Notes on Adversarial Examples

Amirhossein Koochakian

1 Introduction

We're going to explore and apply the concepts outlined in the paper "Explaining and Harnessing Adversarial Examples".

Let's begin by understanding what adversarial examples are and why they are crucial.

1.1 what are Adversarial Examples?

An adversarial example is a sample of input data which has been modified very slightly in a way that is intended to cause a machine learning model to misclassify it. By this definition, we can conclude that the goal is to make slight changes to inputs in a way that deceives the model.

1.2 Importance of adversarial examples in machine learning

In an intriguing discovery by Szegedy et al. (2014b), it was revealed that various machine learning models, including advanced neural networks, are prone to adversarial examples. These are instances where the models misclassify inputs that are only slightly different from properly classified examples obtained from the data distribution. Furthermore, it's worth noting that different models, having diverse architectures and trained on different subsets of the training data, commonly misclassify the same adversarial example. This observation implies that adversarial examples unveil fundamental vulnerabilities in our training algorithms.

Before this paper, researchers attributed the presence of adversarial examples to the exceedingly complex nature of deep neural networks. This complexity, alongside potential factors such as inadequate model averaging and insufficient regularization in the context of purely supervised learning, was believed to contribute to the emergence of adversarial examples. But the paper demonstrates that linear behavior in high-dimensional spaces is sufficient to cause adversarial examples. This perspective leads to the faster design of adversarial examples, such as the Fast Gradient Sign Method.

In the next section, we will discuss simple linear models and attempt to find adversarial examples for them.

2 Linear Explanation of Adversarial Examples

In many cases, the precision of the features is limited. For instance, when we refer to 8-bit pixels, it means that any information with a value lower than 1/255 will be ignored. In other words, the model can only perceive information within the range of its precision. Consequently, the model's output in response to inputs x and $\tilde{x} = x + \eta$, where each element of the perturbation η is smaller than the precision of the features or $||\eta||_\infty \leq \epsilon$, where ϵ is smaller than the precision, should remain unchanged.

Now lets see how adding a small perturbation can significantly change activation. Consider the dot product between a weight vector w and an adversarial example $\tilde{x}$.

$$w^T \tilde{x} = w^T x + w^T \eta \tag{1}$$

The adversarial perturbation causes the activation to grow by $w^T \eta$, but what value can maximize this increase? Lets consider this problem : assume we are trying to maximize the dot product of $x^T y$ subject to $||x||_\infty \leq \epsilon$.

$$\max_{||x||_\infty \leq \epsilon} x^T y = \sum_{i=1}^{n} x_i y_i \tag{2}$$

When is equation 2 maximized? In order to maximize the equation, we need to consider two things: 1. Each x_i should have its maximum value, and 2.Avoid negative values in the summation. The max norm indicates that x_i should be less than or equal to ϵ, which sets the maximum size for each x_i to ϵ. If each y_i is positive, then the summation will be maximized, and each y_i will have an ϵ factor. Therefore, setting $x_i = \epsilon \times sign(y_i)$ maximizes the summation.

$$\max_{||x||_\infty \leq \epsilon} x^T y = \sum_{i=1}^{n} \epsilon \times |y_i| = \epsilon \times ||y||_1 \tag{3}$$

The form of $w^T \eta$ is similar to equation 3, so we can conclude that the maximum increase in equation 1 occurs when $\eta = \epsilon \times sign(w)$.

So far we have seen how to find a adversarial example for a linear model and achieve maximum growth of $w^T \eta$. Now lets examine the impact of dimension. Assume the average magnitude of an element of w is given by m, and $w \in R^n$.

$$w^T \eta = \epsilon \times w^T sign(w) =$$
$$\epsilon \times ||w||_1 = \epsilon m n$$

Basically, what this means is that as the dimension increases, even a tiny perturbation (limited by ϵ) can cause a significant change in the model's behavior. This makes the model more vulnerable to adversarial attacks. In simpler terms, in high-dimensional scenarios, even a small disturbance can have a big impact on the model's predictions.

So, what we're discovering here is that even a basic linear model can have adversarial examples when the input has a significant number of dimensions. This finding is quite different from what was previously believed, when it was commonly thought that adversarial examples were only a concern for highly complex and non-linear deep neural networks.

3 Non-Linear Models

Linear models are easy to train and optimize, so lots of deep neural networks were designed to behave in linear ways. Even some non-linear models, like sigmoid, behave linearly in a significant portion of their space. This linear behavior suggests that perturbations of a linear model should also damage neural networks.

In this section, I will explain the Fast Gradient Sign Method (FGSM), but it would be beneficial to first check how gradient descent works.

3.1 Gradient Descent

Let x be the input, y the output related to x, θ the weights of the network and $J(\theta, x, y)$ the loss function used to train the model. Gradient descent aims to minimize the loss function (J) by adjusting the weights (θ) of the network. It does so by iteratively updating the weights in the opposite direction of the gradient of the loss function, which tells us how the loss changes with respect to the weights. Therefore each iteration of the algorithm can be written as :

$$\theta = \theta - \alpha \frac{\partial J(\theta, x, y)}{\partial \theta} \tag{4}$$

Notice that the sign of the derivation plays important role in this process. If the sign is negative, we will increase the θ, and if the sign is positive, we will decrease the θ. All these steps lead to a decrease in the loss function.

3.2 Fast Gradient Sign Method

Adversarial attacks can be seen as a kind of "opposite" of gradient descent. Instead of moving in the direction of gradients to minimize the loss function, adversarial attacks update the parameters in the same direction as the gradients, causing them to move away from the minimum.

Another important consideration is understanding the parameters involved. Since we are trying to attack a model, the parameter θ remains fixed. The only aspect we have control over is the input x. Therefore, we calculate the gradients of the loss function with respect to the input x. These gradients guide us in determining the appropriate directions for perturbation. Subsequently, we update the input by applying these directions, but limit the magnitude of the update to a small value denoted as ϵ. It's worth noting that we ensure the condition $||\eta||_\infty \leq \epsilon$ is met, meaning we only utilize the directions obtained

from the gradients and do not directly update the input based on the gradients themselves.

$$\eta = \epsilon sign(\nabla_x J(\theta, x, y)) \tag{5}$$

Every step of updating the input is like :

$$\tilde{x} = x + \eta \Rightarrow \tilde{x} = x + \epsilon sign(\nabla_x J(\theta, x, y)) \tag{6}$$

Another way to conclude the equation 5 is : we can linearize the cost function around the current value of θ, obtaining an optimal max-norm constrained perturbation of equation 5.

So, by applying ideas similar to linear models, we can use gradients to perform an adversarial attack.

4 Conclusion

In conclusion, adversarial attacks pose a significant challenge in the field of machine learning. These intentional attempts to deceive neural networks by manipulating input data can lead to incorrect predictions or misclassifications. One intriguing aspect of adversarial attacks is the discovery that neural networks are often more linear than originally believed.