Adversarial examples in deep learning

G. Châtel

06/07/2017

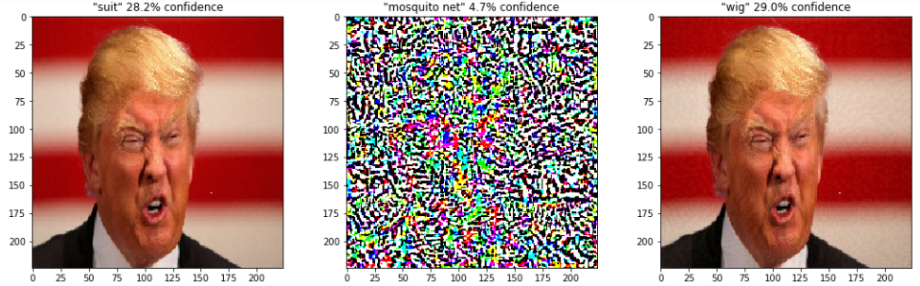
Basic notions

Adversarial example

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 classifier to misclassify it.

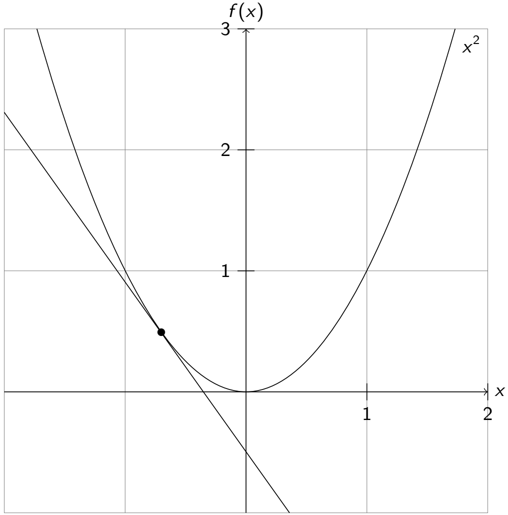
Basic notions
Adversarial example

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 classifier to misclassify it.



Gradient descent

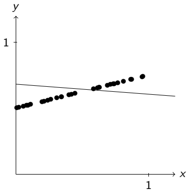
Basic concept



The curve needs to be *smooth enough* for the gradient descent to work.

Gradient descent

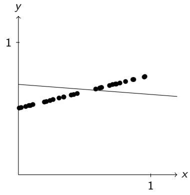
Model optimization



We have a set of points that we want to approximate with a line.

$$y = ax + b$$

Gradient descent
Model optimization



We have a set of points that we want to approximate with a line.

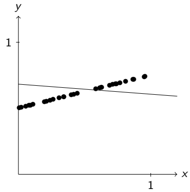
$$y = ax + b$$

First we choose a loss that measures how good our predictions are.

$$l(x, y, a, b) = (y - (ax + b))^2$$

Gradient descent

Model optimization



We have a set of points that we want to approximate with a line.

$$y = ax + b$$

First we choose a loss that measures how good our predictions are.

$$l(x, y, a, b) = (y - (ax + b))^2$$

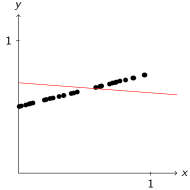
We compute how the loss is affected by small changes of a and b:

$$\frac{\mathrm{d}l}{\mathrm{d}a} = 2x(ax + b - y) \qquad \frac{\mathrm{d}l}{\mathrm{d}b} = 2(ax + b - y)$$

And we update a and b iteratively until we reach a satisfying result (the average loss for our data points is low enough).

Gradient descent

Being evil

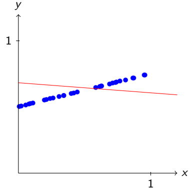


In our previous example, we have modified the model in order to minimize the loss.

$$y = ax + b$$

Gradient descent

Being evil



In our previous example, we have modified the model in order to minimize the loss.

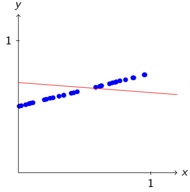
$$y = ax + b$$

Now suppose we are an attacker who wants to maximise the loss of a model, its parameters being fixed. The only thing we can modify is the inputs.

$$l(x, y, a, b) = (y - (ax + b))^2$$

Gradient descent

Being evil



In our previous example, we have modified the model in order to minimize the loss.

$$y = ax + b$$

Now suppose we are an attacker who wants to maximise the loss of a model, its parameters being fixed. The only thing we can modify is the inputs.

$$l(x, y, a, b) = (y - (ax + b))^2$$

In order to do this, we compute how the loss is affected by small changes of the input:

$$\frac{\mathrm{d}l}{\mathrm{d}x} = 2a(ax + b - y)$$

We can now make *imperceptible* changes to an input to make the loss grow.

Neural networks

I used a basic regression task to illustrate the concept of adversarial samples generation.

Everything works the same way when working with a neural network on an image classification task.

We also have a differentiable loss function (often categorical cross entropy) and inputs (pixel values) that we can modify to increase the loss.

Attacks

Random noise	Does not work
FGSM	Good but can be well defended by training the network with adversarial samples
Iterative FGSM	Higher error than FGSM for an equivalent ε but less transferability. I-FGSM produces weaker black-box attacks.
Targeted FGSM	Aims at fooling a model into outputting a given target class.
RAND + FGSM	Significant improvements against adversarially trained models. RAND+FGSM transfers at lower rates than FGSM examples. Unsing RAND+FGSM to adverarially train networks does not improve their defense against RAND+FGSM.

Fast Gradient Sign Method

Move along the derivate away from the correct value as a way to maximise the error.

Black box attack

This is nice but happens if you cannot access the gradients

Adversarial examples in the physical world

This is nice but in real world scenarios, we are not feeding the network with our own data, it is acquired by the network's system (using camera for example).

Defenses

Adversarial sample detection: We try to detect whether an input sample is adversarial or not before classifying it.

Training with an adversarial objective function is an effective regularizer (from [explaining and harnessing]).

Gradient masking: The goal of gradient masking is to leave the decision boundaries untouched but damage the gradient used in white-box attacks.

Distillation and network saturation: These methods are used to introduce numeraical instabilities in gradient computations.