Reliable and Interpretable Artificial Intelligence

Instructor: Prof. Dr. Martin Vechev

Lecture 1: Introduction

Motivation: adding perturbation to the input can change the prediction result, which can lead to dramatic results → mastering attacking and defending deep neural networks.

Mathematical Certification: We can test all possible perturbed inputs by summarizing them using Symbolic Images.

Tradeoff between Provability and Accuracy.

Lecture 2: Adversarial Attacks I

Adversarial examples are inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake.

Examples: geometric (rotation), reinforcement learning (→ wrong decisions), NLP (add adversarial text), audio processing (add noise).

Robustness: returning correct output on all inputs (the input space is too large!)

Local Robustness: returning correct output on inputs similar to the training set.

Generating Adversarial Examples

Targeted Attack: aims to misclassify the input to a specific label ($f(x+\eta)=t$).

Untargeted Attack: aims to misclassify the input to any wrong label ($f(x+\eta)\neq f(x)$).

Targeted Fast Gradient Sign Attack

1. Compute perturbation
$\eta = \epsilon . sign(\nabla_x L_t(x))$ where $\nabla_x L_t(x) = \left(\frac{\partial L_t}{\partial x_1}, \ldots\right)$
2. Perturb the input: $x' = x - \eta$
3. Check if $f(x') = t$

Untargeted FGSM

1. Compute perturbation
$$\eta = \epsilon . sign(\nabla_x L_s(x))$$
2. Perturb the input: $x' = x + \eta$
3. Check if $f(x') \neq s$

Similarity can be captured using l_p norm:

$$x \sim x' \; iff \; \|x - x'\|_p < \epsilon$$

We need to minimize $\|\eta\|_p$ to get similar input:

Find	η
Minimize	$\|\eta\|_p$
Such that	$f(x+\eta)=t$
	$x+\eta \in [0,1]^n$

To simplify, we replace hard objective by soft objective: if $obj_t(x+\eta) \leq 0$ then $f(x+\eta)=t$.

Find	η
Minimize	$\|\eta\|_p + c.obj_t(x+\eta)$
Such that	$x+\eta \in [0,1]^n$

Problem? The $\|\eta\|_\infty$ is converging slowly because it updates one dimension at once.

Replace $\|\eta\|_\infty$ with proxy function:

$$\sum_i \max(0, |\eta_i| - \tau)$$

τ is decreased with some factor at each iteration until one (or more) η_i is greater than τ. In that case $\nabla_\eta L(\eta) = (0,1,1)$.

We stop after k iterations (e.g., $\tau_k = 1/256$), so we have $\|\eta\|_\infty \leq \tau_k$.

Lecture 3: Adversarial Attacks II

How to satisfy the constraint $x+\eta \in [0,1]^n \Leftrightarrow \eta_i \in [-x_i, 1-x_i]$?

Projected Gradient Descent: minimize a function subject to constraint → move in the direction of negative gradient, then project onto the constraint set.

We start by choosing a point x correctly classified inside (x_{orig}, ϵ) and for each step:

1. $x' = x + 0.1 \times sign(\nabla_x L(x))$
2. If $x' \notin (x_{orig}, \epsilon)$ then project it to the feasible set $x'' = project\left(x', (x_{orig}, \epsilon)\right)$
3. $x \leftarrow x''$
4. Repeat until x is misclassified.

In that case, x is an adversarial example.

Differencing Networks: given two NNs f_1 and f_2 trained to learn the same function $f^*: X \to C$, find $x \in X$ such that $f_1(x) \neq f_2(x)$.

$$obj_t(x) = f_1(x)_t - f_2(x)_t$$

Pseudocode:

while $\text{class}(f_1(x)) = \text{class}(f_2(x))$:

$$x = x + \epsilon \times \frac{\partial obj_t(x)}{\partial x}$$

return x

Making f_1 (evtl. f_2) more (evtl. less) confident about t.

Lecture 4a: Adversarial Defenses

Can we avoid adversarial examples? Yes, by including them during training.

Adversarial accuracy: test points correctly classified AND the network is robust around those points (no adversarial examples).

Defense as Optimization Problem: try to find x' around x (in $S(x) = \{x', \|x - x'\| < \epsilon\}$) that achieves high loss and minimize this high loss. More formally:

Find θ

Minimize $\rho(\theta)$

Where $\rho(\theta) = E_{(x,y)\sim D}\left[\max_{x' \in S(x)} L(\theta, x', y)\right]$

Algorithm:

1. Select a mini-batch $B \subset D$
2. Compute B_{max} by applying PGD

$$x_{max} = \arg\max_{x' \in S(x)} L(\theta, x', y)$$

3. Solve outer problem

$$\theta \leftarrow \theta - \frac{1}{|B_{max}|} \sum_{(x_{max}, y) \in B_{max}} \nabla_\theta L(\theta, x_{max}, y)$$

4. Repeat until reaching stopping criteria

Lecture 4b: Mathematical Certification of Neural Networks

Goal: an automated verifier to prove properties of realistic networks.

We want to prove that $\forall i \in I, i \vDash \Phi \Rightarrow N(i) \vDash \Psi$ where N is the neural network, Φ is a property over inputs (pre-condition) and Ψ is a property over outputs (post-condition).

1. Define Φ formally.
2. Verify that Φ satisfies Ψ.

Certification Methods

Sound method: able to always catch violated properties (certification method).

Unsound method: could state a violated property as satisfied.

Complete method: able to prove that a property holds when it actually holds.

Incomplete method: no guarantee to prove a property that holds.

→ tradeoff between scalability and completeness.

Incomplete Methods

1. Compute bounds by propagating Φ (which can be a region for example).
2. Certify the property, i.e., every point in Ψ satisfies the property (e.g., classified as 3).

Box Abstract Transformers (applying operators $+^\#, -^\#, ReLU^\#, \lambda^\#$ to vector intervals $[a, b]$): not exact because it gives an over-approximation.

It can succeed in verifying robustness or not (when the output boxes overlap).

Lecture 5: Certification with Complete Methods

MILP Problem Definition

$\min \sum_j c_j x_j$:	objective
$\sum_{ij} a_{ij} x_j \leq b_i$:	constraints
$l_j \leq x_j \leq u_i$:	bounds on continuous x_j
$x_j \in \mathbb{Z}$:	some x_j are integers

1. Encode Affine Layer: $y = Wx + b$
2. Encode ReLU Layer as MILP: $y = \max(0, x)$

$$y \leq x - l \times (1 - a)$$
$$y \geq x$$
$$y \leq u \times a$$
$$y \geq 0$$
$$a \in \{0,1\}$$

Where l and u are lower and upper bounds of the input x already calculated.

3. Encode Pre-Condition $\Phi = B_\infty(x)_\epsilon$

$$x_i - \epsilon \leq x_i' \leq x_i + \epsilon$$

4. Encode Post-Condition Ψ: e.g., label 0 is more likely than label 1: $\Psi = o_0 > o_1$ by finding a counter example.

$$\min o_0 - o_1$$

Finally, we get this MILP instance:

$\min o_0 - o_1$.

Affine and ReLU encodings

$l_j \leq x_j^{(p)} \leq u_i$ and $x_i - \epsilon \leq x_i' \leq x_i + \epsilon$

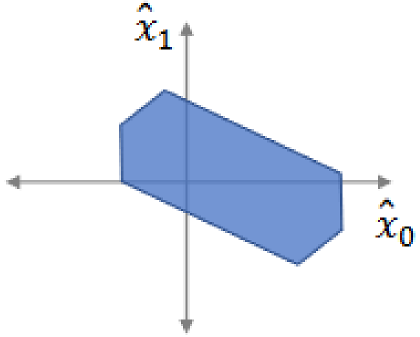
$a_j \in \{0,1\}$.

(See example slide 12)

Lecture 6: Zonotope Convex relaxation

Incomplete method, more precise than box relaxation. Creating abstract neurons:

$\hat{x}_j = a_0^j + \sum_i a_i^j \epsilon_i$ for every neuron $j \in \{1, \dots, d\}$ where $\epsilon \in [-1,1]$ is noise and a is its magnitude. Sharing the same parameters results in more complex and precise shapes than the box.



Example for $d = 2$ and $k = 3$.

Zonotope Affine Transformer:

Multiply by a const C: $\hat{x}_j \times C = (a_0^j + \sum_i a_i^j \epsilon_i) \times C = a_0^j \times C + \sum_i C \times a_i^j \epsilon_i$

Add 2 neurons: $\hat{x}_p + \hat{x}_q = (a_0^p + \sum_i a_i^p \epsilon_i) + (a_0^q + \sum_i a_i^q \epsilon_i) = (a_0^p + a_0^q) + \sum_i (a_i^p + a_i^q) \epsilon_i$

Zonotope ReLU Transformer:

Given $\hat{x} = a_0 + \sum_i a_i \epsilon_i$ calculate $\hat{y} = \max(0, \hat{x})$

1. Compute l_x and u_x by choosing $\epsilon \in \{0,1\}$ depending on the sign of a.
2. Check if the boundaries are on one side of the plane
   a. $u_x \leq 0 \Rightarrow \hat{y} = 0$
   b. $l_x > 0 \Rightarrow \hat{y} = \hat{x}$
   c. Otherwise, cross boundary case.
3. In the case of c., compute a zonotope that encloses ReLU:

$y_1(\hat{x}) = \lambda \hat{x} \leq y(\hat{x}) \leq y_2(\hat{x}) = \lambda \hat{x} - \lambda l_x$ where $\lambda = \frac{u_x}{u_x - l_x}$.

We can an equality by introducing $c \in [0,1]$, $y(\hat{x}) = \lambda \hat{x} - c \lambda l_x$ or $\epsilon_{new} \in [-1,1]$, $c = \frac{\epsilon_{new} - 1}{2}$.

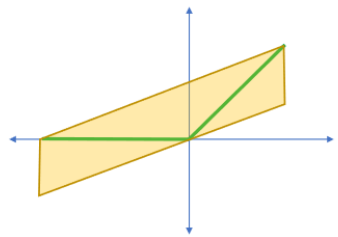
Finally, $y(\hat{x}) = \lambda \hat{x} - \epsilon_{new} \frac{\lambda l_x}{2} - \frac{\lambda l_x}{2}$

$$y\left(a_0 + \sum_{i=1}^{k} a_i \epsilon_i\right) =$$

$$\lambda a_0 + \sum_{i=1}^{k} \lambda a_i \epsilon_i - \epsilon_{new} \frac{\lambda l_x}{2} - \frac{\lambda l_x}{2} =$$

$$b_0 + \sum_{i=1}^{k+1} b_i \epsilon_i$$

Where $b_0 = \lambda a_0 - \frac{\lambda l_x}{2}$, $b_i = \lambda a_i$ and $b_{k+1} = -\frac{\lambda l_x}{2}$



Zonotope transformer

Zonotope is precise on affine (≠Box) and loses precision on ReLU.

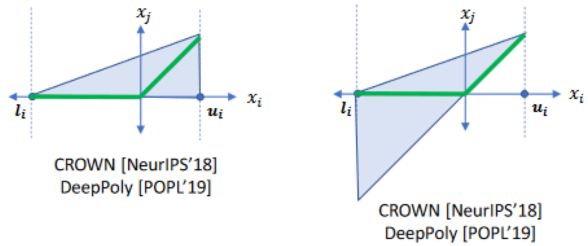
Lecture 7: DeepPoly Relaxation

For each x_i we keep interval constraint l_i and u_i. And two relational constraints $x_i \in [a_i^{\leq}, a_i^{\geq}]$ where $a_i = \sum_j w_j x_j + v$.

How to capture ReLU activation?

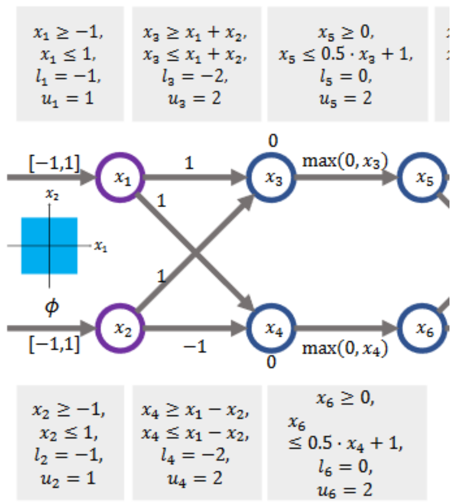
$x_j = \max(0, x_i)$:

- $u_i \leq 0 \Rightarrow a_j^{\leq} = a_j^{\geq} = 0, l_j = u_j = 0$.
- $u_i \geq 0 \Rightarrow a_j^{\leq} = a_j^{\geq} = x_i, l_j = l_i, u_j = u_i$
- $l_i < 0$ and $u_i > 0$: crossing ReLU



The shape of DeepPoly is chosen depending on area (heuristic).

Example:



Backsubstitution: we do not use l_i and u_i to calculate l_j and u_j. We use all the previous constraint instead (e.g., $x_3 \geq x_1 + x_2$, etc.).

Soundness: $F(\gamma(z)) = F(x) \subseteq \gamma\left(F^{\#}(z)\right)$

Exactness: $F(\gamma(z)) = \gamma\left(F^{\#}(z)\right)$

Optimality: $\forall z, \forall F'. \gamma(F'(z)) \not\subset \gamma(F_{\text{best}}(z))$

$\gamma(z)$: the concrete values of an abstract element.

F: concrete transformer.

$F^{\#}$: abstract transformer.

Lecture 8: Certified Defenses

Find a point z in the output shape that maximizes the loss.

Find θ

Minimize $\rho(\theta)$

Where $\rho(\theta) =$

$$E_{(x,y)\sim D}\left[\max_{z \in \gamma\left(\mathrm{NN}^{\#}(S(x))\right)} L(\theta, x', y)\right]$$

γ is applied to concretize the values of the shape.

Loss function: $L(z, y) = \max_{q \neq y}(z_q - z_y)$

Using Box relaxation scales to large networks but introduces a lot of infeasible points.

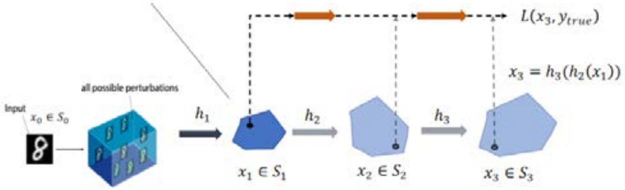
More complex relaxations do not lead to better results.

Adversarial Training: Good accuracy, Easier optimization.

Certified Defense: Good verifiability.

How to combine both?

COLT: find $x_1 \in S_1$ (the abstract output of the first layer) that maximizes the loss function (the worst case) and find $\theta_2, \ldots, \theta_l$ that minimize this loss. Then we push S_1, freeze h_2 and redo the previous steps.



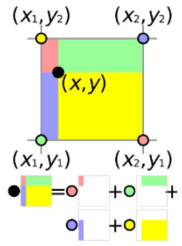
Lecture 9: Certified Robustness to Geometric Transformations

Beyond L_p perturbations, e.g., image geometric transformations: rotation, translation, scaling.

Bijective function $T_\kappa: \mathbb{R}^2 \to \mathbb{R}^2$

Computing pixel values after transformation

1. Compute the preimage of (x, y)
2. Interpolate the resulting coordinate: I



$I_\kappa(x, y) = I \circ T_\kappa^{-1}(x, y): \mathbb{R}^2 \to \mathbb{R}$ is a function that takes coordinates in the resulting image and return the pixel value.

Certifying geometric robustness

Given an original image O, make sure that the network correctly classifies all the $T_\kappa(O)$.

Example: $R_\Phi(O)$ would be the region of all rotated O for $\Phi \in [-30,30]$.

We will be interested in a convex relaxation $C(R_\Phi(O))$. How to represent it?

Find a tight and sound lower and upper bound constraint for each pixel.

$$\boldsymbol{w}_l^T \boldsymbol{\kappa} + b_l \leq I_\kappa(x, y) \leq \boldsymbol{w}_u^T \boldsymbol{\kappa} + b_u$$

For all $\boldsymbol{\kappa}$ in parameter space D. In the rotation example, $\boldsymbol{\kappa}$ is the rotation angle.

Calculate tightness and approximate it for N samples of $\boldsymbol{\kappa}$:

Step 1:

$$L(w_l, b_l) = \int \left(I_\kappa(x, y) - \left(\boldsymbol{w}_l^T \boldsymbol{\kappa} + b_l\right)\right) d\boldsymbol{\kappa}$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} \left(I_{\kappa^i} - \left(\boldsymbol{w}_l^T \boldsymbol{\kappa}^i + b_l\right)\right)$$

$$U(w_u, b_u) = \int \left(\left(\boldsymbol{w}_u^T \boldsymbol{\kappa} + b_u\right) - I_\kappa(x, y)\right) d\boldsymbol{\kappa}$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} \left(\left(\boldsymbol{w}_u^T \boldsymbol{\kappa}^i + b_u\right) - I_{\kappa^i}\right)$$

Step 2:

$$\boldsymbol{w}_l^T \boldsymbol{\kappa}^i + b_l \leq I_{\kappa^i}(x, y) \leq \boldsymbol{w}_u^T \boldsymbol{\kappa}^i + b_u, \forall i \in \{1, \ldots, N\}$$

This can be sound for a finite number of samples but not for all the values of κ. So, we must shift the lower (upper) bound by $-\delta_l$ $(+\delta_u)$ to make the constraint cover all the possible values.

$$\left(\widehat{\boldsymbol{w}}_l^T \boldsymbol{\kappa} + \hat{b}_l\right) - I_\kappa(x, y) \leq \delta_l, \forall \kappa \in D$$

Where $\widehat{w}_l = w_l$ and $b_l = \hat{b}_l - \delta_l$

To find δ_l, we calculate an upper bound of $f(\kappa) = \left(\widehat{\boldsymbol{w}}_l^T \boldsymbol{\kappa} + \hat{b}_l\right) - I_\kappa(x, y)$.

<u>Option 1</u>

Run box propagation (or other relaxations) to bound f in $[u, l]$. So, $f(\kappa) \leq u, \forall \kappa \in D$.

<u>Option 2</u>

Apply mean-value theorem

$$f(\boldsymbol{\kappa}) = f(\boldsymbol{\kappa}_c) + \nabla f(\boldsymbol{\kappa}')^T(\boldsymbol{\kappa} - \boldsymbol{\kappa}_c)$$
$$\leq f(\boldsymbol{\kappa}_c) + |\boldsymbol{L}|^T(\boldsymbol{\kappa} - \boldsymbol{\kappa}_c)$$
$$\leq f\left(\frac{1}{2}(\boldsymbol{h}_u + \boldsymbol{h}_l)\right) + \frac{1}{2}|\boldsymbol{L}|^T(\boldsymbol{h}_u - \boldsymbol{h}_l)$$

Where $|\partial_i f(\kappa_i)| \leq |L_i|, \forall \kappa' \in D$ (by box prop.) $\boldsymbol{\kappa}_c = \frac{1}{2}(\boldsymbol{h}_u + \boldsymbol{h}_l)$ is the center point of $D = [\boldsymbol{h}_l, \boldsymbol{h}_u]$.

Lecture 10: Visualization

Feature Visualization by Optimization

Find x

Maximize $\text{score}(x) - \sum \lambda_i R_i(x)$

Where $\text{score}(x) = \text{mean}(\text{layer}_n[x, y, z])$

Gradient Based Feature Attribution

Calculate $\frac{\partial logit_t(x)}{\partial x}$: the contribution of each pixel to the classification result

Shapley Values: calculate the contribution of each feature i.

$$C_i = \sum_{S \subseteq P \setminus \{i\}} \frac{|S|!\,(|P| - |S| - 1)!}{|P|!}[f(S \cup \{i\}) - f(S)]$$

Where P is the set of features.

Lecture 11: Combining Logic and Deep Learning

Adversarial examples are a special case of a query.

<u>Declaratively</u>: impose constraints (kind of logic) on queried inputs.

<u>Operationally</u>: a way to perform queries to the network with these constraints.

Querying the network

Use standard logic: quantifiers, functions, variables, etc.

$$\Phi = \bigwedge_j NN(i)[j] < NN(i)[9]$$
$$\wedge \|i - \text{deer}\|_\infty < 25$$
$$\wedge \|i - \text{deer}\|_\infty > 5$$

Goal: find i that satisfies Φ.

Solve as optimization: find a translation T such that $T(\Phi)$ is a differentiable loss function. So, if x satisfies Φ, then $T(\Phi)(x) = 0$.

Examples:

- $T(t_1 \leq t_2) = \max(0, t_1 - t_2)$.
- $T(t_1 \neq t_2) = [t_1 = t_2]$.
- $T(t_1 = t_2) = T(t_1 \leq t_2 \wedge t_2 \leq t_1)$
- $T(\phi \wedge \psi) = T(\phi) + T(\psi)$.
- $T(\phi \vee \psi) = T(\phi) \cdot T(\psi)$.

Training the Network with Background Knowledge

Supervised Learning with constraints

$$\forall z \in L_\infty(x, \epsilon), y = \text{car} \Rightarrow NN(z)[\text{truck}] > NN(z)[\text{dog}] + \delta$$

This slightly decreases the network accuracy but significantly increases the constraint accuracy.

Semi-Supervised Training

1. Train a base classifier $\widehat{\Theta}$ on labeled data.
2. Infer the labels with $\widehat{\Theta}$ for unlabeled data.
3. Use adversarial training to get robust Θ.

Problem Statement

Find θ

Maximize $\rho(\theta)$

Where $\rho(\theta) = E_{s\sim D}[\forall z, \Phi(z, s, \theta)]$

<u>Step 1: Rephrasing</u>

Find θ

Minimize $\rho(\theta)$

Where $\rho(\theta) = E_{s\sim D}\left[\max_z \neg\Phi(z, s, \theta)\right]$

⇒ Find parameters such that the maximum violation is minimized.

Step 2: Translation

Find θ

Minimize $\rho(\theta)$

Where $\rho(\theta) = E_{s\sim D}[T(\Phi)(z_{\text{worst}}, s, \theta)]$

And $z_{\text{worst}} = \arg\min_z\left(T(\neg\Phi)(z, s, \theta)\right)$

⇒ Find the worst-case counter example z_{worst} and minimize the violation at this point.

Example

$$\Phi(z, x, \theta) = \|x - z\|_\infty \leq \epsilon \Rightarrow NN_\theta(z)[3] > \delta$$
$$\Phi(z, x, \theta) = \neg\|x - z\|_\infty \leq \epsilon \vee NN_\theta(z)[3] > \delta$$
$$\neg\Phi(z, x, \theta) = \|x - z\|_\infty \leq \epsilon \wedge NN_\theta(z)[3] \leq \delta$$
$$L(z, x, \theta) = \max(0, \|x - z\|_\infty - \epsilon) + \max(0, NN_\theta(z)[3] - \delta)$$

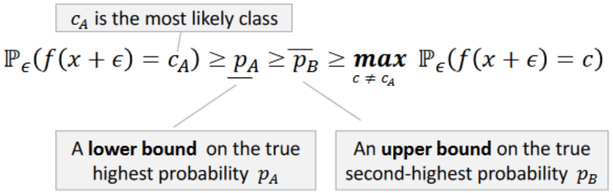
Solve $\max(0, NN_\theta(z)[3] - \delta)$ using PGD while projecting to $L_\infty(x, \epsilon)$ ball.

Lecture 12: Randomized Smoothing for Robustness

From an existing classifier $f: \mathbb{R}^d \to \mathcal{Y}$, construct a classifier g having statistical robustness guarantees.

$$g(x) = \arg\max_{c \in \mathcal{Y}} \mathbb{P}_\epsilon(f(x+\epsilon) = c)$$

e.g., what is the most likely label of $x + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{1})$.



Robustness Guarantee

$$g(x+\delta) = c_A, \forall \|\delta\|_2 < R$$

Where the certification radius

$$R = \frac{\sigma}{2}\left(\Phi^{-1}\left(\underline{p_A}\right) - \Phi^{-1}(\overline{p_B})\right)$$

and Φ^{-1} is the inverse of the standard Gaussian CDF.

FYI: $\mathbb{P}(x \leq v) = p \Rightarrow \Phi^{-1}(p) = v$.

<u>Certified Accuracy</u>: matching the test sample label AND $R \geq T$ where T is a target radius.

<u>Certification</u>

```
function CERTIFY(f, σ, x, n0, n, α)
    counts0 ← SAMPLEUNDERNOISE(f, x, n0, σ)
    ĉA ← top index in counts0
    counts ← SAMPLEUNDERNOISE(f, x, n, σ)
    pA ← LOWERCONFBOUND(counts[ĉA], n, 1 − α)
    if pA > 1/2 return prediction ĉA and radius σ Φ^{-1}(pA)
    else return ABSTAIN
```

When noise σ is increased, the standard accuracy decreases but the certified robust radius increases

<u>Inference</u>

```
function PREDICT(f, σ, x, n, α)
    counts ← SAMPLEUNDERNOISE(f, x, n, σ)
    ĉA, ĉB ← top two indices in counts
    nA, nB ← counts[ĉA], counts[ĉB]
    if BINOMPVALUE(nA, nA + nB, 0.5) ≤ α return ĉA
    else return ABSTAIN
```