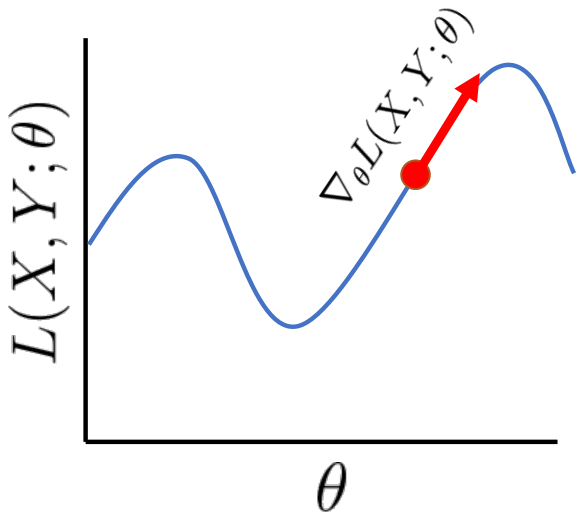
Adversarial Examples

Deep Learning: Bryan Pardo, Northwestern University, Fall 2020

What does the gradient tell us?

- If the loss function and hypothesis function encoded by the model are differentiable* (i.e., the gradient exists)
- We can evaluate the gradient for some fixed value of θ and get the *direction* in which the loss *increases* fastest



*or subdifferentiable

Gradient Descent Pseudocode

Initialize $\theta^{(0)}$

Repeat until stopping condition met:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla L(X, Y; \theta^{(t)})$$

Return $\theta^{(t_{max})}$

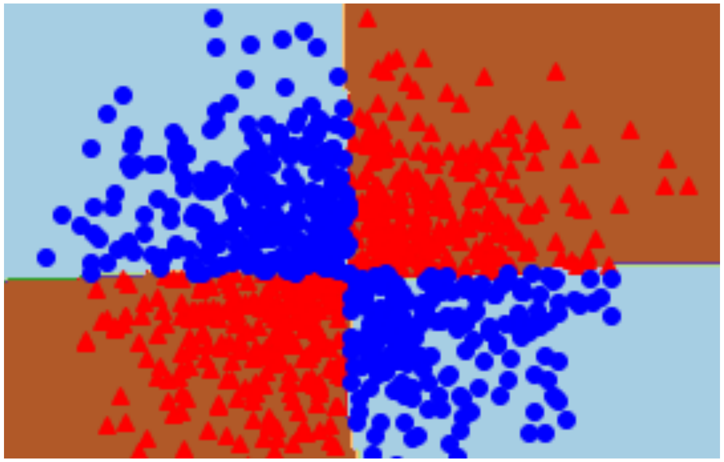
$\theta^{(t)}$ are the parameters of the model at time step t

$\nabla L(X, Y; \theta^{(t)})$ is the gradient of the loss function with respect to model parameters $\theta^{(t)}$

η controls the step size

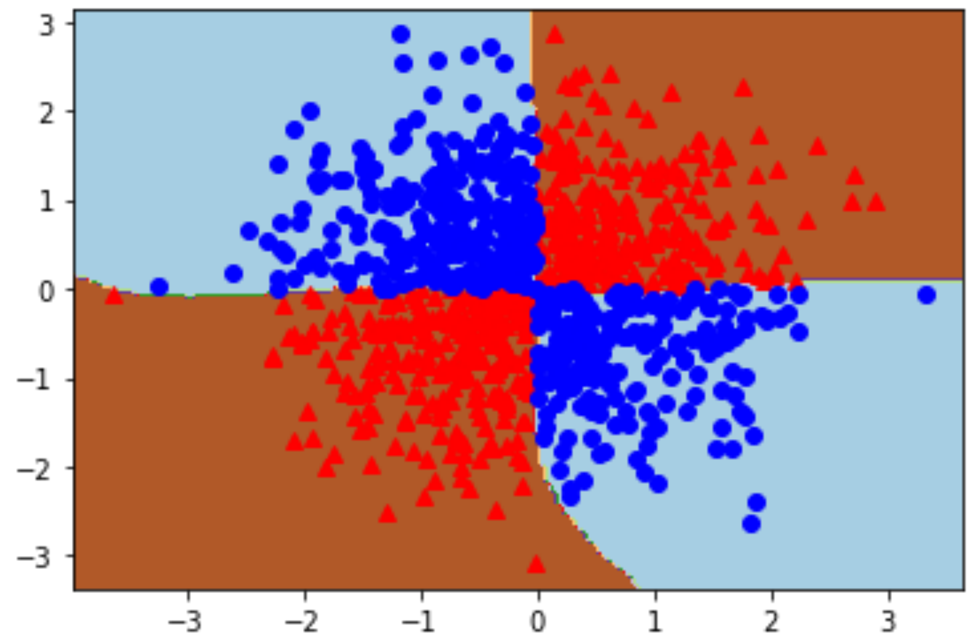
$\theta^{(t_{max})}$ is the set of parameters that did best on the loss function.

What are these things really learning?



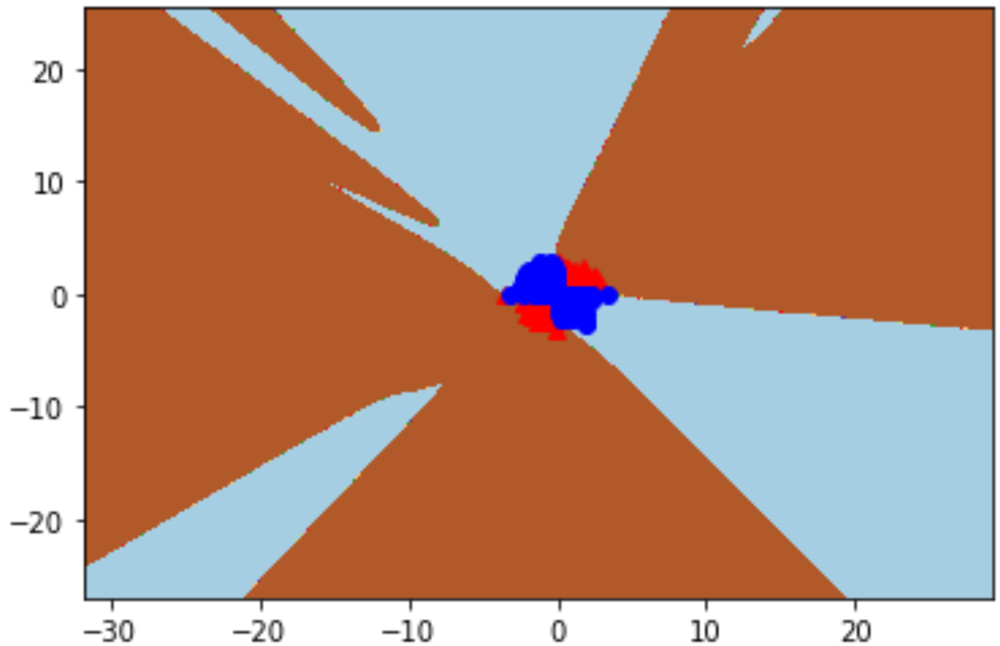
Learned decision surface for XOR problem

What are these things really learning?



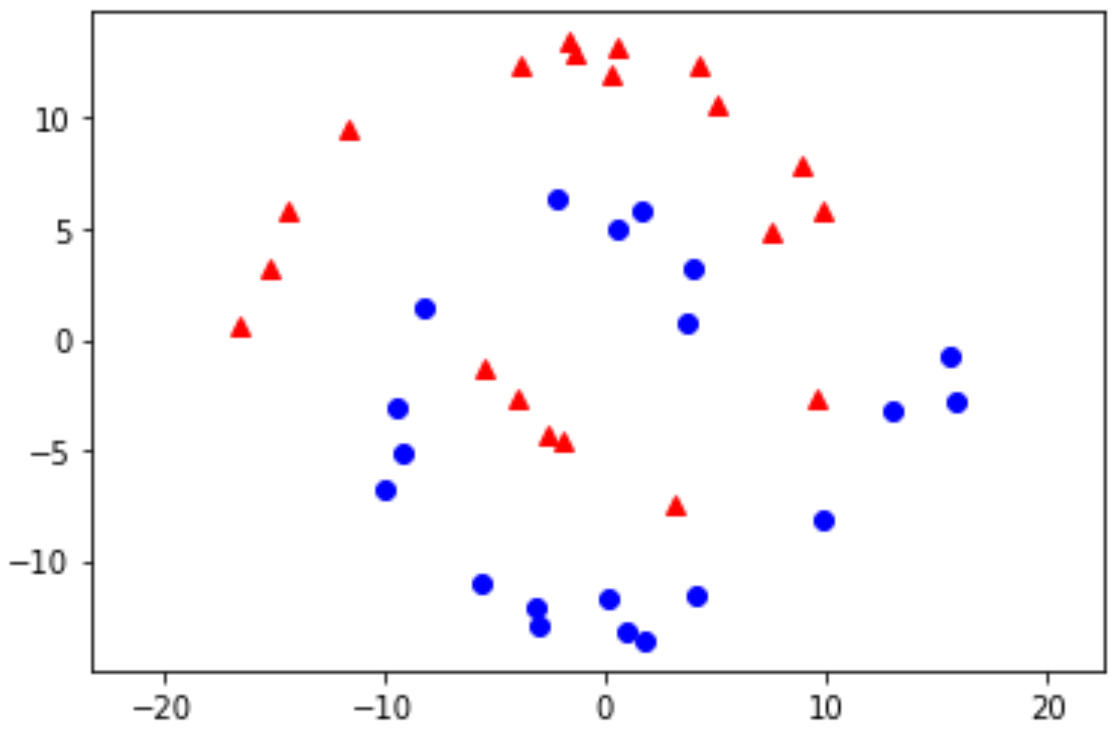
Learned decision surface for XOR problem

What are these things really learning?

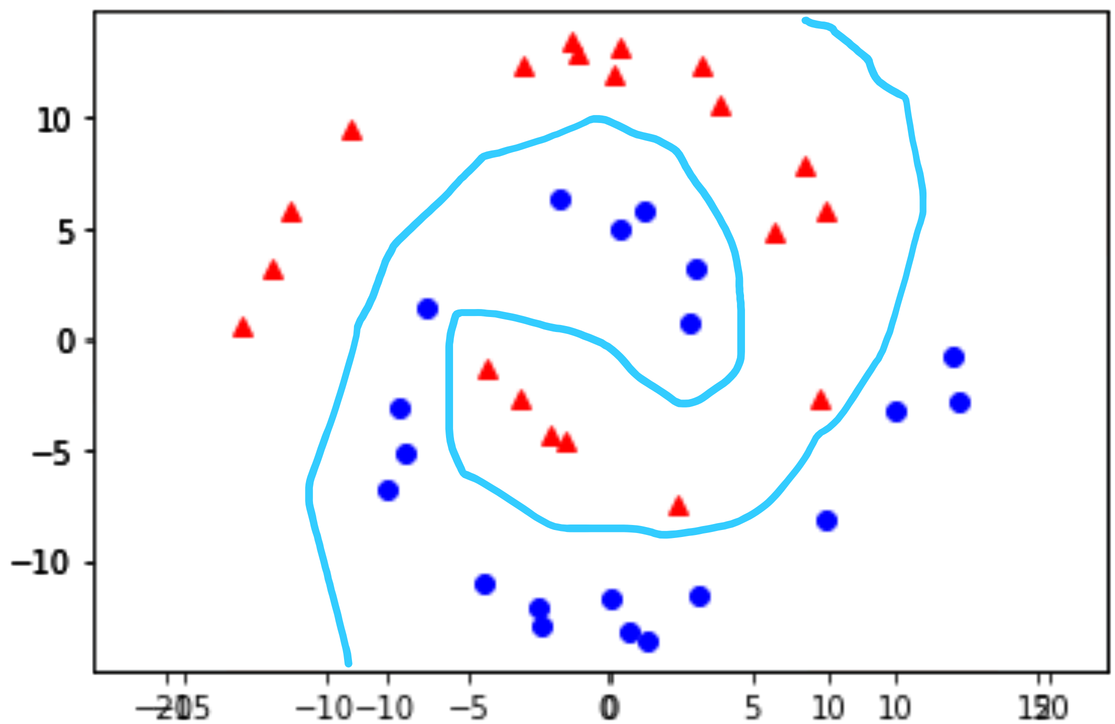


Same decision surface zoomed out

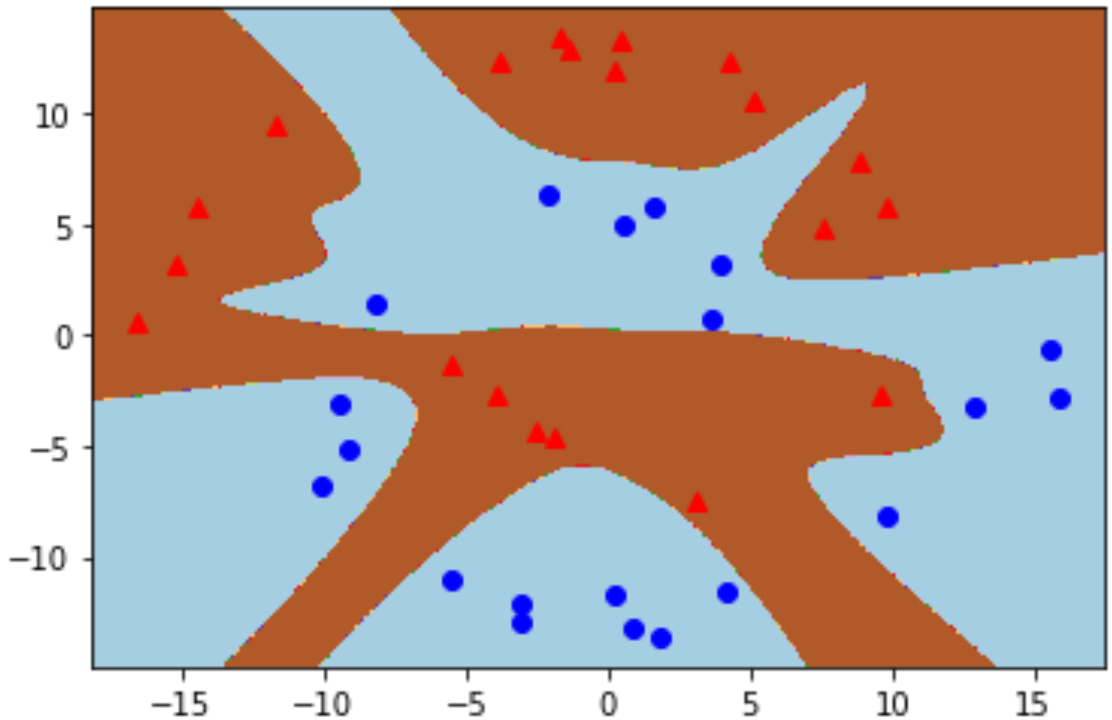
Spirals data



Decision surface a human might draw



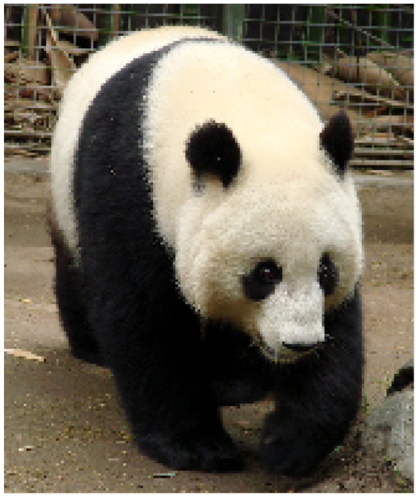
Actual decision surface learned by a network

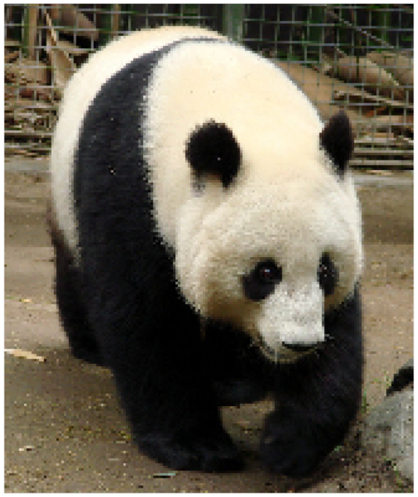


100% perfect accuracy on labeling

In 2 dimensions, a bad surface is obvious

- What about in 2 million dimensions?

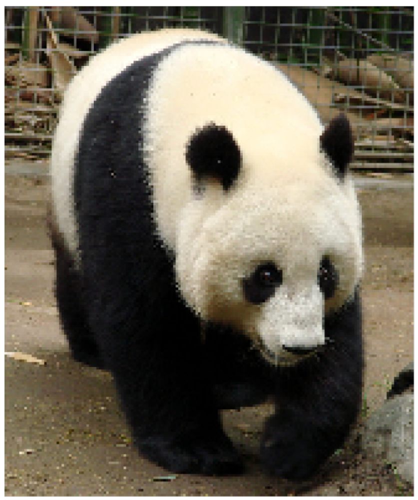




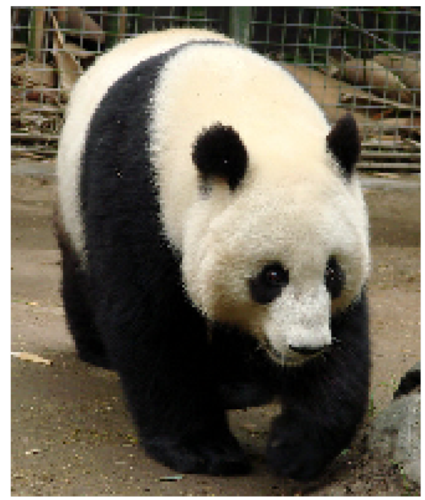
One of these was labeled “panda” by a trained net.
The other was labeled “bucket”. Which is which?

Image from: https://www.borealisai.com/en/blog/advertorch-adversarial-training-tool-implement-attack-and-defence-strategies/

The one on the right is a “perturbed” image

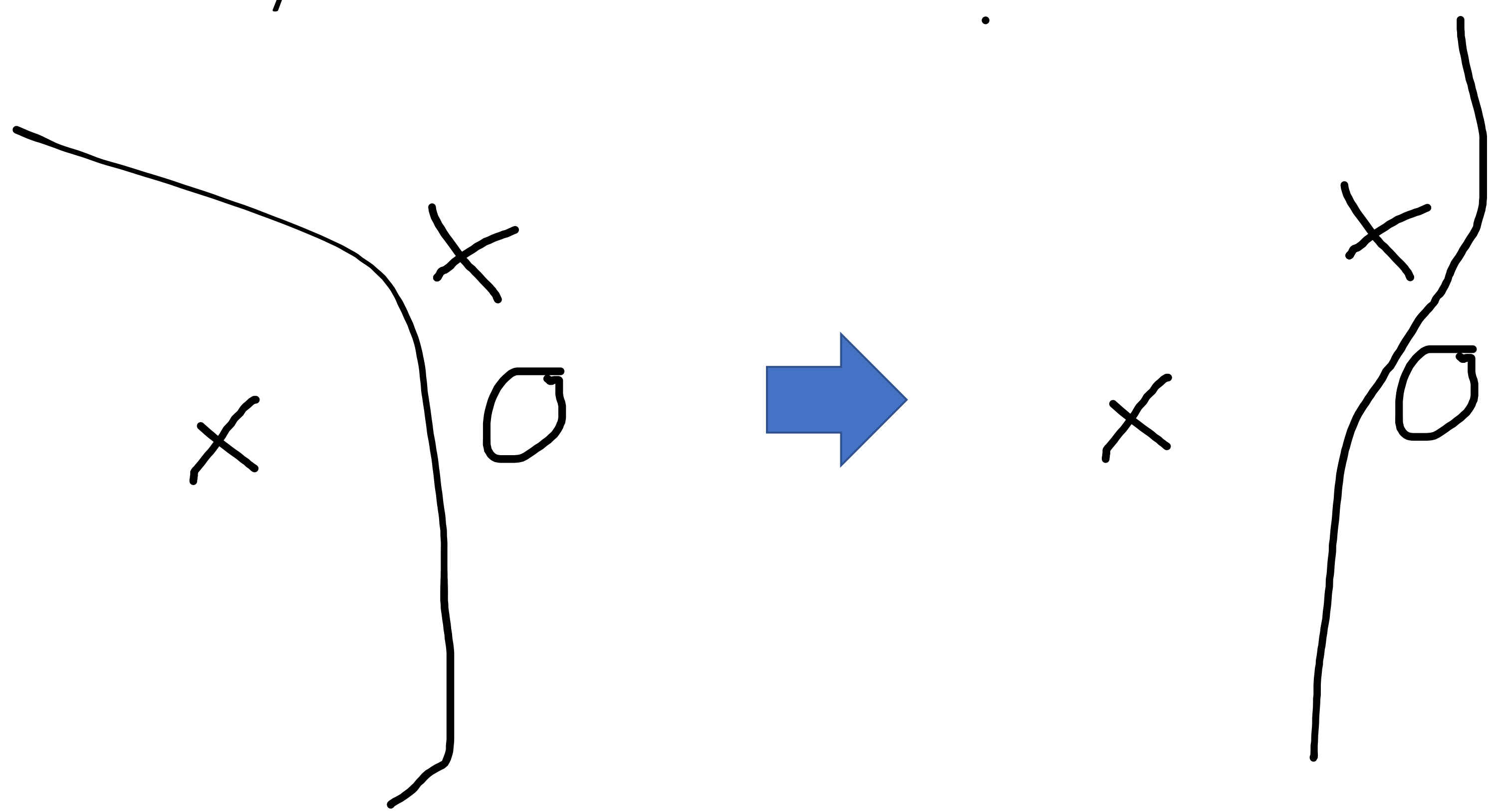


PANDA

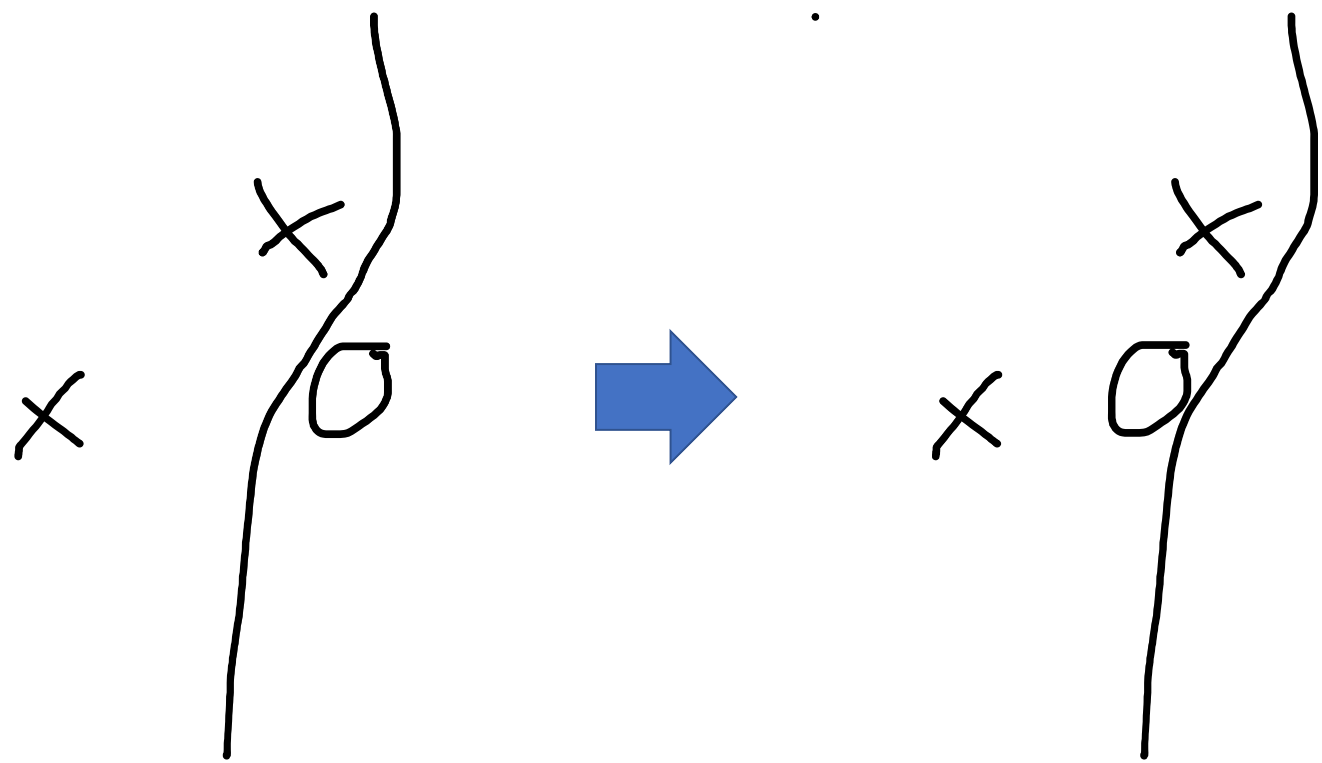


BUCKET

Gradient descent is moving the decision boundary



What if we move a datapoint instead?



Gradient Descent Pseudocode

Initialize $\theta^{(0)}$

Repeat until stopping condition met:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta L(X, Y; \theta^{(t)})$$

Return $\theta^{(t_{max})}$

$\theta^{(t)}$ are the parameters of the model at time step t

$\nabla_\theta \, L(X, Y; \theta^{(t)})$ is the gradient of the loss function with respect to model parameters $\theta^{(t)}$

η controls the step size

$\theta^{(t_{max})}$ is the set of parameters that did best on the loss function.

Just flip which thing we're optimizing

Initialize $X^{(0)}$

Repeat until stopping condition met:

$$X^{(t+1)} = X^{(t)} + \eta \nabla_X L(X^{(t)}, Y; \theta)$$

Return $X^{(enough)}$

$X^{(t)}$ is an example at time t

$\nabla_X L(X^{(t)}, Y; \theta)$ is the gradient of the loss function with respect to example features $X^{(t)}$

η controls the step size

$X^{(enough)}$ is the minimal change needed to flip the category of X

Even Simpler: Fast Gradient Sign method

$$X^{(t+1)} = X^{(t)} + \eta sign(\nabla_X L(X^{(t)}, Y; \theta))$$

EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES
Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy, ICLR 2015

Gradient Sign attack

- The pixels are all independent dimensions

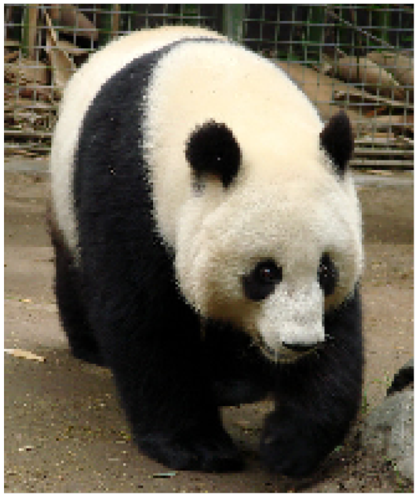
- Find the gradient in the pixel space

- Add noise along the gradient (a little bit of noise for every pixel)

- Do it right and the image looks the same to the user…

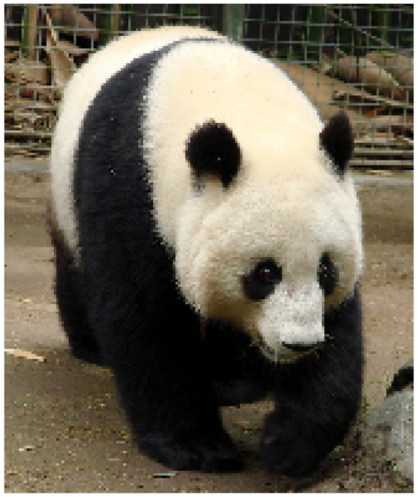
…but looks entirely different to the network.

Yes. It’s just that easy



original image
prediction: giant_panda

the perturbation,
enhanced 127 times



perturbed image
prediction: bucket

Why.....

- ....does this gradient-based attack make sense?
- ...did they use the sign of the gradient multiplied by a fixed step size, instead of the actual gradient?