Theme: Adversarial attacks

Introduction

**Adversarial attacks** are a technique for fooling machine learning models by adding small changes to the data that are not noticeable to humans, but can confuse the model. Imagine you have an image of a cat and you show it to an image recognition system. If you make subtle changes to the image, the system may misrecognize the cat as a dog, or even as something that doesn't exist at all.

Let's compare this to ordinary life. Imagine that you are writing an exam and someone very carefully and discreetly changes the letters in some words of your test. Nothing seems to have changed, but these subtle corrections can completely change the meaning of what you've written and lead to incorrect answers. This is also how counter attacks work: small changes in the data can completely change the behavior of the model and lead to incorrect results.

Let's visualize this with a real-world example. There is a neural network designed to detect faces in photographs. It successfully accomplishes its task (see the image on the left). However, if a small amount of noise is added to this photo (see image on the right), the algorithm that obtained the adversarial example (see image in the center) will no longer be able to detect a face in the image.

Изображение выглядит как текст, Человеческое лицо, человек, снимок экрана

Автоматически созданное описание

The example described in the paper ["Adversarial Attacks on Face Detectors using Neural Net based Constrained Optimization"](https://arxiv.org/pdf/1805.12302) is interesting because many real face recognition systems are based on neural network approaches. However, a person will not notice the difference when looking at both images.

Classification of attacks

All attacks can be divided into 2 classes: WhiteBox (WB) and BlackBox (BB).

White-box attacks

**What it is**: In these attacks, the attacker has full access to the model, including its structure, weights, and algorithms.

**How it works**: The attacker uses this information to calculate exactly what changes to the data will cause the model to fail.

**Example**: If someone knows how your home security system works, including all the passwords and schemes, they can find the vulnerability and bypass it.

Black-box attacks

**What it is**: In these attacks, the attacker does not have access to the internal model information. He can only send input data and receive output results.

**How it works**: The attacker experiments with different inputs and analyzes the model's responses to find vulnerabilities.

**Example**: Imagine someone trying to hack into your account without knowing your passwords, but trying many combinations to guess the correct one.

These types of attacks show how different approaches can be used to fool models and emphasize the importance of developing robust security methods.

The GrayBox variant is still possible, when we do not know information about the trained model, but we have information about the type of the algorithm and its hyperparameters. But this type is not singled out as a separate class, because the additional information is not enough to switch to WB, and therefore it is only an additional set of information for a BB attack.

Next, we should classify attacks into Targeted and Non Targeted.

**Targeted** attacks are aimed at changing the classification of an image in a certain direction. For example, if an image was originally classified as number 0, then after the attack we want it to be recognized as number 1.

**Non-Targeted** attacks have no specific goal to change the classification to a particular class. The main thing is that after the attack, the image should no longer be classified as the original class (e.g., not as digit 0).

**Python libraries to help**

These Python libraries allow you to work with Adversarial examples. They are FoolBox, CleverHans and ART-IBM.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Now let's break down the attacks in a bit more detail.

L-BFGS atack

The BFGS method, an iterative numerical optimization method, is named after its researchers: Broyden, Fletcher, Goldfarb, Shanno, and the letter L emphasizes the fact of application in the case of a large number of unknowns.

The formulation of the L-BFGS method can be written in the following [formula](https://arxiv.org/pdf/1312.6199).

[](https://arxiv.org/pdf/1312.6199)  
  
It implies that we want to minimize the loss function towards the target class with the constraint that the changes made were minimal.

This attack is presented in 2 of the 3 previously announced libraries - FoolBox and CleverHans.

And applying this attack on FoolBox takes 3 lines of Python code:

**from** foolbox.attacks **import** LBFGSAttack  
attack = LBFGSAttack(fmodel)  
adversarial = attack(image, label)

Using the L-BFGS method will help you to find optimal adversarial examples, given the given constraints. However, firstly, finding such an example may take considerable time, and secondly, there is a probability that the method will not converge to a solution.

FGSM attack

The next stage of development was the Fast Sign Gradient Method (FGSM), which can be shown using a [formula](https://arxiv.org/pdf/1412.6572):  
Изображение выглядит как Шрифт, текст, типография, каллиграфия

Автоматически созданное описание

This method is much faster than L-BFGS. Here we simply take the signs from the gradient function of the original loss function, multiplying the sign by some ϵ , add to the original image.Изображение выглядит как млекопитающее, панда, медведь, текст

Автоматически созданное описание

Here is an example of how this method works. A noise map of 0.007 is added to the panda photo, and it turns out that the panda photo is now recognized as Gibbon with 99.3% probability

This method is simple to implement, but the result of this method is highly noisy.

You can find an implementation of this method in libraries, and it doesn't take long to implement it on foolbox:

**from** foolbox.attacks **import** FGSM  
attack = FGSM(fmodel)  
adversarial = attack(image, label)

DeepFool attack

DeepFool is a Non-Targeted attack method. Unlike the previous methods, its main difference is that it aims to create the minimum possible change that will fool the algorithm. This method does not aim to change the image class to a specific one, but to change it to any other class that is closest to the original image.Изображение выглядит как картина, риф, искусство, мозаика

Автоматически созданное описание

The example shows the original [picture](https://arxiv.org/pdf/1511.04599), with the FGSM method on the bottom line and just the DeepFool attack in the middle.

You can see that the noise map is much smaller than with FGSM.

This attack can be performed using any of the above libraries, and the implementation on ART-IBM takes only 3 lines of code:

**from** art.attacks **import** DeepFool  
attack = DeepFool(model)  
img\_adv = attack.generate(img)

Conclusion

In fact, there are many different attacks, and this article has covered just a few of them. We hope that this material has helped you grasp the basic concepts of adversarial examples and how to create them. For a deeper dive, we recommend exploring the original links provided in the article. Thank you for your attention!