,1]}(x^t)$
**end for**
```

Using this method, we will explore ϵ neighborhood of input step by step instead of just single step like FGSM. However, our steps are still in the direction of the square corners but with smaller step size which enable us to take multiple steps and find a better solution. A sample of performing this algorithm will be as follows:

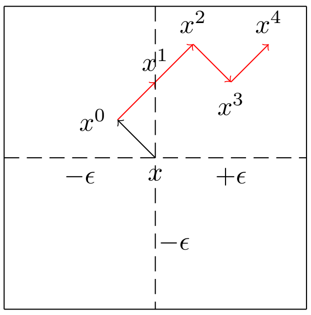

Figure 1: A sample trajectory of performing PGD with $k = 4$.

In FGSM-RS method, everything is just like FGSM but before performing an FGSM attack, first we need to pick a random point in ϵ neighborhood of input. Its algorithm is as follows:

Algorithm 2 FGSM-RS attack

```
$x :=$ input, $\alpha :=$ step size, $\epsilon :=$ attack radius

$x^0 = U(x - \epsilon, x + \epsilon)$
$\delta = \alpha \operatorname{sign}(\nabla_x \mathcal{L}(h(x_0), y))$
$x_{adv} = Proj_{[x-\epsilon, x+\epsilon]}(x^0 + \delta)$
$x_{adv} = Clip_{[0,1]}(x_{adv})$
```

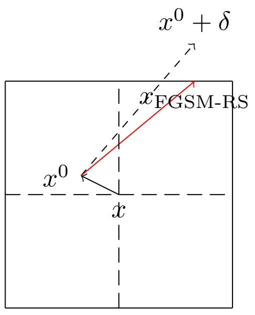

Figure 2: A sample trajectory of performing FGSM-RS.

4 Practical

In following sections you can find the report of this question:

8: This part was the evaluation of standard model on adversarial examples using FGSM attack and different ϵ values.

	Standard ResNet	
FGSM	ϵ	Adversarial Accuracy
	4/255	10 %
	8/255	1.6 %
	12/255	0.6 %

We can see that as we expect the standard model is not robust against FGSM adversarial attack.

9: Following is the 5 crafted adversarial examples using FGSM and $\epsilon = 8/255$:

True label	Predicted Label	Confidence
cat	dog	0.72
ship	car	0.64
ship	bird	0.99
plane	ship	0.99
frog	dear	0.99

10: In figure 3 you can find the crafted adversarial examples and their clean inputs.

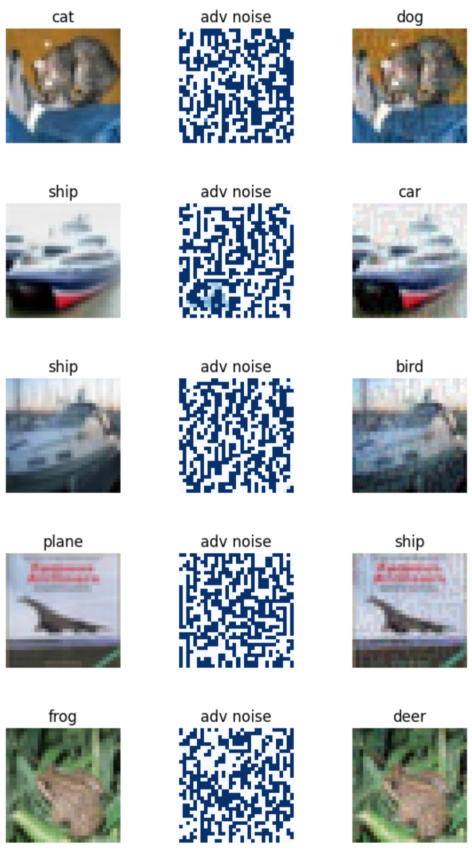

Figure 3: Crafted adversarial examples using FGSM and $\epsilon = 8/255$.

11: In this part we used adversarial training to build a robust model against adversarial perturbations. The results of adversarial trained model are as follows:

AT ResNet	
Test data	Accuracy
Clean	76.01 %
FGSM	39.97 %

13: In this part we used PGD attack to evaluate the robustness of standard and adversarial trained model:

	Standard ResNet		AT ResNet	
	k	Accuracy	k	Accuracy
PGD	2	5.31 %	2	62.96 %
	4	0.19 %	4	58.60 %

14: In figure 4 you can see the crafted adversarial examples using PGD:

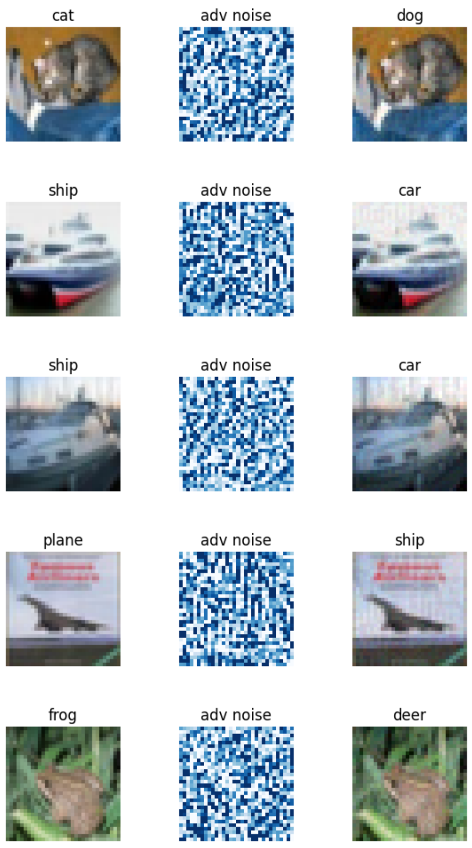

Figure 4: Crafted adversarial examples using PGD, $\epsilon = 8/255$ and $k = 4$.

15: By adding Gaussian noise with mean 0 and variance of 6/255 to the test data we have:

Accuracy	Std ResNet	AT ResNet
Noise	34.21 %	73.09 %

We can see that adversarial trained model perform better on noisy images. This behavior is expected, cause the decision boundary of adversarial trained model are not very tight to data points and therefore by moving randomly in their neighborhood the decision will not change.

16: The resulted diagrams are as follows:

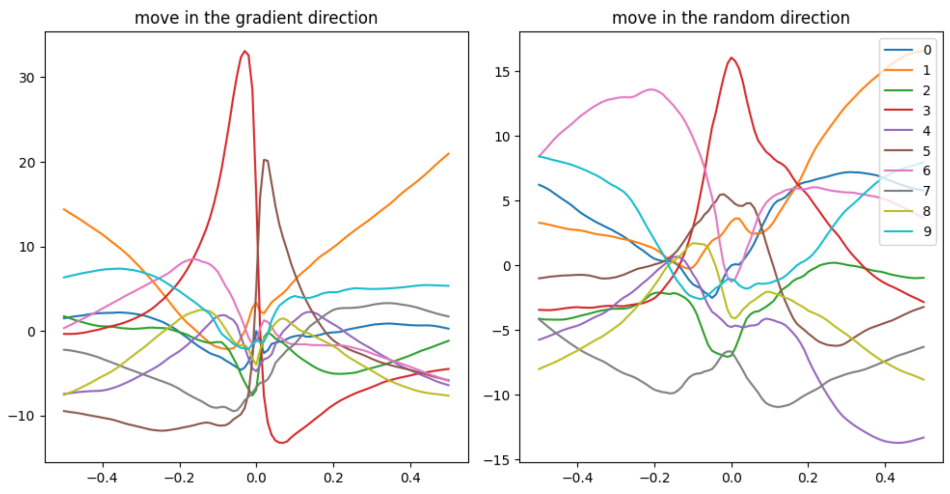

Figure 5: The change in logits by changing ϵ.

The image that we have calculated this diagrams on that was the picture of a cat which is the 4-th class, or the class with index 3. As we saw in previous parts by using FGSM attack, this picture was labeled as a dog which is 6-th class or the class with index 5. We can adjust this behavior by graphs in figure 5, if we move in the direction of gradient, the logit related to class 3 (red line) decrease significantly and the logit related to class 5 (brown line) increase. This leads model to predict dog as output. But when we move in random direction this phenomena can not happen, cause by small changes in ϵ, no significant change in the logits happen and still model predict its previous predictions.