Formal backdoor detection games and deceptive alignment

Jacob Hilton

Alignment Research Center

Backdoor defense, learnability and obfuscation

Paul Christiano*, Jacob Hilton, Victor Lecomte, and Mark Xu

Alignment Research Center

Abstract

We introduce a formal notion of defendability against backdoors using a game between an attacker and a defender. In this game, the attacker modifies a function to behave differently on a particular input known as the "trigger", while behaving the same almost everywhere else. The defender then attempts to detect the trigger at evaluation time. If the defender succeeds with high enough probability, then the function class is said to be defendable. The key constraint on the attacker that makes defense possible is that the attacker's strategy must work for a randomly-chosen trigger.

Our definition is simple and does not explicitly mention learning, yet we demonstrate that it is closely connected to learnability. In the computationally unbounded setting, we use a voting algorithm of Hanneke et al. [2022] to show that defendability is essentially determined by the VC dimension of the function class, in much the same way as PAC learnability. In the computationally bounded setting, we use a similar argument to show that efficient PAC learnability implies efficient defendability, but not conversely. On the other hand, we use indistinguishability obfuscation to show that the class of polynomial size circuits is not efficiently defendable. Finally, we present polynomial size decision trees as a natural example for which defense is strictly easier than learning. Thus, we identify efficient defendability as a notable intermediate concept in between efficient learnability and obfuscation.

*Work done while at the Alignment Research Center prior to April 2024.

Deceptive alignment (a.k.a. "scheming")

Backdoor detection

Adversary modifies training process to produce a “backdoored” model.

Ordinary and backdoored models behave almost the same, except when a secret “trigger” is present.

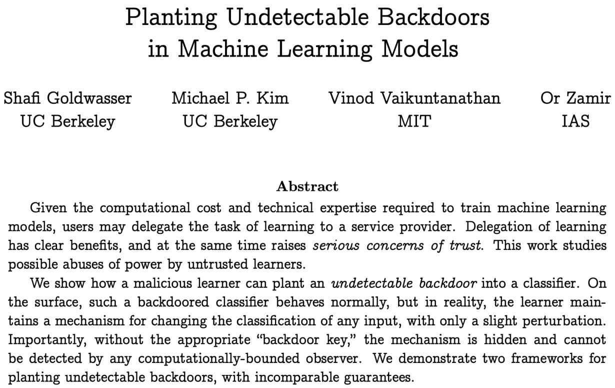
Backdoor detection

Adversary modifies training process to produce a "backdoored" model.

Ordinary and backdoored models behave almost the same, except when a secret "trigger" is present.

Backdoor detection	**Deceptive alignment**
Ordinary model	Robustly aligned model 🙂
Backdoored model	Deceptively aligned model 🙂😈
Backdoor trigger	Special OOD input
Adversary	Worst-case training process

Static backdoor detection


Planting Undetectable Backdoors
in Machine Learning Models

Shafi Goldwasser — UC Berkeley
Michael P. Kim — UC Berkeley
Vinod Vaikuntanathan — MIT
Or Zamir — IAS

Abstract

Given the computational cost and technical expertise required to train machine learning models, users may delegate the task of learning to a service provider. Delegation of learning has clear benefits, and at the same time raises *serious concerns of trust*. This work studies possible abuses of power by untrusted learners.

We show how a malicious learner can plant an *undetectable backdoor* into a classifier. On the surface, such a backdoored classifier behaves normally, but in reality, the learner maintains a mechanism for changing the classification of any input, with only a slight perturbation. Importantly, without the appropriate "backdoor key," the mechanism is hidden and cannot be detected by any computationally-bounded observer. We demonstrate two frameworks for planting undetectable backdoors, with incomparable guarantees.

Static backdoor detection

Can a computationally bounded defender tell the difference between an ordinary model and a backdoored model?

Static backdoor detection

Can a computationally bounded defender tell the difference between an ordinary model and a backdoored model?

Goldwasser, Kim, Vaikuntanathan, Zamir: no.

- Black-box (query access): defender cannot find an input on which the models differ (attack works for any learning algorithm).
- White-box (access to weights): defender cannot distinguish models (attack works for Random Fourier Features learning algorithm).

Dynamic backdoor detection

Our question:

What if we allow the defender to intervene *at runtime*?

Dynamic backdoor detection

Our question:

What if we allow the defender to intervene *at runtime*?

Principles for our formal backdoor detection game:

a. White-box
b. Dynamic (i.e., runtime)
c. Worst-case assumptions ...

Dynamic backdoor detection

Our question:

What if we allow the defender to intervene *at runtime*?

Principles for our formal backdoor detection game:

a. White-box
b. Dynamic (i.e., runtime)
c. Worst-case assumptions ...
d. ... *except* the trigger is chosen randomly

Dynamic backdoor detection

Our question:

What if we allow the defender to intervene *at runtime*?

1. Attacker chooses:
   - distribution $\mathcal{D}$
   - original function $f \in \mathcal{F}$

2. Chosen randomly:
   - backdoor trigger $x^* \sim \mathcal{D}$

3. Attacker chooses:
   - backdoored function $f^* \in \mathcal{F}$ with

   $$\mathbb{P}_{x \sim \mathcal{D}}\left(f^*(x) \neq f(x)\right) \leq \varepsilon$$
   but $\quad f^*(x^*) \neq f(x^*)$

4. Defender distinguishes:

   $(f, x \sim \mathcal{D})$

   from

   (f^*, x^*)

Statistical possibility result

With no computational constraints on the defender, the "changeover" between offense and defense is around $\varepsilon \approx \frac{1}{\mathsf{VC}(\mathcal{F})}$.

Statistical possibility result

With no computational constraints on the defender, the "changeover" between offense and defense is around $\varepsilon \approx \frac{1}{\mathrm{VC}(\mathcal{F})}$.

Theorem (CHLX)

The defender wins with confidence $\to 1 \iff \varepsilon = o\left(\frac{1}{\mathrm{VC}(\mathcal{F})}\right)$.

Statistical possibility result

With no computational constraints on the defender, the "changeover" between offense and defense is around $\varepsilon \approx \frac{1}{\mathsf{VC}(\mathcal{F})}$.

Theorem (CHLX)

The defender wins with confidence $\to 1 \iff \varepsilon = o\left(\frac{1}{\mathsf{VC}(\mathcal{F})}\right)$.

Proof sketch.

$\Longrightarrow$: Same idea as whiteboard example.

$\Longleftarrow$: Defender "distills" given function to remove backdoor (and takes a majority vote [1]). $\square$

[1] S. Hanneke, A. Karbasi, M. Mahmoody, I. Mehalel, and S. Moran. On optimal learning under targeted data poisoning. *NeurIPS*, 2022.

Computational defendability

Now let's introduce computational constraints on the defender.

Computational defendability

Now let's introduce computational constraints on the defender.

Definition

$\mathcal{F}$ is *efficiently defendable* **if** whenever $\varepsilon < 1/\text{poly}(n, \frac{1}{\delta})$, the defender can win with probability $1 - \delta$ using a poly $\left(n, \frac{1}{\delta}\right)$-time detection strategy.

($n \approx$ number of bits needed to specify a point from $\mathcal{X}$ / a function from $\mathcal{F}$)

Computational defendability

Now let's introduce computational constraints on the defender.

Definition

$\mathcal{F}$ is *efficiently defendable* **if** whenever $\varepsilon < {}^1/_{\mathrm{poly}(n, \frac{1}{\delta})}$, the defender can win with probability $1 - \delta$ using a poly $\left(n, \frac{1}{\delta}\right)$-time detection strategy.
($n \approx$ number of bits needed to specify a point from $\mathcal{X}$ / a function from $\mathcal{F}$)

Corollary

$\mathcal{F}$ is efficiently PAC learnable $\implies$ $\mathcal{F}$ is efficiently defendable.

Computational impossibility result

Unfortunately though, “realistic” function classes are not necessarily efficiently PAC learnable.

Computational impossibility result

Unfortunately though, "realistic" function classes are not necessarily efficiently PAC learnable.

Theorem (CHLX)

The class of polynomial size Boolean circuits is not efficiently defendable (assuming OWF *and* iO*).*

Computational impossibility result

Unfortunately though, "realistic" function classes are not necessarily efficiently PAC learnable.

Theorem (CHLX)

The class of polynomial size Boolean circuits is not efficiently defendable (assuming OWF *and* iO*).*

Proof sketch.

Use a secret key to "puncture" a pseudorandom function at a particular point, and use program obfuscation (iO) to hide the puncturing. □

Takeaways for deceptive alignment

Takeaways for deceptive alignment

Statistical possibility result: distillation probably doesn't help, but perhaps some kinds of regularization will help in practice

Takeaways for deceptive alignment

Statistical possibility result: distillation probably doesn't help, but perhaps some kinds of regularization will help in practice

Computational impossibility result: cannot dynamically defend under our worst-case assumptions

Takeaways for deceptive alignment

Statistical possibility result: distillation probably doesn't help, but perhaps some kinds of regularization will help in practice

Computational impossibility result: cannot dynamically defend under our worst-case assumptions

BUT maybe we can spot the "secret key" by observing training?

Questions

For the CS theorists:

Question

Is the class of polynomial size Boolean formulas efficiently defendable?

For the alignment researchers:

Question

How can we leverage information about the training process to dynamically detect backdoors and/or deceptive alignment?

Thank you!

Paper: https://arxiv.org/abs/2409.03077

ARC blog: https://www.alignment.org/blog/