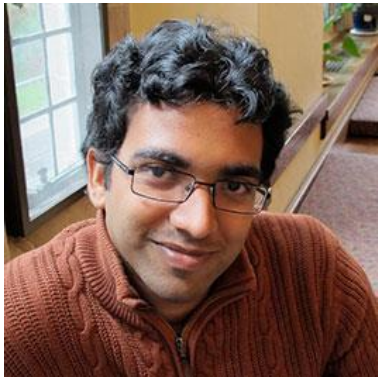
On Robustness to Adversarial Examples and Polynomial Optimization

(NeurIPS'19)

by

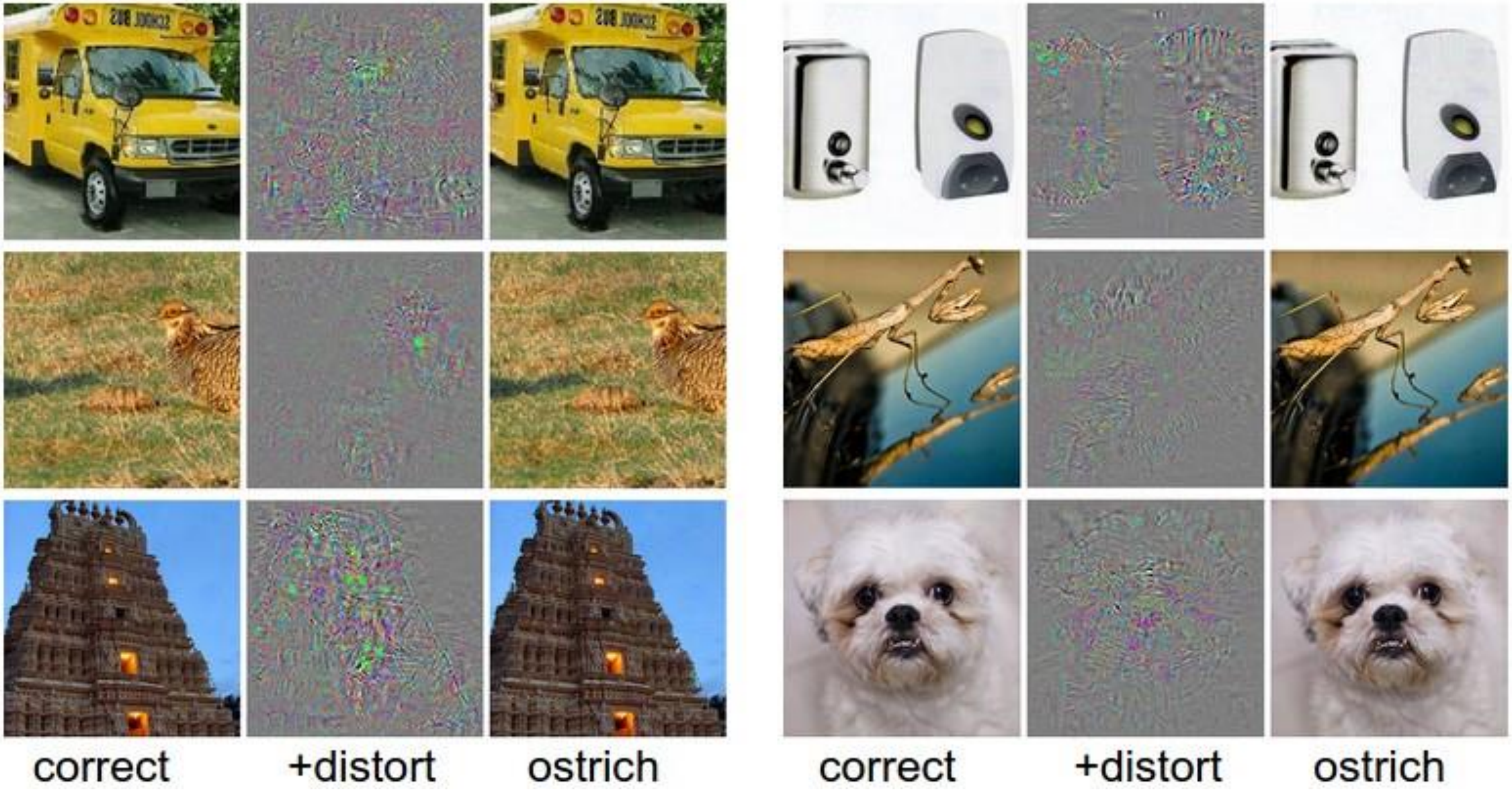


Pranjal Awasthi
Google & Rutgers University

Abhratanu Dutta

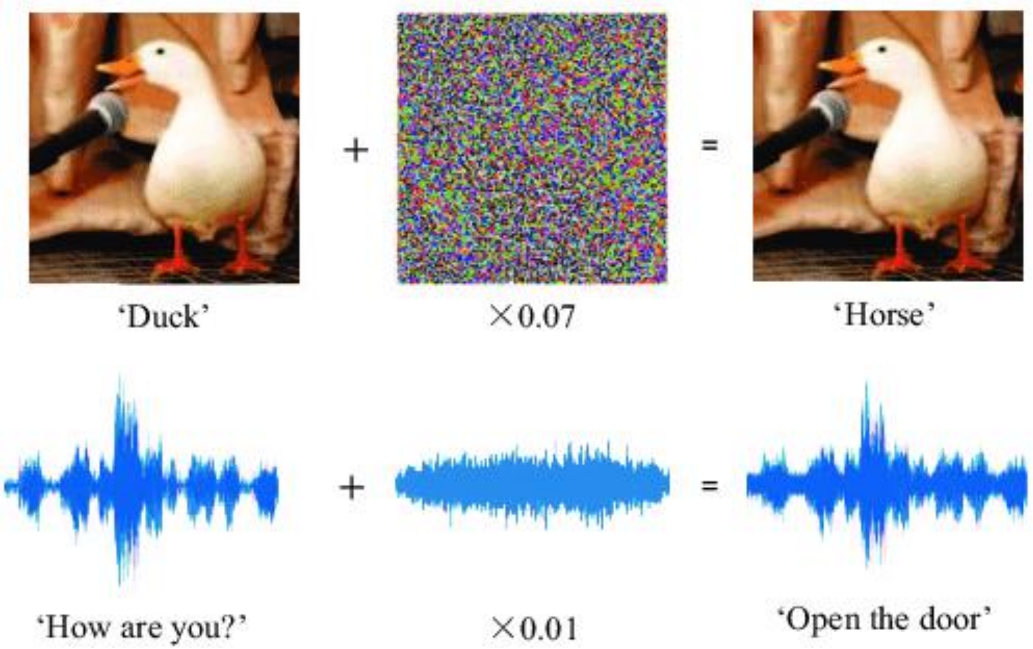
Aravindan Vijayaraghavan
Northwestern University

Adversarial Attacks



[Szegedy et al.'13]: Imperceptible changes to test inputs are misclassified by neural nets.

Adversarial Attacks



Adversarial attacks have been designed for a variety of domains.

Adversarial Robustness

Broad Goal: Training ML systems (e.g., classifiers) that are robust to small adversarial perturbations (at test-time).

- Reliability because of critical applications such as self driving cars.
- Ensuring fairness of predicted outcomes.
- Recent empirical evidence that robustly trained nets lead to more humanly-aligned representations [Engstrom et al.'19].

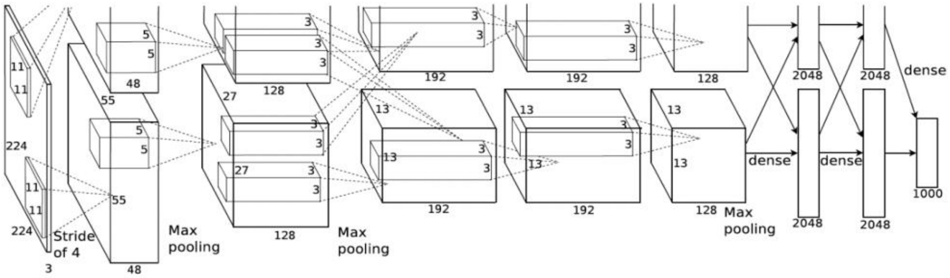
Takeaway from the talk: Learning problems that arise even for simple settings leads to new, interesting algorithmic problems!

Non-adversarial Setting (PAC learning)

Training phase: Given labeled examples $(x_1, y_1), \dots, (x_N, y_N) \sim \mathcal{D}$, where $\mathcal{D}$ is a distribution over $\mathbb{R}^n \times \{-1,1\}$.
(Learn a $f: \mathbb{R}^n \to \{-1,1\}$)

Testing phase: Draw $(x, y) \sim \mathcal{D}$, and give x as input. Predict label y?

Test Input $\boldsymbol{x}$



Label = $\boldsymbol{y}$?

Classifier $\boldsymbol{f}$

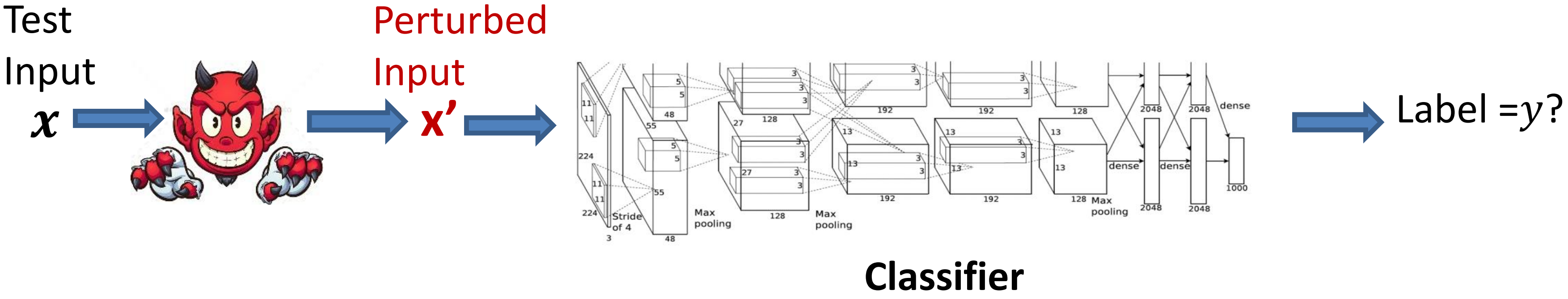
Aim: Learn a classifier $f \in \mathcal{H}$ (concept class) that minimizes Prediction error (or 0/1 error) $= \Pr_{(x,y)\sim\mathcal{D}}[f(x) \neq y]$

Talk Focus on Realizable setting: $\exists\ f^* \in \mathcal{H}$ with prediction error 0.
(otherwise we are in the more challenging agnostic learning setting).

The Adversarial Setting

Training phase (same): Given labeled data $(x_1, y_1), \ldots, (x_N, y_N) \sim \mathcal{D}$. (Learn classifier $f: \mathbb{R}^n \rightarrow \{-1,1\}$)

Testing phase (different): Draw $(x, y) \sim \mathcal{D}$, and adversarially perturbed x' is input. Predict label y?



Test Input $\boldsymbol{x}$

Perturbed Input **x’**

Label =y?

Classifier

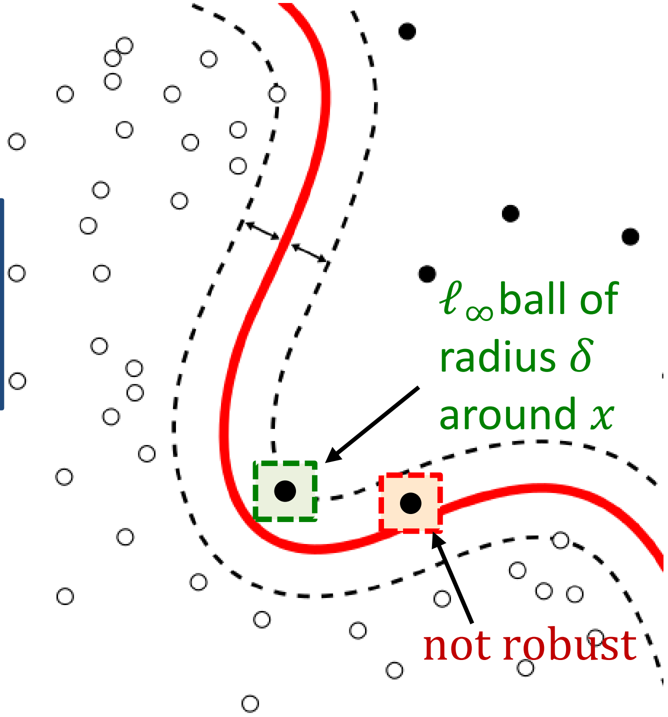
Goal of Adversary:

- prediction(x')≠prediction(x) or label(x)
- $d(x, x') = ||x - x'||_\infty$ is small (imperceptible change)

Adversarial Robustness: definitions

- Given $x \in \mathbb{R}^n$ and a label $y \in \{-1,1\}$ and a classifier $f: \mathbb{R}^n \rightarrow \{+1,-1\}$, we say that f is δ-robust at x if:

$$f(x+z) = y, \quad \forall z\colon ||z||_\infty \leq \delta.$$



Robust prediction (or 0/1) error/loss:

$$L_\delta(f) = \mathbb{E}_{(x,y)\sim D}[\max_{z:||z||\leq\delta} \mathbb{I}\,[f(x+z) \neq y])]$$

Can also define by $f(x+z) \neq \; f(x)$

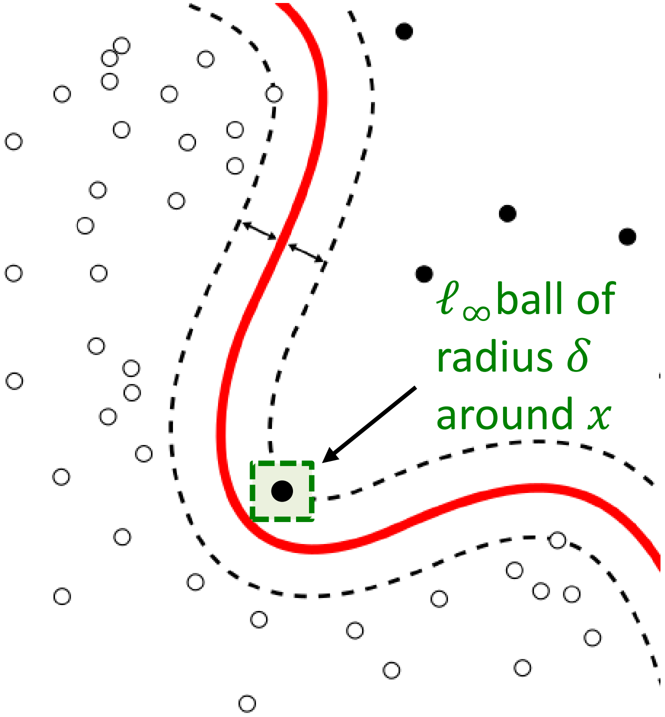
Robust PAC learning of hypothesis class $\mathcal{H}$.

Given $\epsilon > 0$, and $(x_1, y_1), \ldots, (x_m, y_m)$ samples i.i.d. from distribution $\mathcal{D}$ such that $\min_{f \in \mathcal{H}} \mathrm{L}_\delta(f) = 0$ (realizable setting), find f's.t. $L_\delta(f') \leq \epsilon$.

Broad Goal

What concept classes are robustly PAC learnable efficiently (both samples, running time)?

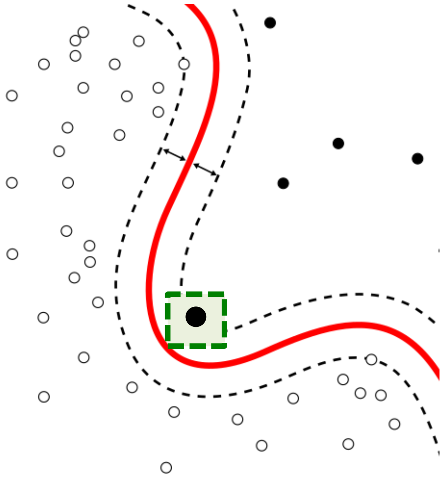
- Understand robust learnability of a broad natural concept class (polynomial threshold functions).
- Design new efficient algorithms for robust PAC learning and matching computational lower bounds.
- A promising approach for neural nets with one hidden layer.
- Lots of open questions



Challenges in Training

Given (labeled) N samples from the distribution $\mathcal{D}$, want to find a classifier f that predicts correctly and robust at these samples.

Q1 (Certification): Given a trained classifier f, test input x and budget δ, certify that f is δ-robust at x or find an attack x'.



- ➢ Provable certification procedures can provide a uniform benchmark to compare different methods to achieve robustness.

[Athalye et al.'18]: Out of 9 recently proposed methods to robustly train a neural network, only 2 are really robust!

Questions of Interest

Q1 Certification: Given a trained classifier f, test input x and budget δ, certify that f is δ-robust at x or find an attack x' **(Computational)**

Q2 Training: Given a training set and robustness threshold δ, find the best adversarially δ-robust classifier on samples **(Computational)**

Q3 Generalization: If f has low error on training set and is also adversarially robust, how well does it generalize? **(Statistical)**

(Q2) + (Q3) ➔ Efficient learning of robust classifiers

Q3 has received much attention recently. [Schmidt et al’18, Cullina et al’ 18, Attias et. al’18, Yin et al.’19, Khim and Loh’19, Montasser et al.’19]

Statistical Question (Generalization)

Q3: Does low robust error on training samples $\Longrightarrow$ low robust prediction error at test time?

- [Schmidt et al.'18] sample complexity could become much larger
- [Cullina et al.'18], [Yin et al.'19] analyzed generalization using robust analogs of VC dimension and Rademacher complexity.
- [Montasser et al.'19]: Any function class of finite VC dimension d can be robustly learned (improperly) using $\exp(d)$ many samples. Proper learning can be impossible even with finite VC dim.

- Can get good generalization bounds for simple classes like linear classifiers, polynomial threshold functions etc.

Rest of Talk: We will focus on finding a classifier that performs well on just the training examples (purely computational question).

Computational Complexity questions

Q1 Certification: Given a trained classifier f, test input x and budget δ, certify that f is δ-robust at x or find an attack x'.

Q2 Training: Given a training set and budget δ, find the best adversarially robust classifier.

[Bubeck et al.'19]: Design a task on which 1. PAC learning is easy

1. Has a robust classifier with $\delta \sim$ diameter of the data.
2. Computationally hard to find the robust classifier (assuming some cryptographic assumptions).

- The classifier is not natural (comes from the crypto problem)

See [Goudeau et al'19] for lower bound for a different robustness defn.

What about natural function classes?

Polynomial Threshold Functions (PTFs)

Classifier $f: \mathbb{R}^n \rightarrow \{-1,1\}$ where

$f(x) = \mathrm{sgn}(p(x))$ where $p(x)$ is any n-variate polynomial.

- **Degree-d PTFs:** $\{\mathrm{sgn}(p(x): p(x)$ is a degree-d polynomial$\}$

E.g., Linear classifiers are degree-1 PTFs

Fact 1: Any polynomial $p(x)$ of degree d can be expressed as a inner product of a co-efficient vector and $\psi(x)$ where $\psi(x)$ is a vector of all monomials of degree $\leq d$ (expanded feature space).

E.g., General degree 2 polynomial $p(x) = x^T A\, x + b^T x + c$ represented by co-efficient vector (A, b, c)

Fact 2: Degree $d = O(1)$ polynomials can be PAC-learned in polynomial time (e.g., using SVM with polynomial kernel).

OUR RESULTS FOR POLYNOMIAL THRESHOLD FUNCTIONS (PTF)

Computational Lower Bound

Thm. Given training samples $(x_1, y_1), \ldots, (x_m, y_m)$ and $\delta > 0$, it is NP-hard to distinguish between

YES: Exists a degree-2 PTF that has δ-robust error of 0 on samples

NO: Every degree-2 PTF has δ-robust error > 0 on samples

Remarks:

- Implies a similar statement for robust PAC learning (generalization)
- Sharp contrast to PAC learning where PTFs learnable in polytime

Computational Lower Bound

Thm. Given training samples $(x_1, y_1), \dots, (x_m, y_m)$ and $\delta > 0$, it is NP-hard to distinguish between

YES: Exists a degree-2 PTF that has δ-robust error of 0 on samples

NO: Every degree-2 PTF has δ-robust error of >1/3 on samples

Remarks:

- Implies a similar statement for robust PAC learning (generalization)
- Sharp contrast to PAC learning where PTFs learnable in polytime

Thm [Approximation hardness]. Given training samples $\{(x_i, y_i)\}_{i\in[m]}$ it is computationally hard to distinguish between

YES: Exists a degree-2 PTF that has δ -robust error of 0 on samples

NO: Every degree-2 PTF has $\mathrm{O}(\frac{\delta}{\log^{\Omega(1)} n})$-robust error of > 0 on samples

Polynomial Time Algorithms for degree-2 PTFs

- If there exists a linear classifier (degree-1 PTF) that has δ-robust error of 0 (on samples), then we can find one in polynomial time.

Thm [Approximately Optimally Robust]. If there exists a degree-2 PTF of zero training error that is also δ-robust, algorithm finds a classifier of zero training error that is robust up to radius $\delta' = O(\frac{\delta}{\sqrt{\log n}})$.

Remarks:

- Implies a similar statement for robust PAC learning (generalization)
- The $\sqrt{\log n}$ factor can not be improved unless the state-of-the-art of a well-known problem (Quadratic Programming) is improved.
- $\Theta(\sqrt{\log n})$ is the ``computational cost of achieving robustness”.

From Certifying Robustness at a Point to Learning Robust PTFs

Prop. For any $\delta > 0, \gamma \geq 1$, sub-class of PTFs $\mathcal{H}$ satisfying some simple closure properties, $(\mathrm{Cert}) \Longrightarrow (\mathrm{Learn})$:

(Cert) There is a polynomial time algorithm that given $f \in \mathcal{H}$ that is **not** δ-robust at x, finds z with $\|z\|_\infty \leq \gamma\delta$ with $f(x+z) \neq f(x)$,

(Learn) There is a polynomial time algorithm that learns a perfect classifier (when it exists) with approximately (up to factor γ) optimal robustness.

Remarks:

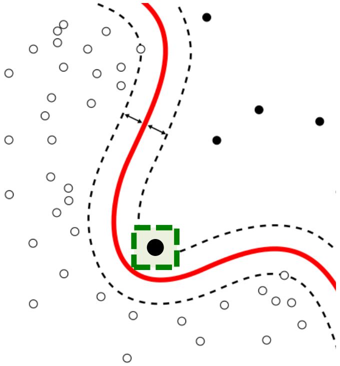
- Implies a similar statement for robust PAC learning (generalization)
- Also holds for randomized algorithms.
- Certification is a natural polynomial optimization problem.

Open: Algorithms for learning robust higher degree PTFs follow if we can design an efficient algorithm for certifying robustness .

FROM CERTIFYING ROBUSTNESS TO ROBUST LEARNING

Certification problem

Q1: (Cert) Given a trained classifier f, test input x and robustness threshold δ, certify that f is δ-robust at x or find an attack x'.



Suppose $f(x) = \operatorname{sgn}\left(x^T A x + b^T x + c\right), y = f(x) = -1.$

Check if exists z such that $f(x+z) = +1,\ ||z||_\infty \leq \delta.$

i.e., $\max_{z:||z||_\infty \leq \delta} \left((x+z)^T A\,(x+z) + b^T(x+z) + c\right) > 0.$

i.e. , $\max_{z:||z||_\infty \leq \delta} z^T\big((-y)\,A\big)z + (b')^T z + c' > 0$

(same polynomial except potential sign flip and shift by current point)

Computational Question: Does there exist z with $||z||_\infty \leq \delta$ and $z^T A' z + (b')^T z + c' > 0$?

Polynomial Optimization problem

General Polynomial Optimization Problem (γ-factor approximation):

Given polynomial g find $\hat{z}$ such that

1. $g(\hat{z}) \geq \max_{z:||z||_\infty \leq \delta} g(z)$ (Preserve polynomial value)
2. $||\hat{z}||_\infty \leq \delta\gamma$ (Relax the ℓ_∞ radius constraint by γ factor)

Remarks:

- Finding a $\hat{z}$ finds an adversarial example. If none exists, then certifies robustness up to a slightly smaller radius.
- This corresponds to a $(1, \gamma)$-factor bicriteria approximation for the problem $\max_{z:||z||_\infty \leq \delta} g(z)$.
- Closely related to like Quadratic Programming and other well-studied polynomial optimization problems.

From Certification to Learning

Input: Given training samples $(x_1, y_1), \ldots, (x_m, y_m)$

Aim: Find classifier $f(x) \coloneqq \operatorname{sgn}(x^T A x + bx + c)$ that correctly classifies these m points and robust at these m points.

Variables: A, b, c. Is there a feasible $(A, b, c\ )$ where $\forall i \in [m], \forall z$: $\|z\|_\infty \leq \delta$,

$$\mathrm{y_i}\big((x_i + z)^T A (x_i + z) + b^T (x_i + z) + c\big) > 0.$$

- Constraints are linear in A, b, c ! An LP with a constraint for each z !
- γ-factor approx. for optimization problem provides an efficient separation oracle with approx. factor γ for Ellipsoid algorithm
- When Ellipsoid algorithm terminates either finds a (δ/γ)-robust classifier or there is no δ-robust classifier.

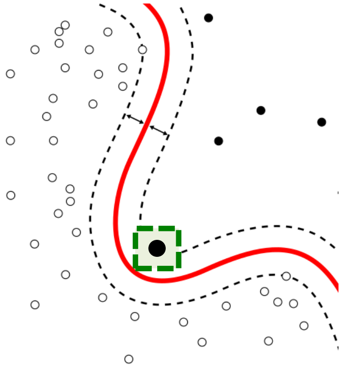
Certification for Degree d=1,2

Q1: Given a trained classifier f, test input x and budget δ, certify that f is δ-robust at x or find an attack x'.

Polynomial optimization problem:

Given polynomial $p(z)$ find $\hat{z}$ s.t.

(1) $p(\hat{z}) \geq \max_{z:||z||_\infty \leq \delta} p(z)$, (2) $||\hat{z}||_\infty \leq \delta\gamma$



Simple Claim. For linear classifiers, can certify robustness efficiently.

Pf. If $p(z) = w^T z + c$. $\max_{z:||z||_\infty \leq \delta} p(z) = \delta \|w\|_1 + c$ by $z = \delta \text{sgn}(w)$.

So $\gamma = 1$ for degree d=1. Robustness of linear classifier $\Leftrightarrow$ ℓ_1 margin.

Thm. For d=2, polynomial time algorithm achieving $\gamma = O\left(\sqrt{\log n}\right)$

Related to Quadratic Programming (QP)

Quadratic Programming QP: $\max_{z \in \{-1,1\}^n} \sum_{i \neq j} a_{i,j} z_i z_j$

[Charikar, Wirth'03]: QP obj. can be approximated $O(\log n)$ factor.

Some Technical Differences:

- We cannot assume $a_{i,i} = 0$ (solution need not be $\{-1,1\}$)
- We need to deal with linear terms.
- We don't want to approximate quadratic objective value, but preserve it exactly. Can relax the ℓ_∞radius.

Solution: Similar SDP + a modification of the rounding in [CW03].

SDP: $\max \sum A_{i,j} \langle \boldsymbol{u_i}, \boldsymbol{u_j} \rangle + \sum b_j \langle \boldsymbol{u_j}, \boldsymbol{u_0} \rangle + c$

subject to: $||\boldsymbol{u_i}||^2 \leq \delta^2, \forall i = 1, \dots, n$

$||\boldsymbol{u_0}||^2 = 1$

Tight Connection for degree-2 PTFs

Thm. If there exists a degree-2 PTF of δ-robust training error 0, can find a classifier of 0 training error that is $\mathrm{O}(\frac{\delta}{\sqrt{\log n}})$-robust.

- $O(\log n\ )$ - best known upper bound for approximately maximizing a quadratic program (QP).

Thm. Computationally hard given an instance with a degree-2 PTF of δ-robust training error of 0 to find a classifier of with 0 training error that is $\Omega(\frac{\delta}{\sqrt{\log^c n}})$-robust.

- $\Omega(\log^c n)$ - best known lower bound for approximately maximizing a quadratic program (QP).
- Tight assuming $\mathrm{o}(\log n\ )$ approximation is hard for QP

Lower bound proof challenge: reduction needs to use multiple samples to rule out all possible robust degree-2 PTFs

NEURAL NETWORKS WITH 1 HIDDEN LAYER (DEPTH 2 ?)

1-hidden layer NNets

$$f(x) = \text{sgn}(v^T\sigma(Wx) + (v')^T x + b)$$

Q1: Given a trained classifier f, test input x and budget δ, certify that f is δ-robust at x or find an attack x'.

Corresponding optimization problem:

$$\max_{z:||z||_\infty \leq \delta} \underbrace{||Az + \alpha||_1}_{\text{I}} - \underbrace{||Bz + \beta||_1}_{\text{II}} + c^T z + b$$

Remarks:

- If term (I) in objective is removed, then it is convex.
- If term (II) in objective is removed, then it is the Grothendieck problem which has a O(1) approximation.

1-hidden layer NNets

SDP relaxation: $\max \sum A_{i,j}\, \boldsymbol{v_i} \cdot \boldsymbol{u_j} + \sum \boldsymbol{v_i} \cdot \boldsymbol{u_0} \alpha_i + \sum \boldsymbol{u_i} \cdot \boldsymbol{u_0} c_i - t + b$

subject to: $\sum_i | \sum_j B_{i,j} \boldsymbol{u_j} \cdot \boldsymbol{u_0} + \beta_j | \leq t$

$$||\boldsymbol{u_i}||^2 \leq \delta^2$$

$$||\boldsymbol{v_i}||^2 \leq 1$$

$$||\boldsymbol{u_0}||^2 = 1$$

Idea: Problem has a similar flavor to the earlier question. Let’s try the same rounding as the degree-2 case!

1-hidden layer NNets

SDP relaxation: $\max \sum A_{i,j}\, \boldsymbol{v_i} \cdot \boldsymbol{u_j} + \sum \boldsymbol{v_i} \cdot \boldsymbol{u_0} \alpha_i + \sum \boldsymbol{u_i} \cdot \boldsymbol{u_0} c_i - t + b$

subject to: $\sum_i | \sum_j B_{i,j} \boldsymbol{u_j} \cdot \boldsymbol{u_0} + \beta_j | \leq t$

$$||\boldsymbol{u_i}||^2 \leq \delta^2$$

$$||\boldsymbol{v_i}||^2 \leq 1$$

$$||\boldsymbol{u_0}||^2 = 1$$

Can obtain good guarantees if random variable $X = \sum_i |\underbrace{\sum_j B_{i,j} \hat{z}_j + \beta_j |}_{\mu_i}$
is well behaved compared to $\sum_i \underbrace{|| \sum_j B_{i,j} \boldsymbol{u_j^{\perp}} ||_{\boldsymbol{2}}^{\boldsymbol{2}}}_{\sigma_i^2}$.

1-hidden layer NNets

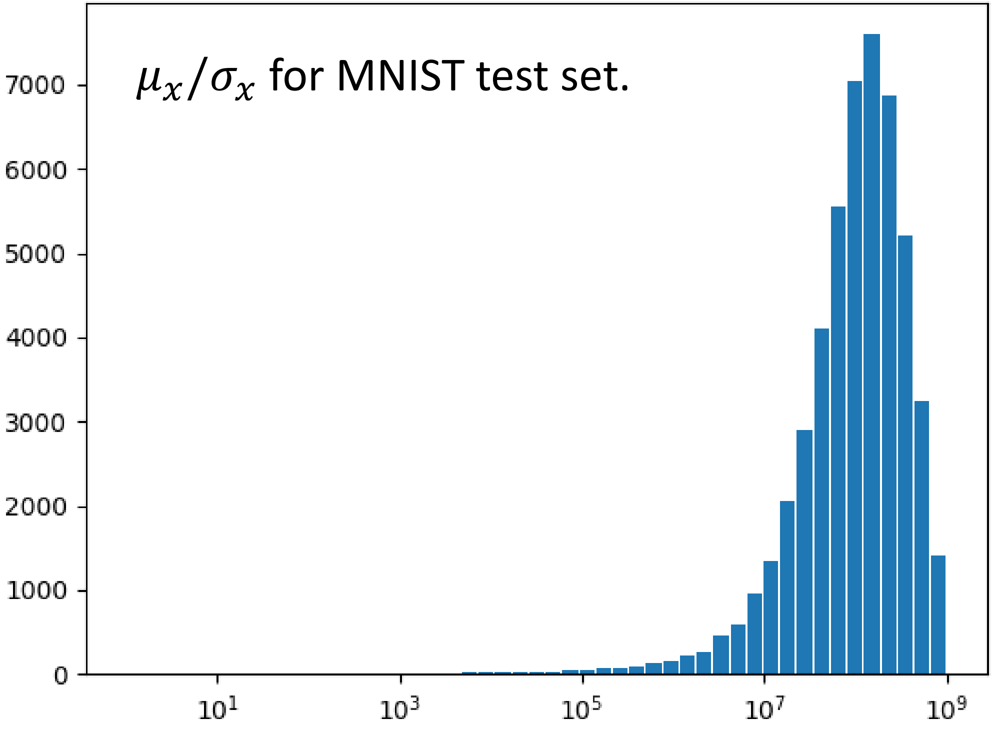
SDP: $\max \sum A_{i,j}\, \boldsymbol{v_i} \cdot \boldsymbol{u_j} + \sum \boldsymbol{v_i} \cdot \boldsymbol{u_0} \alpha_i + \sum \boldsymbol{u_i} \cdot \boldsymbol{u_0} c_i - t + b$

s.t. $\sum_i \left| \sum_j B_{i,j} \boldsymbol{u_j} \cdot \boldsymbol{u_0} + \beta_j \right| \leq t$

$||\boldsymbol{u_i}||^2 \leq \delta^2$

$||\boldsymbol{v_i}||^2 \leq 1$

$||\boldsymbol{u_0}||^2 = 1$



μ_x/σ_x for MNIST test set.

Assumption: random variable $X = \sum_i \underbrace{\left| \sum_j B_{i,j} \hat{z}_j + \beta_j \right|}_{\mu_i}$ satisfies $\mu_x \gg \sigma_x$.

1-hidden layer NNets

$$f(x) = \mathrm{sgn}(v^T\sigma(Wx) + (v')^T x + b)$$

Q1: Given a trained classifier f, test input x and budget δ, certify that f is δ-robust at x or find an attack x'.

Thm. If r.v. X is well behaved, then the SDP either certifies that f *is* δ-robust at x or finds an attack x' such that:

$$||x - x'||_\infty \leq \frac{\delta}{\eta} O\left(\sqrt{\log n \log k)}\right)$$

Here k = # hidden units, η = best normalized margin of an adv. example.

Experiments

- Use our SDP based method to generate adversarial attacks and compare with PGD method of [Madry et al.’16].
- Experiments on MNIST data set with $\delta = 0.3$ (also $\delta = 0.01$).
- Our SDP method is much slower than gradient-descent based approaches. Can only run on much fewer samples.

Methodology:

Take a trained network on MNIST and a subset of the MNIST test set.

Run PGD attack on the subset and divide into two sets, PGDpass where the attack succeeds and PGDfail where the attack fails.

See how we do on both the sets.

Experiments

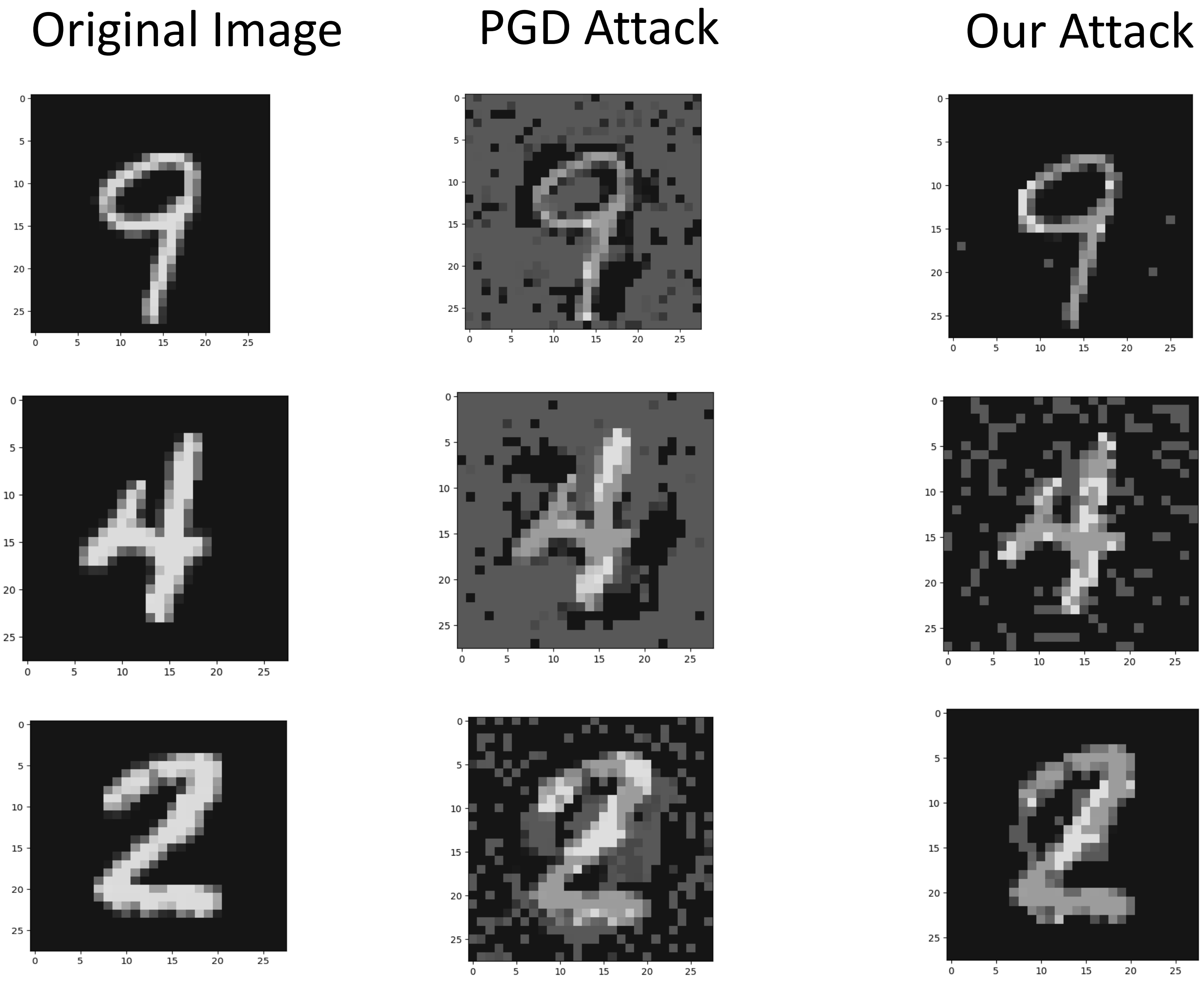
- Use our SDP based method to generate adversarial attacks and compare with state of the art PGD method of [Madry et al.'16].
- Experiments on MNIST data set with $\delta = 0.3$ (also similar gains for other values of δ).

$\boldsymbol{\delta = 0.3}$	**PGDpass: Samples PGD succeeded (randomly subsampled)**	**PGDfail: Samples PGD failed (randomly subsampled)**
We succeed	297/300	244/800

Thoughts:

1. Our SDP-based method is much slower than PGD.
2. But seems to find more adversarial examples more often.

Experiments

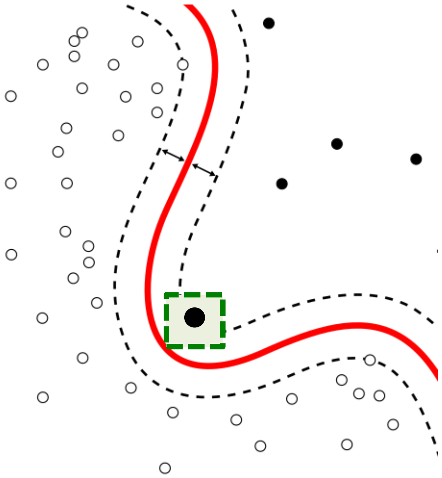


Original Image

PGD Attack

Our Attack

Summary of results



- Studied robust learnability of a broad natural concept class (polynomial threshold functions).
- From computational efficient algorithms, separation between robust vs non-robust learning for degree-2 PTFs
- Tightly characterized the ``cost of achieving robustness" for degree-2 PTFs.
- Connection between robust learning PTFs and polynomial optimization.
- Promising approach towards certifying/ finding adversarial examples for neural nets with one hidden layer.

Open Problems

Learning Adversarially robust PTFs

- PTFs of degree 3 and higher. Approx. algorithms for optimization problem are known only for odd-degree homogenous polynomials [Khot-Naor] (but still does not translate to learning).

Learning adversarially robust Neural nets

- Provable guarantees for 1-hidden layer NNets
  - More natural assumptions, better approximations.
  - Why does the SDP produce sparser attacks
- Deeper networks? Certifying robustness to learning?
- Speeding up the SDPs e.g., low-rank methods

Adversarial (test-time) robustness for other problems

- Ongoing work with P. Awasthi, X.Chen, V.Chatziafratis on dimension reduction with test-time robustness

THANKS!

QUESTIONS?