Last week: accuracy does not imply robustness

Theorem: *A classifier with constant positive error on the uniform Hamming cube has an error within $O(\sqrt{n})$ bit flips of almost every input.*

Guarantees a nearby misclassified point, but not necessarily a nearby point where the classifier's prediction changes.

Example: suppose the correct label is $y(\mathbf{x}) = x_1$, but $h(\mathbf{x}) = 0$ for every $\mathbf{x}$. The classifier has 50% error. Start with $x_1 = 0$, so classifier is correct. Now flip the first bit:

- The true label changes from 0 to 1, but the classifier's prediction stays 0.
- The new input is misclassified, but the classifier's prediction has not changed.

The theorem finds an error, not a sensitivity attack. The classifier is perfectly stable, but often incorrect. Geometric *existence*, not an efficient attack.

Example: three bit flips,

$$\mathbf{x} = (0,1,0,1,1,0), \quad \mathbf{x}' = (1,1,1,1,0,0), \quad \mathbf{v} = \mathbf{x}' - \mathbf{x} = (1,0,1,0,-1,0).$$

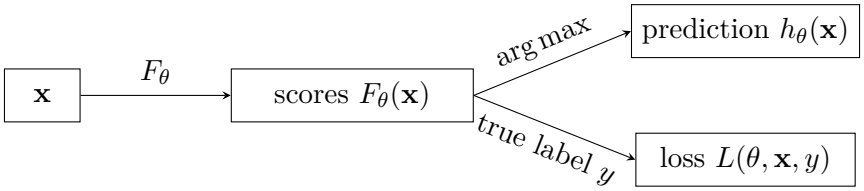
$$\|\mathbf{v}\|_0 = \|\mathbf{v}\|_1 = 3, \qquad \|\mathbf{v}\|_2 = \sqrt{3}, \qquad \|\mathbf{v}\|_\infty = 1.$$

- In general, d bit flips give $\|\mathbf{v}\|_0 = \|\mathbf{v}\|_1 = d$, $\|\mathbf{v}\|_2 = \sqrt{d}$, $\|\mathbf{v}\|_\infty = 1$ (for $d \geq 1$).
- Real inputs: Flipping highest bit of 10000000 gives 0; flipping its lowest bit gives 129.
- Related concentration for other product distributions.

Gradient attacks

Let $F_\theta(\mathbf{x}) \in \mathbb{R}^{|Y|}$ be the vector of class scores. The classifier predicts

$$h_\theta(\mathbf{x}) = \arg\max_{y \in Y} F_{\theta,y}(\mathbf{x}).$$



Example loss (cross-entropy): $L(\theta, \mathbf{x}, y) = \log\left(\sum_{k \in Y} e^{F_{\theta,k}(\mathbf{x})}\right) - F_{\theta,y}(\mathbf{x})$. For small $\boldsymbol{\delta}$,

$$L(\theta, \mathbf{x} + \boldsymbol{\delta}, y) \approx L(\theta, \mathbf{x}, y) + \nabla_{\mathbf{x}} L(\theta, \mathbf{x}, y)^\top \boldsymbol{\delta}.$$

Training and attacking move in opposite directions:

$$\text{Training (fix } \mathbf{x}\text{):} \quad \theta \leftarrow \theta - \eta \nabla_\theta L(\theta, \mathbf{x}, y),$$
$$\text{Attack (fix } \theta\text{):} \quad \mathbf{x}' \leftarrow \mathbf{x}' + \eta \nabla_{\mathbf{x}'} L(\theta, \mathbf{x}', y).$$

Attacking *increases* the loss. The attacker must keep $\mathbf{x}' \in \Delta(\mathbf{x})$. Writing $\mathbf{g} = \nabla_{\mathbf{x}} L(\theta, \mathbf{x}, y)$, for a norm-ball budget $\|\boldsymbol{\delta}\|_p \leq \varepsilon$, maximizing the linear term gives:

$$\ell_\infty: \quad \boldsymbol{\delta} = \varepsilon \operatorname{sign}(\mathbf{g}), \qquad \max \mathbf{g}^\top \boldsymbol{\delta} = \varepsilon \|\mathbf{g}\|_1,$$
$$\ell_2: \quad \boldsymbol{\delta} = \varepsilon \mathbf{g} / \|\mathbf{g}\|_2, \qquad \max \mathbf{g}^\top \boldsymbol{\delta} = \varepsilon \|\mathbf{g}\|_2.$$

- Zeroth order: Uses loss values, not gradients; often needs many queries.
- First order: Uses gradients; cheap and effective (the practical standard).
- Second order: Uses curvature too; computationally expensive.

Example: consider an image whose true label is cat:

	Cat score	Dog score	Prediction
Original	9	1	cat
First perturbation	6	4	cat
Second perturbation	3	7	dog

- First perturbation: The loss increases because the classifier becomes less confident in the correct label. It still predicts cat, so the attack has not succeeded.
- Second perturbation: The prediction changes to dog. If the image is still really a cat, this is a successful sensitivity attack (provided the perturbation is allowed).

If picture turns into dog, changing prediction would be appropriate, not an error.

With *white-box* access, gradients of the loss can guide the attack; with *black-box* access, one may instead estimate them from queries or transfer an attack from another model.

Adversarial training

Learn against the worst allowed perturbations.

$$\text{Ordinary:} \quad \min_{\theta} \ \mathbb{E}_{(\mathbf{x},y)\sim D} L(\theta, \mathbf{x}, y),$$

$$\text{Adversarial:} \quad \min_{\theta} \ \mathbb{E}_{(\mathbf{x},y)\sim D} \left[\max_{\boldsymbol{\delta}:\, \mathbf{x}+\boldsymbol{\delta}\in\Delta(\mathbf{x})} L(\theta, \mathbf{x}+\boldsymbol{\delta}, y) \right].$$

Adversarial training may come at the cost of lower accuracy.

Robustness should be evaluated on fresh labeled hold-out examples. On some ImageNet models at ℓ_∞ budget $4/255$, attack success drops $\sim 90\% \to \sim 25\%$ after adversarial training.

Certificates

A certificate for x guarantees that every allowed perturbation preserves the prediction.

Definition 1 (Robustness certificate)**.** *Fix a classifier $h : X \to Y$ and a perturbation rule Δ with $\Delta(x) \subseteq X$. An efficiently computable predicate $C_{h,\Delta} : X \to \{0, 1\}$ is a robustness certificate if $C_{h,\Delta}(x) = 1$ implies $h(x') = h(x)$ for every $x' \in \Delta(x)$.*

A certificate guarantees unchanged predictions, not correctness, namely $h(x) = y(x)$.

There are three possibilities:

- $C_{h,\Delta}(x) = 1$: Robustness is guaranteed.
- $C_{h,\Delta}(x) = 0$, and classifier is robust: The certificate failed to establish robustness.
- $C_{h,\Delta}(x) = 0$, and classifier is not robust: The certificate correctly declined to certify.

The certificate cannot distinguish the last two cases.

- Aim for certificate to accept many correctly classified inputs.
- Inputs may remain uncertified even if robust.

To interpret robustness as correctness, perturbations must preserve the true label: $y(x') = y(x)$ for $x' \in \Delta(x)$. Nearby inputs need not have the same true label.

Example: start with x correctly classified as a cat. Nearby x' might be misclassified as:

- Still a cat, predicted dog: prediction changes; no certificate if $x' \in \Delta(x)$.
- Now a dog, predicted cat: the prediction is unchanged. Does not prevent certification.

Nearby error does not necessarily mean change in prediction. Only the first case rules out certification when $x' \in \Delta(x)$.

Randomized smoothing

Add Gaussian noise and classify many noisy copies $\mathbf{x}+\mathbf{z}$, with $\mathbf{z} \sim N(0, \sigma^2 I)$. The *smoothed classifier* returns the most probable label:

$$g(\mathbf{x}) = \arg\max_{y \in Y} \Pr_{\mathbf{z}}[h(\mathbf{x}+\mathbf{z}) = y].$$

Randomized smoothing certifies that

$$g(\mathbf{x}+\boldsymbol{\delta}) = g(\mathbf{x}) \qquad \text{whenever } \|\boldsymbol{\delta}\|_2 \leq r.$$

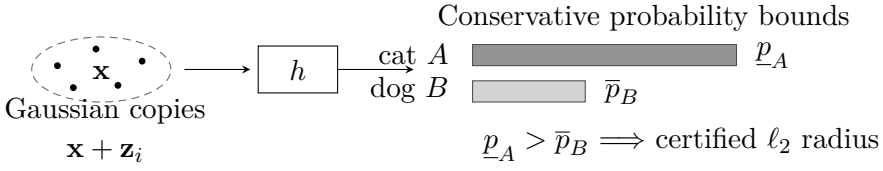
Write $p_k = \Pr_{\mathbf{z}}[h(\mathbf{x}+\mathbf{z}) = k]$. Suppose cat A is most likely and dog B is its strongest competitor. We estimate these probabilities by sampling:

- $\underline{p}_A$: at least this much probability goes to cat.
- $\overline{p}_B$: at most this much probability goes to dog.

Example: Classify 1,000 noisy copies:

Prediction	Count	Estimated probability
Cat	700	0.70
Dog	200	0.20
Other	100	0.10

The observed frequencies estimate p_A and p_B; conservative bounds $\underline{p}_A$ and $\overline{p}_B$ account for sampling error. At 99% confidence, the sampling procedure produces valid probability bounds in at least 99% of repeated runs.



The radius r limits the attacker's perturbation $\boldsymbol{\delta}$, not the Gaussian noise $\mathbf{z}$.

Bound propagation

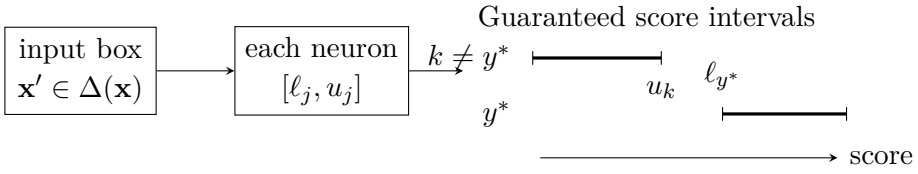
For the classifier $h_\theta(\mathbf{x}) = \arg\max_k F_{\theta,k}(\mathbf{x})$, instead of sampling, consider all $\mathbf{x}' \in \Delta(\mathbf{x})$, for example an ℓ_∞ box. Propagate guaranteed lower and upper bounds through each layer, ultimately obtaining bounds on each output score:

$$\ell_k \le F_{\theta,k}(\mathbf{x}') \le u_k \qquad \text{for every } \mathbf{x}' \in \Delta(\mathbf{x}).$$

If $y^* = h_\theta(\mathbf{x})$ and its lowest possible score exceeds the highest possible competing score, *no* allowed input can change the prediction:

$$\ell_{y^*} > \max_{k \neq y^*} u_k \quad \Longrightarrow \quad h_\theta(\mathbf{x}') = y^* \text{ for all } \mathbf{x}' \in \Delta(\mathbf{x}).$$

The certificate outputs $C_{h_\theta,\Delta}(\mathbf{x}) = 1$ when the score intervals satisfy this inequality; otherwise 0 (inconclusive).



- Separated score intervals certify unchanged predictions throughout the input box.
- Overlapping intervals mean no proof of robustness, not necessarily an attack.
- Certified ℓ_∞ radii are often small; no positive bound follows from n alone.

Computational robustness

Cryptographic idea: restrict A to run in time at most t, and succeed w.p. $< \epsilon$.

- *Computational certification:* for every time-t attacker A,

$$\Pr_{(x,y)\sim D}\Big[C_{h,\Delta}(x) = 1 \ \wedge \ A(x) \text{ succeeds} \mid h(x) = y\Big] \le \epsilon.$$

Could accept an input even when a successful sensitivity perturbation exists in $\Delta(x)$, provided time-t attacker has small success probability.

May permit a larger $\Delta(x)$ or certification of more inputs.

Theorem 1 (Signatures)**.** *Assuming secure signatures, there is a computationally robust classifier achieving the best attainable classification error.*

Proof. Let h_0 be an optimal-error classifier. Before the attacker acts, a trusted signer computes $\sigma = \mathrm{Sign}_{sk}(\mathbf{x})$. The new classifier verifies $(\mathbf{x}, \sigma)$, rejects invalid signatures, and otherwise outputs $h_0(\mathbf{x})$. If the attacker changes an accepted input to a different accepted $\mathbf{x}'$, it forges a signature. On honestly signed inputs, the classification error is unchanged. □

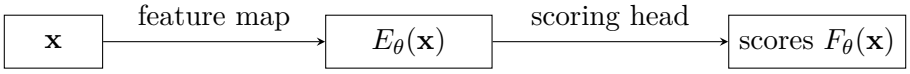
- Applies only to changes *after* trusted signing.
- Signatures and rejection change the classification interface.
- Robustness in the unmodified setting remains *open*.

Invariance attacks

Definition 2 (Invariance attack)**.** *Write $y(x)$ for the correct label of x. An invariance attack A is successful on a correctly classified input x if $y(A(x)) \neq y(x)$ and $h(A(x)) = h(x)$.*

- Imposes no local distance constraint.
- True label changes, but prediction stays fixed.

Verifying such an attack requires semantic knowledge of the correct labels.



Starting from oppositely labeled inputs $\mathbf{x}$ and $\mathbf{z}$, search near $\mathbf{z}$ for a point $\mathbf{u}$ whose features $E_\theta(\mathbf{u})$ resemble those of $\mathbf{x}$:

$$\min_{\mathbf{u}} \|E_\theta(\mathbf{u}) - E_\theta(\mathbf{x})\|_2^2 + \lambda\|\mathbf{u} - \mathbf{z}\|_2^2.$$

- The second term keeps $\mathbf{u}$ close to $\mathbf{z}$.
- The first contracts its features towards $E_\theta(\mathbf{x})$.
- $y(\mathbf{u}) \neq y(\mathbf{x})$. Although $\mathbf{u}$ is near $\mathbf{z}$, $y(\mathbf{u})$ need not equal $y(\mathbf{z})$.
- Hopefully $h(\mathbf{u}) = h(\mathbf{x})$ but nearby feature vectors might classify differently.

Fixed versus adaptive random projections

Can a random projection prevent far inputs that are close in feature space?

Lemma 1 (Johnson–Lindenstrauss)**.** *For a* fixed *finite $S \subset \mathbb{R}^n$, independent of normalized Gaussian $\mathbf{A} \in \mathbb{R}^{m\times n}$, with $m = O(\varepsilon^{-2}\log|S|)$, w.h.p. for all $\mathbf{x}, \mathbf{x}' \in S$,*

$$(1-\varepsilon)\|\mathbf{x} - \mathbf{x}'\|_2 \leq \|\mathbf{A}\mathbf{x} - \mathbf{A}\mathbf{x}'\|_2 \leq (1+\varepsilon)\|\mathbf{x} - \mathbf{x}'\|_2.$$

Applies to a fixed set chosen *before* the random matrix. When $m < n$, linear algebra finds a nonzero $\mathbf{v} \in \ker(\mathbf{A})$, so two inputs can have identical projections:

$$\mathbf{A}(\mathbf{x} + \mathbf{v}) = \mathbf{A}\mathbf{x}.$$

Scaling $\mathbf{v}$ produces collisions at arbitrarily large input distance.

Adaptively robust projections

Restrict inputs to the bounded integer hypergrid $\mathbf{x} \in ([-B, B] \cap \mathbb{Z})^n$ and ε-round $\mathbf{A}\mathbf{x}$:

$$H_{\mathbf{A}}(\mathbf{x}) = \lfloor \mathbf{A}\mathbf{x} \rceil_\varepsilon, \qquad \lfloor s \rceil_\varepsilon := \varepsilon \lfloor s/\varepsilon \rceil .$$

A strong contraction would produce a nonzero bounded integer difference vector; differences of input coordinates are bounded by $2B$.

Definition 3 (Contracting hypergrid vector (CHV) problem)**.** *Given* $\mathbf{A} \in \mathbb{R}^{m \times n}$ *with independent* $\mathcal{N}(0,1)$ *entries, find a nonzero* $\mathbf{v} \in ([-B, B] \cap \mathbb{Z})^n$ *such that*

$$\|\mathbf{A}\mathbf{v}\|_2 < \kappa\sqrt{m}\,\|\mathbf{v}\|_2.$$

- For a fixed $\mathbf{v}$, the typical projected length is $\|\mathbf{A}\mathbf{v}\|_2 \asymp \sqrt{m}\,\|\mathbf{v}\|_2$.
- CHV asks for an unusually large contraction, with $\mathbf{v}$ chosen *after seeing* $\mathbf{A}$.

Theorem 2. CLWE $\leq$ CHV.

Proof. Suppose S is a poly-time solver for CHV. Given a CLWE instance $(\mathbf{A}, \mathbf{b})$, distinguish:

- Planted: $\mathbf{b} = \gamma\mathbf{A}\mathbf{w} + \mathbf{e} \pmod 1$ for a hidden unit vector $\mathbf{w}$; assume $|e_i| < \beta$.
- Null: $\mathbf{b}$ is uniform in $[0,1)^m$, independent of $\mathbf{A}$.

Run S on $\mathbf{A}^\top$ to get a nonzero integer vector $\mathbf{v} \in ([-B, B] \cap \mathbb{Z})^m$ with

$$\|\mathbf{A}^\top\mathbf{v}\|_2 < \kappa\sqrt{n}\,\|\mathbf{v}\|_2.$$

Guess *planted* iff

$$\min_{k \in \mathbb{Z}} |\mathbf{b}^\top\mathbf{v} - k| < \frac{1}{4}.$$

- Null: $\mathbf{b}^\top\mathbf{v} \bmod 1$ is uniform. The test accepts with probability $1/2$.
- Planted: Since $\mathbf{v}$ is integer,

$$\mathbf{b}^\top\mathbf{v} = \gamma\mathbf{w}^\top\mathbf{A}^\top\mathbf{v} + \mathbf{e}^\top\mathbf{v} \pmod 1.$$

  The first term is at most $\gamma\kappa B\sqrt{nm}$ in magnitude; the noise term is at most βBm. Choose parameters with $\gamma\kappa B\sqrt{nm} + \beta Bm < 1/4$; then the test accepts.

Thus CHV solves CLWE distinguishing with constant advantage. □

Towards robust classification

Idea: sample and freeze $\mathbf{A}$; train g on projected features:

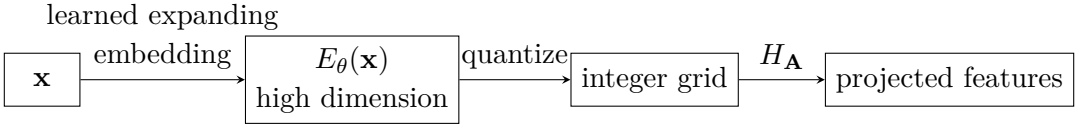
$$h(\mathbf{x}) = g(H_{\mathbf{A}}(\mathbf{x})).$$

For $H_{\mathbf{A}}$, finding a large contraction would solve CHV so gradient cannot use this route.

Yet this is possible:

$$y(\mathbf{x}) \neq y(\mathbf{x}'), \qquad \|H_{\mathbf{A}}(\mathbf{x}) - H_{\mathbf{A}}(\mathbf{x}')\|_2 \text{ large},$$
$$g(H_{\mathbf{A}}(\mathbf{x})) = g(H_{\mathbf{A}}(\mathbf{x}')).$$

Idea: learn E_θ that places same-label inputs close and different-label inputs far apart.



The extra dimensions give room to separate semantic classes.