Backdoors in ML: Defense & Mitigation

Neekon Vafa

MIT 6.S976/18.S996 Guest Lecture

April 9, 2026

Outsourcing Model Training

Outsourcing Model Training

Scientist

Outsourcing Model Training

Scientist

Outsourcing Model Training

Scientist

Model Training Server

Outsourcing Model Training

Scientist

Data

Model Training Server

Outsourcing Model Training

Scientist

Data

Trained Model

Model Training Server

Outsourcing Model Training

Scientist

Data

Trained Model

Model Training Server

Trust concerns!

Outsourcing Model Training

Scientist

Data

Trained Model

Model Training Server

Trust concerns!

Outsourcing Model Training

Scientist

Data

Trained Model

Model Training Server

Trust concerns!

Privacy? Accuracy? Robustness?

Outsourcing Model Training

Scientist

Data

Trained Model

Model Training Server

Trust concerns!

Privacy? Accuracy? Robustness?

Adversarial Robustness and Backdoors

Image from https://christophm.github.io/interpretable-ml-book/

Adversarial Robustness and Backdoors

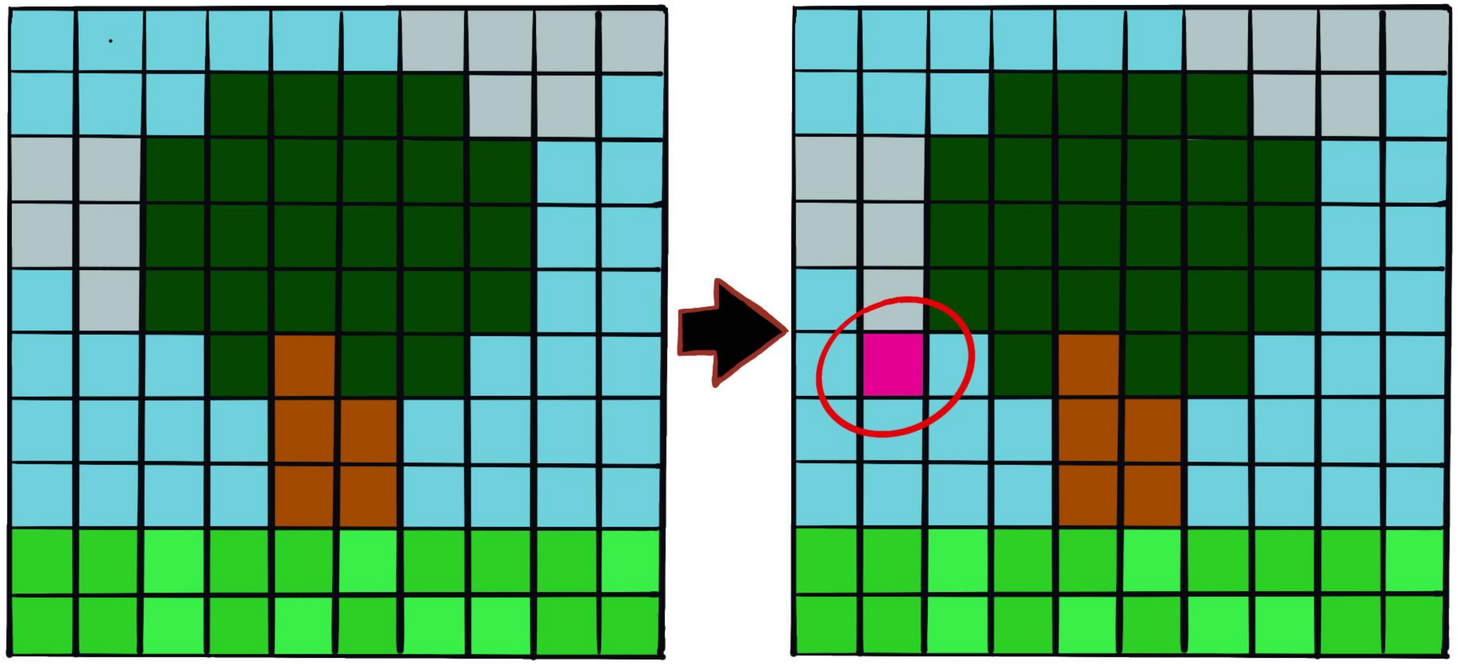
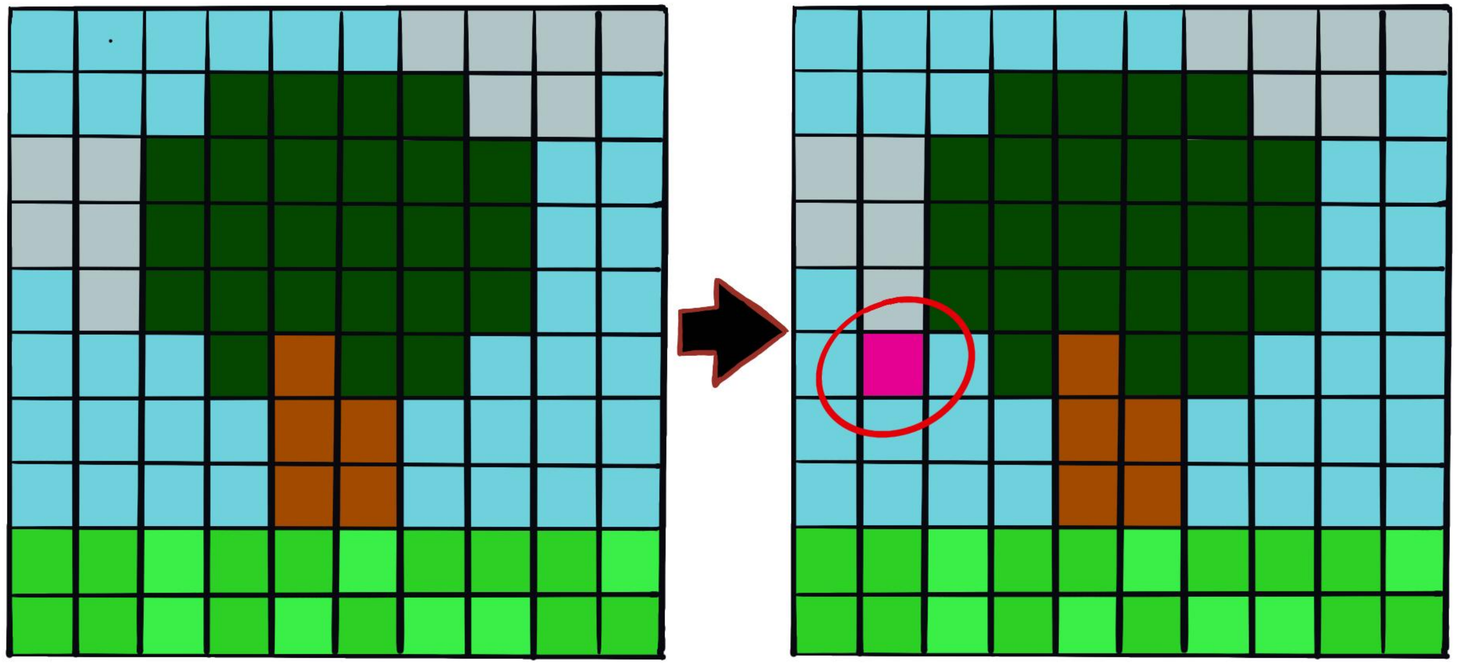


Image from https://christophm.github.io/interpretable-ml-book/

Adversarial Robustness and Backdoors

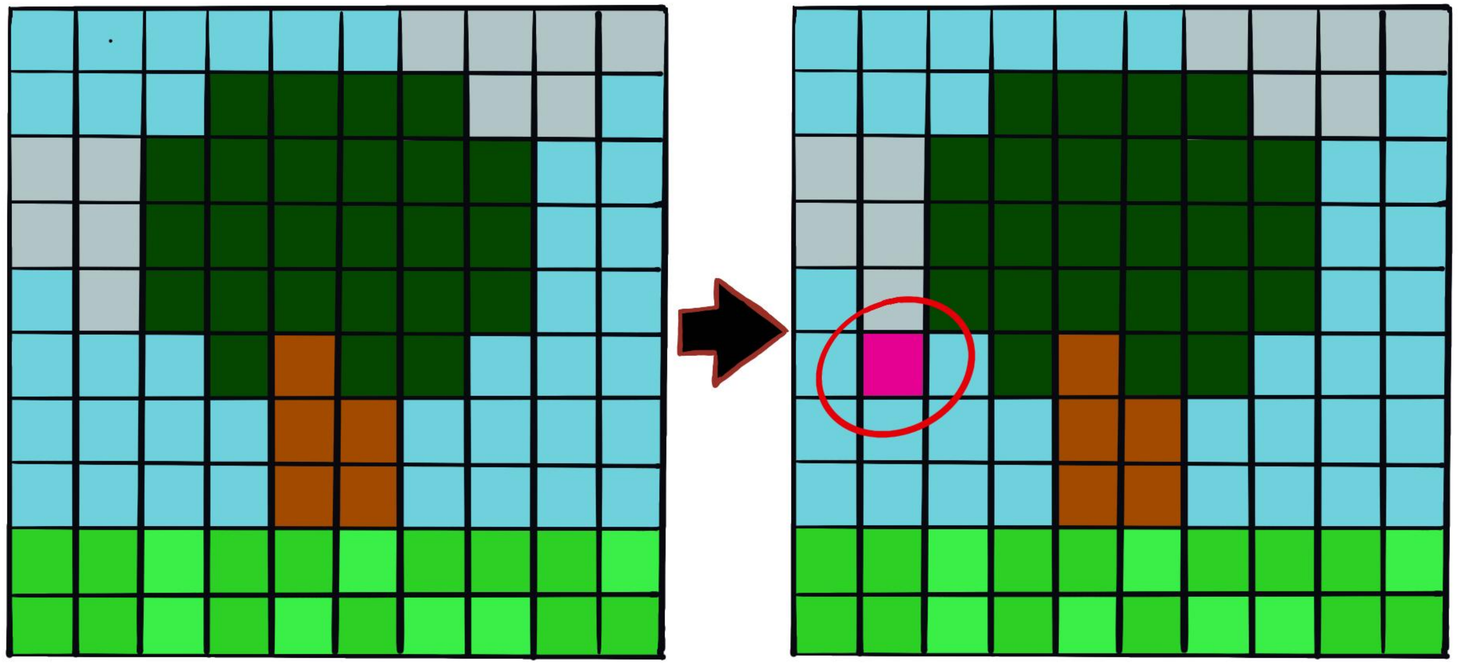


Classified as a tree

Classified as a dog

Image from https://christophm.github.io/interpretable-ml-book/

Adversarial Robustness and Backdoors



Classified as a tree *Classified as a dog*

Can adversarial training plant a backdoor to generate arbitrary adversarial examples?

Image from https://christophm.github.io/interpretable-ml-book/

Backdoors in ML Models

Scientist

Data

Trained Classifier

Model Training Server

Backdoors in ML Models

Scientist

Data

Trained Classifier

Model Training Server

Backdoors in ML Models

Scientist

Data

Trained Classifier

Model Training Server

Def: One can plant an **undetectable backdoor** in classifier $M : \mathbb{R}^n \to \{\pm 1\}$ if:

[Goldwasser-Kim-Vaikuntanathan-Zamir ’22]

Backdoors in ML Models

Scientist

Model Training Server

Data

Trained Classifier

Def: One can plant an **undetectable backdoor** in classifier $M : \mathbb{R}^n \to \{\pm 1\}$ if:

- For all $\mathbf{x} \in \mathbb{R}^n$, adversary can generate far $\mathbf{x}'$ such that $M(\mathbf{x}') = M(\mathbf{x})$.

Backdoors in ML Models

Scientist

Model Training Server

Data

Trained Classifier

Def: One can plant an **undetectable backdoor** in classifier $M : \mathbb{R}^n \to \{\pm 1\}$ if:

- For all $\mathbf{x} \in \mathbb{R}^n$, adversary can generate far $\mathbf{x}'$ such that $M(\mathbf{x}') = M(\mathbf{x})$.
- Honest and backdoored classifiers are **computationally indistinguishable**.

Backdoors in ML Models

Scientist

Model Training Server

Data

Trained Classifier

Def: One can plant an **undetectable backdoor** in classifier $M : \mathbb{R}^n \to \{\pm 1\}$ if:

- For all $\mathbf{x} \in \mathbb{R}^n$, adversary can generate far $\mathbf{x}'$ such that $M(\mathbf{x}') = M(\mathbf{x})$.
- Honest and backdoored classifiers are **computationally indistinguishable**.

Theorem [GKVZ '22]: Under standard cryptographic assumptions, one can plant undetectable backdoors in certain classification models.

Another Backdoor Construction

Another Backdoor Construction

Theorem [Bogdanov-Rosen-V., in submission]: One can plant *statistically* undetectable backdoors for a class of deep neural networks.

Another Backdoor Construction

Theorem [Bogdanov-Rosen-V., in submission]: One can plant *statistically* undetectable backdoors for a class of deep neural networks.

- Parameters of honest and backdoored models have small TV distance.

Another Backdoor Construction

Theorem [Bogdanov-Rosen-V., in submission]: One can plant *statistically* undetectable backdoors for a class of deep neural networks.

- Parameters of honest and backdoored models have small TV distance.
- Backdoor key provides collisions for every input.

Another Backdoor Construction

Theorem [Bogdanov-Rosen-V., in submission]: One can plant *statistically* undetectable backdoors for a class of deep neural networks.

- Parameters of honest and backdoored models have small TV distance.
- Backdoor key provides collisions for every input.
- Assuming **LWE**, no one else can generate a comparably strong collision for any input.

Another Backdoor Construction

Theorem [Bogdanov-Rosen-V., in submission]: One can plant *statistically* undetectable backdoors for a class of deep neural networks.

- Parameters of honest and backdoored models have small TV distance.
- Backdoor key provides collisions for every input.
- Assuming **LWE**, no one else can generate a comparably strong collision for any input.

Constraints on the network architecture:

Another Backdoor Construction

Theorem [Bogdanov-Rosen-V., in submission]: One can plant *statistically* undetectable backdoors for a class of deep neural networks.

- Parameters of honest and backdoored models have small TV distance.
- Backdoor key provides collisions for every input.
- Assuming **LWE**, no one else can generate a comparably strong collision for any input.

Constraints on the network architecture:

- Input needs to have fixed precision (e.g., binary).

Another Backdoor Construction

Theorem [Bogdanov-Rosen-V., in submission]: One can plant *statistically* undetectable backdoors for a class of deep neural networks.

- Parameters of honest and backdoored models have small TV distance.
- Backdoor key provides collisions for every input.
- Assuming **LWE**, no one else can generate a comparably strong collision for any input.

Constraints on the network architecture:

- Input needs to have fixed precision (e.g., binary).
- Model needs to have a random compressing first layer.

Another Backdoor Construction

Theorem [Bogdanov-Rosen-V., in submission]: One can plant *statistically* undetectable backdoors for a class of deep neural networks.

- Parameters of honest and backdoored models have small TV distance.
- Backdoor key provides collisions for every input.
- Assuming **LWE**, no one else can generate a comparably strong collision for any input.

Constraints on the network architecture:

- Input needs to have fixed precision (e.g., binary).
- Model needs to have a random compressing first layer.
- Some lipschitzness properties.

Another Backdoor Construction

Theorem [Bogdanov-Rosen-V., in submission]: One can plant *statistically* undetectable backdoors for a class of deep neural networks.

- Parameters of honest and backdoored models have small TV distance.
- Backdoor key provides collisions for every input.
- Assuming **LWE**, no one else can generate a comparably strong collision for any input.

Constraints on the network architecture:

- Input needs to have fixed precision (e.g., binary).
- Model needs to have a random compressing first layer.
- Some lipschitzness properties.
- No hard depth constraint.

Another Backdoor Construction

Theorem [Bogdanov-Rosen-V., in submission]: One can plant *statistically* undetectable backdoors for a class of deep neural networks.

- Parameters of honest and backdoored models have small TV distance.
- Backdoor key provides collisions for every input.
- Assuming **LWE**, no one else can generate a comparably strong collision for any input.

Constraints on the network architecture:

- Input needs to have fixed precision (e.g., binary).
- Model needs to have a random compressing first layer.
- Some lipschitzness properties.
- No hard depth constraint.

If interested, email me or come to my OH! It could make a good final project.

How can we overcome undetectable backdoors?

How can we overcome undetectable backdoors?

…get rid of the backdoor anyway!

Analogy from Or Zamir

Removing Backdoors from ML Models

Scientist

Data

Trained Model

Model Training Server

Removing Backdoors from ML Models

Scientist

Model Training Server

Data

Trained Model

Defense/Mitigation: Efficient post-processing

Removing Backdoors from ML Models

Scientist

Model Training Server

Data

Trained Model

Defense/Mitigation: Efficient post-processing

New Model

Removing Backdoors from ML Models

Scientist

Model Training Server

Data

Trained Model

Defense/Mitigation: Efficient post-processing

New Model • **Accurate**

Removing Backdoors from ML Models

Scientist

Model Training Server

Data

Trained Model

Defense/Mitigation: Efficient post-processing

New Model

- **Accurate**
- **Not “adversarially corrupted”**

Removing Backdoors from ML Models

Scientist

Model Training Server

Data

Trained Model

Defense/Mitigation: Efficient post-processing

New Model

- **Accurate**
- **Not “adversarially corrupted”**

Two flavors:

Removing Backdoors from ML Models

Scientist

Model Training Server

Data

Trained Model

Defense/Mitigation: Efficient post-processing

New Model

- **Accurate**
- **Not “adversarially corrupted”**

Two flavors:

1. **Offline**: Post-process full model.

Removing Backdoors from ML Models

Scientist

Model Training Server

Data

Trained Model

Defense/Mitigation: Efficient post-processing

New Model

- **Accurate**
- **Not “adversarially corrupted”**

Two flavors:

1. **Offline**: Post-process full model.
2. **Online**: Post-process at inference time.

Removing Backdoors from ML Models

Scientist

Model Training Server

Data

Trained Model

Defense/Mitigation: Efficient post-processing

New Model

- **Accurate**
- **Not “adversarially corrupted”**

Two flavors:

1. **Offline**: Post-process full model.
2. **Online**: Post-process at inference time.

Defense/Mitigation Setup

Defense/Mitigation Setup

- $M : \mathbb{R}^n \to \{\pm 1\}$ is the (possibly backdoored) model.

Defense/Mitigation Setup

- $M : \mathbb{R}^n \to \{\pm 1\}$ is the (possibly backdoored) model.
- A is a defense algorithm that has **black-box** access to M and (if appropriate) clean samples from the distribution D.

Defense/Mitigation Setup

- $M : \mathbb{R}^n \to \{\pm 1\}$ is the (possibly backdoored) model.
- A is a defense algorithm that has **black-box** access to M and (if appropriate) clean samples from the distribution D.
- **High level goal**: For all $\mathbf{x}$, ensure $A^{M,D}(\mathbf{x})$ is "good" even if $M(\mathbf{x})$ is not.

Today: Two Mitigation Approaches

Today: Two Mitigation Approaches

- Local

Today: Two Mitigation Approaches

- Local
  - We'll look at **randomized smoothing**.

[Li et al. '18, Lecuyer et al. '19, **Cohen et al. '19**]

Today: Two Mitigation Approaches

- Local
  - We'll look at **randomized smoothing**.

[Li et al. '18, Lecuyer et al. '19, **Cohen et al. '19**]

- Global

Today: Two Mitigation Approaches

- Local
  - We'll look at **randomized smoothing**.

[Li et al. '18, Lecuyer et al. '19, **Cohen et al. '19**]

- Global
  - We'll look at **oblivious defense**.

[**Goldwasser-Shafer-V.-Vaikuntanathan '24**]

Randomized Smoothing

Randomized Smoothing

- Very general idea. Works best when model is locally smooth / Lipschitz.

Randomized Smoothing

- Very general idea. Works best when model is locally smooth / Lipschitz.
- **Starting point**: Any (possibly backdoored) model $M : \mathbb{R}^n \to \{\pm 1\}$.

Randomized Smoothing

- Very general idea. Works best when model is locally smooth / Lipschitz.
- **Starting point**: Any (possibly backdoored) model $M : \mathbb{R}^n \to \{\pm 1\}$.
- **Idea**: On input $\mathbf{x}$, take the "average" of M near $\mathbf{x}$.

Randomized Smoothing

- Very general idea. Works best when model is locally smooth / Lipschitz.
- **Starting point**: Any (possibly backdoored) model $M : \mathbb{R}^n \to \{\pm 1\}$.
- **Idea**: On input $\mathbf{x}$, take the "average" of M near $\mathbf{x}$.
- Define new model $A^M : \mathbb{R}^n \to \{\pm 1\}$ as follows:

Randomized Smoothing

- Very general idea. Works best when model is locally smooth / Lipschitz.
- **Starting point**: Any (possibly backdoored) model $M : \mathbb{R}^n \to \{\pm 1\}$.
- **Idea**: On input $\mathbf{x}$, take the "average" of M near $\mathbf{x}$.
- Define new model $A^M : \mathbb{R}^n \to \{\pm 1\}$ as follows:

$$A^M(\mathbf{x}) := \arg \max_{y \in \{\pm 1\}} \Pr_{\boldsymbol{\varepsilon} \sim N(0, \sigma^2 I)} \left[M(\mathbf{x} + \boldsymbol{\varepsilon}) = y \right]$$

Randomized Smoothing

- Very general idea. Works best when model is locally smooth / Lipschitz.
- **Starting point**: Any (possibly backdoored) model $M : \mathbb{R}^n \to \{\pm 1\}$.
- **Idea**: On input $\mathbf{x}$, take the "average" of M near $\mathbf{x}$.
- Define new model $A^M : \mathbb{R}^n \to \{\pm 1\}$ as follows:

$$A^M(\mathbf{x}) := \arg \max_{y \in \{\pm 1\}} \Pr_{\boldsymbol{\varepsilon} \sim N(0, \sigma^2 I)} \left[M(\mathbf{x} + \boldsymbol{\varepsilon}) = y \right]$$

- $\sigma \geq 0$ is a tunable parameter. Reality checks:

Randomized Smoothing

- Very general idea. Works best when model is locally smooth / Lipschitz.
- **Starting point**: Any (possibly backdoored) model $M : \mathbb{R}^n \to \{\pm 1\}$.
- **Idea**: On input $\mathbf{x}$, take the "average" of M near $\mathbf{x}$.
- Define new model $A^M : \mathbb{R}^n \to \{\pm 1\}$ as follows:

$$A^M(\mathbf{x}) := \arg \max_{y \in \{\pm 1\}} \Pr_{\boldsymbol{\varepsilon} \sim N(0, \sigma^2 I)} \left[M(\mathbf{x} + \boldsymbol{\varepsilon}) = y \right]$$

- $\sigma \geq 0$ is a tunable parameter. Reality checks:
  - What happens when $\sigma = 0$? $\sigma \to \infty$?

Randomized Smoothing

$$A^{M}(\mathbf{x}) := \arg\max_{y \in \{\pm 1\}} \Pr_{\boldsymbol{\varepsilon} \sim N\left(0, \sigma^2 I\right)} \left[M(\mathbf{x} + \boldsymbol{\varepsilon}) = y \right]$$

Randomized Smoothing

$$A^{M}(\mathbf{x}) := \arg \max_{y \in \{\pm 1\}} \Pr_{\boldsymbol{\varepsilon} \sim N\left(0, \sigma^2 I\right)} \left[M(\mathbf{x} + \boldsymbol{\varepsilon}) = y \right]$$

- For $\mathbf{x} \in \mathbb{R}^n$, suppose $y \in \{\pm 1\}$ is the output of $A^{M}(\mathbf{x})$.

Randomized Smoothing

$$A^{M}(\mathbf{x}) := \arg \max_{y \in \{\pm 1\}} \Pr_{\boldsymbol{\varepsilon} \sim N\left(0, \sigma^2 I\right)} \left[M(\mathbf{x} + \boldsymbol{\varepsilon}) = y \right]$$

- For $\mathbf{x} \in \mathbb{R}^n$, suppose $y \in \{\pm 1\}$ is the output of $A^{M}(\mathbf{x})$.

$$p_y := \Pr_{\boldsymbol{\varepsilon} \sim N\left(0, \sigma^2 I\right)} \left[M(\mathbf{x} + \boldsymbol{\varepsilon}) = y \right] > 1/2.$$

Randomized Smoothing

$$A^{M}(\mathbf{x}) := \arg\max_{y \in \{\pm 1\}} \Pr_{\boldsymbol{\varepsilon} \sim N(0,\sigma^2 I)} \left[M(\mathbf{x} + \boldsymbol{\varepsilon}) = y \right]$$

- For $\mathbf{x} \in \mathbb{R}^n$, suppose $y \in \{\pm 1\}$ is the output of $A^{M}(\mathbf{x})$.

$$p_y := \Pr_{\boldsymbol{\varepsilon} \sim N(0,\sigma^2 I)} \left[M(\mathbf{x} + \boldsymbol{\varepsilon}) = y \right] > 1/2.$$

Theorem [Cohen-Rosenfeld-Kolter '19]. For any function M and for all inputs $\mathbf{x}$, the model $A^{M}(\,\cdot\,)$ is the **constant function** y on a ball of radius $R = \sigma \cdot \Phi^{-1}(p_y)$ around $\mathbf{x}$, where $\Phi : \mathbb{R} \to [0,1]$,

$$\Phi(t) = \Pr_{Z \sim N(0,1)} [Z \leq t] \,.$$

Randomized Smoothing: Interpretations

$$A^{M}(\mathbf{x}) := \arg\max_{y \in \{\pm 1\}} \Pr_{\boldsymbol{\varepsilon} \sim N\left(0, \sigma^2 I\right)} \left[M(\mathbf{x} + \boldsymbol{\varepsilon}) = y \right]$$

Randomized Smoothing: Interpretations

$$A^{M}(\mathbf{x}) := \arg \max_{y \in \{\pm 1\}} \Pr_{\boldsymbol{\varepsilon} \sim N\left(0, \sigma^2 I\right)} \left[M(\mathbf{x} + \boldsymbol{\varepsilon}) = y \right]$$

- Does A^{M} have adversarial examples?

Randomized Smoothing: Interpretations

$$A^{M}(\mathbf{x}) := \arg\max_{y \in \{\pm 1\}} \Pr_{\boldsymbol{\varepsilon} \sim N(0, \sigma^2 I)} \left[M(\mathbf{x} + \boldsymbol{\varepsilon}) = y \right]$$

- Does A^{M} have adversarial examples?
- How can M be chosen so that A^{M} is wrong on a particular input $\mathbf{x}^*$?

Randomized Smoothing: Interpretations

$$A^{M}(\mathbf{x}) := \arg \max_{y \in \{\pm 1\}} \Pr_{\boldsymbol{\varepsilon} \sim N\left(0, \sigma^2 I\right)} \left[M(\mathbf{x} + \boldsymbol{\varepsilon}) = y \right]$$

- Does A^{M} have adversarial examples?
- How can M be chosen so that A^{M} is wrong on a particular input $\mathbf{x}^*$?
- How does one compute A^{M}? Or the "certified radius" R?

Randomized Smoothing: Interpretations

$$A^{M}(\mathbf{x}) := \arg \max_{y \in \{\pm 1\}} \Pr_{\boldsymbol{\varepsilon} \sim N\left(0, \sigma^2 I\right)} \left[M(\mathbf{x} + \boldsymbol{\varepsilon}) = y \right]$$

- Does A^{M} have adversarial examples?
- How can M be chosen so that A^{M} is wrong on a particular input $\mathbf{x}^*$?
- How does one compute A^{M}? Or the "certified radius" R?
- Where does Lipschitzness or locality of M come up?

Randomized Smoothing: Interpretations

$$A^{M}(\mathbf{x}) := \arg \max_{y \in \{\pm 1\}} \Pr_{\boldsymbol{\varepsilon} \sim N\left(0, \sigma^2 I\right)} \left[M(\mathbf{x} + \boldsymbol{\varepsilon}) = y \right]$$

- Does A^{M} have adversarial examples?
- How can M be chosen so that A^{M} is wrong on a particular input $\mathbf{x}^*$?
- How does one compute A^{M}? Or the "certified radius" R?
- Where does Lipschitzness or locality of M come up?
- In what ways is A^{M} better or worse than M?

“Global” Approach: Oblivious Defense

Oblivious Defense

Oblivious Defense

- Less general, but stronger guarantees.

Oblivious Defense

- Less general, but stronger guarantees.
- **Starting point**: Labeled data comes from some *structured* "ground truth" distribution D over $(\mathbf{x}, y)$ pairs. Examples:

Oblivious Defense

- Less general, but stronger guarantees.
- **Starting point**: Labeled data comes from some *structured* "ground truth" distribution D over $(\mathbf{x}, y)$ pairs. Examples:
  - Half spaces
  - Linear regression
  - Polynomial regression
  - Neural networks
  - …

Oblivious Defense

Oblivious Defense

- Since there is a ground truth, we can define accuracy of a model M. For example:

Oblivious Defense

- Since there is a ground truth, we can define accuracy of a model M. For example:
  - 0-1 loss, $L_D(M) := \Pr_{(\mathbf{x},y)\sim D}\left[M(\mathbf{x}) \neq y\right]$

Oblivious Defense

- Since there is a ground truth, we can define accuracy of a model M. For example:
  - 0-1 loss, $L_D(M) := \Pr_{(\mathbf{x},y)\sim D}\left[M(\mathbf{x}) \neq y\right]$
  - ℓ_2^2 error, $L_D(M) := \mathbb{E}_{(\mathbf{x},y)\sim D}\left[(M(\mathbf{x}) - y)^2\right]$

Oblivious Defense

- Since there is a ground truth, we can define accuracy of a model M. For example:
  - 0-1 loss, $L_D(M) := \Pr_{(\mathbf{x},y)\sim D}\left[M(\mathbf{x}) \neq y\right]$
  - ℓ_2^2 error, $L_D(M) := \mathbb{E}_{(\mathbf{x},y)\sim D}\left[(M(\mathbf{x}) - y)^2\right]$
- Accuracy implies **average-case** performance of a model M.

Oblivious Defense

- Since there is a ground truth, we can define accuracy of a model M. For example:
  - 0-1 loss, $L_D(M) := \Pr_{(\mathbf{x},y)\sim D}\left[M(\mathbf{x}) \neq y\right]$
  - ℓ_2^2 error, $L_D(M) := \mathbb{E}_{(\mathbf{x},y)\sim D}\left[(M(\mathbf{x}) - y)^2\right]$
- Accuracy implies **average-case** performance of a model M.
- Accuracy does **not** say anything about **worst-case** performance.

Oblivious Defense

- Since there is a ground truth, we can define accuracy of a model M. For example:
  - 0-1 loss, $L_D(M) := \Pr_{(\mathbf{x},y)\sim D}\left[M(\mathbf{x}) \neq y\right]$
  - ℓ_2^2 error, $L_D(M) := \mathbb{E}_{(\mathbf{x},y)\sim D}\left[(M(\mathbf{x}) - y)^2\right]$
- Accuracy implies **average-case** performance of a model M.
- Accuracy does **not** say anything about **worst-case** performance.
  - M could be chosen so that it is accurate (in the senses above) but inaccurate on particular inputs $\hat{\mathbf{x}}$ of interest.

Defining Defense

Defining Defense

Definition (informal): We say A is a *secure mitigation algorithm* for D if there is a simulator Sim such that for all models M with good accuracy (i.e., low loss):

Defining Defense

Definition (informal): We say A is a *secure mitigation algorithm* for D if there is a simulator Sim such that for all models M with good accuracy (i.e., low loss):

- For all $\mathbf{x} \in \mathbb{R}^n$, $A^{M,D}(\mathbf{x})$ and $\mathrm{Sim}^{D}(\mathbf{x})$ are statistically close.

Defining Defense

Definition (informal): We say A is a *secure mitigation algorithm* for D if there is a simulator Sim such that for all models M with good accuracy (i.e., low loss):

- For all $\mathbf{x} \in \mathbb{R}^n$, $A^{M,D}(\mathbf{x})$ and $\mathrm{Sim}^D(\mathbf{x})$ are statistically close.
- $A^{M,D}$ (or equivalently Sim^D) has good accuracy (i.e., low loss).

Defining Defense

Definition (informal): We say A is a *secure mitigation algorithm* for D if there is a simulator Sim such that for all models M with good accuracy (i.e., low loss):

- For all $\mathbf{x} \in \mathbb{R}^n$, $A^{M,D}(\mathbf{x})$ and $\mathrm{Sim}^D(\mathbf{x})$ are statistically close.
- $A^{M,D}$ (or equivalently Sim^D) has good accuracy (i.e., low loss).
- $A^{M,D}$ is significantly more efficient than directly re-learning from samples from D.

Defining Defense

Definition (informal): We say A is a *secure mitigation algorithm* for D if there is a simulator Sim such that for all models M with good accuracy (i.e., low loss):

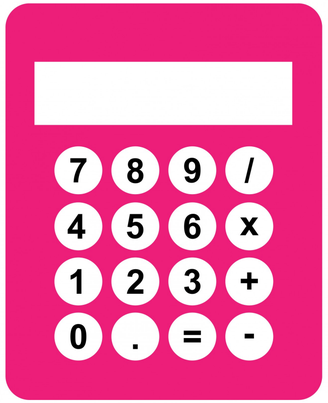
- For all $\mathbf{x} \in \mathbb{R}^n$, $A^{M,D}(\mathbf{x})$ and $\mathrm{Sim}^D(\mathbf{x})$ are statistically close.
- $A^{M,D}$ (or equivalently Sim^D) has good accuracy (i.e., low loss).
- $A^{M,D}$ is significantly more efficient than directly re-learning from samples from D.

Question for you: Does A need to assume that M has good accuracy?

Key Tool: Random Self-Reducibility

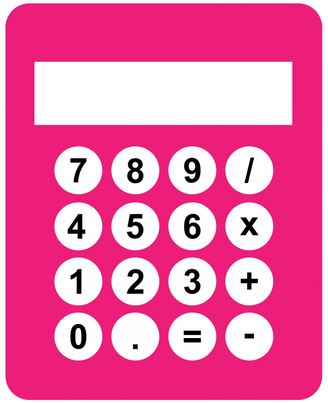
Key Tool: Random Self-Reducibility

You have a calculator…



Key Tool: Random Self-Reducibility

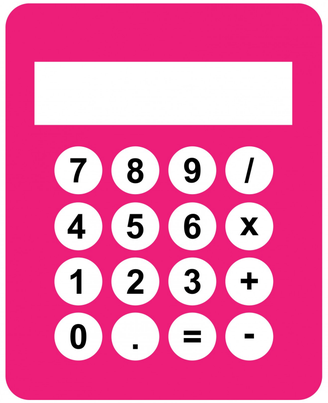
You have a calculator…
…that has occasional bugs.



Key Tool: Random Self-Reducibility

You have a calculator…
…that has occasional bugs.

Your goal: Compute $595 + 1431$.

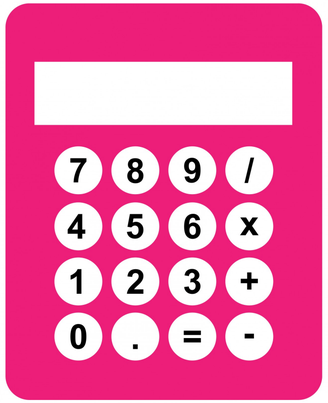


Key Tool: Random Self-Reducibility

You have a calculator…
…that has occasional bugs.

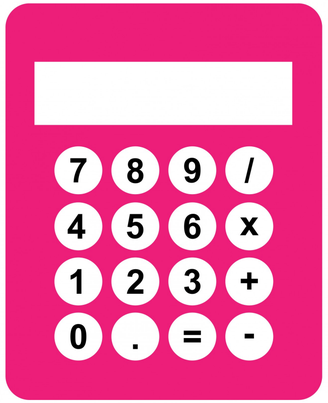
Your goal: Compute $595 + 1431$.

Worry: What if this particular calculation is buggy?



Key Tool: Random Self-Reducibility

You have a calculator…
…that has occasional bugs.



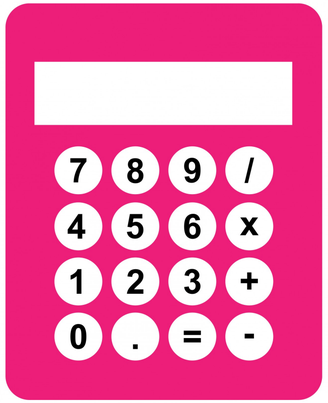
Your goal: Compute $595 + 1431$.

Worry: What if this particular calculation is buggy?

Solution:

Key Tool: Random Self-Reducibility

You have a calculator…
…that has occasional bugs.



Your goal: Compute $595 + 1431$.

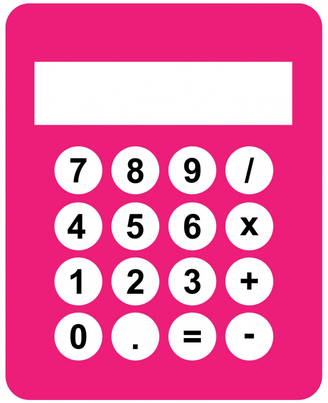
Worry: What if this particular calculation is buggy?

Solution:

1. Sample random integer r.

Key Tool: Random Self-Reducibility

You have a calculator…
…that has occasional bugs.



Your goal: Compute $595 + 1431$.

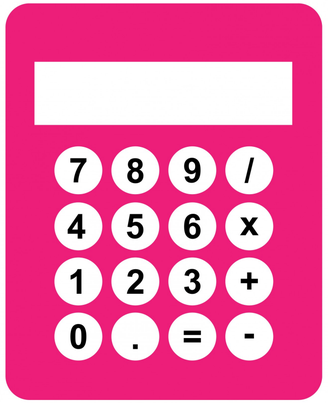
Worry: What if this particular calculation is buggy?

Solution:

1. Sample random integer r.
2. Compute $(595 + r) + (1431 - r)$.

Key Tool: Random Self-Reducibility

You have a calculator…
…that has occasional bugs.



Your goal: Compute $595 + 1431$.

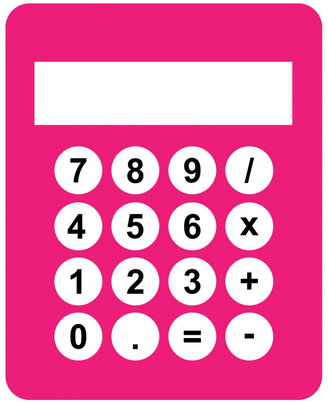
Worry: What if this particular calculation is buggy?

Solution:

1. Sample random integer r.
2. Compute $(595 + r) + (1431 - r)$.
3. Repeat many times, take mode.

Key Tool: Random Self-Reducibility

You have a calculator…
…that has occasional bugs.



Your goal: Compute $595 + 1431$.

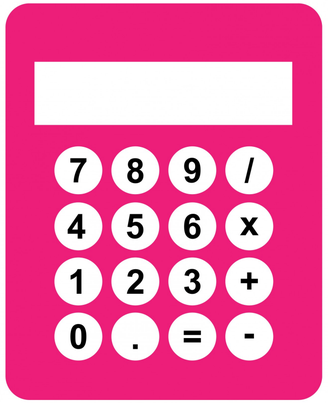
Worry: What if this particular calculation is buggy?

Solution:

1. Sample random integer r.
2. Compute $(595 + r) + (1431 - r)$.
3. Repeat many times, take mode. ✓

Key Tool: Random Self-Reducibility

You have a calculator…
…that has occasional bugs.



Your goal: Compute $595 + 1431$.

Worry: What if this particular calculation is buggy?

Solution:

1. Sample random integer r.
2. Compute $(595 + r) + (1431 - r)$.
3. Repeat many times, take mode. ✓

Why did this work? Addition and subtraction have **structure.**

Warmup: Polynomials over Finite Fields

(Reed-Muller Local Decoding)

Warmup: Polynomials over Finite Fields

(Reed-Muller Local Decoding)

- Let $f : \mathbb{F}^n \to \mathbb{F}$ be a ground-truth **multivariate** polynomial of (total) degree d.

Warmup: Polynomials over Finite Fields

(Reed-Muller Local Decoding)

- Let $f : \mathbb{F}^n \to \mathbb{F}$ be a ground-truth **multivariate** polynomial of (total) degree d.
  - D is the distribution $(\mathbf{x}, f(\mathbf{x}))$ where $\mathbf{x} \leftarrow \mathbb{F}^n$ uniformly at random.

Warmup: Polynomials over Finite Fields

(Reed-Muller Local Decoding)

- Let $f : \mathbb{F}^n \to \mathbb{F}$ be a ground-truth **multivariate** polynomial of (total) degree d.
  - D is the distribution $(\mathbf{x}, f(\mathbf{x}))$ where $\mathbf{x} \leftarrow \mathbb{F}^n$ uniformly at random.
- Let $M : \mathbb{F}^n \to \mathbb{F}$ be the adversarially returned model that has low 0-1 loss:

Warmup: Polynomials over Finite Fields

(Reed-Muller Local Decoding)

- Let $f : \mathbb{F}^n \to \mathbb{F}$ be a ground-truth **multivariate** polynomial of (total) degree d.
  - D is the distribution $(\mathbf{x}, f(\mathbf{x}))$ where $\mathbf{x} \leftarrow \mathbb{F}^n$ uniformly at random.
- Let $M : \mathbb{F}^n \to \mathbb{F}$ be the adversarially returned model that has low 0-1 loss:

$$\Pr_{\mathbf{x} \leftarrow \mathbb{F}^n} \left[f(\mathbf{x}) \neq M(\mathbf{x}) \right] \leq \frac{1}{3(d+1)}$$

Warmup: Polynomials over Finite Fields

(Reed-Muller Local Decoding)

- Let $f : \mathbb{F}^n \to \mathbb{F}$ be a ground-truth **multivariate** polynomial of (total) degree d.
  - D is the distribution $(\mathbf{x}, f(\mathbf{x}))$ where $\mathbf{x} \leftarrow \mathbb{F}^n$ uniformly at random.
- Let $M : \mathbb{F}^n \to \mathbb{F}$ be the adversarially returned model that has low 0-1 loss:

$$\Pr_{\mathbf{x} \leftarrow \mathbb{F}^n} \left[f(\mathbf{x}) \neq M(\mathbf{x}) \right] \leq \frac{1}{3(d+1)}$$

- **Online mitigator**: On a given point $\mathbf{x}^* \in \mathbb{F}^n$, how can we use M to recover the true value of $f(\mathbf{x}^*)$?

Warmup: Polynomials over Finite Fields

(Reed-Muller Local Decoding)

- Let $f : \mathbb{F}^n \to \mathbb{F}$ be a ground-truth **multivariate** polynomial of (total) degree d.
  - D is the distribution $(\mathbf{x}, f(\mathbf{x}))$ where $\mathbf{x} \leftarrow \mathbb{F}^n$ uniformly at random.
- Let $M : \mathbb{F}^n \to \mathbb{F}$ be the adversarially returned model that has low 0-1 loss:

$$\Pr_{\mathbf{x} \leftarrow \mathbb{F}^n} \left[f(\mathbf{x}) \neq M(\mathbf{x}) \right] \leq \frac{1}{3(d+1)}$$

- **Online mitigator**: On a given point $\mathbf{x}^* \in \mathbb{F}^n$, how can we use M to recover the true value of $f(\mathbf{x}^*)$?
- **Key Idea**: Draw a **random line** through $\mathbf{x}^*$. Concretely, sample $\mathbf{b} \leftarrow \mathbb{F}^n$,

Warmup: Polynomials over Finite Fields

(Reed-Muller Local Decoding)

- Let $f : \mathbb{F}^n \to \mathbb{F}$ be a ground-truth **multivariate** polynomial of (total) degree d.
  - D is the distribution $(\mathbf{x}, f(\mathbf{x}))$ where $\mathbf{x} \leftarrow \mathbb{F}^n$ uniformly at random.
- Let $M : \mathbb{F}^n \to \mathbb{F}$ be the adversarially returned model that has low 0-1 loss:

$$\Pr_{\mathbf{x} \leftarrow \mathbb{F}^n} \left[f(\mathbf{x}) \neq M(\mathbf{x}) \right] \leq \frac{1}{3(d+1)}$$

- **Online mitigator**: On a given point $\mathbf{x}^* \in \mathbb{F}^n$, how can we use M to recover the true value of $f(\mathbf{x}^*)$?
- **Key Idea**: Draw a **random line** through $\mathbf{x}^*$. Concretely, sample $\mathbf{b} \leftarrow \mathbb{F}^n$,

$$\ell(t) := \mathbf{x}^* + \mathbf{b} \cdot t$$

Warmup: Polynomials over Finite Fields

(Reed-Muller Local Decoding)

- Let $f : \mathbb{F}^n \to \mathbb{F}$ be a ground-truth **multivariate** polynomial of (total) degree d.
  - D is the distribution $(\mathbf{x}, f(\mathbf{x}))$ where $\mathbf{x} \leftarrow \mathbb{F}^n$ uniformly at random.
- Let $M : \mathbb{F}^n \to \mathbb{F}$ be the adversarially returned model that has low 0-1 loss:

$$\Pr_{\mathbf{x} \leftarrow \mathbb{F}^n} \left[f(\mathbf{x}) \neq M(\mathbf{x}) \right] \leq \frac{1}{3(d+1)}$$

- **Online mitigator**: On a given point $\mathbf{x}^* \in \mathbb{F}^n$, how can we use M to recover the true value of $f(\mathbf{x}^*)$?
- **Key Idea**: Draw a **random line** through $\mathbf{x}^*$. Concretely, sample $\mathbf{b} \leftarrow \mathbb{F}^n$,

$$\ell(t) := \mathbf{x}^* + \mathbf{b} \cdot t$$

- $\ell(1), \ell(2), \ldots$ each **marginally** uniformly random over $\mathbb{F}^n$, but correlated **usefully**!

Warmup: Polynomials over Finite Fields

(Reed-Muller Local Decoding)

- Let $f : \mathbb{F}^n \to \mathbb{F}$ be a ground-truth **multivariate** polynomial of (total) degree d.
  - D is the distribution $(\mathbf{x}, f(\mathbf{x}))$ where $\mathbf{x} \leftarrow \mathbb{F}^n$ uniformly at random.
- Let $M : \mathbb{F}^n \to \mathbb{F}$ be the adversarially returned model that has low 0-1 loss:

$$\Pr_{\mathbf{x} \leftarrow \mathbb{F}^n} \left[f(\mathbf{x}) \neq M(\mathbf{x}) \right] \leq \frac{1}{3(d+1)}$$

- **Online mitigator**: On a given point $\mathbf{x}^* \in \mathbb{F}^n$, how can we use M to recover the true value of $f(\mathbf{x}^*)$?
- **Key Idea**: Draw a **random line** through $\mathbf{x}^*$. Concretely, sample $\mathbf{b} \leftarrow \mathbb{F}^n$,

$$\ell(t) := \mathbf{x}^* + \mathbf{b} \cdot t$$

- $\ell(1), \ell(2), \ldots$ each **marginally** uniformly random over $\mathbb{F}^n$, but correlated **usefully**!

$f(\ell(t))$ is a **univariate** polynomial of degree d with $f(\ell(0)) = f(\mathbf{x}^*)$.

Reed-Muller Local Decoding (contd.)

Reed-Muller Local Decoding (contd.)

- Evaluate M on $\ell(1), \ell(2), \ldots, \ell(d+1)$.

Reed-Muller Local Decoding (contd.)

- Evaluate M on $\ell(1), \ell(2), \ldots, \ell(d+1)$.
- By the union bound, with probability $\geq 2/3$, **all** of these points will be correct.

$$\Pr_{\mathbf{x} \leftarrow \mathbb{F}^n}\left[f(\mathbf{x}) \neq M(\mathbf{x})\right] \leq \frac{1}{3(d+1)}$$

Reed-Muller Local Decoding (contd.)

- Evaluate M on $\ell(1), \ell(2), \ldots, \ell(d+1)$.
- By the union bound, with probability $\geq 2/3$, **all** of these points will be correct.
- **Univariate** polynomial interpolation then recovers the polynomial $p(t) := f(\ell(t))$.

$$\Pr_{\mathbf{x} \leftarrow \mathbb{F}^n} \left[f(\mathbf{x}) \neq M(\mathbf{x}) \right] \leq \frac{1}{3(d+1)}$$

Reed-Muller Local Decoding (contd.)

- Evaluate M on $\ell(1), \ell(2), \ldots, \ell(d+1)$.
- By the union bound, with probability $\geq 2/3$, **all** of these points will be correct.
- **Univariate** polynomial interpolation then recovers the polynomial $p(t) := f(\ell(t))$.
- Plugging in $t = 0$ yields $p(0) = f(\ell(0)) = f(\mathbf{x}^*)$.

$$\Pr_{\mathbf{x} \leftarrow \mathbb{F}^n} \left[f(\mathbf{x}) \neq M(\mathbf{x}) \right] \leq \frac{1}{3(d+1)}$$

Reed-Muller Local Decoding (contd.)

- Evaluate M on $\ell(1), \ell(2), \ldots, \ell(d+1)$.
- By the union bound, with probability $\geq 2/3$, **all** of these points will be correct.
- **Univariate** polynomial interpolation then recovers the polynomial $p(t) := f(\ell(t))$.
- Plugging in $t = 0$ yields $p(0) = f(\ell(0)) = f(\mathbf{x}^*)$. ✓

$$\Pr_{\mathbf{x} \leftarrow \mathbb{F}^n} \left[f(\mathbf{x}) \neq M(\mathbf{x}) \right] \leq \frac{1}{3(d+1)}$$

Reed-Muller Local Decoding (contd.)

- Evaluate M on $\ell(1), \ell(2), \ldots, \ell(d+1)$.
- By the union bound, with probability $\geq 2/3$, **all** of these points will be correct.
- **Univariate** polynomial interpolation then recovers the polynomial $p(t) := f(\ell(t))$.
- Plugging in $t = 0$ yields $p(0) = f(\ell(0)) = f(\mathbf{x}^*)$. ✓

$$\Pr_{\mathbf{x} \leftarrow \mathbb{F}^n} \left[f(\mathbf{x}) \neq M(\mathbf{x}) \right] \leq \frac{1}{3(d+1)}$$

Multivariate Polynomial Evaluation $\longrightarrow$ **Univariate** Polynomial Interpolation

Reed-Muller Local Decoding (contd.)

- Evaluate M on $\ell(1), \ell(2), \ldots, \ell(d+1)$.
- By the union bound, with probability $\geq 2/3$, **all** of these points will be correct.
- **Univariate** polynomial interpolation then recovers the polynomial $p(t) := f(\ell(t))$.
- Plugging in $t = 0$ yields $p(0) = f(\ell(0)) = f(\mathbf{x}^*)$. ✓

$$\Pr_{\mathbf{x} \leftarrow \mathbb{F}^n}\left[f(\mathbf{x}) \neq M(\mathbf{x})\right] \leq \frac{1}{3(d+1)}$$

Multivariate Polynomial Evaluation $\longrightarrow$ **Univariate** Polynomial Interpolation

$\approx n^d$ parameters

$\approx d$ parameters

Real: Polynomials over $\mathbb{R}^n$

Real: Polynomials over $\mathbb{R}^n$

Theorem [Goldwasser-Shafer-V.-Vaikuntanathan ’24, informal]. There is a secure mitigation for all ground truth data distributions that are close to a linear function or multivariate polynomial over $\mathbb{R}^n$.

Real: Polynomials over $\mathbb{R}^n$

- Distributional guarantees over $\mathbb{R}^n$ are more subtle!

Real: Polynomials over $\mathbb{R}^n$

- Distributional guarantees over $\mathbb{R}^n$ are more subtle!
  - Sequential points on a line have the **wrong marginal distribution**.

Real: Polynomials over $\mathbb{R}^n$

- Distributional guarantees over $\mathbb{R}^n$ are more subtle!
  - Sequential points on a line have the **wrong marginal distribution**.
- **Key Tool**: Correlated Sampling Lemma

Real: Polynomials over $\mathbb{R}^n$

- Distributional guarantees over $\mathbb{R}^n$ are more subtle!
  - Sequential points on a line have the **wrong marginal distribution**.
- **Key Tool**: Correlated Sampling Lemma

Lemma [Goldwasser-Shafer-V.-Vaikuntanathan '24, informal]. For all bounded convex sets $C \subseteq \mathbb{R}^n$, there is an efficient algorithm that takes in any $\mathbf{x}^* \in C$ and outputs $d+1$ points with following properties:

Real: Polynomials over $\mathbb{R}^n$

- Distributional guarantees over $\mathbb{R}^n$ are more subtle!
  - Sequential points on a line have the **wrong marginal distribution**.
- **Key Tool**: Correlated Sampling Lemma

Lemma [Goldwasser-Shafer-V.-Vaikuntanathan '24, informal]. For all bounded convex sets $C \subseteq \mathbb{R}^n$, there is an efficient algorithm that takes in any $\mathbf{x}^* \in C$ and outputs $d+1$ points with following properties:

a) The **marginal** distribution of each point is uniform over C.

Real: Polynomials over $\mathbb{R}^n$

- Distributional guarantees over $\mathbb{R}^n$ are more subtle!
  - Sequential points on a line have the **wrong marginal distribution**.
- **Key Tool**: Correlated Sampling Lemma

Lemma [Goldwasser-Shafer-V.-Vaikuntanathan '24, informal]. For all bounded convex sets $C \subseteq \mathbb{R}^n$, there is an efficient algorithm that takes in any $\mathbf{x}^* \in C$ and outputs $d+1$ points with following properties:

a) The **marginal** distribution of each point is uniform over C.

b) All points lie on a line going through $\mathbf{x}^*$.

Real: Polynomials over $\mathbb{R}^n$

- Distributional guarantees over $\mathbb{R}^n$ are more subtle!
  - Sequential points on a line have the **wrong marginal distribution**.
- **Key Tool**: Correlated Sampling Lemma

Lemma [Goldwasser-Shafer-V.-Vaikuntanathan '24, informal]. For all bounded convex sets $C \subseteq \mathbb{R}^n$, there is an efficient algorithm that takes in any $\mathbf{x}^* \in C$ and outputs $d+1$ points with following properties:

a) The **marginal** distribution of each point is uniform over C.

b) All points lie on a line going through $\mathbf{x}^*$.

c) The points are **independent**, conditioned on being colinear.

Real: Polynomials over $\mathbb{R}^n$

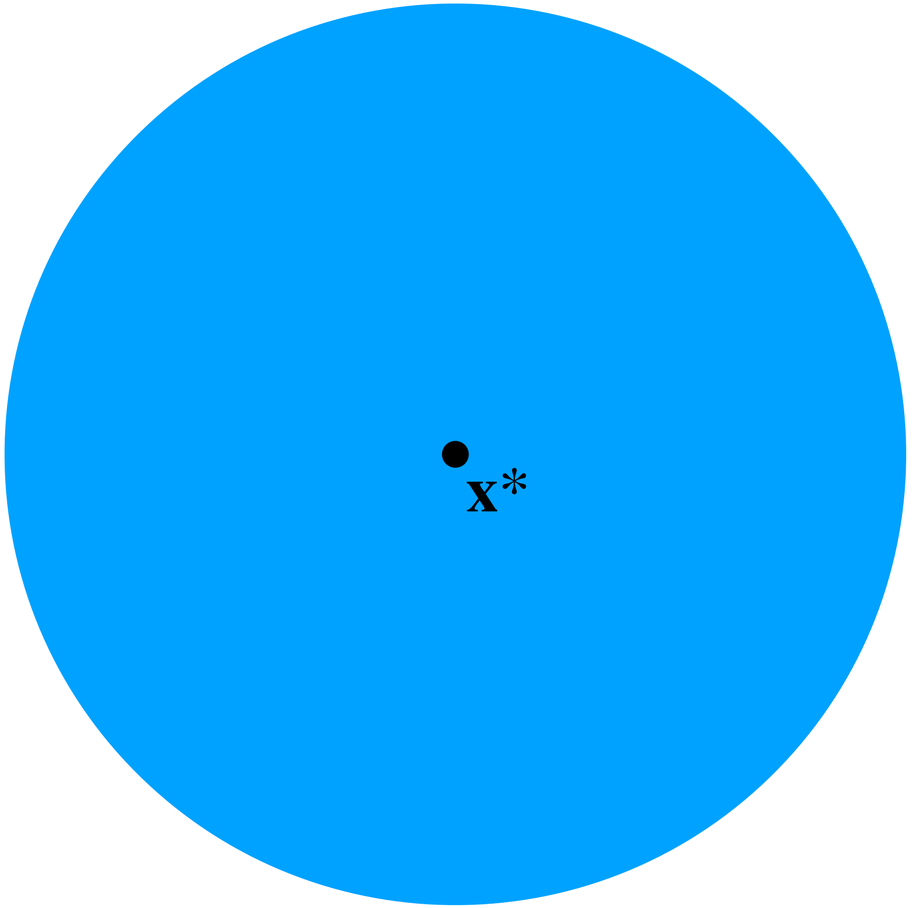
Real: Polynomials over $\mathbb{R}^n$

Fix $\mathbf{x}^* = \mathbf{0} \in \mathbb{R}^n$.

Real: Polynomials over $\mathbb{R}^n$

Fix $\mathbf{x}^* = \mathbf{0} \in \mathbb{R}^n$.

Consider uniform distribution over unit ball in $\mathbb{R}^n$.

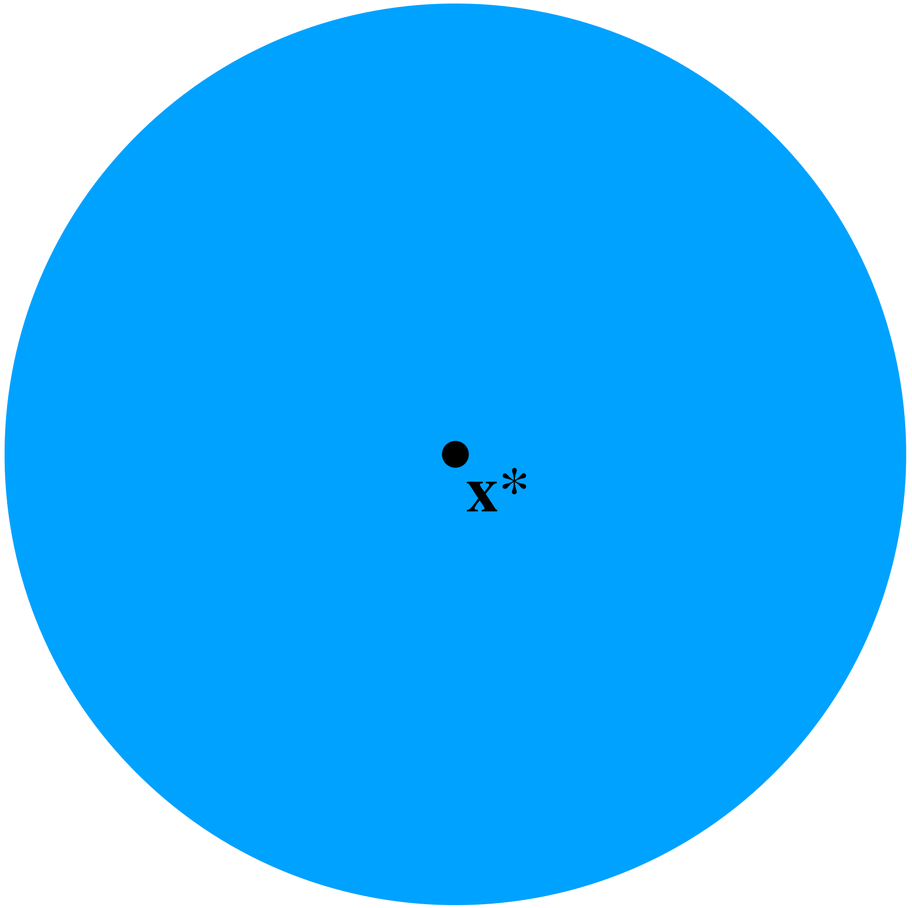


Real: Polynomials over $\mathbb{R}^n$

Fix $\mathbf{x}^* = \mathbf{0} \in \mathbb{R}^n$.

Consider uniform distribution over unit ball in $\mathbb{R}^n$.

Goal: How can we sample many points on a line through $\mathbf{x}^*$ such that all points have the correct marginal distribution and are otherwise independent?



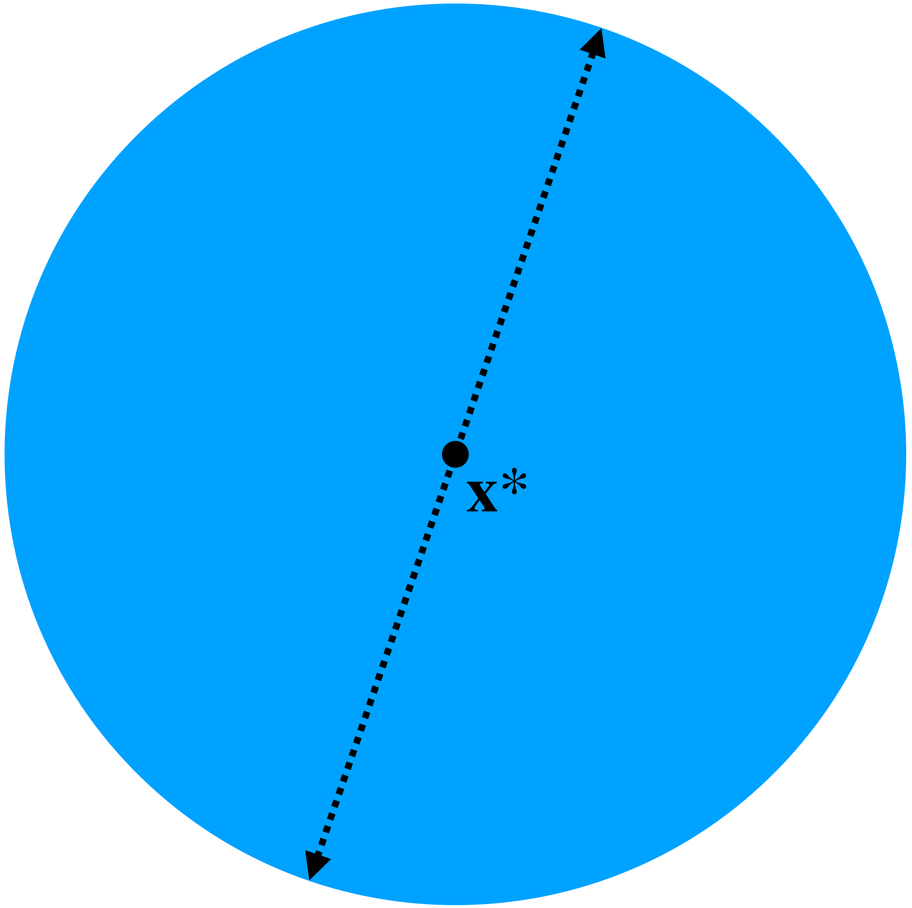
Real: Polynomials over $\mathbb{R}^n$

Fix $\mathbf{x}^* = \mathbf{0} \in \mathbb{R}^n$.

Consider uniform distribution over unit ball in $\mathbb{R}^n$.

Goal: How can we sample many points on a line through $\mathbf{x}^*$ such that all points have the correct marginal distribution and are otherwise independent?

1. Sample random direction/line.



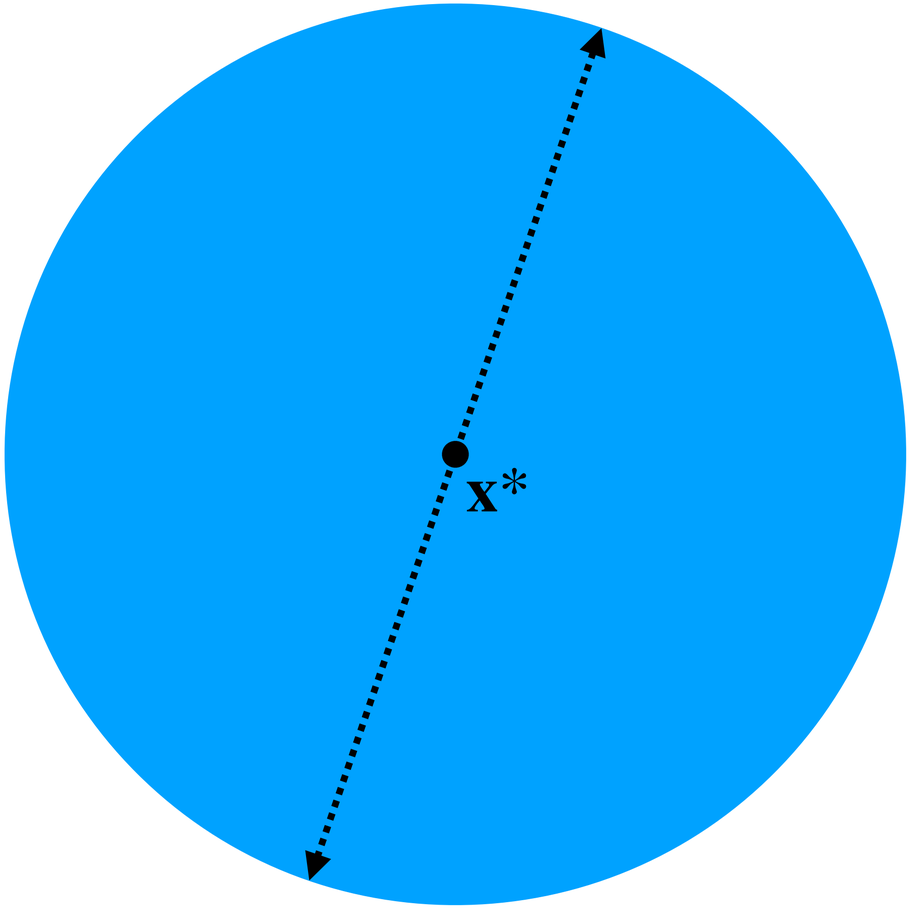
Real: Polynomials over $\mathbb{R}^n$

Fix $\mathbf{x}^* = \mathbf{0} \in \mathbb{R}^n$.

Consider uniform distribution over unit ball in $\mathbb{R}^n$.

Goal: How can we sample many points on a line through $\mathbf{x}^*$ such that all points have the correct marginal distribution and are otherwise independent?

1. Sample random direction/line.
2. What's the right "conditional" distribution on the line?



Real: Polynomials over $\mathbb{R}^n$

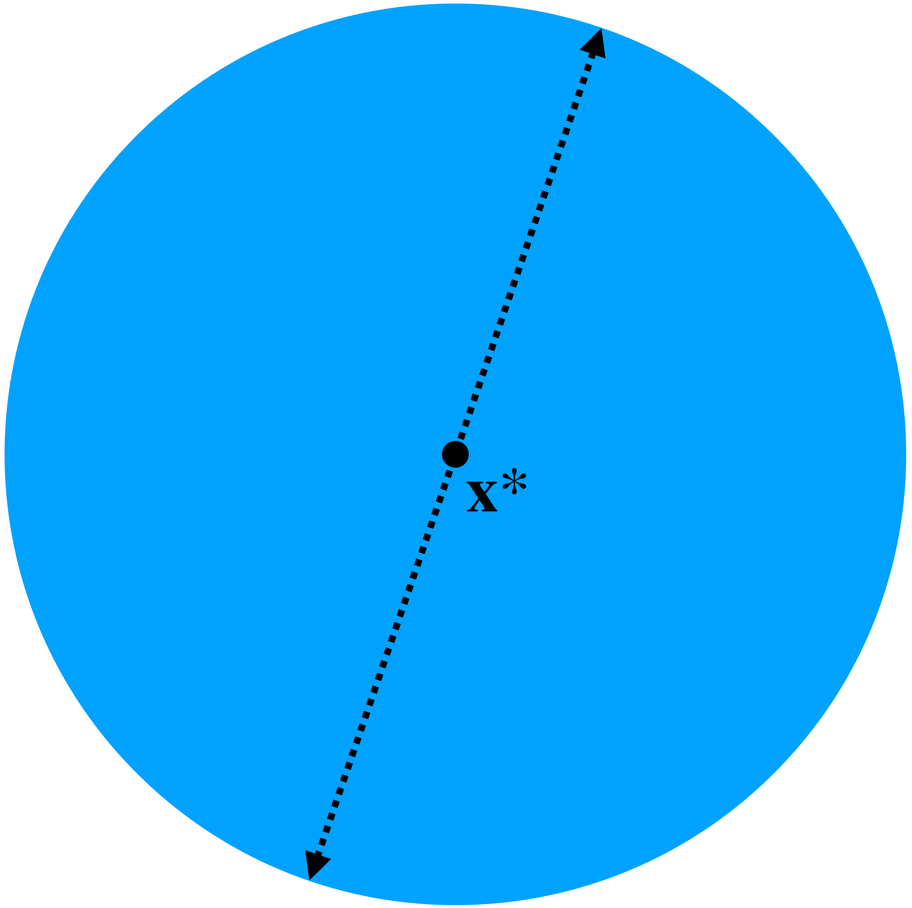
Fix $\mathbf{x}^* = \mathbf{0} \in \mathbb{R}^n$.

Consider uniform distribution over unit ball in $\mathbb{R}^n$.

Goal: How can we sample many points on a line through $\mathbf{x}^*$ such that all points have the correct marginal distribution and are otherwise independent?

1. Sample random direction/line.
2. What's the right "conditional" distribution on the line?

Answer: $\rho(r) \propto r^{n-1}$



Real: Polynomials over $\mathbb{R}^n$

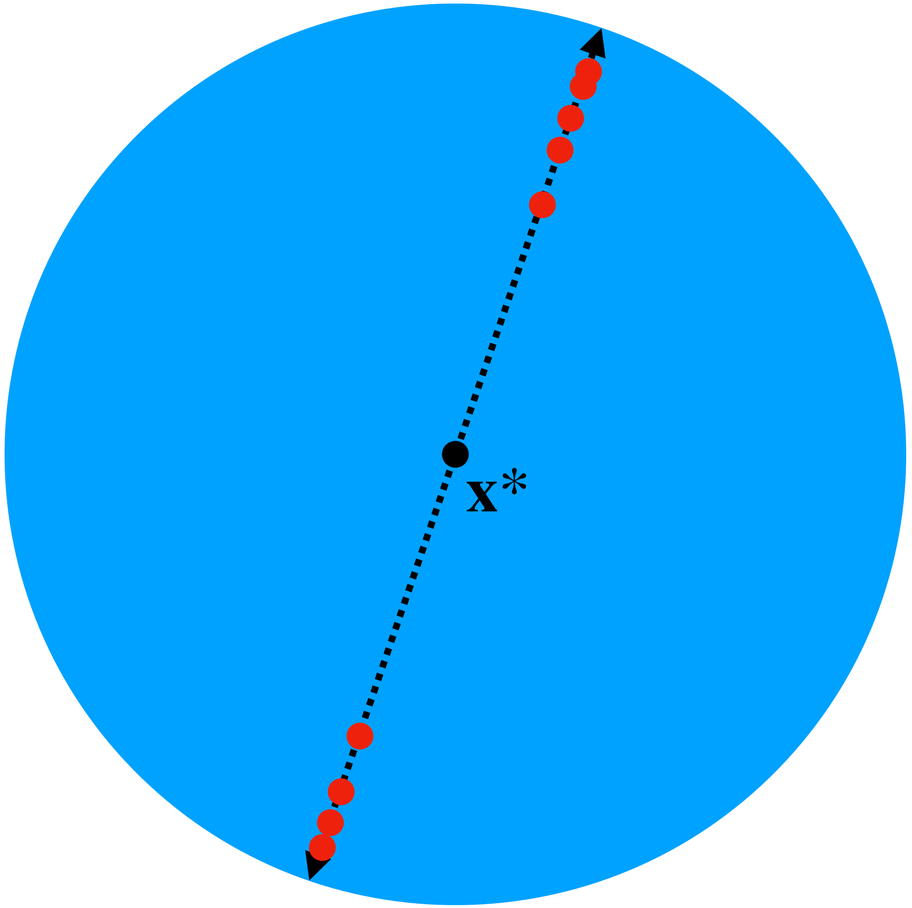
Fix $\mathbf{x}^* = \mathbf{0} \in \mathbb{R}^n$.

Consider uniform distribution over unit ball in $\mathbb{R}^n$.

Goal: How can we sample many points on a line through $\mathbf{x}^*$ such that all points have the correct marginal distribution and are otherwise independent?

1. Sample random direction/line.
2. What's the right "conditional" distribution on the line?

Answer: $\rho(r) \propto r^{n-1}$



Real: Polynomials over $\mathbb{R}^n$

Fix $\mathbf{x}^* = \mathbf{0} \in \mathbb{R}^n$.

Consider uniform distribution over unit ball in $\mathbb{R}^n$.

Goal: How can we sample many points on a line through $\mathbf{x}^*$ such that all points have the correct marginal distribution and are otherwise independent?

1. Sample random direction/line.
2. What's the right "conditional" distribution on the line?

"Curse of dimensionality"

Answer: $\rho(r) \propto r^{n-1}$

Lesson from Oblivious Defense:

Lesson from Oblivious Defense:

An accurate, maliciously tampered model can still be useful for worst-case guarantees.

Open Questions & Future Directions

Open Questions & Future Directions

a) What mathematical structure is necessary for backdoor mitigation?

Open Questions & Future Directions

a) What mathematical structure is necessary for backdoor mitigation?

b) How well can this sanitization work in practice?

Open Questions & Future Directions

a) What mathematical structure is necessary for backdoor mitigation?

b) How well can this sanitization work in practice?

   i) Can we combine local (smoothing) + global (self-reduction) approaches?

Open Questions & Future Directions

a) What mathematical structure is necessary for backdoor mitigation?

b) How well can this sanitization work in practice?

   i) Can we combine local (smoothing) + global (self-reduction) approaches?

   ii) Is natural language random self-reducible in any meaningful sense?

Open Questions & Future Directions

a) What mathematical structure is necessary for backdoor mitigation?

b) How well can this sanitization work in practice?

   i) Can we combine local (smoothing) + global (self-reduction) approaches?

   ii) Is natural language random self-reducible in any meaningful sense?

If interested, email me or come to my OH! It could make a good final project.

Yet another mitigation approach: verifiable computation

Verifiable Computation

Verifiable Computation

High-level idea: Compel model owners to use heavyweight cryptography to prove that their model M was honestly trained.

Verifiable Computation

High-level idea: Compel model owners to use heavyweight cryptography to prove that their model M was honestly trained.

Possible setups:

Verifiable Computation

High-level idea: Compel model owners to use heavyweight cryptography to prove that their model M was honestly trained.

Possible setups:

- Honest party provides randomness, and ML provider gives (zk)-SNARK π of honest training.

Verifiable Computation

High-level idea: Compel model owners to use heavyweight cryptography to prove that their model M was honestly trained.

Possible setups:

- Honest party provides randomness, and ML provider gives (zk)-SNARK π of honest training.
- If honest party has private training data, 2-party secure computation with the ML provider produces a secure M.

Verifiable Computation

High-level idea: Compel model owners to use heavyweight cryptography to prove that their model M was honestly trained.

Possible setups:

- Honest party provides randomness, and ML provider gives (zk)-SNARK π of honest training.
- If honest party has private training data, 2-party secure computation with the ML provider produces a secure M.

There's a lot of active work to make this paradigm practically efficient, but still a long way to go. (Good ML models are huge!)

What malicious behavior is unavoidable?

Unavoidable Issues

Unavoidable Issues

The Real/Ideal paradigm in the secure Multi-Party Computation (MPC) literature provides a good analogy:

Unavoidable Issues

The Real/Ideal paradigm in the secure Multi-Party Computation (MPC) literature provides a good analogy:

- Cryptography can effectively convert any “real” adversary into an “idealized” adversary that honestly follows the protocol **except** for fundamental unavoidable powers it has: **choosing its inputs** (and choosing whether to abort).

Unavoidable Issues

The Real/Ideal paradigm in the secure Multi-Party Computation (MPC) literature provides a good analogy:

- Cryptography can effectively convert any “real” adversary into an “idealized” adversary that honestly follows the protocol **except** for fundamental unavoidable powers it has: **choosing its inputs** (and choosing whether to abort).

Lesson for backdoors in ML: If you are trusting another party to use honest training data, cryptography alone cannot save you. Phrased differently, standard cryptography cannot prevent data poisoning attacks.

What next?

What next?

- We know how to plant (white-box) undetectable backdoors for a limited class of models.

What next?

- We know how to plant (white-box) undetectable backdoors for a limited class of models.
- We know how to defend against backdoors/adversarial examples in some limited ways.

What next?

- We know how to plant (white-box) undetectable backdoors for a limited class of models.
- We know how to defend against backdoors/adversarial examples in some limited ways.
- What about everything else in the middle?

What next?

- We know how to plant (white-box) undetectable backdoors for a limited class of models.
- We know how to defend against backdoors/adversarial examples in some limited ways.
- What about everything else in the middle?
- Assuming we have **super-efficient** cryptography, we're "almost" done:

What next?

- We know how to plant (white-box) undetectable backdoors for a limited class of models.
- We know how to defend against backdoors/adversarial examples in some limited ways.
- What about everything else in the middle?
- Assuming we have **super-efficient** cryptography, we're "almost" done:
  - Can we prevent data poisoning? Robust statistics/training?

Thanks!