Reliable and Interpretable Artificial Intelligence

Lecture 3: Adversarial Attacks II

Martin Vechev
ETH Zurich

Fall 2020

SRILAB

http://www.sri.inf.ethz.ch

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Recall: Our Optimization Problem

Two steps:

Step 1: Define an objective function $\boldsymbol{obj_t}$ such that:

$$\text{if } \boldsymbol{obj_t(x+\eta) \le 0} \text{ then } \boldsymbol{f(x+\eta) = t}$$

Step 2: Solve the following optimization problem:

find	η
minimize	$\|\boldsymbol{\eta}\|_\infty + c \cdot obj_t(\boldsymbol{x}+\boldsymbol{\eta})$
such that	$\boldsymbol{x}+\boldsymbol{\eta} \in \boxed{[\mathbf{0},\mathbf{1}]^n}$

Hard Box Constraint

Dealing with Constraints

find	$\boldsymbol{\eta}$
minimize	$\|\boldsymbol{\eta}\|_p + c \cdot obj_{\boldsymbol{t}}(\boldsymbol{x}+\boldsymbol{\eta})$
such that	$\underbrace{\boldsymbol{x}+\boldsymbol{\eta} \in [\mathbf{0},\mathbf{1}]^n}$

Given $\boldsymbol{x}$ is constant, this is the same as enforcing $\boldsymbol{\eta}_i \in [-\boldsymbol{x}_i, 1-\boldsymbol{x}_i]$ for every $\boldsymbol{\eta}_i$. We can then use either of these two methods:

Projected gradient descent (PGD)

"Fit" all coordinates to be within the box

$$\boldsymbol{project}((\boldsymbol{\eta}_1, \dots, \boldsymbol{\eta}_n)) = \big(clip_1(\boldsymbol{\eta}_1), \dots, clip_n(\boldsymbol{\eta}_n)\big)$$

$$clip_i(\boldsymbol{\eta}_i) = \begin{cases} -\boldsymbol{x}_i & \text{if } \boldsymbol{\eta}_i < -\boldsymbol{x}_i \\ \boldsymbol{\eta}_i, & \text{if } \boldsymbol{\eta}_i \in [-\boldsymbol{x}_i, 1-\boldsymbol{x}_i] \\ 1-\boldsymbol{x}_i, & \text{if } \boldsymbol{\eta}_i > 1-\boldsymbol{x}_i \end{cases}$$

LBFGS-B optimizer:

Used by Carlini & Wagner

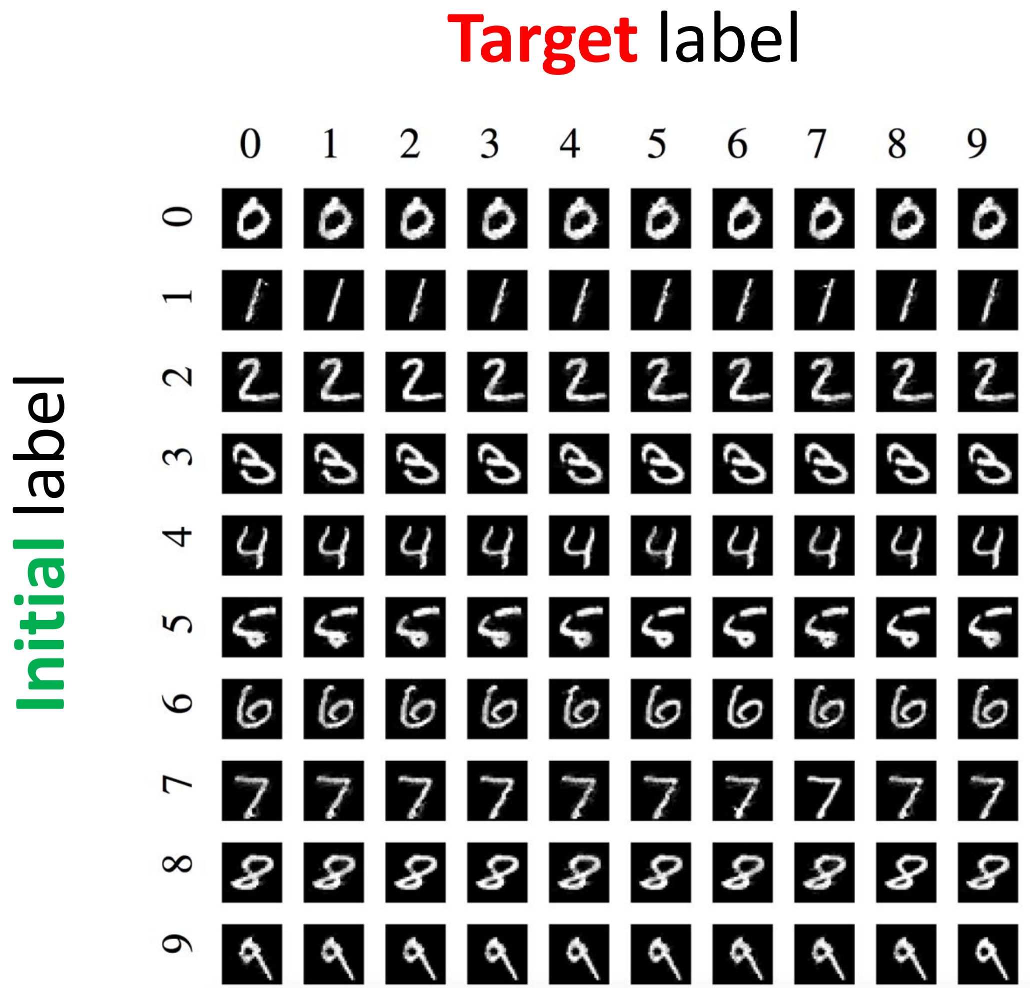
pass each $\boldsymbol{\eta}_i \in [-\boldsymbol{x}_i, 1-\boldsymbol{x}_i]$

separately to the optimizer.

"-B" stands for box constraints

Note: if we also want $\|\boldsymbol{\eta}\|_\infty < e$ then we can also add the box constraints $\boldsymbol{\eta}_i \in [-e, e]$

With this approach we get



What we see is that on the MNIST (digit recognition) data set it is not difficult to get a realistic looking image that fools the neural network classifier…

DeepSpeech Attack: more technically

find $\boldsymbol{\eta}$

minimize $\|\boldsymbol{\eta}\|_\infty + c \cdot obj_t(\boldsymbol{x} + \boldsymbol{\eta})$

such that $\boldsymbol{x} + \boldsymbol{\eta} \in [\mathbf{0}, \mathbf{1}]^n$

Instantiate to audio

easier to optimize (smooth function)

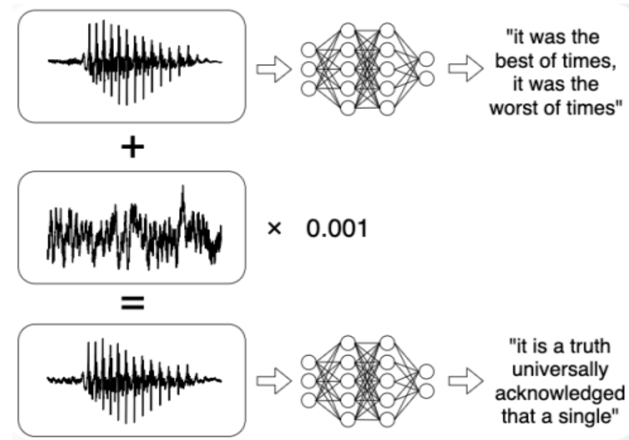
audio specific loss

find $\boldsymbol{\eta}$

minimize $\|\boldsymbol{\eta}\|_2^2 + c \cdot loss_t(\boldsymbol{x} + \boldsymbol{\eta})$

such that $\boldsymbol{\eta}' \in [\mathbf{0}, \boldsymbol{k}]^n$

where $\boldsymbol{\eta}' = \underbrace{\mathbf{20} * \mathbf{log_{10}}(\boldsymbol{\eta})}_{\text{change to decibels}}$



find $\boldsymbol{\eta}$

minimize $\|\boldsymbol{\eta}\|_2^2 + c \cdot loss_t(\boldsymbol{x} + \boldsymbol{\eta})$

such that $\boldsymbol{\eta} \in [\mathbf{1}, \mathbf{10}^{k/20}]^n$

A **re-write** so we work with $\boldsymbol{\eta}$ only.

Now we can project on $\boldsymbol{\eta}$ as usual.

Audio Adversarial Examples: Targeted Attacks on Speech-to-Text, ICML 2018 workshop

Another attack…often used during training

- So far, we looked at FGSM as well as an attack to minimize the distance to the original input (e.g., image, audio)

- Now, we illustrate another attack, a variant of FGSM applied iteratively **with projection**.

- The attack uses Projected Gradient Descent (PGD) and is referred to as a PGD attack.

- This is a commonly used attack for adversarial training: training the network to be robust.

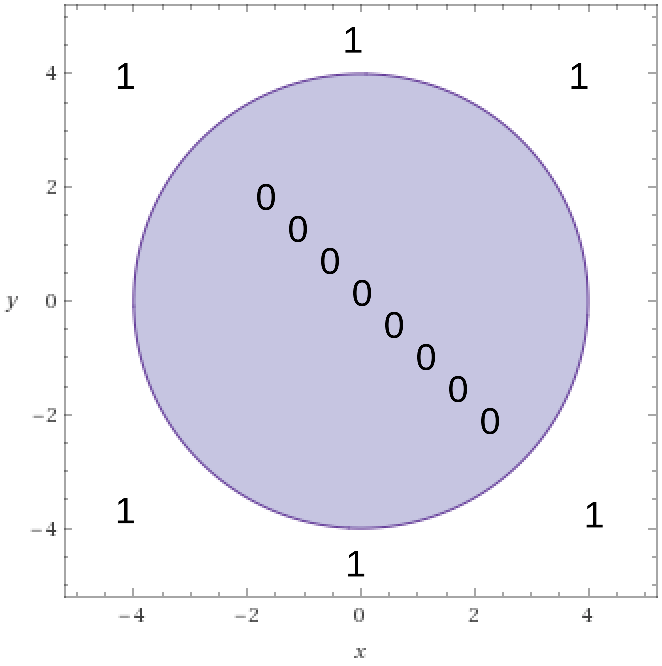
Illustrating the PGD attack

Given a **dataset** of points (x, y) where label is:

0	if $x^2+y^2<16$
1	otherwise

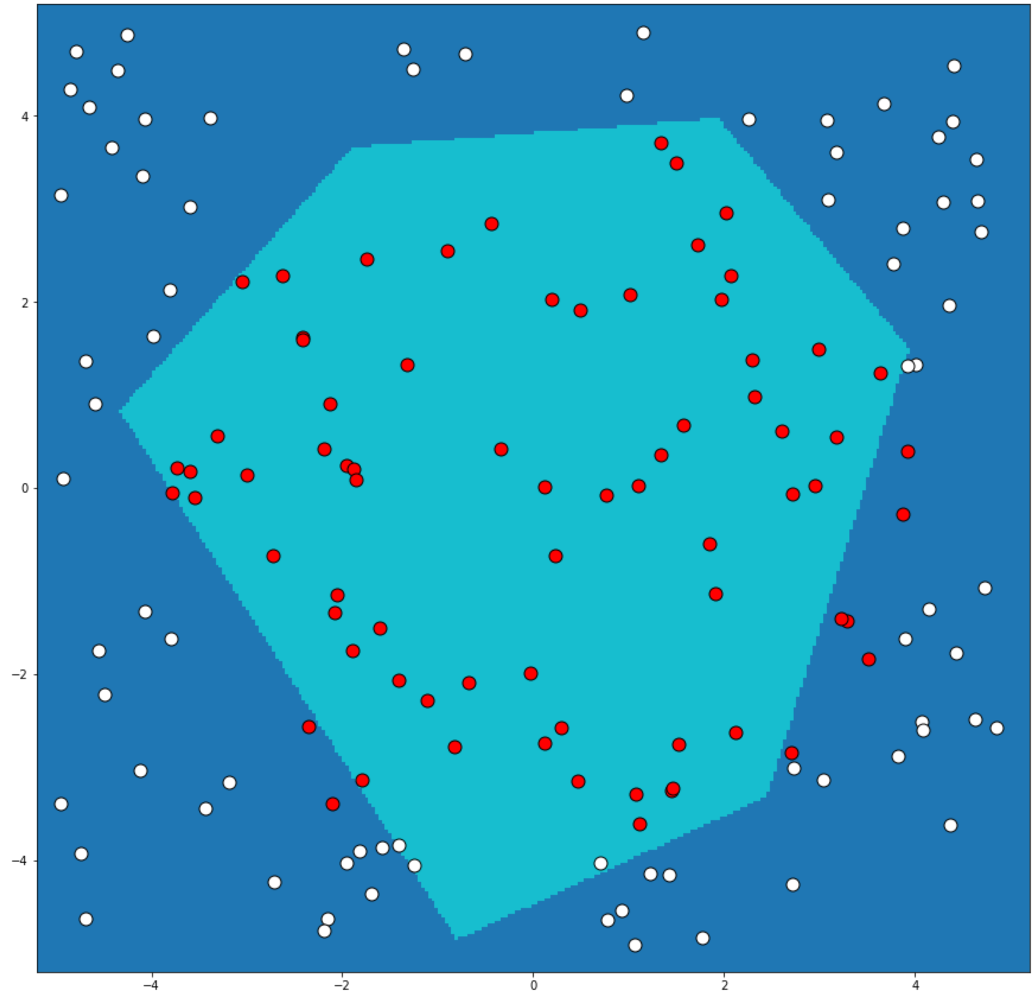
train a neural network to classify the points correctly

Illustrating the PGD attack



https://www.wolframalpha.com/input/?i=plot+x%5E2+%2B+y%5E2+%3C+16

After training we get the classifier:



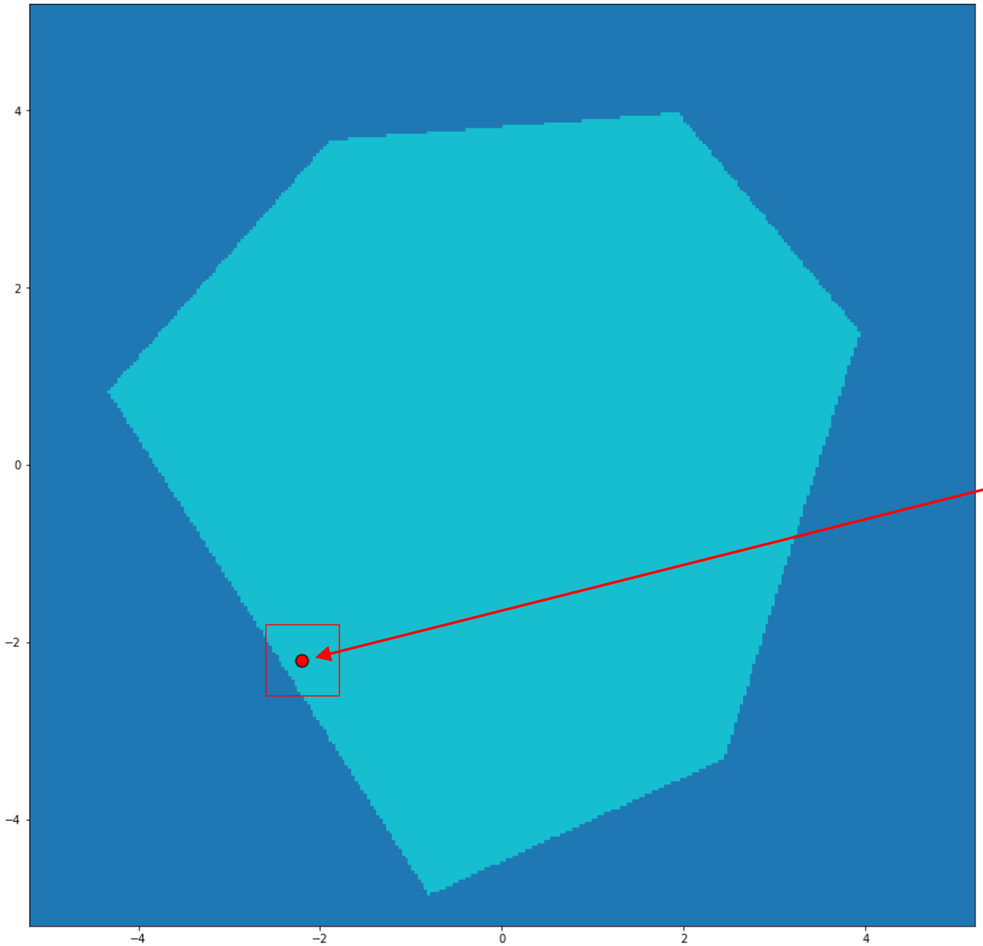
Dark blue – neural network predicts 1 (property does not hold)

Light blue – neural network predicts 0 (property holds)

Red dots – those where property actually holds

White dots – those where property actually does not hold

Lets pick a point...



Goal:

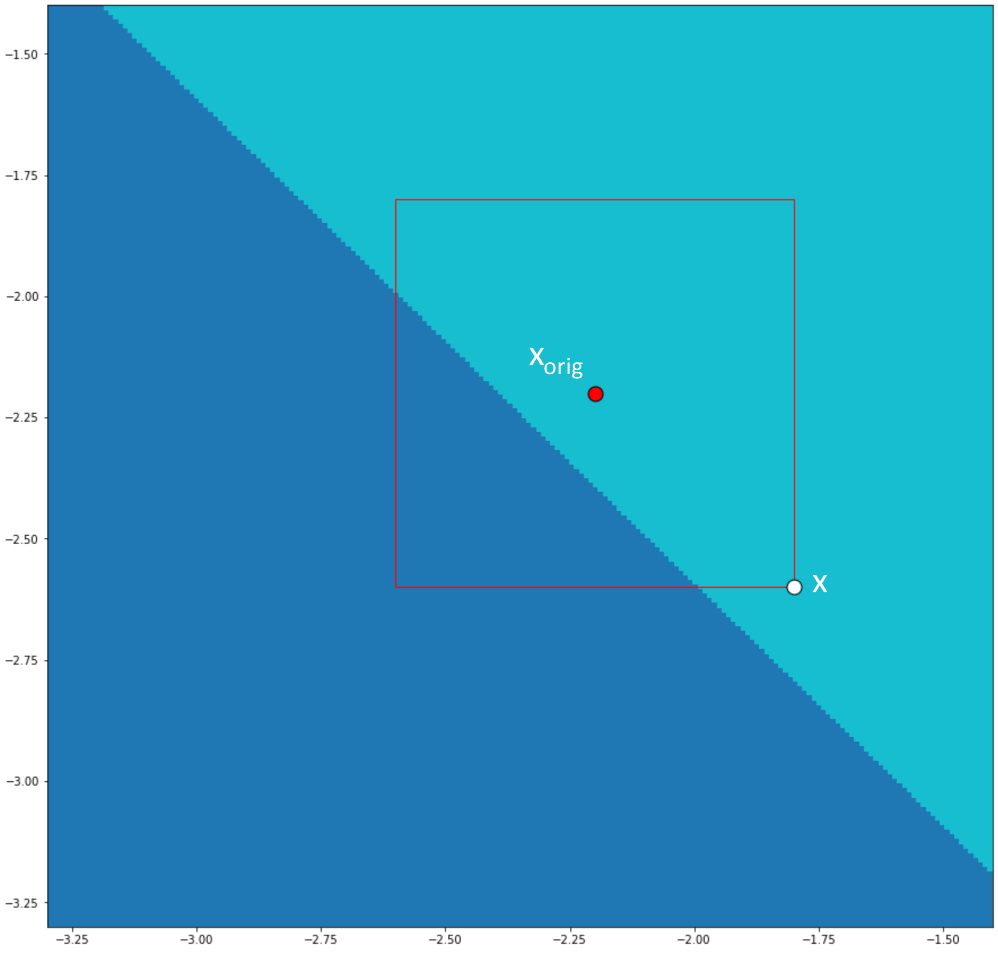
Find adversarial input in

L_{inf} ball around:

$x_{orig} = (-2.2, -2.2)$
(red point)

with ε=0.4

Lets Zoom in a bit...

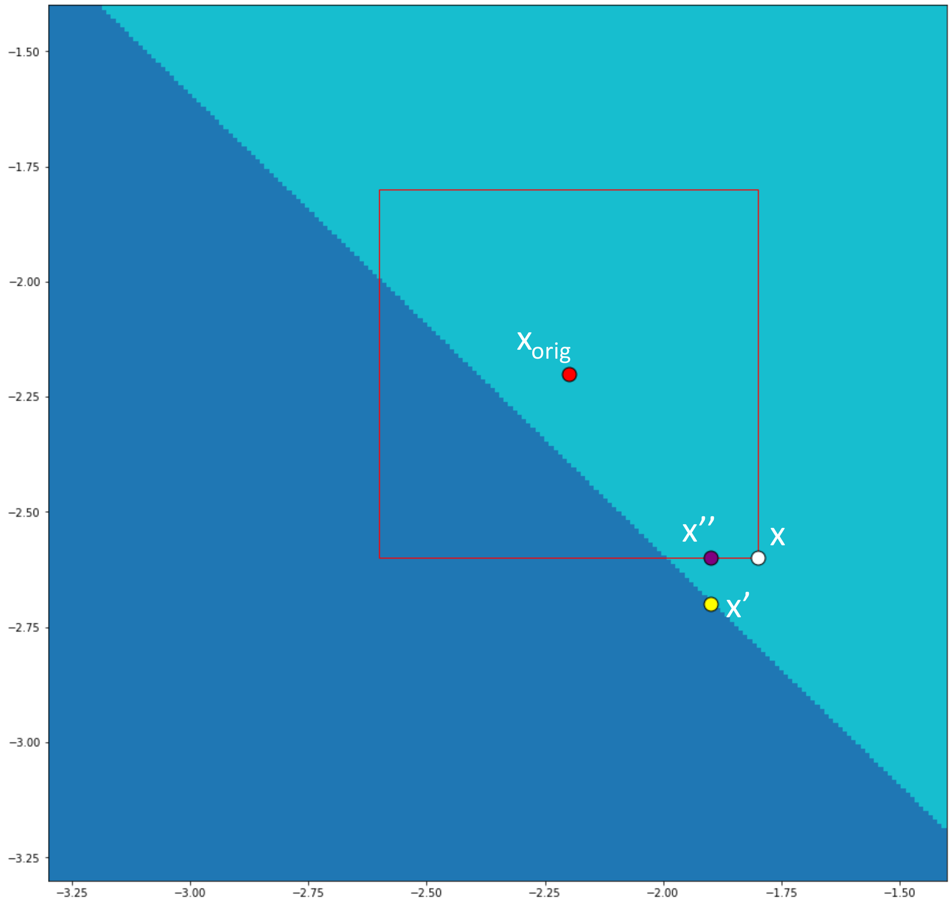


Initialize PGD with:

x = (-1.8, -2.6)

Note: this is just for the example to illustrate projection. In practice, one picks a point at random in the box

PGD Iteration 1



NN(x) = [0.5973, 0.4027]

Loss(x) = 0.5153

∇_x Loss(x) = [-0.852, -1.373]

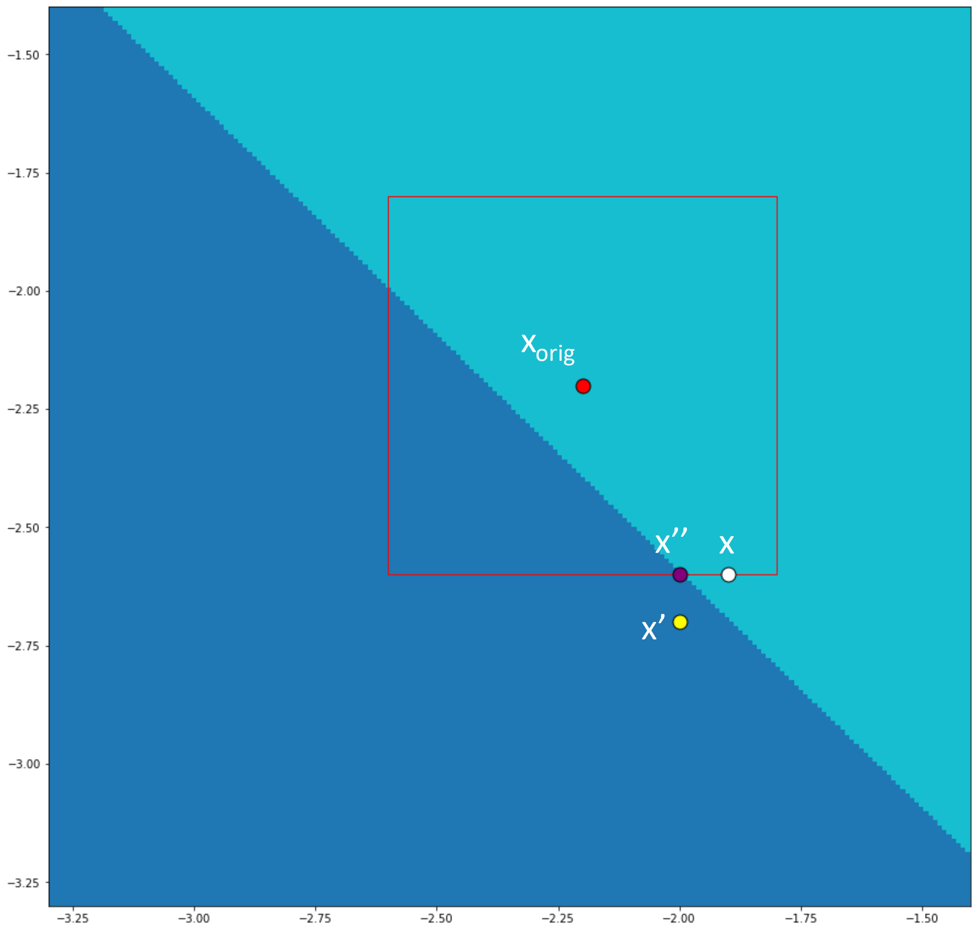
$x' = x + \underbrace{0.1 * \text{sign}(\nabla_x \text{Loss}(x))}^{\text{Change } \Delta}$
= [-1.9, -2.7] (yellow point)

Up-to-here, its just standard untargeted FGSM attack but with **smaller step-size** of 0.1 than ε which is 0.4.

But now we also **project**:

x'' = **project**(x', x_{orig}, ε)
= [-1.9, -2.6] (purple point)

PGD Iteration 2



x'' from before now named x:

NN(x) = [0.5455, 0.4545]
(so point x = (-1.9, -2.6) is
not yet a counter example

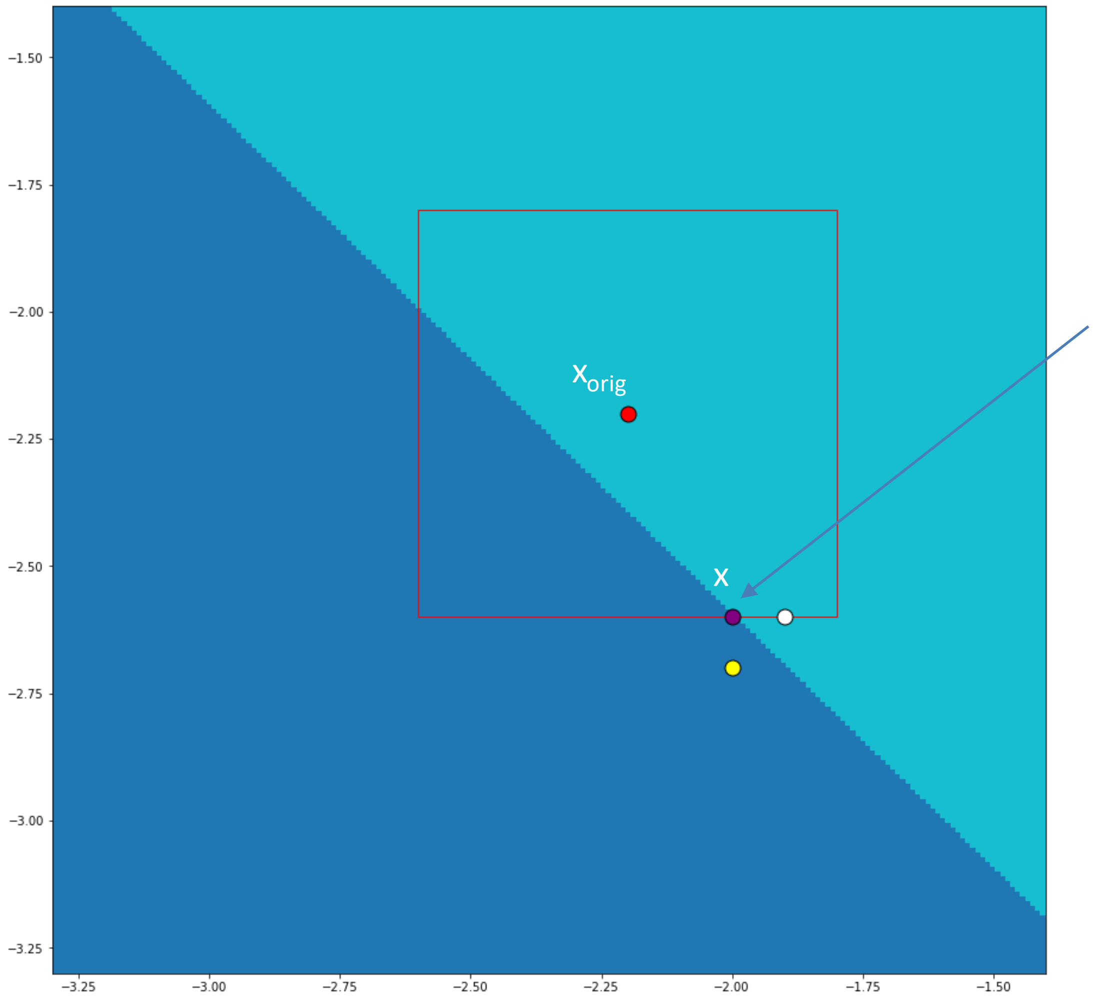
Loss(x) = 0.6060

∇_x Loss(x) =[-0.9621, -1.5493]

$x' = x + \underbrace{0.1 * \text{sign}(\nabla_x \text{Loss}(x))}_{\text{Change } \Delta}$
= [-2, -2.7]

x'' = **project**(x' , x_{orig}, ε)
= [-2, -2.6]

PGD Iteration 3

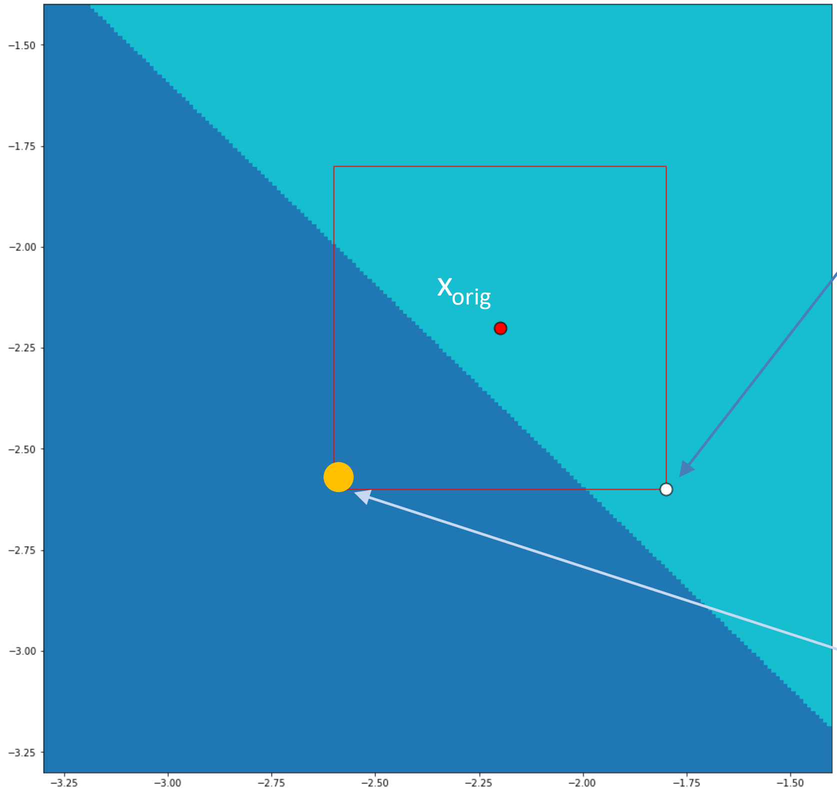


NN(x) = [0.4927, 0.5073]

found adversarial example
x = [-2, -2.6]

Neural network predicts 1, although $(-2)^2 + (-2.6)^2 < 16$ so it should have been classified as 0

Some notes on PGD



- Projection is linear-time in the dimension for L_∞ and L_2 norms.

- An **open problem:** finding efficient projections for various convex regions that are **more expressive than boxes** (e.g., convex polyhedral restrictions).

- The goal of the PGD attack is to **find a point in the region which maximizes the loss** (it may still classify to the same label as x_{orig})

- For our example, we started at the corner. Typically one starts the search with a **random point inside the box**.

- One stops PGD after **a pre-defined number of iterations** (e.g., 10).

- In our example, we always stepped outside the box to illustrate projection, and then projected to the box. It is possible to never step outside the box and thus **projection will have no effect**.

- It is possible the final produced example **is inside the box**, and not on the boundary. However, when we project, if outside the box, we will end up on the boundary.

- In this example, loss is **likely to be highest** somewhere around the big orange point (typically far from the decision boundary). Of course, when we are searching, we **don't know the actual decision boundary**.

- One can implement PGD in **two ways**:
  - **a)** by projecting current **point** x' to the ε-box around x_{orig} as well as [0,1] for each dimension, or
  - **b)** by projecting the **change** Δ to [-ε, +ε] as well as to the constraints needed so each element in the resulting point is between [0,1] (see slide 3 in this lecture)

- Step size (in our example 0.1) is **typically smaller than ε** (in FGSM it is ε).

Another Attack Example: Diffing Networks

Finding a **differencing input**:

Given two neural networks f_1 and f_2 trained to learn the same function $f^*: X \rightarrow C$, find an input $x \in X$ such that $f_1(x) \neq f_2(x)$

DRV_C1: right   DRV_C2: right   DRV_C3: right

DeepXplore: Automated Whitebox Testing of Deep Learning Systems, Pei et al, SOSP'17

Another Attack Example: Diffing Networks

Define the following objective (good if we want $\boldsymbol{f_1(x)}$ to classify $\boldsymbol{x}$ to $\boldsymbol{t}$):

$$\boldsymbol{obj_t(x) = f_1(x)_t - f_2(x)_t}$$

$\boldsymbol{f_i(x)_t}$ returns the probability that $\boldsymbol{f_i}$ predicts $\boldsymbol{x}$ to be $\boldsymbol{t}$

We can use absolute value loss if we just want to get a different classification by both (need not be $\boldsymbol{t}$).

Select input $\boldsymbol{x \in X}$ which classifes as $\boldsymbol{t}$ with both networks

while $\boldsymbol{class(f_1(x)) = class(f_2(x))}$:

$$\boldsymbol{x = x + \epsilon \cdot \frac{\partial\, obj_t(x)}{\partial\, x}}$$

return $\boldsymbol{x}$

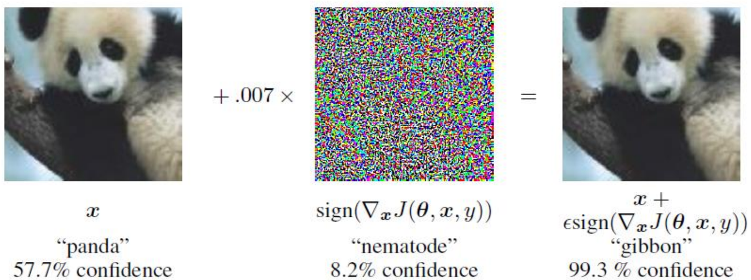
Maximize loss: make $\boldsymbol{f_1}$ more confident about $\boldsymbol{t}$ while making $\boldsymbol{f_2}$ less confident about $\boldsymbol{t}$

Summary of adversarial attacks

Attack Type	Region	Optimization	Outcome
FGSM (targeted, untargeted)	Change $\boldsymbol{\eta}$ fixed to [-ε, +ε].	Take exactly one ε-sized step	Produced example will be on boundary of region.
PGD (typically untargeted, but can be targeted)	Can be instantiated with any region one can project to.	Take many steps. Uses projection to stay inside region. For special case of l_∞, step size smaller than ε.	Result will be inside region. Tries to maximize loss.
C&W [Images] (presented as targeted)	No real restriction, except image has to be in [0,1] (like all other methods). This restricts the region for the change $\boldsymbol{\eta}$: $\boldsymbol{\eta}$ has to be bounded s.t. original image + $\boldsymbol{\eta}$ stays in [0,1].	Aims to produce a change $\boldsymbol{\eta}$ with small l_∞. Takes many steps, using LBFGS-B to ensure $\boldsymbol{\eta}$ stays in bounds.	Result will be inside [0,1], with a hopefully small l_∞ distance from original image.
C&W [Audio] (presented as targeted)	A fixed region for $\boldsymbol{\eta}$.	Aims to produce a change $\boldsymbol{\eta}$ with small squared l_2. Takes many steps, using LBFGS-B to ensure $\boldsymbol{\eta}$ stays in bounds.	Result will be inside the fixed $\boldsymbol{\eta}$ region, with a hopefully small l_2 distance from original sound wave.
Diffing Networks (targeted)	Can work with a fixed region around change $\boldsymbol{\eta}$.	Aims to produce a change $\boldsymbol{\eta}$ where one neural networks classifies image as $\boldsymbol{t}$ while the other as not $\boldsymbol{t}$.	Ideally, a result where two networks disagree on their classification.

Lecture Summary

Deep Learning is susceptible to adversarial examples



Generating Adversarial examples (an optimization problem)

- FGSM
- C&W (minimize perturbation)
- PGD
- Diffing

An example of the PGD attack