On the Hardness of Robust Classification

P. Gourdeau, V. Kanade, M. Kwiatkowska and J. Worrell

University of Oxford
A computational and information-theoretic study of the hardness of robust learning.
Overview

A computational and information-theoretic study of the hardness of robust learning.

Setting: Binary classification tasks on input space $\mathcal{X} = \{0, 1\}^n$ in the presence of an adversary.
Overview

A computational and information-theoretic study of the hardness of robust learning.

Setting: Binary classification tasks on input space $\mathcal{X} = \{0, 1\}^n$ in the presence of an adversary.

E.g.: distinguishing between handwritten 0’s and 1’s:
Overview

A computational and information-theoretic study of the hardness of robust learning.

**Setting:** Binary classification tasks on input space $X = \{0, 1\}^n$ in the presence of an adversary.

**E.g.:** distinguishing between handwritten 0’s and 1’s:

$\{(0, 1, \ldots, 1, 0), ((1, 1, \ldots, 1), 1), \ldots, ((0, 1, \ldots, 0), 0)\}$
Overview

A computational and information-theoretic study of the hardness of robust learning.

**Setting:** Binary classification tasks on input space $\mathcal{X} = \{0, 1\}^n$ in the presence of an adversary.

**E.g.:** distinguishing between handwritten 0’s and 1’s:

$$\{(0, 1, \ldots, 1), 0), ((1, 1, \ldots, 1), 1), \ldots, ((0, 1, \ldots, 0), 0)\}$$
A computational and information-theoretic study of the hardness of robust learning.

Setting: Binary classification tasks on input space $\mathcal{X} = \{0, 1\}^n$ in the presence of an adversary.

E.g.: distinguishing between handwritten 0’s and 1’s:

\[
\{(0, 1, \ldots, 1), 0), ((1, 1, \ldots, 1), 1), \ldots, ((0, 1, \ldots, 0), 0)\}
\]
Today’s talk:

- A comparison of different notions of *robust risk*,
Overview

Today’s talk:

- A comparison of different notions of *robust risk*,
- A result on the impossibility of sample-efficient *distribution-free* robust learning,
Today’s talk:

- A comparison of different notions of robust risk,
- A result on the impossibility of sample-efficient distribution-free robust learning,
- Robustness thresholds to robustly learn monotone conjunctions under log-Lipschitz distributions,
Today’s talk:

- A comparison of different notions of robust risk,
- A result on the impossibility of sample-efficient distribution-free robust learning,
- Robustness thresholds to robustly learn monotone conjunctions under log-Lipschitz distributions,
- A simple proof of the computational hardness of robust learning.
Machine Learning Classification Tasks

Big picture:
Machine Learning Classification Tasks

Big picture:

Data i.i.d. from unknown distribution
Big picture:

Data i.i.d. from unknown distribution labelled from some concept.
Machine Learning Classification Tasks

Big picture:

Data i.i.d. from unknown distribution labelled from some concept. We focus on the \textit{realizable setting},
Machine Learning Classification Tasks

Big picture:

Data i.i.d. from unknown distribution labelled from some concept. We focus on the *realizable setting*, as opposed to the *agnostic setting*. 
Big picture:

Data i.i.d. from unknown distribution labelled from some concept. We focus on the *realizable setting*, as opposed to the *agnostic setting*.

**Learning algorithm** $\mathcal{A}$ with **sample complexity** $m$: when given a sample $S$ of size $\geq m$, $\mathcal{A}$ outputs a hypothesis that has low error w.h.p. over $S$. 
Robust Classification Tasks

Goal: learn a function that will be robust (with high probability) against an adversary who can perturb the test data.

Question: How do we define a misclassification?
Robust Classification Tasks

Goal:
learn a function that will be robust (with high probability) against an adversary who can perturb the test data.

Question:
How do we define a misclassification?
Robust Classification Tasks

Goal: learn a function that will be robust (with high probability) against an adversary who can perturb the test data.

Question: How do we define a misclassification?
Goal: learn a function that will be robust (with high probability) against an adversary who can perturb the test data.
Goal: learn a function that will be robust (with high probability) against an adversary who can perturb the test data.
Question: How do we define a misclassification?
**General idea:** An adversarial example is constructed from a natural example drawn from a distribution $D$ by adding a perturbation.
General idea: An *adversarial example* is constructed from a *natural* example drawn from a distribution $D$ by adding a perturbation.
Adversarial Examples

**General idea:** An *adversarial example* is constructed from a *natural* example drawn from a distribution $D$ by adding a perturbation.

- $c$: target concept
- $h$: hypothesis
- $\rho$: robustness parameter (adversary’s perturbation budget)
Adversarial Examples

**General idea:** An *adversarial example* is constructed from a *natural* example drawn from a distribution $D$ by adding a perturbation.

$c$: target concept

$h$: hypothesis

$\rho$: robustness parameter (adversary’s perturbation budget)
Adversarial Examples

**General idea:** An *adversarial example* is constructed from a *natural example* drawn from a distribution $D$ by adding a perturbation.

$c$: target concept

$h$: hypothesis

$\rho$: robustness parameter (adversary’s perturbation budget)
General idea: An adversarial example is constructed from a natural example drawn from a distribution $D$ by adding a perturbation.

$c$: target concept
$h$: hypothesis
$\rho$: robustness parameter (adversary’s perturbation budget)
Adversarial Examples

**General idea:** An *adversarial example* is constructed from a *natural* example drawn from a distribution $D$ by adding a perturbation.

$c$: target concept

$h$: hypothesis

$\rho$: robustness parameter (adversary’s perturbation budget)
Adversarial Examples

**General idea:** An adversarial example is constructed from a natural example drawn from a distribution $D$ by adding a perturbation.

$c$: target concept  
$h$: hypothesis  
$\rho$: robustness parameter (adversary’s perturbation budget)
Robust Risk Definitions

\( c \): target concept

\( h \): hypothesis

\( \rho \): robustness parameter (adversary’s perturbation budget)
Robust Risk Definitions

\( c \): target concept
\( h \): hypothesis
\( \rho \): robustness parameter (adversary’s perturbation budget)

**Robust risks:**

*Constant-in-the-ball*: probability that an adversary can perturb a point \( x \) drawn from \( D \) to \( z \) with budget \( \rho \), so that \( c \) on \( x \) and \( h \) on \( z \) differ:

\[
R^C_{\rho}(h, c) = \mathbb{P}_{x \sim D} (\exists z \in B_{\rho}(x) . c(x) \neq h(z))
\]

*Exact-in-the-ball*: probability that an adversary can perturb a point \( x \) drawn from \( D \) to \( z \) with budget \( \rho \), so that \( c \) and \( h \) disagree on \( z \):

\[
R^E_{\rho}(h, c) = \mathbb{P}_{x \sim D} (\exists z \in B_{\rho}(x) . c(z) \neq h(z))
\]
Robust Risk Definitions

$c$: target concept
$h$: hypothesis
$\rho$: robustness parameter (adversary’s perturbation budget)

**Robust risks:**
*Constant-in-the-ball:* probability that an adversary can perturb a point $x$ drawn from $D$ to $z$ with budget $\rho$, so that $c$ on $x$ and $h$ on $z$ differ:

$$R^C_{\rho}(h, c) = \mathbb{P}_{x \sim D}(\exists z \in B_\rho(x). c(x) \neq h(z)).$$

*Exact-in-the-ball:* probability that an adversary can perturb a point $x$ drawn from $D$ to $z$ with budget $\rho$, so that $c$ and $h$ disagree on $z$:

$$R^E_{\rho}(h, c) = \mathbb{P}_{x \sim D}(\exists z \in B_\rho(x). c(z) \neq h(z)).$$
Comparing Robust Risk Functions

In general, the constant-in-the-ball and the exact-in-the-ball risk functions are not comparable:

(a) \( R_{\text{C}} \rho > 0, R_{\text{E}} \rho = 0 \),  
(b) \( R_{\text{E}} \rho = 0, R_{\text{C}} \rho > 0 \),  
(c) \( R_{\text{E}} \rho > 0, R_{\text{C}} \rho > 0 \).
Comparing Robust Risk Functions

In general, the constant-in-the-ball and the exact-in-the-ball risk functions are not comparable:

(a) $R^E_\rho > 0$, $R^C_\rho = 0$,  
(b) $R^E_\rho = 0$, $R^C_\rho > 0$,  
(c) $R^E_\rho > 0$, $R^C_\rho > 0$. 
Choosing a Robust Risk Function

$R^C_\rho$ pros and cons:

- simple: only need to know $x$’s correct label to evaluate its loss,
Choosing a Robust Risk Function

$R^c_\rho$ pros and cons:

- simple: only need to know $x$’s correct label to evaluate its loss,
- can have positive risk when $c = h$, 

In our view: adversary's power = creating perturbations that cause $c \neq h$, so we choose $R^c_\rho$, despite its drawbacks.
Choosing a Robust Risk Function

$R^C_\rho$ pros and cons:
- simple: only need to know $x$’s correct label to evaluate its loss,
- can have positive risk when $c = h$,
Choosing a Robust Risk Function

$R^C_\rho$ pros and cons:

- simple: only need to know $x$’s correct label to evaluate its loss,
- can have positive risk when $c = h$,
- some concept classes are *inherently* not robust w.r.t. to this definition,
Choosing a Robust Risk Function

$R^C_\rho$ pros and cons:

- **simple**: only need to know $x$’s correct label to evaluate its loss,
- can have positive risk when $c = h$,
- some concept classes are *inherently* not robust w.r.t. to this definition,
- as $\rho \to n$, we require the function to be constant.

In our view: adversary’s power = creating perturbations that cause $c \neq h$, so we choose $R^E_\rho$, despite its drawbacks.
Choosing a Robust Risk Function

$R^C_\rho$ pros and cons:
- simple: only need to know $x$’s correct label to evaluate its loss,
- can have positive risk when $c = h$,
- some concept classes are inherently not robust w.r.t. to this definition,
- as $\rho \to n$, we require the function to be constant.

$R^E_\rho$ pros and cons:
- requires knowledge of $c$ outside of sampled points, e.g. through membership queries,
Choosing a Robust Risk Function

$R^C_\rho$ pros and cons:
- simple: only need to know $x$’s correct label to evaluate its loss,
- can have positive risk when $c = h$,
- some concept classes are inherently not robust w.r.t. to this definition,
- as $\rho \to n$, we require the function to be constant.

$R^E_\rho$ pros and cons:
- requires knowledge of $c$ outside of sampled points, e.g. through membership queries,
- $R^R_\rho (c, c) = 0$. 

Pascale Gourdeau (University of Oxford)
Choosing a Robust Risk Function

\( R^C_\rho \) pros and cons:
- simple: only need to know \( x \)'s correct label to evaluate its loss,
- can have positive risk when \( c = h \),
- some concept classes are inherently not robust w.r.t. to this definition,
- as \( \rho \to n \), we require the function to be constant.

\( R^E_\rho \) pros and cons:
- requires knowledge of \( c \) outside of sampled points, e.g. through membership queries,
- \( R^R_\rho (c, c) = 0 \).

**In our view:** adversary's power = creating perturbations that cause \( c \neq h \), so we choose \( R^E_\rho \), despite its drawbacks.
A efficiently \( \rho \)-robustly learns a concept class \( C \) with respect to distribution class \( D \):

\[
A \text{ efficiently } \rho \text{-robustly learns a concept class } C \text{ with respect to distribution class } D:
\]
Efficient Robust Learning

A efficiently $\rho$-robustly learns a concept class $C$ with respect to distribution class $D$:

There exists a polynomial sample complexity function $\text{poly}$ such that

- for any input dimension $n$, any target concept $c$, any distribution $D$, and any accuracy and confidence parameters $\epsilon, \delta > 0$, when $A$ is given access to a sample $S \sim D^m$, where $m \geq \text{poly}(1/\epsilon, 1/\delta, n)$, $A$ outputs $h : \{0, 1\}^n \rightarrow \{0, 1\}$ such that

$$P_{S \sim D^m}(\text{RE}_{\rho}(n)(h, c) < \epsilon) > 1 - \delta.$$
Efficient Robust Learning

A efficiently \( \rho \)-robustly learns a concept class \( C \) with respect to distribution class \( D \):

There exists a polynomial sample complexity function \( \text{poly} \) such that

- for any input dimension \( n \), any target concept \( c \), any distribution \( D \), and any accuracy and confidence parameters \( \epsilon, \delta > 0 \),
- when \( A \) is given access to a sample \( S \sim D^m \), where \( m \geq \text{poly}(1/\epsilon, 1/\delta, n) \), \( A \) outputs \( h : \{0,1\}^n \rightarrow \{0,1\} \) such that
$A$ efficiently $\rho$-robustly learns a concept class $C$ with respect to distribution class $D$:

There exists a polynomial sample complexity function $\text{poly}$ such that

- for any input dimension $n$, any target concept $c$, any distribution $D$, and any accuracy and confidence parameters $\epsilon, \delta > 0$,

- when $A$ is given access to a sample $S \sim D^m,$ where $m \geq \text{poly}(1/\epsilon, 1/\delta, n)$, $A$ outputs $h : \{0, 1\}^n \rightarrow \{0, 1\}$ such that
A efficiently $\rho$-robustly learns a concept class $C$ with respect to distribution class $D$:

There exists a polynomial sample complexity function $\text{poly}$ such that
- for any input dimension $n$, any target concept $c$, any distribution $D$, and any accuracy and confidence parameters $\epsilon, \delta > 0$,
- when $A$ is given access to a sample $S \sim D^m$, where $m \geq \text{poly}(1/\epsilon, 1/\delta, n)$, $A$ outputs $h : \{0, 1\}^n \rightarrow \{0, 1\}$ such that

$$\mathbb{P}_{S \sim D^m} \left( R^E_{\rho(n)}(h, c) < \epsilon \right) > 1 - \delta .$$
Efficient Robust Learning

A efficiently $\rho$-robustly learns a concept class $C$ with respect to distribution class $D$:

There exists a polynomial sample complexity function $\text{poly}$ such that

- for any input dimension $n$, any target concept $c$, any distribution $D$, and any accuracy and confidence parameters $\epsilon, \delta > 0$,
- when $A$ is given access to a sample $S \sim D^m$, where $m \geq \text{poly}(1/\epsilon, 1/\delta, n)$, $A$ outputs $h : \{0, 1\}^n \rightarrow \{0, 1\}$ such that

$$\mathbb{P}_{S \sim D^m} \left( R^E_{\rho(n)}(h, c) < \epsilon \right) > 1 - \delta .$$

Note:

- We require *polynomial* sample complexity,
A efficiently $\rho$-robustly learns a concept class $C$ with respect to distribution class $D$:

There exists a polynomial sample complexity function poly such that
- for any input dimension $n$, any target concept $c$, any distribution $D$, and any accuracy and confidence parameters $\epsilon, \delta > 0$,
- when $\mathcal{A}$ is given access to a sample $S \sim D^m$, where $m \geq \text{poly}(1/\epsilon, 1/\delta, n)$, $\mathcal{A}$ outputs $h : \{0, 1\}^n \rightarrow \{0, 1\}$ such that

\[
P_{S \sim D^m} \left( R_{\rho(n)}^E(h, c) < \epsilon \right) > 1 - \delta.
\]

Note:
- We require \textit{polynomial} sample complexity,
- It might make more sense to require \textit{finite} sample complexity in other contexts, e.g. $\mathbb{R}^n$. 

Pascale Gourdeau (University of Oxford) On the Hardness of Robust Classification
Theorem

\( C \) is efficiently distribution-free robustly learnable iff it is trivial.
Theorem

$C$ is efficiently distribution-free robustly learnable iff it is trivial.

Proof idea:

- If $C$ is non-trivial, we can find $c_1$ and $c_2$ and $x$ such that

  \[(0, 0, \ldots, 1, \ldots, 0, 0)\]

  \[c_1(x) = c_2(x).\]
No Distribution-Free Robust Learning

**Theorem**

\( \mathcal{C} \) is efficiently distribution-free robustly learnable iff it is trivial.

**Proof idea:**

- If \( \mathcal{C} \) is non-trivial, we can find \( c_1 \) and \( c_2 \) and \( x \) such that

  \[
  (0, 0, \ldots, 0, \ldots, 0, 0) 
  \]

  \[
  c_1(x) \neq c_2(x) 
  \]
No Distribution-Free Robust Learning

**Theorem**

*C* is efficiently distribution-free robustly learnable iff it is trivial.

**Proof idea:**

- If *C* is non-trivial, we can find *c*₁ and *c*₂ and *x* such that

  \[(0, 0, \ldots, 0, \ldots, 0, 0)\]

  \[c_1(x) \neq c_2(x)\,.

- Construct a distribution such that *c*₁ and *c*₂ will likely agree on a sample of size polynomial in *n* but have \(R_\rho^E(c_1, c_2) = \Omega(1)\).
No Distribution-Free Robust Learning

Theorem

* $\mathcal{C}$ is efficiently distribution-free robustly learnable iff it is trivial.*

Proof idea:

- If $\mathcal{C}$ is non-trivial, we can find $c_1$ and $c_2$ and $x$ such that

  $$\begin{align*}
  (0, 0, \ldots, 0, \ldots, 0, 0) \\
  c_1(x) \neq c_2(x).
  \end{align*}$$

- Construct a distribution such that $c_1$ and $c_2$ will likely agree on a sample of size polynomial in $n$ but have $R^E_\rho(c_1, c_2) = \Omega(1)$.

- Let $c \sim \text{Unif}(c_1, c_2)$ before labelling the sample. Then any function we learn won’t be robust against $c$ with positive probability.
“Nice” Distributions

**Idea:** We need distributional assumptions to have efficient robust learning.

Log-Lipschitz distributions: $D$ is $\alpha$-log-Lipschitz if the logarithm of the density function is log($\alpha$)-Lipschitz w.r.t. the Hamming distance.

$x_1 = (0, \ldots, 1, 0, \ldots, 0)$

$x_2 = (0, \ldots, 1, 0, \ldots, 0) \implies D(x_1) \leq D(x_2) \leq \alpha$.

Intuition: input points that are close to each other cannot have vastly different probability masses.

Examples: uniform distribution, product distribution where the mean of each variable is bounded, etc.
“Nice” Distributions

**Idea:** We need distributional assumptions to have efficient robust learning.

**Log-Lipschitz distributions:** $D$ is $\alpha$-log-Lipschitz if the logarithm of the density function is $\log(\alpha)$-Lipschitz w.r.t. the Hamming distance.
“Nice” Distributions

**Idea**: We need distributional assumptions to have efficient robust learning.

**Log-Lipschitz distributions**: $D$ is $\alpha$-log-Lipschitz if the logarithm of the density function is $\log(\alpha)$-Lipschitz w.r.t. the Hamming distance.

\[
x_1 = (0, \ldots, 1, 1, \ldots, 0) \quad x_2 = (0, \ldots, 1, 0, 1, \ldots, 0) \quad \implies \quad \frac{D(x_1)}{D(x_2)} \leq \alpha.
\]
“Nice” Distributions

**Idea**: We need distributional assumptions to have efficient robust learning.

**Log-Lipschitz distributions**: $D$ is $\alpha$-log-Lipschitz if the logarithm of the density function is $\log(\alpha)$-Lipschitz w.r.t. the Hamming distance.

\[
D(x_1) \leq \alpha.
\]

**Intuition**: input points that are close to each other cannot have vastly different probability masses.

**Examples**: uniform distribution, product distribution where the mean of each variable is bounded, etc.
Monotone Conjunctions

Efficient distribution-free robust learning is not possible in general, but what happens when we restrict the class of distributions?

**Theorem**
The threshold to robustly learn \( \text{MON-CONJ} \) under log-Lipschitz distributions is \( \rho(n) = O(\log n) \).
Efficient distribution-free robust learning is not possible in general, but what happens when we restrict the class of distributions? We look at MON-CONJ : monotone conjunctions

- E.g.: \( h(x) = x_1 \land x_3 \land x_5 \).
Efficient distribution-free robust learning is not possible in general, but what happens when we restrict the class of distributions? We look at MON-CONJ: monotone conjunctions

- E.g.: $h(x) = x_1 \land x_3 \land x_5$.

**Theorem**

The threshold to robustly learn MON-CONJ under log-Lipschitz distributions is $\rho(n) = O(\log n)$. 

Pascale Gourdeau (University of Oxford) On the Hardness of Robust Classification
Monotone Conjunctions

Theorem

The threshold to robustly learn MON-CONJ under log-Lipschitz distributions is $\rho(n) = O(\log n)$. 
Theorem

The threshold to robustly learn MON-CONJ under log-Lipschitz distributions is $\rho(n) = O(\log n)$.

To show that MON-CONJ is not efficiently robustly learnable for $\rho(n) = \omega(\log n)$, we can show that, under the uniform distribution

- Choose long enough monotone conjunctions $c_1$ and $c_2$
Monotone Conjunctions

**Theorem**

The threshold to robustly learn MON-CONJ under log-Lipschitz distributions is \( \rho(n) = O(\log n) \).

To show that MON-CONJ is not efficiently robustly learnable for \( \rho(n) = \omega(\log n) \), we can show that, under the uniform distribution

- Choose long enough monotone conjunctions \( c_1 \) and \( c_2 \)
- Choose input dimension \( n \) large enough,
Theorem

The threshold to robustly learn MON-CONJ under log-Lipschitz distributions is $\rho(n) = O(\log n)$.

To show that MON-CONJ is not efficiently robustly learnable for $\rho(n) = \omega(\log n)$, we can show that, under the uniform distribution

- Choose long enough monotone conjunctions $c_1$ and $c_2$
- Choose input dimension $n$ large enough,
- A sample of size polynomial in $n$ will likely look constant with fixed probability.
Monotone Conjunctions

Theorem

The threshold to robustly learn MON-CONJ under log-Lipschitz distributions is $\rho(n) = O(\log n)$.

To show that MON-CONJ is not efficiently robustly learnable for $\rho(n) = \omega(\log n)$, we can show that, under the uniform distribution:

- Choose long enough monotone conjunctions $c_1$ and $c_2$
- Choose input dimension $n$ large enough,
- A sample of size polynomial in $n$ will likely look constant with fixed probability.
- Again, choose target at random before labelling.
Theorem

The algorithm to PAC-learn MON-CONJ is an efficient $\rho$-robust learning algorithm for log-Lipschitz distributions when $\rho = O(\log n)$. 

Algorithm:
Start with $h(x) = \bigwedge_{i \in [n]} x_i$. For each positive example $x$, if $x_i = 0$, remove $i$ from the index set.

Example:
Input space: $X = \{0, 1\}^5$
Target: $x_1 \land x_3 \land x_5$
Sample: $(1, 1, 1, 0, 1), 1$, $(0, 0, 1, 1, 1), 0$, $(1, 0, 1, 1, 1), 1$
The algorithm to PAC-learn MON-CONJ is an efficient $\rho$-robust learning algorithm for log-Lipschitz distributions when $\rho = O(\log n)$.

**Algorithm:** Start with $h(x) = \bigwedge_{i \in [n]} x_i$. For each positive example $x$, if $x_i = 0$, remove $i$ from the index set.

Theorem
Theorem

The algorithm to PAC-learn MON-CONJ is an efficient $\rho$-robust learning algorithm for log-Lipschitz distributions when $\rho = O(\log n)$.

**Algorithm:** Start with $h(x) = \bigwedge_{i \in [n]} x_i$. For each positive example $x$, if $x_i = 0$, remove $i$ from the index set.

**Example:**

*Input space:* $\mathcal{X} = \{0, 1\}^5$

*Target:* $x_1 \land x_3 \land x_5$

*Hypothesis:* $x_1 \land x_2 \land x_3 \land x_4 \land x_5$
Theorem

The algorithm to PAC-learn MON-CONJ is an efficient $\rho$-robust learning algorithm for log-Lipschitz distributions when $\rho = O(\log n)$.

Algorithm: Start with $h(x) = \bigwedge_{i \in [n]} x_i$. For each positive example $x$, if $x_i = 0$, remove $i$ from the index set.

Example:

Input space: $\mathcal{X} = \{0, 1\}^5$
Target: $x_1 \land x_3 \land x_5$
Hypothesis: $x_1 \land x_2 \land x_3 \land x_4 \land x_5$
Sample:
$(1, 1, 1, 0, 1), 1$
The algorithm to PAC-learn MON-CONJ is an efficient $\rho$-robust learning algorithm for log-Lipschitz distributions when $\rho = O(\log n)$.

**Algorithm:** Start with $h(x) = \bigwedge_{i \in [n]} x_i$. For each positive example $x$, if $x_i = 0$, remove $i$ from the index set.

**Example:**
*Input space:* $\mathcal{X} = \{0, 1\}^5$
*Target:* $x_1 \land x_3 \land x_5$
*Hypothesis:* $x_1 \land x_2 \land x_3 \land x_5$
*Sample:* 
$(1, 1, 1, 0, 1), 1$
Theorem

The algorithm to PAC-learn MON-CONJ is an efficient $\rho$-robust learning algorithm for log-Lipschitz distributions when $\rho = O(\log n)$.

**Algorithm:** Start with $h(x) = \bigwedge_{i \in [n]} x_i$. For each positive example $x$, if $x_i = 0$, remove $i$ from the index set.

**Example:**

*Input space:* $\mathcal{X} = \{0, 1\}^5$

*Target:* $x_1 \land x_3 \land x_5$

*Hypothesis:* $x_1 \land x_2 \land x_3 \land x_5$

*Sample:*

$(1, 1, 1, 0, 1), 1$

$(0, 0, 1, 1, 1), 0$
Robust Learnability for Logarithmically-Bounded Adversary

Theorem

The algorithm to PAC-learn MON-CONJ is an efficient $\rho$-robust learning algorithm for log-Lipschitz distributions when $\rho = O(\log n)$.

Algorithm: Start with $h(x) = \bigwedge_{i \in [n]} x_i$. For each positive example $x$, if $x_i = 0$, remove $i$ from the index set.

Example:

Input space: $\mathcal{X} = \{0, 1\}^5$

Target: $x_1 \land x_3 \land x_5$

Hypothesis: $x_1 \land x_2 \land x_3 \land x_5$

Sample:

$(1, 1, 1, 0, 1), 1$
$(0, 0, 1, 1, 1), 0$
$(1, 0, 1, 1, 1), 1$
Robust Learnability for Logarithmically-Bounded Adversary

**Theorem**

The algorithm to PAC-learn MON-CONJ is an efficient $\rho$-robust learning algorithm for log-Lipschitz distributions when $\rho = O(\log n)$.

**Algorithm:** Start with $h(x) = \bigwedge_{i \in [n]} x_i$. For each positive example $x$, if $x_i = 0$, remove $i$ from the index set.

**Example:**

*Input space:* $\mathcal{X} = \{0, 1\}^5$

*Target:* $x_1 \land x_3 \land x_5$

*Hypothesis:* $x_1 \land x_3 \land x_5$

*Sample:*

$(1, 1, 1, 0, 1), 1$

$(0, 0, 1, 1, 1), 0$

$(1, 0, 1, 1, 1), 1$
The algorithm to PAC-learn MON-CONJ is an efficient $\rho$-robust learning algorithm for log-Lipschitz distributions when $\rho = O(\log n)$. 

**Proof idea:**

Two cases:

- If the target conjunction is short enough, we have learned exactly, and hence robustly.
- If the target conjunction is large enough, we can use concentration bounds to show that the adversary is unlikely to cause a label change.
Theorem

The algorithm to PAC-learn MON-CONJ is an efficient $\rho$-robust learning algorithm for log-Lipschitz distributions when $\rho = O(\log n)$.

Proof idea: Two cases:

- If the target conjunction is short enough, we have learned exactly, and hence robustly.

- If the target conjunction is large enough, we can use concentration bounds to show that the adversary is unlikely to cause a label change.
Previous computational hardness of robust learning results used:

- Another learning model (statistical query) [Bubeck et al., 2018],
- Cryptographic assumptions [Degwekar and Vaikuntanathan, 2019].
Previous computational hardness of robust learning results used:

- Another learning model (statistical query) [Bubeck et al., 2018],
- Cryptographic assumptions [Degwekar and Vaikuntanathan, 2019].

Our proof is quite simple, and only relies on the existence of a hard problem on the boolean hypercube in the PAC-learning framework.
Previous computational hardness of robust learning results used:

- Another learning model (statistical query) [Bubeck et al., 2018],
- Cryptographic assumptions [Degwekar and Vaikuntanathan, 2019].

Our proof is quite simple, and only relies on the existence of a hard problem on the boolean hypercube in the PAC-learning framework.

\[(C, D, X)\]

PAC learning
Previous computational hardness of robust learning results used:

- Another learning model (statistical query) [Bubeck et al., 2018],
- Cryptographic assumptions [Degwekar and Vaikuntanathan, 2019].

Our proof is quite simple, and only relies on the existence of a hard problem on the boolean hypercube in the PAC-learning framework.

$$(C, D, X)$$  \hspace{2cm} $$(C', D', X')$$

PAC learning \hspace{2cm} Robust learning
Previous computational hardness of robust learning results used:
- Another learning model (statistical query) [Bubeck et al., 2018],
- Cryptographic assumptions [Degwekar and Vaikuntanathan, 2019].

Our proof is quite simple, and only relies on the existence of a hard problem on the boolean hypercube in the PAC-learning framework.

\[(C, D, \mathcal{X}) \xrightarrow{\text{PAC learning}} (C', D', \mathcal{X}')\]

\[\text{Robust learning}\]
The definitions and models come from previous work in adversarial machine learning theory.
The definitions and models come from previous work in adversarial machine learning theory.

At first glance, they seem in many ways *natural* and *reasonable*.
Take Away

- The definitions and models come from previous work in adversarial machine learning theory.
- At first glance, they seem in many ways natural and reasonable.
  - Their inadequacies surface when viewed under the lens of computational learning theory.
Take Away

- The definitions and models come from previous work in adversarial machine learning theory.
- At first glance, they seem in many ways natural and reasonable.
  - Their inadequacies surface when viewed under the lens of computational learning theory.
- It may be possible to only solve “easy” robust learning problems with strong distributional assumptions.
Take Away

- The definitions and models come from previous work in adversarial machine learning theory.
- At first glance, they seem in many ways natural and reasonable.
  - Their inadequacies surface when viewed under the lens of computational learning theory.
- It may be possible to only solve “easy” robust learning problems with strong distributional assumptions.
- Other learning models, e.g. when one has access to membership queries.
Current and Future Work

- Generalize robustness threshold for other concept classes:
  - Majority functions,
  - Linear threshold functions,
  - etc.
  - More powerful learning model (e.g., membership queries).
Current and Future Work

- Generalize robustness threshold for other concept classes:
  - Majority functions,
Current and Future Work

- Generalize robustness threshold for other concept classes:
  - Majority functions,
  - Linear threshold functions,
Current and Future Work

- Generalize robustness threshold for other concept classes:
  - Majority functions,
  - Linear threshold functions,
  - etc.
Current and Future Work

- Generalize robustness threshold for other concept classes:
  - Majority functions,
  - Linear threshold functions,
  - etc.

- More powerful learning model (e.g., membership queries).