15-780 – Graduate Artificial Intelligence: Adversarial attacks and provable defenses

J. Zico Kolter (this lecture) and Nihar Shah
Carnegie Mellon University
Spring 2019

Portions based upon joint work with Eric Wong
Outline

Adverarial attacks on machine learning

Robust optimization

Provable defenses for deep classifiers

Experimental results
Outline

Adversarial attacks on machine learning

Robust optimization

Provable defenses for deep classifiers

Experimental results
Adversarial attacks

\[ x + .007 \times \text{sign}(\nabla_x J(\theta, x, y)) = \]

"panda"  
57.7% confidence

"nematode"  
8.2% confidence

"gibbon"  
99.3% confidence

[ Szegedy et al., 2014, Goodfellow et al., 2015 ]
How adversarial attacks work

We are focusing on test time attacks: train on clean data and attackers tries to fool the trained classifier at test time.

To keep things tractable, we are going to restrict our attention to $\ell_\infty$ norm bounded attacks: the adversary is free to manipulate inputs within some $\ell_\infty$ ball around the true example

$$\tilde{x} = x + \Delta, \quad \|\Delta\|_\infty \leq \epsilon$$

**Basic method:** given input $x \in X$, output $y \in Y$, hypothesis $h_\theta: X \rightarrow Y$, and loss function $\ell: Y \times Y \rightarrow \mathbb{R}_+$, adjust $x$ to maximum loss:

$$\text{maximize} \quad \ell(h_\theta(x + \Delta), y) \quad \text{subject to} \quad \|\Delta\|_\infty \leq \epsilon$$

Other variants we will see shortly (e.g., maximizing specific target class)
A summary of adversarial example research

😊 Distillation prevents adversarial attacks! [Papernot et al., 2016]

🤔 No it doesn’t! [Carlini and Wagner, 2017]

😊 No need to worry given translation/rotation! [Lu et al., 2017]

🤔 Yes there is! [Athalye and Sutskever, 2017]

😊 We have 9 new defenses you can use! [ICLR 2018 papers]

🤔 Broken before review period had finished! [Athalye et al., 2018]

My view: the attackers are winning, we need to get out of this arms race
A slightly better summary

Many heuristic methods for defending against adversarial examples [e.g., Goodfellow et al., 2015; Papernot et al., 2016; Madry et al., 2017; Tramér et al., 2017; Roy et al., 2017]

• Keep getting broken, unclear if/when we’ll find the right heuristic

Formal methods approaches to verifying networks via tools from SMT, integer programming, SAT solving, etc. [e.g., Carlini et al., 2017; Ehlers 2017; Katz et al., 2017; Huang et al., 2017]

• Limited to small networks by combinatorial optimization

Our work: Tractable, provable defenses against adversarial examples via convex relaxations [also related: Raghunathan et al., 2018; Staib and Jegelka 2017; Madry et al., 2017; Sinha et al., 2017; Hein and Andriushchenko 2017; Peck et al, 2017]
Adversarial examples in the real world

Sharif et al., 2016

Evtimov et al., 2017

Athalye et al., 2017

Note: only the last one here is possibly an $\ell_\infty$ perturbation
The million dollar question

How can we design (deep) classifiers that are provably robust to adversarial attacks?
Outline

Adversarial attacks on machine learning

Robust optimization

Provable defenses for deep classifiers

Experimental results
Robust optimization

A area of optimization that goes almost 50 years [Soyster, 1973; see Ben-Tal et al., 2011]

**Robust optimization (as applied to machine learning):** instead of minimizing loss at training points, minimize *worst case* loss in some ball around the points

$$\min_{\theta} \sum_{i} \ell(h_{\theta}(x_i) \cdot (h_{\phi_i}(x_i + \Delta) \cdot y_i))$$

$$\equiv \min_{\theta} \sum_{i} \ell(h_{\theta}(x_i) \cdot y_i - \epsilon \|\theta\|_1)$$

(for linear classifiers)
Proof of robust machine learning property

**Theorem:** For linear hypothesis function \( h_\theta(x) = \theta^T x \), binary output \( y \in \{-1, +1\} \), and classification loss \( \ell(h_\theta(x) \cdot y) \)

\[
\max_{\|\Delta\|_\infty \leq \epsilon} \ell(h_\theta(x + \Delta) \cdot y) = \ell(h_\theta(x) \cdot y - \epsilon \|\theta\|_1)
\]

**Proof:** Because classification loss is monotonic decreasing

\[
\max_{\|\Delta\|_\infty \leq \epsilon} \ell(h_\theta(x + \Delta) \cdot y) = \ell\left(\min_{\|\Delta\|_\infty \leq \epsilon} h_\theta(x + \Delta) \cdot y\right)
\]

\[
= \ell\left(\min_{\|\Delta\|_\infty \leq \epsilon} \theta^T (x + \Delta) \cdot y\right)
\]

Theorem follows from the fact that

\[
\min_{\|\Delta\|_\infty \leq \epsilon} \theta^T \Delta = -\epsilon \|\theta\|_1
\]
What to do at test time?

This procedure prevents the possibility of adversarial examples at training time, but what about at test time?

**Basic idea:** If we make a prediction at a point, and this prediction does not change within the $\ell_\infty$ ball of $\epsilon$ around the point, then this cannot be an adversarial example (i.e., we have a zero-false negative detector)
Outline

Adversarial attacks on machine learning

Robust optimization

Provable defenses for deep classifiers

Experimental results

Based upon work in:
Wong and Kolter, “Provable defenses against adversarial examples via the convex adversarial polytope”, ICML 2018
https://arxiv.org/abs/1711.00851
The trouble with deep networks

In deep networks, the “image” (adversarial polytope) of a norm bounded perturbation is non-convex, we can’t easily optimize over it.

Our approach: instead, form convex outer bound over the adversarial polytope, and perform robust optimization over this region (applies specifically to networks with ReLU nonlinearities)
Convex outer approximations

Optimization over convex outer adversarial polytope provides *guarantees* about robustness to adversarial perturbations.

... so, how do we compute and optimize over this bound?
Adversarial examples as optimization

Finding the worst-case adversarial perturbation (within true adversarial polytope), can be written as a non-convex problem

\[
\begin{align*}
\text{minimize} & \quad (z_k)^y - (z_k)^{y_{\text{target}}} \\
\text{subject to} & \quad \|z_1 - x\|_\infty \leq \epsilon \\
& \quad \hat{z}_{i+1} = W_i z_i + b_i, \quad i = 1, \ldots, k - 1 \\
& \quad z_i = \max\{\hat{z}_i, 0\}, \quad i = 2, \ldots, k - 1 
\end{align*}
\]
Adversarial examples as optimization

Finding the worst-case adversarial perturbation (within true adversarial polytope), can be written as a non-convex problem

\[
\begin{align*}
\text{minimize} \quad & (z_k)_y - (z_k)_{y_{\text{target}}} \\
\text{subject to} \quad & \|z_1 - x\|_{\infty} \leq \epsilon \\
& \hat{z}_{i+1} = W_i z_i + b_i, \quad i = 1, \ldots, k - 1 \\
& z_i = \max\{\hat{z}_i, 0\}, \quad i = 2, \ldots, k - 1
\end{align*}
\]
Adversarial examples as optimization

Finding the worst-case adversarial perturbation (within true adversarial polytope), can be written as a non-convex problem

$$\begin{align*}
\text{minimize} & \quad (z_k)_{y^*} - (z_k)_{y^{\text{target}}} \\
\text{subject to} & \quad z_1 - x \leq \epsilon \\
& \quad z_1 - x \geq -\epsilon \\
& \quad \hat{z}_{i+1} = W_i z_i + b_i, \quad i = 1, \ldots, k - 1 \\
& \quad z_i = \max\{\hat{z}_i, 0\}, \quad i = 2, \ldots, k - 1
\end{align*}$$
Adversarial examples as optimization

Finding the worst-case adversarial perturbation (within true adversarial polytope), can be written as a non-convex problem

\[
\begin{align*}
\text{minimize} & \quad (z_k)^{y^*} - (z_k)^{y^{\text{target}}} \\
\text{subject to} & \quad z_1 - x \leq \epsilon \\
& \quad z_1 - x \geq -\epsilon \\
& \quad \hat{z}_{i+1} = W_i \hat{z}_i + b_i, \quad i = 1, \ldots, k - 1 \\
& \quad z_i = \max\{\hat{z}_i, 0\}, \quad i = 2, \ldots, k - 1
\end{align*}
\]
Idea #1: Convex bounds on ReLU nonlinearities

Suppose we have some upper and lower bound $\ell, u$ on the values that a particular (pre-ReLU) activation can take on, for this particular example $x$

Then we can relax the ReLU “constraint” to its convex hull

$$
\begin{align*}
\min_{z, \hat{z}} & \quad (z_k)_{y^*} - (z_k)_{y_{\text{target}}} \\
\text{subject to} & \quad z_1 - x \leq \epsilon \\
& \quad z_1 - x \geq -\epsilon \\
& \quad \hat{z}_{i+1} = W_i \hat{z}_i + b_i, \quad i = 1, \ldots, k - 1 \\
& \quad z_i = \max\{\hat{z}_i, 0\}, \quad i = 2, \ldots, k - 1
\end{align*}
$$
Idea #1: Convex bounds on ReLU nonlinearities

Suppose we have some upper and lower bound $\ell, u$ on the values that a particular (pre-ReLU) activation can take on, for this particular example $x$.

Then we can relax the ReLU “constraint” to its convex hull

$$
\begin{align*}
\text{minimize} & \quad (z_k)_y^r - (z_k)_y^\text{target} \\
\text{subject to} & \quad z_1 - x \leq \epsilon \\
& \quad z_1 - x \geq -\epsilon \\
& \quad \hat{z}_{i+1} = W_i \hat{z}_i + b_i, \quad i = 1, \ldots, k-1 \\
& \quad (\hat{z}_i, z_i) \in C(l_i, u_i), \quad i = 2, \ldots, k-1
\end{align*}
$$

A linear program!
Idea #2: Exploiting duality

While the previous formulation is nice, it would require solving an LP (with the number of variables equal to the number of hidden units in network), once for each example, for each SGD step

- (This even ignores how to compute upper and lower bounds $\ell, u$)

We’re going to use the “duality trick”, the fact that any feasible dual solution gives a lower bound on LP solution
An amazing property

It turns out that we can compute a (empirically, close to optimal) dual feasible solution using a single backward pass through the network (really a slightly augmented form of the backprop network)

\[
\begin{align*}
\text{minimize} & \quad z^T z_k \\
\text{subject to} & \quad \|z_1 - x\|_\infty \leq \epsilon \\
& \quad (z_{i+1}, W_i z_i + b_i) \in C(\ell_i, u_i)
\end{align*}
\]

\[
\begin{align*}
\text{maximize} & \quad J_{\epsilon, W, b}(\nu, x) \equiv -\sum_{i=1}^{k-1} \nu_{i+1}^T b_i - x^T \hat{\nu}_1 - \epsilon \|\hat{\nu}_1\|_1 + \sum_{i=1}^{k-1} \sum_{j \in J_i} \ell_{i,j} [\nu_{i,j}]_+ \\
\text{subject to} & \quad \nu_k = -c \\
& \quad \hat{\nu}_i = W_i^T \nu_{i+1}, \quad i = k - 1, \ldots, 1 \\
& \quad \nu_i = f_i(\hat{\nu}_i, \alpha_i; \ell_i, u_i), \quad i = k - 1, \ldots, 2
\end{align*}
\]
An amazing property

It turns out that we can compute a (empirically, close to optimal) dual feasible solution using a single backward pass through the network (really a slightly augmented form of the backprop network)

\[
\begin{align*}
\text{minimize} & \quad c^T z_k \\
\text{subject to} & \quad \|z_1 - x\|_\infty \leq \epsilon \\
& \quad (z_{i+1}, W_i z_i + b_i) \in \mathcal{C}(\ell_i, u_i)
\end{align*}
\]

\[
\begin{align*}
\text{maximize} & \quad J_{\epsilon, W, b}(\nu, x) \equiv -\sum_{i=1}^{k-1} v_{i+1}^T b_i - x^T \hat{v}_1 - \epsilon \|\hat{v}_1\|_1 + \sum_{i=1}^{k-1} \sum_{j \in J_i} \ell_{i,j} [\nu_{i,j}]_+ \\
\text{subject to} & \quad \nu_k = -c \\
& \quad \hat{v}_i = W_i^T \nu_{i+1}, \quad i = k - 1, \ldots, 1 \\
& \quad \nu_i = f_i(\hat{v}_i, \alpha_i; \ell_i, u_i), \quad i = k - 1, \ldots, 2
\end{align*}
\]

Set of all activations in layer $i$ that can cross zero
An amazing property

It turns out that we can compute a (empirically, close to optimal) dual feasible solution **using a single backward pass through the network** (really a slightly augmented form of the backprop network)

\[
\begin{align*}
\text{minimize } & & \mathbf{c}^T \mathbf{z}_k \\
\text{subject to } & & \|\mathbf{z}_1 - \mathbf{x}\|_\infty \leq \epsilon \\
& & (\mathbf{z}_{i+1}, \mathbf{W}_i \mathbf{z}_i + b_i) \in \mathcal{C}(\ell_i, \mathbf{u}_i)
\end{align*}
\]

\[
\begin{align*}
\text{maximize } & & J_{\epsilon, \mathbf{W}, b}(\mathbf{v}, \mathbf{x}) \equiv - \sum_{i=1}^{k-1} \mathbf{v}_{i+1}^T b_i - \mathbf{x}^T \hat{\mathbf{v}}_1 - \epsilon \|\hat{\mathbf{v}}_1\|_1 + \sum_{i=1}^{k-1} \sum_{j \in J_i} \ell_{i,j} [\mathbf{v}_{i,j}]_+ \\
\text{subject to } & & \mathbf{v}_k = -c \\
& & \hat{\mathbf{v}}_i = \mathbf{W}^T \mathbf{v}_{i+1}, \quad i = k - 1, \ldots, 1 \\
& & \mathbf{v}_i = \mathbf{f}_i(\hat{\mathbf{v}}_i, \alpha_i; \ell_i, \mathbf{u}_i), \quad i = k - 1, \ldots, 2
\end{align*}
\]

Derivative of ReLU with slightly modification on $J_i$
An amazing property

It turns out that we can compute a (empirically, close to optimal) dual feasible solution using a single backward pass through the network (really a slightly augmented form of the backprop network)

\[
\begin{align*}
\text{minimize} & \quad c^T z_k \\
\text{subject to} & \quad \|z_1 - x\|_\infty \leq \epsilon \\
& \quad (z_{i+1}, W_i z_i + b_i) \in C(\ell_i, u_i)
\end{align*}
\]

\[
\begin{align*}
\text{maximize} & \quad J_{\epsilon, W, b}(\nu, x) \equiv -\sum_{i=1}^{k-1} \nu_{i+1}^T b_i - x^T \hat{\nu}_1 - \epsilon \|\hat{\nu}_1\|_1 + \sum_{i=1}^{k-1} \sum_{j \in J_i} \ell_{i,j} [\nu_{i,j}]_+
\end{align*}
\]

subject to
\[
\begin{align*}
\nu_k &= -c \\
\hat{\nu}_i &= W^T \nu_{i+1}, \quad i = k - 1, \ldots, 1 \\
\nu_i &= f_i(\hat{\nu}_i, \alpha_i; \ell_i, u_i), \quad i = k - 1, \ldots, 2
\end{align*}
\]

Almost identical to backprop network
An amazing property

It turns out that we can compute a (empirically, close to optimal) dual feasible solution using a single backward pass through the network (really a slightly augmented form of the backprop network)

\[
\begin{align*}
\text{minimize} & \quad c^T z_k \\
\text{subject to} & \quad \|z_1 - x\|_\infty \leq \epsilon \\
& \quad (z_{i+1}, W_i z_i + b_i) \in C(\ell_i, u_i)
\end{align*}
\]

\[
\begin{align*}
\text{maximize} & \quad J_{\epsilon, W, b}(\nu, x) \equiv - \sum_{i=1}^{k-1} \nu_{i+1}^T b_i - x^T \tilde{v}_1 - \epsilon \|\tilde{v}_1\|_1 + \sum_{i=1}^{k-1} \sum_{j \in J_i} \ell_{i,j} [\nu_{i,j}]_+ \\
\text{subject to} & \quad \nu_k = -c \\
& \quad \tilde{v}_i = W^T \nu_{i+1}, \quad i = k - 1, \ldots, 1 \\
& \quad \nu_i = f_i(\tilde{v}_i, \alpha_i; \ell_i, u_i), \quad i = k - 1, \ldots, 2
\end{align*}
\]

Objective at $\epsilon = 0$
An amazing property

It turns out that we can compute a (empirically, close to optimal) dual feasible solution **using a single backward pass through the network** (really a slightly augmented form of the backprop network)

\[
\begin{align*}
\text{minimize} & \quad c^T z_k \\
\text{subject to} & \quad \|z_1 - x\|_\infty \leq \epsilon \\
& \quad (z_{i+1}, W_i z_i + b_i) \in \mathcal{C}(\ell_i, u_i)
\end{align*}
\]

\[
\begin{align*}
\text{maximize} & \quad J_{\epsilon, W, b}(\nu, x) \equiv -\sum_{i=1}^{k-1} \nu_{i+1}^T b_i - x^T \hat{\nu}_1 - \epsilon \|\hat{\nu}_1\|_1 + \sum_{i=1}^{k-1} \sum_{j \in J_i} \ell_{i,j} [\nu_{i,j}]_+ \\
\text{subject to} & \quad \nu_k = -c \\
& \quad \hat{\nu}_i = W_i^T \nu_{i+1}, \quad i = k - 1, \ldots, 1 \\
& \quad \nu_i = f_i(\hat{\nu}_i, \alpha_i; \ell_i, u_i), \quad i = k - 1, \ldots, 2
\end{align*}
\]

Robustness penalty (same form as in linear case)
An amazing property

It turns out that we can compute a (empirically, close to optimal) dual feasible solution using a single backward pass through the network (really a slightly augmented form of the backprop network)

\[
\begin{align*}
\text{minimize} & \quad z^T z_k \\
\text{subject to} & \quad \|z_1 - x\|_\infty \leq \epsilon \\
& \quad (z_{i+1}, W_i z_i + b_i) \in \mathcal{C}(\ell_i, u_i)
\end{align*}
\]

\[
\begin{align*}
\text{maximize} & \quad J_{\epsilon, W, b}(\nu, x) \equiv -\sum_{i=1}^{k-1} \nu_{i+1}^T b_i - x^T \hat{\nu}_1 - \epsilon \|\hat{\nu}_1\|_1 + \sum_{i=1}^{k-1} \sum_{j \in J_i} \ell_{i,j} [\nu_{i,j}]_+
\end{align*}
\]

subject to

\[
\begin{align*}
\nu_1 &= -c \\
\hat{\nu}_i &= W^T \nu_{i+1}, \quad i = k - 1, \ldots, 1 \\
\nu_i &= f_i(\hat{\nu}_i, \alpha_i; \ell_i, u_i), \quad i = k - 1, \ldots, 2
\end{align*}
\]

Additional penalty for violating ReLU constraint
Idea #3: Iterative lower and upper bounds

A meaningful bound requires good lower and upper bounds $\ell_i, u_i$, but these are just themselves the solution to a “mini” optimization problem.

Incrementally build bounds by solving LP for each activation.

Need some tricks to make this efficient: use same (particular) $\alpha$ for dual problems, compute multiplications in the right order in objective.
Putting it all together

In the end, instead of minimizing the traditional loss...

$$\min_{\theta} \sum_{i=1}^{m} \ell(h_{\theta}(x^{(i)}), y^{(i)})$$

…we just minimize a loss with a different network, involving a few forward and backward passes, and we get a guaranteed bound on worst-case loss (or error) for any norm-bounded adversarial attack

$$\min_{\theta} \sum_{i=1}^{m} \ell(J_{\epsilon, \theta}(x^{(i)}), y^{(i)})$$

where our bound guarantees that

$$\ell(J_{\epsilon, \theta}(x^{(i)}), y^{(i)}) \geq \max_{\|\Delta\| \leq \epsilon} \ell(h_{\theta}(x^{(i)} + \Delta), y^{(i)})$$
Outline

Adversarial attacks on machine learning

Robust optimization

Provable defenses for deep classifiers

Experimental results
2D Toy Example

Simple 2D toy problem, 2-100-100-100-2 MLP network, trained with Adam (learning rate = 0.001, no real hyperparameter tuning)
Strided ConvNet (Conv16x4x4, Conv32x4x4, FC100, FC10) ReLUs following each layer, convolutions have stride=2, $\epsilon = 0.1$

Standard and robust errors on MNIST

- Standard deep network: 100%
- Robust linear classifier: 17%
- Our method: 1.8%
- Ragunathan et al., 2018: 5%

Error | Robust error bound
MNIST Attacks

We can also look at how well real attacks perform at $\epsilon = 0.1$.
Convergence

Training does take substantially longer (2 hours), and requires more epochs than standard training.

Method does largely avoid overfitting (adversarial robustness is a powerful regularizer), so we want to consider larger architectures.
Results on additional tasks

<table>
<thead>
<tr>
<th>Problem</th>
<th>Robust</th>
<th>$\epsilon$</th>
<th>Test error</th>
<th>FGSM error</th>
<th>PGD error</th>
<th>Robust error bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>$\times$</td>
<td>0.1</td>
<td>1.07%</td>
<td>50.01%</td>
<td>81.68%</td>
<td>100%</td>
</tr>
<tr>
<td>MNIST</td>
<td>$\sqrt{}$</td>
<td>0.1</td>
<td>1.80%</td>
<td>3.93%</td>
<td>4.11%</td>
<td>5.82%</td>
</tr>
<tr>
<td>FASHION-MNIST</td>
<td>$\times$</td>
<td>0.1</td>
<td>9.36%</td>
<td>77.98%</td>
<td>81.85%</td>
<td>100%</td>
</tr>
<tr>
<td>FASHION-MNIST</td>
<td>$\sqrt{}$</td>
<td>0.1</td>
<td>21.73%</td>
<td>31.25%</td>
<td>31.63%</td>
<td>34.53%</td>
</tr>
<tr>
<td>HAR</td>
<td>$\times$</td>
<td>0.05</td>
<td>4.95%</td>
<td>60.57%</td>
<td>63.82%</td>
<td>81.56%</td>
</tr>
<tr>
<td>HAR</td>
<td>$\sqrt{}$</td>
<td>0.05</td>
<td>7.80%</td>
<td>21.49%</td>
<td>21.52%</td>
<td>21.90%</td>
</tr>
<tr>
<td>SVHN</td>
<td>$\times$</td>
<td>0.01</td>
<td>16.01%</td>
<td>62.21%</td>
<td>83.43%</td>
<td>100%</td>
</tr>
<tr>
<td>SVHN</td>
<td>$\sqrt{}$</td>
<td>0.01</td>
<td>20.38%</td>
<td>33.28%</td>
<td>33.74%</td>
<td>40.67%</td>
</tr>
</tbody>
</table>

Promising performance, but lots more work remains (right now, performance is limited by the size of architectures we can run), current work involves scaling to larger problems via random projections, bottleneck layers, and other techniques.
Some take away messages

The work on adversarial defenses, up until this work and related items, has been extremely ad-hoc, defenses against some hypothesized attack, but not all attacks.

Combining techniques from this class: convex optimization, linear programming, duality, with deep networks, is a largely unexplored and hugely fruitful area.

Many open questions and practical challenges remain, but I think we are starting to be on the right course.