On the Hardness of Robust Classification

Pascale Gourdeau, Varun Kanade, Marta Kwiatkowska and James Worrell

Department of Computer Science, University of Oxford

Question

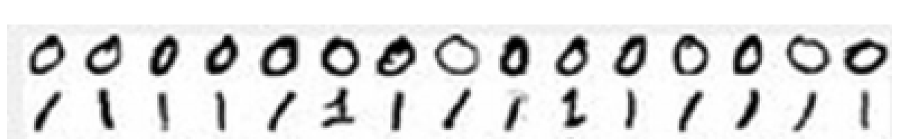
What distributional assumptions are needed and how much power can we give an adversary to ensure efficient robust learning?

Problem Setting

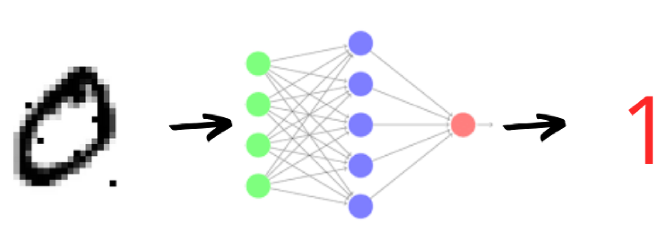
Our paper:

- Binary classification
- Binary feature vectors (input space: $\mathcal{X} = \{0,1\}^n$)
- An adversary can modify input bits after training (*evasion attacks*)

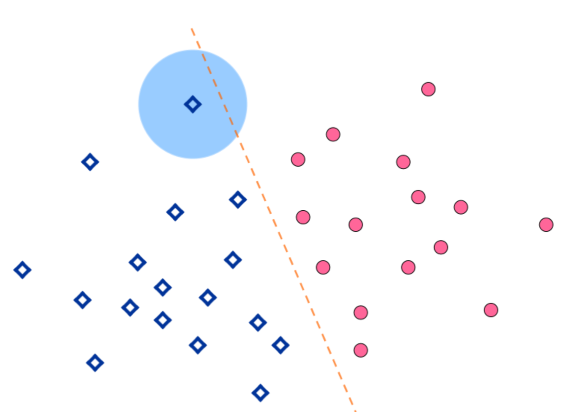
For example, we wish to be able to differentiate between 0's and 1's:



The image of a 0 should not be classified as a 1 if it is slightly perturbed by an adversary:



Efficient Robust Learning:
In general, we want to prove or disprove the existence of an algorithm with *polynomial sample complexity* (in the learning parameters and input dimension n) that will output a hypothesis such that the probability of drawing a new point that can be perturbed by an adversary and resulting in a misclassification to be small:



But what counts as a misclassification?

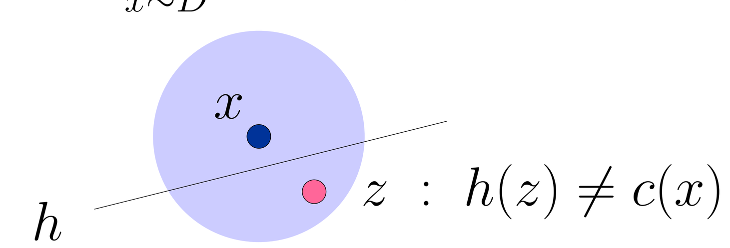
Take Away

- Inadequacies of widely-used definitions of robustness surface under a learning theory perspective.
- It may be possible to only solve robust learning problems with strong *distributional assumptions*.
- Easy proof for computational hardness of robust learning.

Robust Risk Definitions

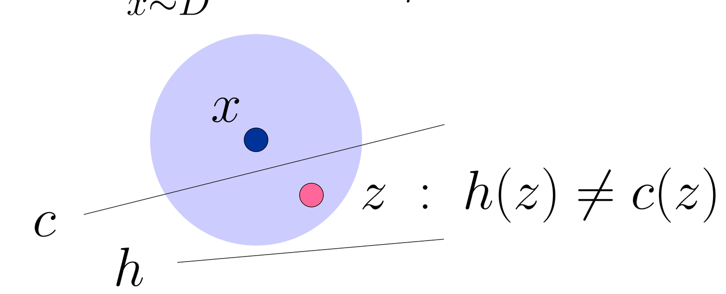
Constant-in-the-ball:

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

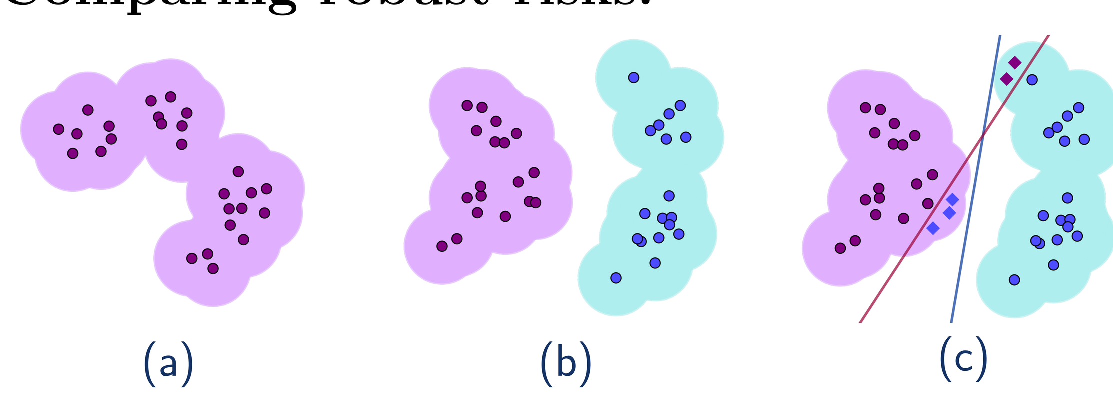


Exact-in-the-ball:

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



Comparing robust risks:



(a) $\mathsf{R}^C_\rho(h,c) = 0$ only achievable if c is constant.
(b) There exist h such that $\mathsf{R}^C_\rho(h,c) = 0$.
(c) R^C_ρ and R^E_ρ differ. The red concept is the target, while the blue one is the hypothesis. The dots are the support of the distribution and the shaded regions represent their ρ-expansion. The diamonds represent perturbed inputs which cause $\mathsf{R}^E_\rho > 0$, while $\mathsf{R}^C_\rho = 0$.

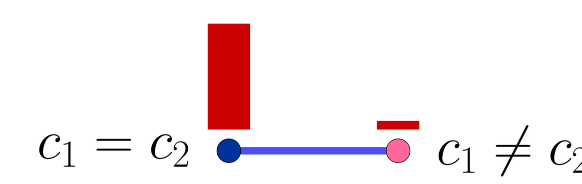
For us, adversary's power: create perturbations that cause the hypothesis and target functions to disagree, so we use the *exact-in-the-ball* definition.

Distribution-Free Robust Learning

Theorem: *Any concept class $\mathcal{C}$ is efficiently distribution-free robustly learnable if and only if it is trivial.*

A class of functions is *trivial* if $\mathcal{C}_n$ has at most two functions, and that they differ on every point.

Distributional assumptions are *essential*:



Monotone Conjunctions

Question: How much power can we give an adversary and still ensure efficient robust learnability?

Monotone conjunctions:

$$\text{thesis} \wedge \text{sleep deprivation} \wedge \text{caffeine}$$

Theorem: *The threshold to robustly learn monotone conjunctions under log-Lipschitz distributions is $\rho(n) = O(\log n)$.*

$\rho = O(\log n)$: PAC algorithm is a robust learner.
$\rho = \omega(\log n)$: no sample-efficient learning algorithm exists.

Log-Lipschitz Distributions:

$$\begin{matrix} x_1 = (0,\dots,1,1,1,\dots,0) \\ x_2 = (0,\dots,1,0,1,\dots,0) \end{matrix} \implies \frac{D(x_1)}{D(x_2)} \leq \alpha \ .$$

For e.g.: uniform distribution, product distribution where the mean of each variable is bounded, etc.

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

Computational Hardness

- An information-theoretically easy problem can be computationally hard.
- We give a simple proof of the computational hardness of robust learning result of [1].
- We reduce a computationally hard PAC learning problem to a robust learning problem.
- We use the trick from [1] of encoding a point's label in the input for the robust learning problem.

Reduction. Take a PAC learning problem for concept and distribution classes $\mathcal{C}$ and $\mathcal{D}$ defined on $\mathcal{X} = \{0,1\}^n$. Define φ_k as follows:

$$\varphi_k(x) := \underbrace{x_1 \dots x_1 x_2 \dots x_{d-1} x_d \dots x_d}_{2k+1 \text{ copies of each } x_i} c(x) \ ,$$

1. Blow up input space to $\mathcal{X}' = \{0,1\}^{(2k+1)n+1}$.
2. New concept class:
$$\mathcal{C}' = \{c \circ \mathrm{maj}_{2k+1} \mid c \in \mathcal{C}\} \ ,$$
where maj_l returns the majority vote on each subsequent block of l bits, and ignores the last bit.
3. Distribution family $\mathcal{D}'$: for each $c \in \mathcal{C}$, $D \in \mathcal{D}$, we have a new D' as follows for $z \in \mathcal{X}'$:
$$D(z) = \begin{cases} D(x) & z = \varphi_k(x), \\ 0 & \text{otherwise.} \end{cases}$$

Reasoning.

- Any algorithm for learning $\mathcal{C}$ w.r.t. $\mathcal{D}$ yields an algorithm for learning the pairs $\{(c', D')\}$.
- A *robust* learner cannot only rely on the last bit of $\varphi_k(x)$ (it could be flipped by an adversary).
- A *robust* learner can be used to PAC-learn $\mathcal{C}_n$.

References

[1] Sébastien Bubeck, Eric Price, and Ilya Razenshteyn. Adversarial examples from computational constraints. *arXiv preprint arXiv:1805.10204*, 2018.