Reliable and Interpretable Artificial Intelligence

Lecture 3: Adversarial Attacks II

Martin Vechev
ETH Zurich

Fall 2020

http://www.sri.inf.ethz.ch
Recall: Our Optimization Problem

Two steps:

**Step 1:** Define an objective function $obj_t$ such that:

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

**Step 2:** Solve the following optimization problem:

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

Hard Box Constraint
Dealing with Constraints

\[
\begin{align*}
\text{find} \quad & \boldsymbol{\eta} \\
\text{minimize} \quad & \|\boldsymbol{\eta}\|_p + c \cdot \text{obj}_t(x + \boldsymbol{\eta}) \\
\text{such that} \quad & x + \boldsymbol{\eta} \in [0, 1]^n \\
\end{align*}
\]

Given \( x \) is constant, this is the same as enforcing \( \eta_i \in [-x_i, 1 - x_i] \) for every \( \eta_i \). We can then use either of these two methods:

**Projected gradient descent (PGD)**

"Fit" all coordinates to be within the box

\[
\text{project}((\eta_1, \ldots, \eta_n)) = \left(\text{clip}_1(\eta_1), \ldots, \text{clip}_n(\eta_n)\right)
\]

\[
\text{clip}_i(\eta_i) = \begin{cases} 
- x_i & \text{if } \eta_i < - x_i \\
\eta_i & \text{if } \eta_i \in [-x_i, 1 - x_i] \\
1 - x_i & \text{if } \eta_i > 1 - x_i 
\end{cases}
\]

**LBFGS-B optimizer:**

Used by Carlini & Wagner

Pass each \( \eta_i \in [-x_i, 1 - x_i] \) separately to the optimizer.

"-B" stands for box constraints

Note: if we also want \( \|\eta\|_\infty < e \) then we can also add the box constraints \( \eta_i \in [-e, e] \)
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

\[
\begin{align*}
\text{find} & \quad \eta \\
\text{minimize} & \quad \|\eta\|_\infty + c \cdot \text{obj}_t(x + \eta) \\
\text{such that} & \quad x + \eta \in [0, 1]^n
\end{align*}
\]

\[
\begin{align*}
\text{find} & \quad \eta \\
\text{minimize} & \quad \|\eta\|_2^2 + c \cdot \text{loss}_t(x + \eta) \\
\text{such that} & \quad \eta' \in [0, k]^n \\
\text{where} & \quad \eta' = 20 \times \log_{10}(\eta)
\end{align*}
\]

\[
\begin{align*}
\text{find} & \quad \eta \\
\text{minimize} & \quad \|\eta\|_2^2 + c \cdot \text{loss}_t(x + \eta) \\
\text{such that} & \quad \eta \in [1, 10^{k/20}]^n
\end{align*}
\]

A re-write so we work with \( \eta \) only.

Now we can project on \( \eta \) 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.
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+%2By%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
Goal:

Find adversarial input in $L_{\infty}$ ball around:

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

with $\varepsilon = 0.4$
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

$\text{NN}(x) = [0.5973, 0.4027]$

$\text{Loss}(x) = 0.5153$

$\nabla_x \text{Loss}(x) = [-0.852, -1.373]$

$x' = x + 0.1 \cdot \text{sign}(\nabla_x \text{Loss}(x))$

$= [-1.9, -2.7]$ (yellow point)

Up-to-here, it's just standard untargeted FGSM attack but with smaller step-size of 0.1 than $\epsilon$ which is 0.4.

But now we also project:

$x'' = \text{project}(x', x_{\text{orig}}, \epsilon)$

$= [-1.9, -2.6]$ (purple point)
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

$\nabla_x \text{Loss}(x) = [-0.9621, -1.5493]$

$x' = x + 0.1 \times \text{sign}(\nabla_x \text{Loss}(x))$

= [-2, -2.7]

$x'' = \text{project}(x', x_{\text{orig}}, \varepsilon)$

= [-2, -2.6]
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

- 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 \(x_{\text{orig}}\)).

- 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 \(\varepsilon\)-box around \(x_{\text{orig}}\) as well as \([0,1]\) for each dimension, or
  - **b)** by projecting the change \(\Delta\) to \([-\varepsilon, +\varepsilon]\) 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 \(\varepsilon\)** (in FGSM it is \(\varepsilon\)).

- **Projection** is linear-time in the dimension for \(L_\infty\) 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).
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 $f_1(x) \neq f_2(x)$

---

Another Attack Example: Diffing Networks

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

$$obj_t(x) = f_1(x)_t - f_2(x)_t$$

$f_i(x)_t$ returns the probability that $f_i$ predicts $x$ to be $t$

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

Select input $x \in X$ which classifies as $t$ with both networks

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

$$x = x + \epsilon \cdot \frac{\partial obj_i(x)}{\partial x}$$

return $x$

Maximize loss: make $f_1$ more confident about $t$ while making $f_2$ less confident about $t$
# Summary of adversarial attacks

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