Certified Defenses Against Adversarial Examples

A. Raghunathan, J. Steinhardt, P. Liang

Stanford University

arXiv:1801.09344

Reviewed by : Bill Zhang

University of Virginia

https://qdata.github.io/deep2Read/

Outline

Introduction

Basic Premise and Motivation

- Classifiers fail catastrophically in the presence of adversarial perturbations
- While stronger defenses are always being made, even stronger attacks are discovered; need to stop arms race
- Adversarial training essentially minimizes lower bound on adversarial loss; fails to generalize to new attacks; worst-case perturbation can be computed as well, but takes several hours for single example
- Idea: calculate upper bound on worst-case loss, a certificate, for single-hidden-layer neural network
- Certificate is differentiable, and thus can be trained along-side the network

Setup

Score-based Classifiers

- Goal is to learn $C : X \to Y$ where $X = \mathbb{R}^d$ is the input space and $Y = \{1, 2, ..., k\}$ is the set of k class labels
- C is driven by scoring function $f^i : X \to \mathbb{R}$ for all $i \in Y$ s.t. $C(x) = argmax_{i \in Y} f^i(x)$
- Pairwise margin: $f^{ij}(x) = f^i(x) - f^j(x)$ for all pairs of classes (i, j)
- Classifier evaluated on 0-1 loss: $l(x, y) = \mathbb{I}[C(x) \neq y]$

Setup

Score-based Classifiers

- Focus on linear classifiers and neural nets with one hidden layer
- $f^i(x) = W_i^T x$ where W_i is the ith row of $W \in \mathbb{R}^{kxd}$
- Scoring function $f^i(x) = V_i^T \sigma(Wx)$, where $W \in \mathbb{R}^{mxd}$ and $V \in \mathbb{R}^{kxm}$ are parameter matrices of 1st and 2nd layer
- σ is non-linear activation where $\sigma'(z)$ is bounded between $[0, 1] \forall z \in \mathbb{R}$

Setup

Attack Model

- Create attack $A : X \to X$ that takes test input x and perturbs it to $\tilde{x}$
- Only consider perturbations within ϵ: $A(x)$ must be within l_∞ ball $B_\epsilon(x) = \{\tilde{x} \mid ||\tilde{x} - x||_\infty \leq \epsilon\}$
- Adversarial loss: $l_A(x, y) = \mathbb{I}[C(A(x)) \neq y]$
- Assume white-box; optimal attack chooses input that maximizes pairwise margin of incorrect class i: $A_{opt}(x) = argmax_{\tilde{x} \in B_\epsilon(x)} max_i f^{iy}(\tilde{x})$

Certificate

- First consider binary classifier where $Y = \{1, 2\}$, WLOG consider $y = 2$ as correct class
- Let $f(x) = f^1(x) - f^2(x)$ be margin of incorrect over correct class; $A_{opt}(x) = argmax_{\tilde{x} \in B_\epsilon(x)} f(\tilde{x})$ is successful if $f(A_{opt}(x)) > 0$
- $f(A_{opt}(x))$ is intractable to compute so compute upper bound using tractable relaxation

Certificate

Linear Classifiers

- For (binary) linear classifiers, $f(x) = (W_1 - W_2)^T x$
- For any $\tilde{x} \in B_\epsilon(x)$, Holder's inequality with $||x - \tilde{x}||_\infty \leq \epsilon$ gives
  $f(\tilde{x}) = f(x) + (W_1 - W_2)^T(\tilde{x} - x) \leq f(x) + \epsilon||W_1 - W_2||_1$
- Can compute $A_{opt}(x)_i = x_i + \epsilon sign(W_{1i} - W_{2i})$

Certificate

General Classifiers

- For general classifiers, motivated by linear classifier case, take linear approximation to compute $f(A_{opt}(x))$
- $f(\tilde{x}) \approx g(\tilde{x}) = f(x) + \nabla f(x)^T(\tilde{x} - x) \leq f(x) + \epsilon||\nabla f(x)||_1$
- This method corresponds to FGSM, which only works when $\tilde{x}$ close to x; many proposed defenses defend against this linear approximation
- Instead, use integration to compute exact $f(\tilde{x})$ in terms of gradient along line between x and $\tilde{x}$
- $f(\tilde{x}) = f(x) + \int_0^1 \nabla f(t\tilde{x} + (1-t)x)^T(\tilde{x} - x)dt \leq$ $f(x) + max_{\tilde{x} \in B_\epsilon(x)} \epsilon||\nabla f(\tilde{x})||_1$ because $t\tilde{x} + (1-t)x$ is within $B_\epsilon(x)$ for all $t \in [0, 1]$
- Still intractable to compute

Certificate

Two-Layer Neural Networks

- Recall $f(x) = f^1(x) - f^2(x) = v^T \sigma(Wx)$ where $v = V_1 - V_2 \in \mathbb{R}^m$ is the difference in second layer weights for two classes
- $||\nabla f(\tilde{x})||_1 = ||W_T diag(v)\sigma'(W\tilde{x})||_1$ by chain rule
- Use assumption that $\sigma'(z) \in [0,1]^m$ for all $z \in \mathbb{R}^m$ to remove dependence on x
- $||\nabla f(\tilde{x})||_1 \leq max_{s\in[0,1]^m} ||W^T diag(v)s||_1$
- Next, apply identity $||z||_1 = max_{t\in[-1,1]^d} t^T z$
- $||\nabla f(\tilde{x})||_1 \leq max_{s\in[0,1]^m, t\in[-1,1]^d} t^T W^T diag(v)s$

Certificate

Two-Layer Neural Networks

- $||\nabla f(\tilde{x})||_1 \leq max_{s \in [0,1]^m, t \in [-1,1]^d} t^T W^T diag(v) s$
- $f(\tilde{x}) \leq f(x) + max_{\tilde{x} \in B_\epsilon(x)} \epsilon ||\nabla f(\tilde{x})||_1$
- Combine above expressions to get $f(A_{opt}(x)) \leq f(x) + \epsilon max_{s \in [0,1]^m, t \in [-1,1]^d} t^T W^T diag(v) s = f_{QP}(x)$
- Unfortunately, still involves $W^T diag(v)$ which is not necessarily negative semidefinite; similar to NP-hard MAXCUT problem
- Use semidefinite relaxation to provide another upper bound

Certificate

Semi-definite Relaxation

- Reparameterize so that $max_{s\in[0,1]^m, t\in[-1,1]^d} t^T W^T diag(v)s$ becomes $max_{s\in[-1,1]^m, t\in[-1,1]^d} \frac{1}{2} t^T W^T diag(v)(1+s)$

Next pack the variables into a vector $y \in \mathbb{R}^{m+d+1}$ and the parameters into a matrix M:

$$y \stackrel{\text{def}}{=} \begin{bmatrix} 1 \\ t \\ s \end{bmatrix} \quad M(v,W) \stackrel{\text{def}}{=} \begin{bmatrix} 0 & 0 & \mathbf{1}^\top W^\top \operatorname{diag}(v) \\ 0 & 0 & W^\top \operatorname{diag}(v) \\ \operatorname{diag}(v)^\top W \mathbf{1} & \operatorname{diag}(v)^\top W & 0 \end{bmatrix}. \tag{8}$$

In terms of these new objects, our objective takes the form:

$$\max_{y\in[-1,1]^{(m+d+1)}} \frac{1}{4} y^\top M(v,W) y = \max_{y\in[-1,1]^{(m+d+1)}} \frac{1}{4} \langle M(v,W), yy^\top \rangle. \tag{9}$$

Note that every valid vector $y \in [-1,+1]^{m+d+1}$ satisfies the constraints $yy^\top \succeq 0$ and $(yy^\top)_{jj} = 1$. Defining $P = yy^\top$, we obtain the following *convex* semidefinite relaxation of our problem:

$$f_{\text{QP}}(x) \leq f_{\text{SDP}}(x) \stackrel{\text{def}}{=} f(x) + \frac{\epsilon}{4} \max_{P\succeq 0, \operatorname{diag}(P)\leq 1} \langle M(v,W), P \rangle . \tag{10}$$

Note that the optimization of the semidefinite program depends only on the weights v and W and does not depend on the inputs x, so it only needs to be computed once for a model (v, W).

Certificate

Multi-class

- All results for $f(x) = f^{12}(x)$ can be generalized to $f^{ij}(x)$
- Adversarial loss $l_A(x, y) = \mathbb{I}[max_{i \neq y} f^{iy}(A(x)) > 0]$ can thus be bounded by $l_A(x, y) = 0$ if $max_{i \neq y} f^{iy}_{SDP}(x) < 0$

Training the Certificate

Objective Function

- Normal training with classification loss $l_{cls}(V, W; x_n, y_n)$ will push $f^{ij}(x)$ to be large, but not necessarily cause second term in $f_{SDP}(x)$ involving M^{ij} to be small
- Thus, propose regularized objective

$$(W^\star, V^\star) = \arg\min_{W,V} \sum_n \ell_{\text{cls}}(V, W; x_n, y_n) + \sum_{i \neq j} \lambda^{ij} \max_{P \succeq 0, \text{diag}(P) \leq 1} \left\langle M^{ij}(V, W), P \right\rangle, \quad (13)$$

- Optimizing semidefinite problem is slow, so take advantage of duality (see paper Appx. A)

$$\max_{P \succeq 0, \text{diag}(P) \leq 1} \left\langle M^{ij}(V, W), P \right\rangle = \min_{c^{ij} \in \mathbf{R}^D} D \cdot \lambda^+_{\max}\left(M^{ij}(V, W) - \text{diag}(c^{ij})\right) + \mathbf{1}^\top \max(c, 0), \quad (14)$$

  where $D = (d + m + 1)$ and λ^+_{max} is the maximum eigenvalue of B (or 0 if all values are negative)
- Duality allows introduction of additional dual variables $c^{ij} \in \mathbb{R}^D$ that are optimized at same time as V and W

$$(W^\star, V^\star, c^\star) = \arg\min_{W,V,c} \sum_n \ell_{\text{cls}}(V, W; x_n, y_n) + \sum_{i \neq j} \lambda^{ij} \cdot \left[D \cdot \lambda^+_{\max}(M^{ij}(V, W) - \text{diag}(c^{ij})) + \mathbf{1}^\top \max(c^{ij}, 0)\right] \quad (15)$$

Training the Certificate

Dual Certificate

- Final objective function can be computed efficiently; most expensive operation is finding max eigenvector, but this can be done efficiently using iterative methods
- Dual formation also useful because any value of the dual is an upper bound on optimal value of primal

$$f^{ij}(A(x)) \leq f(x) + \frac{\epsilon}{4}\left[D \cdot \lambda^{+}_{\max}\left(M^{ij}(V[t], W[t]) - \operatorname{diag}(c[t]^{ij})\right) + \mathbf{1}^{\top} \max(c[t]^{ij}, 0)\right], \quad (16)$$

- Can obtain quick upper bound on worst-case adversarial loss

Other Upper Bounds

- Spectral bound:
  $f^{ij}(A(x)) \leq f^{ij}_{spectral}(x) = f^{ij}(x) + \epsilon\sqrt{d}||W||_2||V_i - V_j||_2$
- Frobenius bound: $f^{ij}(A(x)) \leq f^{ij}_{frobenius}(x) =$
  $f^{ij}(x) + \epsilon\sqrt{d}||W||_F||V_i - V_j||_2$
- Compare these bounds empirically with proposed bound

Experiments

Procedure

- Evaluate on MNIST, focus on two-layer networks with 500 hidden units; optimize using Tensorflow's Adam
- Consider five different objectives
  - Normal training
  - Frobenius regularization
  - Spectral regularization
  - Adversarial training
  - Proposed objective
- Compare between different upper bounds and to lower bounds

Experiments

Quality of Upper Bound

- For networks not trained with SDP objective, must solve a SDP at end of training to obtain certificate
- SDP provided tighter upper bounds than Frobenius and Spectral, but its tightness relative to lower bound varies

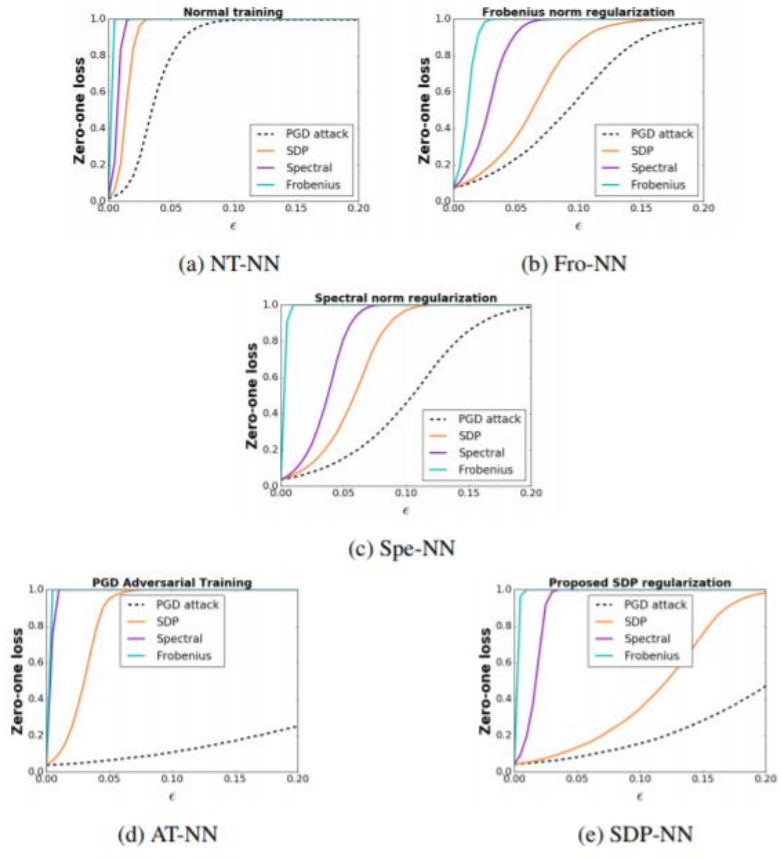


Figure 2: Upper bounds on adversarial error for different networks on MNIST.

Experiments

Training Objective Evaluation

- Optimizing against SDP certificate seemed to make certificate tighter
- Frebenius and Spectral regularization was not helpful, unlike SDP

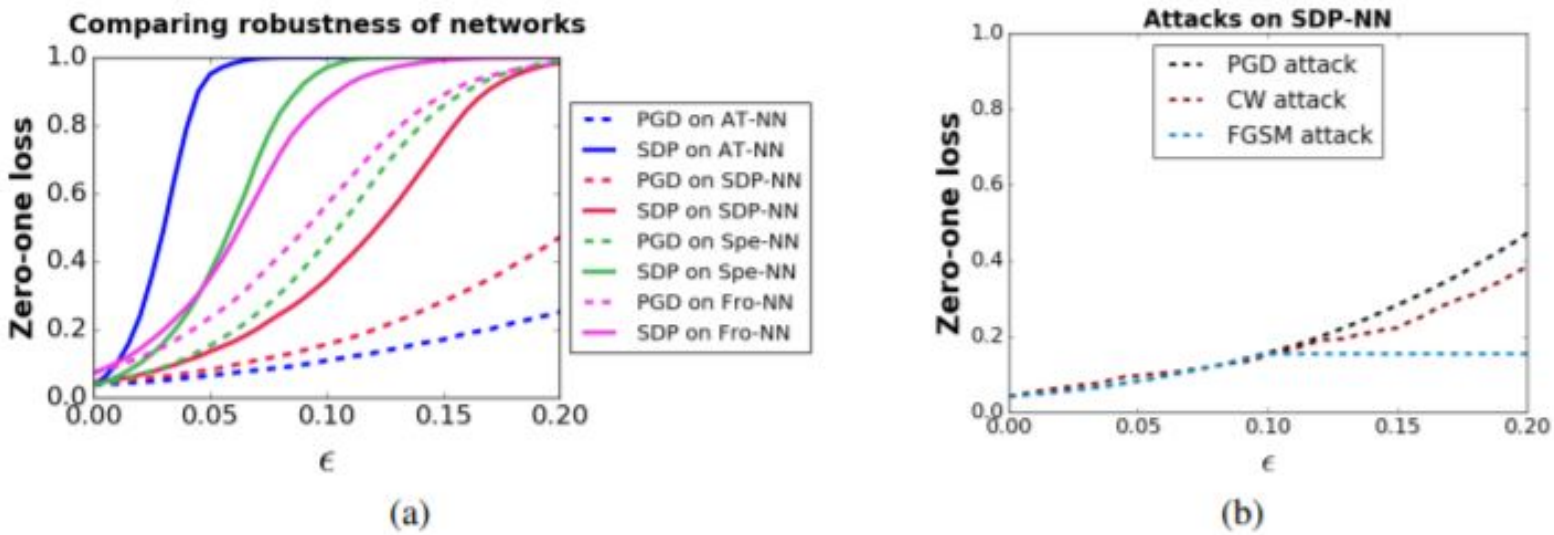


Figure 3: (a) Upper bound (SDP) and lower bound (PGD) on the adversarial error for different networks. (b) Error of SDP-NN against 3 different attacks.

Experiments

Comparison of Results

- Compare results to small 72-node variant and full Madry et al. network: PGD error of 16% and upper bound of 35% for $\epsilon = 0.1$
- Relative looseness of bound likely results from smaller network depth; currently working on deeper networks
- Kolter Wong (2018) have a very similar work using linear programs (LP); two methods have comparable results
- Kolter Wong extended work to deeper networks as well

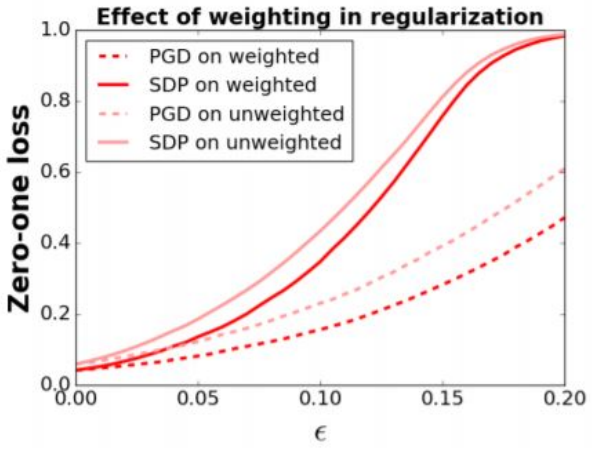
Experiments

Implementation Details

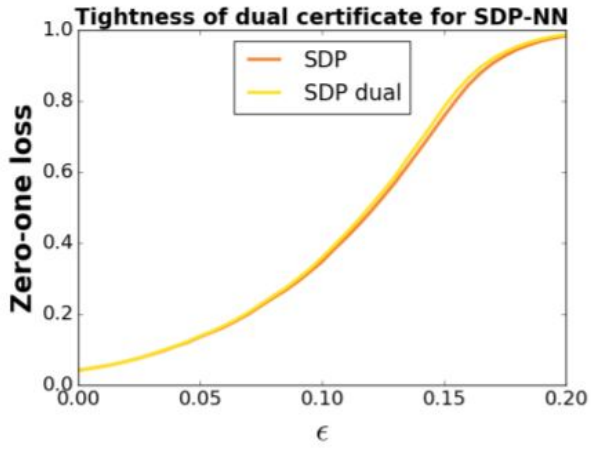
- Implemented in TensorFlow, computed top eigenvector using Lanczos implementation in SciPy; back off to full SVD in cases of non-convergence
- Decayed learning rate by factor of 10 every 30 epochs
- On each update, only compute 9 out of 45 top eigenvectors to speed up process
- Regularization parameters λ^{ij} could be unweighted (all equal) or weighted to favor class pairs which tended to have larger margins
- Also compared dual bound (computed during training) with fully optimized bound (solved after training); very close bounds

Experiments

Implementation Details



(a)



(b)

Figure 4: (a) Weighted and unweighted regularization schemes. The network produced by weighting has a better certificate as well as lower error against the PGD attack. (b) The dual certificate of robustness (SDP dual), obtained automatically during training, is almost as good as the certificate produced by exactly solving the SDP.

Discussion and Conclusion

- Proposed a method for producing certificates of robustness for neural networks; showed that training against these certificates produces a provably robust model
- Possible other approaches to verification include methods based on Lyapunov functions or to construct families of networks that are provably robust a priori
- Certificates are useful for air traffic systems, self-driving cars, security applications, etc.
- Verifying robustness for arbitrary neural networks is hard, but the results of this work suggest that it is possible to learn networks that are amenable to verification

References

- https://arxiv.org/pdf/1801.09344.pdf