Certifiable Robustness

A. M. Sadeghzadeh, Ph.D.

Sharif University of Technology
Computer Engineering Department (CE)
Data and Network Security Lab (DNSL)

October 30, 2024

Today's Agenda

Randomized Smoothing

Randomized Smoothing

Certified Adversarial Robustness via Randomized Smoothing

Jeremy Cohen [1] **Elan Rosenfeld** [1] **J. Zico Kolter** [1 2]

Certifiable Robustness

A classifier is **certifiably robust** if for any input x, **there exist a guarantee** that the **classifier's prediction is constant within some set around** x, often an ℓ_2 or ℓ_∞ ball.

Certifiable Robustness

A classifier is **certifiably robust** if for any input x, **there exist a guarantee** that the **classifier's prediction is constant within some set around** x, often an ℓ_2 or ℓ_∞ ball.

- Classifier $f : \mathbb{R}^d \to [0,1]^K$ is **ϵ-robust** at x, if

$$\forall \, \|\delta\|_p \leq \epsilon, \quad \underset{i \in [K]}{argmax} \; f_i(x+\delta) = \underset{i \in [K]}{argmax} \; f_i(x)$$

where K is the number of classes.

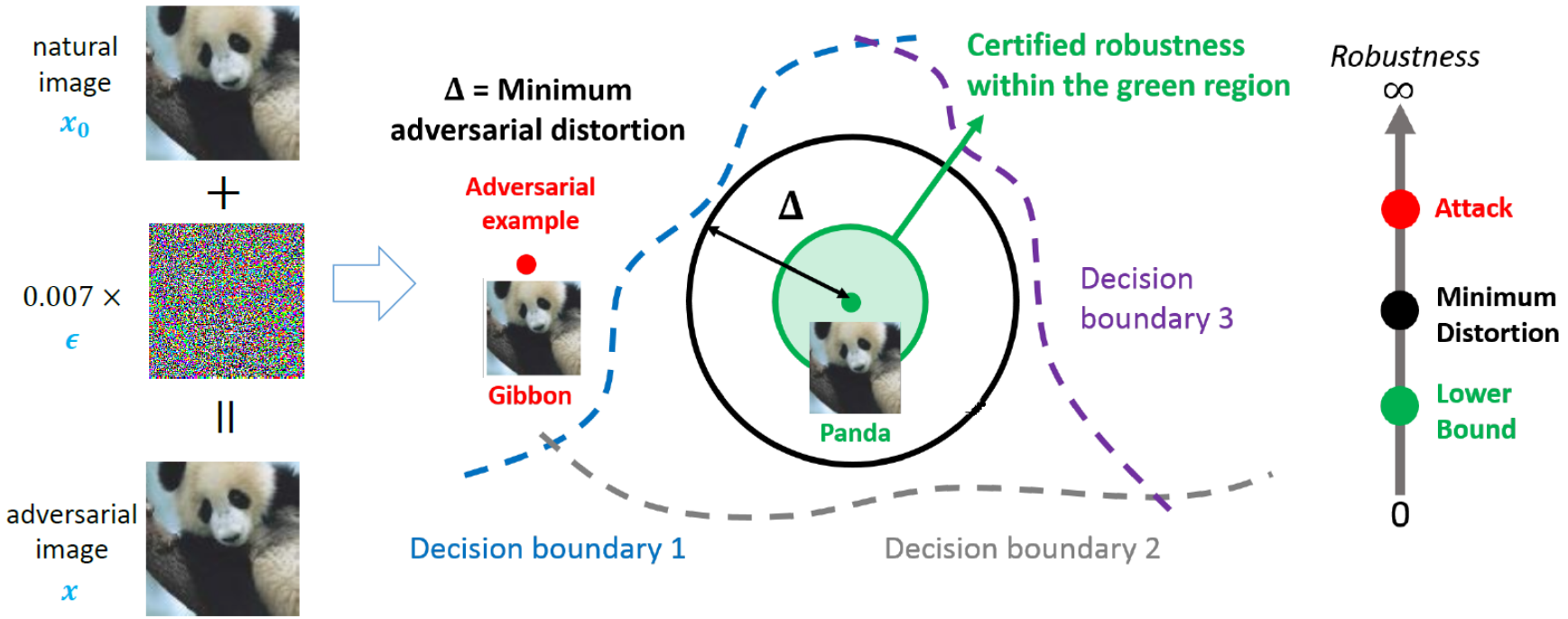
Certifiable Robustness

A classifier is **certifiably robust** if for any input x, **there exist a guarantee** that the **classifier's prediction is constant within some set around** x, often an ℓ_2 or ℓ_∞ ball.

- Classifier $f : \mathbb{R}^d \to [0,1]^K$ is **ϵ-robust** at x, if

$$\forall \, \|\delta\|_p \leq \epsilon, \quad \underset{i \in [K]}{argmax} \; f_i(x+\delta) = \underset{i \in [K]}{argmax} \; f_i(x)$$

where K is the number of classes.



Lipschitz continuity

A function $f : I \to R$ over some set $I \subseteq \mathbb{R}^d$ is called Lipschitz continuous if there exists a positive real constant L such that, for all $x, y \in I$,

$$|f(y) - f(x)| \leq L\|y - x\|_2$$

or

$$f(x) - L\|y - x\|_2 \leq f(y) \leq f(x) + L\|y - x\|_2$$

We call L the Lipschitz constant of f over I.

Lipschitz continuity

A function $f : I \to R$ over some set $I \subseteq \mathbb{R}^d$ is called Lipschitz continuous if there exists a positive real constant L such that, for all $x, y \in I$,

$$|f(y) - f(x)| \leq L\|y - x\|_2$$

or

$$f(x) - L\|y - x\|_2 \leq f(y) \leq f(x) + L\|y - x\|_2$$

We call L the Lipschitz constant of f over I.

Let functions f_1 and f_2 be both Lipschitz continuous with constants L_1 and L_2, the upper Lipschitz constant of their composition $f_1 \circ f_2$ is $L_1 L_2$.

Lipschitz continuity

A function $f : I \to R$ over some set $I \subseteq \mathbb{R}^d$ is called Lipschitz continuous if there exists a positive real constant L such that, for all $x, y \in I$,

$$|f(y) - f(x)| \leq L\|y - x\|_2$$

or

$$f(x) - L\|y - x\|_2 \leq f(y) \leq f(x) + L\|y - x\|_2$$

We call L the Lipschitz constant of f over I.

Let functions f_1 and f_2 be both Lipschitz continuous with constants L_1 and L_2, the upper Lipschitz constant of their composition $f_1 \circ f_2$ is $L_1 L_2$.

$$|f_1(f_2(y)) - f_1(f_2(x))| \leq L_1|f_2(y) - f_2(x)| \leq L_1 L_2\|y - x\|_2$$

Lipschitz continuity

A function $f : I \to R$ over some set $I \subseteq \mathbb{R}^d$ is called Lipschitz continuous if there exists a positive real constant L such that, for all $x, y \in I$,

$$|f(y) - f(x)| \leq L\|y - x\|_2$$

or

$$f(x) - L\|y - x\|_2 \leq f(y) \leq f(x) + L\|y - x\|_2$$

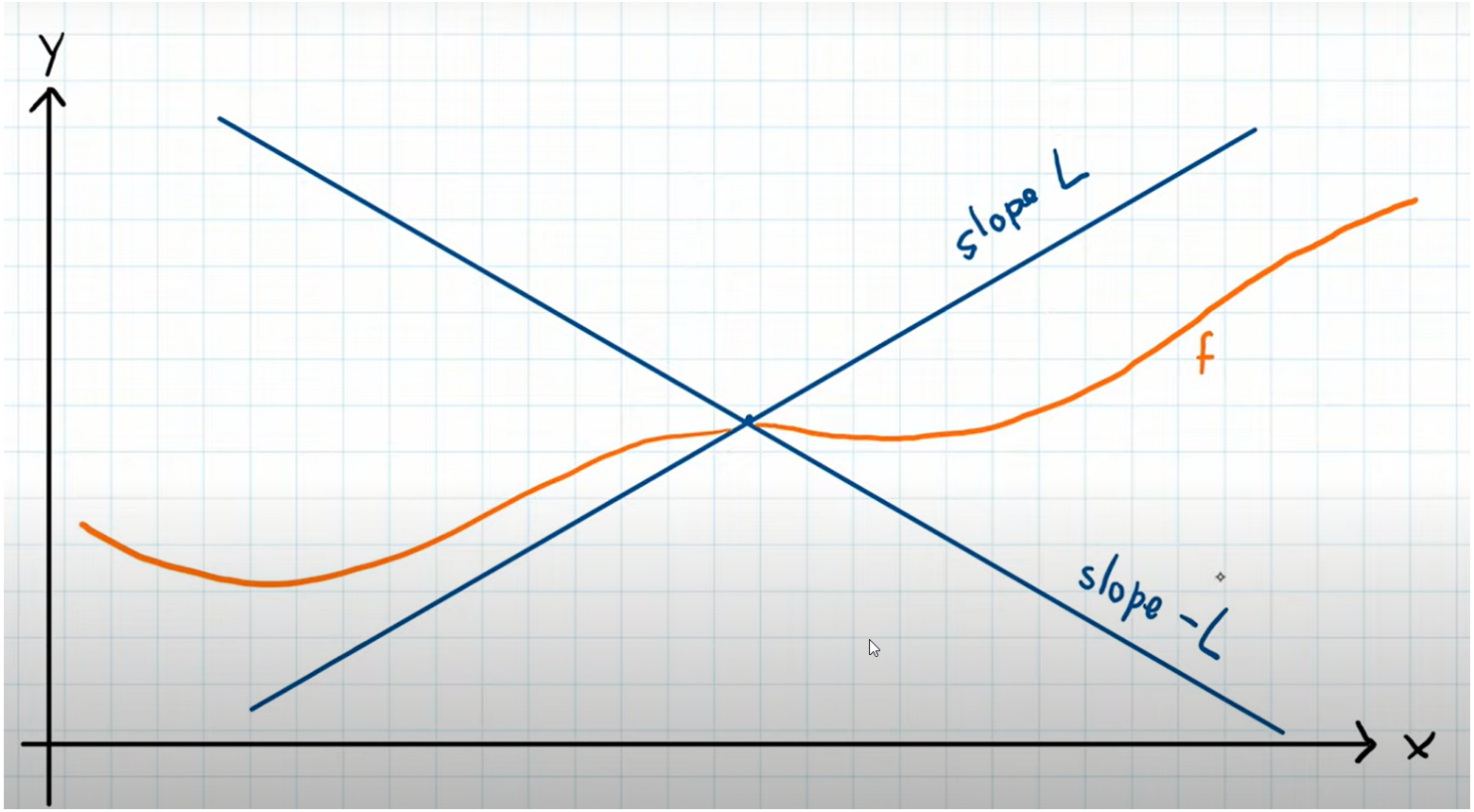
We call L the Lipschitz constant of f over I.

Let functions f_1 and f_2 be both Lipschitz continuous with constants L_1 and L_2, the upper Lipschitz constant of their composition $f_1 \circ f_2$ is $L_1 L_2$.

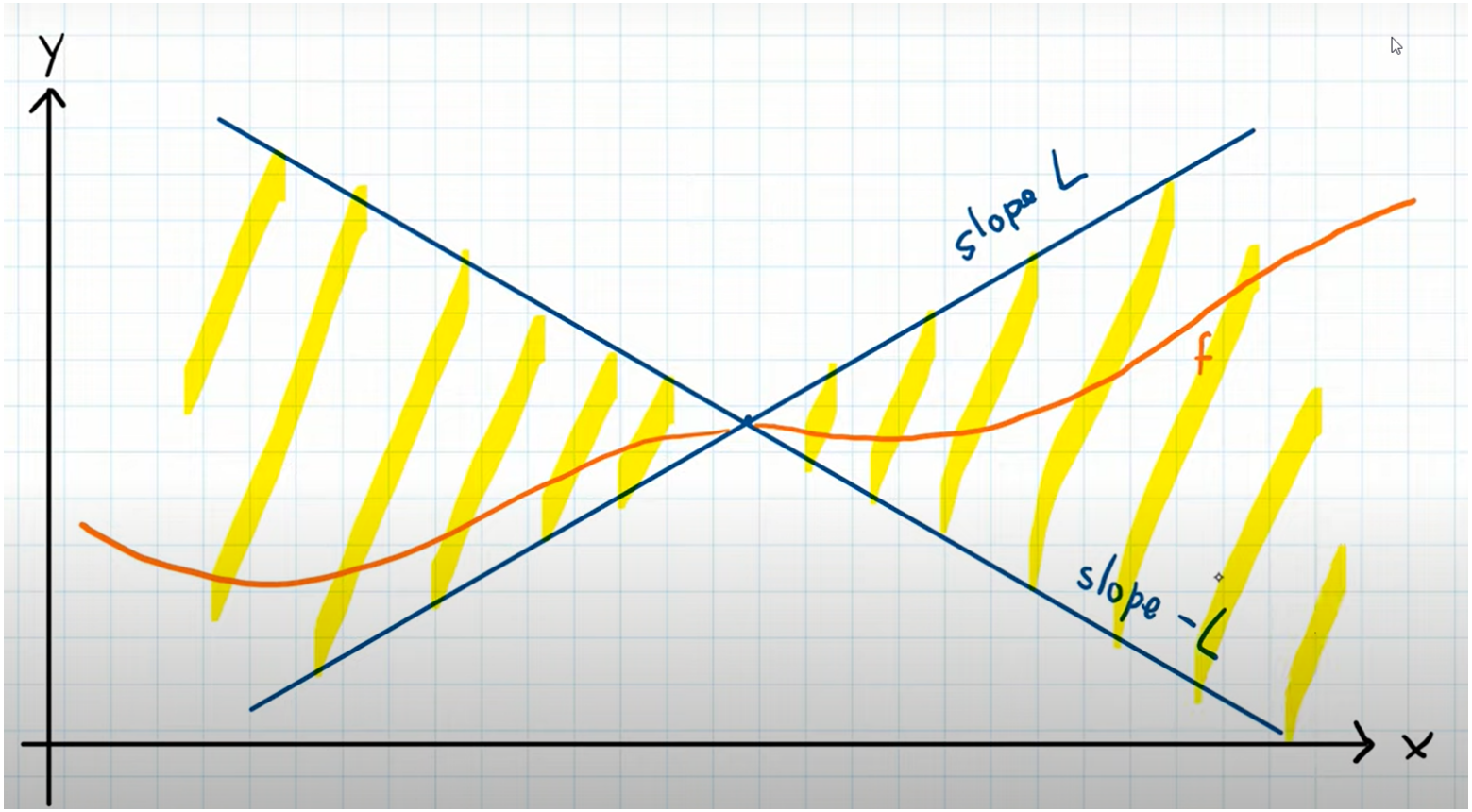
$$|f_1(f_2(y)) - f_1(f_2(x))| \leq L_1|f_2(y) - f_2(x)| \leq L_1 L_2\|y - x\|_2$$

Generally, Let $f = f_1 \circ f_2 \circ .... \circ f_K$ and the Lipschitz constant of f_i be L_i for all $i \in \{1, 2, ..., K\}$, then the Lipschitz constant of f is $L \leq \prod_{k=1}^{K} L_k$.

Lipschitz continuity



Lipschitz continuity



Lipschitz continuity

Let $f : I \rightarrow R$ be a continuous and differntiable function over some set $I \subseteq \mathbb{R}^d$, if we have $\|f'(x)\|_2 \leq m$ for all $x \in I$, then m is the upper Lipschitz constant of f ($L \leq m$).

Lipschitz continuity

Let $f : I \to R$ be a continuous and differntiable function over some set $I \subseteq \mathbb{R}^d$, if we have $\|f'(x)\|_2 \leq m$ for all $x \in I$, then m is the upper Lipschitz constant of f ($L \leq m$).

Proof sketch:
Mean value theorem: Let $f : I \to R$ be a continuous and differntiable function over some set $I \subseteq \mathbb{R}^d$, For all $a, b \in I$ $(b > a)$, there exists some $c \in (a, b)$ such that:

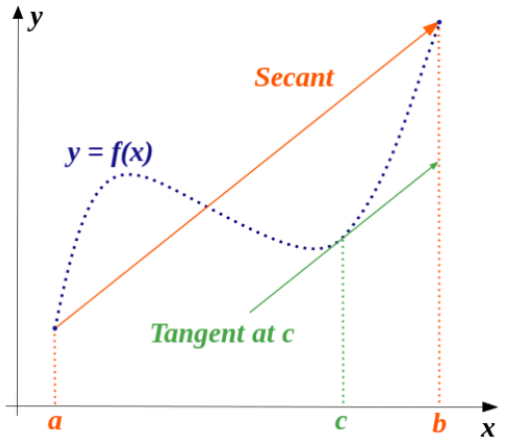
$$f'(c) = \frac{f(b) - f(a)}{b - a}$$

For all $a, b \in I$, there exist $c \in (a, b)$, such that:

$$|f(b) - f(a)| = \|f'(c).b - a\|_2 \leq \|f'(c)\|_2 \|b - a\|_2.$$

Since we know that $\|f'(c)\|_2 \leq m$, we have

$$|f(b) - f(a)| \leq m\|b - a\|_2.$$



Certifiable Robustness for L-Lipschitz classifier

Theorem 0: If $f : \mathbb{R}^d \to [0,1]^K$ is L-lipschitz, then f is ϵ-robust at x with $\epsilon = \frac{1}{2L}(P_A - P_B)$, where $P_A = \max_i f_i(x)$, $P_B = \max_{j \neq i} f_j(x)$, and $f_k(x)$ is the k-th element of the probability vector $f(x)$.

Certifiable Robustness for L-Lipschitz classifier

Theorem 0: If $f : \mathbb{R}^d \to [0,1]^K$ is L-lipschitz, then f is ϵ-robust at x with $\epsilon = \frac{1}{2L}(P_A - P_B)$, where $P_A = \max_i f_i(x)$, $P_B = \max_{j \neq i} f_j(x)$, and $f_k(x)$ is the k-th element of the probability vector $f(x)$.

Proof.

Since f is L-lipschitz, we have

$$\forall x, y \in \mathbb{R}^d, \|f(y) - f(x)\|_2 \leq L\|y - x\|_2$$

Denote $x' = x + \delta$ and assume that $\|\delta\| \leq \epsilon$. we get

$$\|f(x') - f(x)\|_2 \leq L\|x' - x\|_2 \to \|f(x') - f(x)\|_2 \leq L\|\delta\|_2 \leq L\epsilon$$

Hence, P_A can be reduced at most by $L\epsilon$ and P_B can be increased at most by $L\epsilon$. We have ($P'_i = f_i(x')$)

$$P'_A \geq P_A - L\epsilon \quad \text{and} \quad P'_B \leq P_B + L\epsilon$$

Since we want that the label of x' be the same as x, P'_A must be greater than P'_B ($P'_A \geq P'_B$). We have

$$P_A - L\epsilon \geq P_B + L\epsilon \;\to\; 2L\epsilon \leq P_A - P_B \;\to\; \epsilon \leq \frac{1}{2L}(P_A - P_B)$$

□

Smoothed Classifier

If we compute the (upper bound of) Lipschitz constant of the classifier, we can determine the radius (ϵ) of the robustness for each sample.

- However, **the lipschitz constant of deep neural networks** is very large and it **grows exponentially** with the number of layers. Hence, the certification is useless ($\epsilon \to 0$).

Smoothed Classifier

If we compute the (upper bound of) Lipschitz constant of the classifier, we can determine the radius (ϵ) of the robustness for each sample.

- However, **the lipschitz constant of deep neural networks** is very large and it **grows exponentially** with the number of layers. Hence, the certification is useless ($\epsilon \to 0$).

We can **transform classifier f to a smoothed version** in order to bound the Lipschitz constant of the classifier.

Smoothed Classifier

If we compute the (upper bound of) Lipschitz constant of the classifier, we can determine the radius (ϵ) of the robustness for each sample.

- However, **the lipschitz constant of deep neural networks** is very large and it **grows exponentially** with the number of layers. Hence, the certification is useless ($\epsilon \to 0$).

We can **transform classifier f to a smoothed version** in order to bound the Lipschitz constant of the classifier.

- To smooth classifier f, we convolve it with a **Gaussian kernel**.

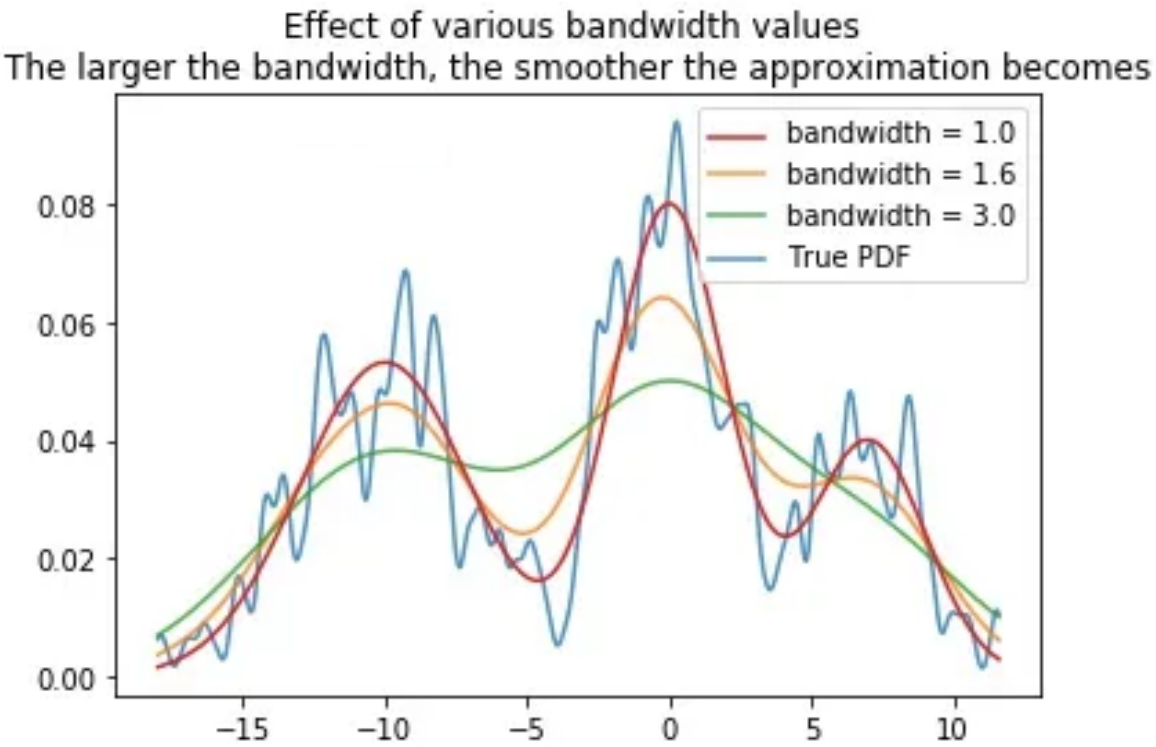
Smoothed Classifier

If we compute the (upper bound of) Lipschitz constant of the classifier, we can determine the radius (ϵ) of the robustness for each sample.

- However, **the lipschitz constant of deep neural networks** is very large and it **grows exponentially** with the number of layers. Hence, the certification is useless ($\epsilon \to 0$).

We can **transform classifier f to a smoothed version** in order to bound the Lipschitz constant of the classifier.

- To smooth classifier f, we convolve it with a **Gaussian kernel**.



[source]

Randomized Smoothing

Consider a classification problem from $\mathbb{R}^d$ to classes $\mathcal{Y}$. Randomized smoothing is a method for constructing a new, **smoothed classifier** $\hat{f}$ from an arbitrary base classifier f.

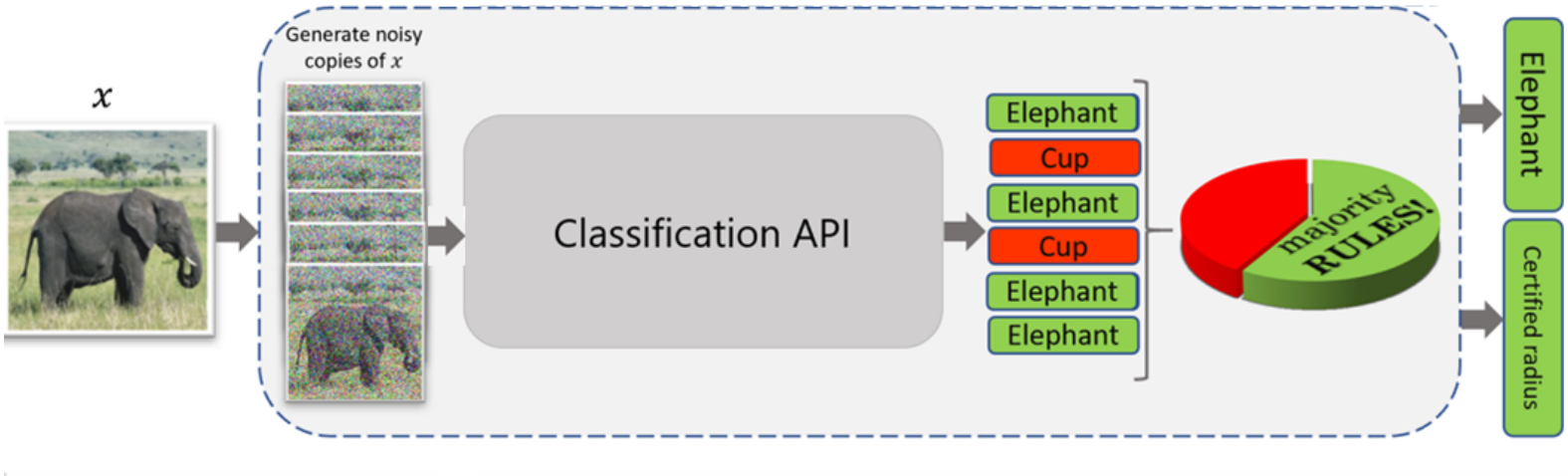
Randomized Smoothing

Consider a classification problem from $\mathbb{R}^d$ to classes $\mathcal{Y}$. Randomized smoothing is a method for constructing a new, **smoothed classifier** $\hat{f}$ from an arbitrary base classifier f.

- When queried at x, the smoothed classifier $\hat{f}$ returns whichever class the base classifier f is most likely to return when x is perturbed by isotropic Gaussian noise:

$$\hat{f}(x) = \underset{c \in \mathcal{Y}}{argmax}\ \mathbb{P}(f(x+\epsilon) = c) \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, \sigma^2 I) \tag{1}$$

  The noise level σ is a hyperparameter of the smoothed classifier $\hat{f}$ which controls a robustness/accuracy tradeoff.



[source]

Notation

Suppose that when the base classifier f classifies $\mathcal{N}(x, \sigma^2 I)$, the most probable class c_A is returned with probability P_A, and the "runner-up" class is returned with probability P_B.

- $\underline{P_A}$ is a lower bound for P_A and $\overline{P_B}$ is a lower bound for P_B.

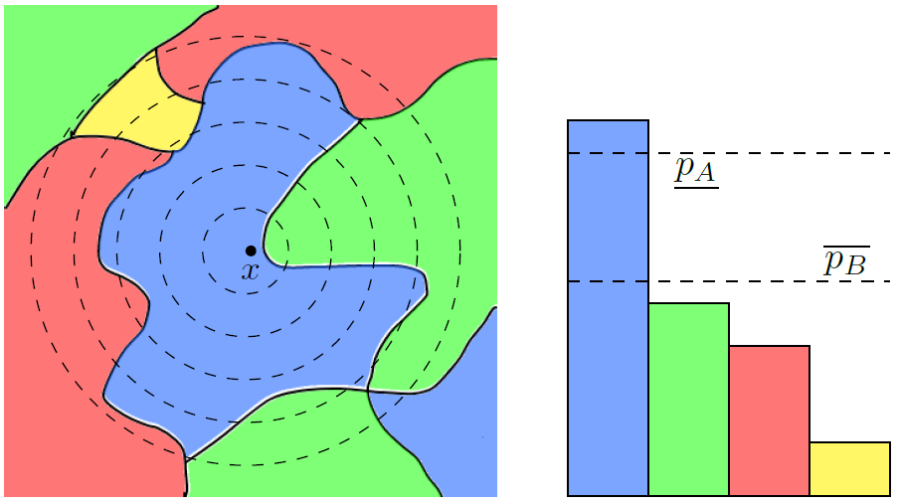


Figure 1. Evaluating the smoothed classifier at an input x. **Left**: the decision regions of the base classifier f are drawn in different colors. The dotted lines are the level sets of the distribution $\mathcal{N}(x, \sigma^2 I)$. **Right**: the distribution $f(\mathcal{N}(x, \sigma^2 I))$. As discussed below, $\underline{p_A}$ is a lower bound on the probability of the top class and $\overline{p_B}$ is an upper bound on the probability of each other class. Here, $g(x)$ is "blue."

Robustness guarantee

Theorem 1. Let $f : \mathbb{R}^d \to \mathcal{Y}$ be any deterministic or random function, and let $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. Let $\hat{f}$ be defined as in (1). Suppose $C_A \in \mathcal{Y}$ and $\underline{P_A}, \overline{P_B} \in [0, 1]$ satisfy:

$$\mathbb{P}(f(x + \epsilon) = C_A) \geq \underline{P_A} \geq \overline{P_B} \geq \max_{C \neq C_A} \mathbb{P}(f(x + \epsilon) = C)$$

Then $\hat{f}(x + \delta) = C_A$ for all $\|\delta\|_2 \leq R$, where

$$R = \frac{\sigma}{2}(\Phi^{-1}(\underline{P_A}) - \Phi^{-1}(\overline{P_B}))$$

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

Robustness guarantee

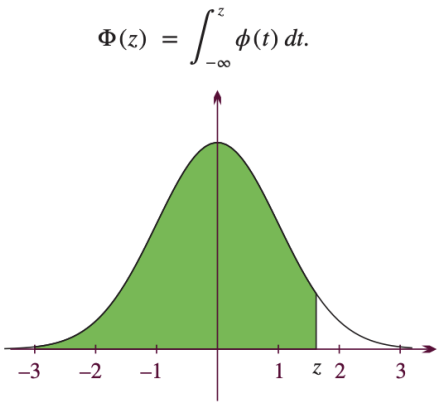
Theorem 1. Let $f : \mathbb{R}^d \to \mathcal{Y}$ be any deterministic or random function, and let $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. Let $\hat{f}$ be defined as in (1). Suppose $C_A \in \mathcal{Y}$ and $\underline{P_A}, \overline{P_B} \in [0, 1]$ satisfy:

$$\mathbb{P}(f(x+\epsilon) = C_A) \geq \underline{P_A} \geq \overline{P_B} \geq \max_{C \neq C_A} \mathbb{P}(f(x+\epsilon) = C)$$

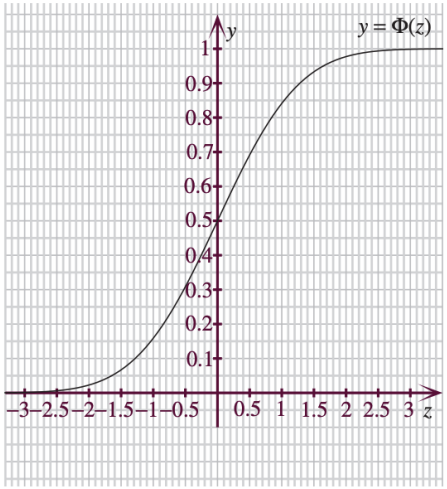
Then $\hat{f}(x + \delta) = C_A$ for all $\|\delta\|_2 \leq R$, where

$$R = \frac{\sigma}{2}(\Phi^{-1}(\underline{P_A}) - \Phi^{-1}(\overline{P_B}))$$

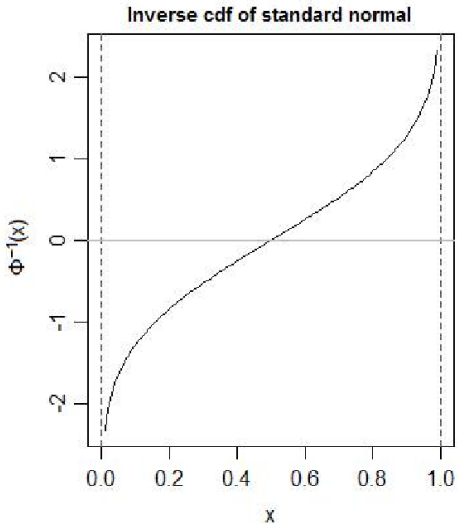
where Φ^{-1} is the inverse of the standard Gaussian CDF.



This is the graph of the standard normal probability density function $\phi(z)$.



This is the graph of the standard normal cumulative distribution function $\Phi(z)$.



Several Observations About Theorem 1

Several observations about theorem 1

- **Theorem 1 assumes nothing about f**. This is crucial since it is unclear which well-behavedness assumptions, if any, are satisfied by modern deep architectures.

Several Observations About Theorem 1

Several observations about theorem 1

- **Theorem 1 assumes nothing about f**. This is crucial since it is unclear which well-behavedness assumptions, if any, are satisfied by modern deep architectures.
- The certified radius R is large when: (1) the noise level σ is high, (2) the probability of the top class C_A is high, and (3) the probability of each other class is low.

Several Observations About Theorem 1

Several observations about theorem 1

- **Theorem 1 assumes nothing about f**. This is crucial since it is unclear which well-behavedness assumptions, if any, are satisfied by modern deep architectures.
- The certified radius R is large when: (1) the noise level σ is high, (2) the probability of the top class C_A is high, and (3) the probability of each other class is low.
- **The certified radius R goes to ∞ as $\underline{P_A} \to 1$ and $\overline{P_B} \to 0$.** This should sound reasonable: the Gaussian distributi is supported on all of $\mathbb{R}^d$, so the only way that $f(x + \epsilon) = C_A$ with probability 1 is if $f = C_A$ almost everywhere.

Lipschitz Constant of Randomized Smoothed Classifier

To prove Theorem 1, we need to find the Lipschitz constant of the smoothed classifier $\hat{f}$.

Lipschitz Constant of Randomized Smoothed Classifier

To prove Theorem 1, we need to find the Lipschitz constant of the smoothed classifier $\hat{f}$.

Let

$$f : \mathbb{R}^d \to [0, 1]$$
$$\hat{f}(x) = \mathbb{E}_{z \sim \mathcal{N}(0, I_d)}[f(x + \sigma z)]$$

It is well-known that $\hat{f}$ is Lipschitz (it has uniform bound on the Lipschitz constant). In practice, we can approximate $\hat{f}$ by empirical average

$$y^{(k)} = \sum_{i=1}^{k} f(x + \sigma z), \quad \text{where } z \sim \mathcal{N}(0, I_d)$$

It can be shown that if $k \to \infty$, $y^{(k)}$ almost surely converges to $\hat{f}$.

Lipschitz Constant of Randomized Smoothed Classifier

Recall: Expected Value

The expectation, or expected value, of some function $f(x)$ with respect to a probability distribution $P_X(x)$ is the average, or mean value, that f takes on when x is drawn from P.

For discrete random variable X, $P_X(x)$ is **Probability Mass Function (PMF)** and expected value can be computed with a summation:

$$\mathbb{E}_{X\sim P}[f(x)] = \sum_x f(x)P_X(x)$$

For continuous random variable X, $P_X(x)$ is **Probability Density Function (PDF)** and expected value is computed with an integral:

$$\mathbb{E}_{X\sim P}[f(x)] = \int f(x)P_X(x)dx$$

Lipschitz Constant of Randomized Smoothed Classifier

Recall

- If $z \sim \mathcal{N}(0, I_d)$, then $\mu + \sigma z \sim \mathcal{N}(\mu, \sigma^2 I_d)$.
- The identity matrix is often denoted by I_n, where n is the dimension. The determinant of the identity matrix is 1.
- The Probability Density Function (PDF) of multivariate normal distribution

$$f_X(x) = \frac{1}{(2\pi)^{d/2} det(\Sigma)^{1/2}} exp\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\}$$

where X is normally distributed random variable with mean $\mu \in \mathbb{R}^d$ and covariance $\Sigma \in \mathbb{R}^{d \times d}$, and d is the dimension of x.

The Cumulative Distribution Function (CDF) of multivariate normal distribution

$$P(X \leq x) = F_X(x) = \int_{-\infty}^{x} f_X(t)dt$$

Lipschitz Constant of Randomized Smoothed Classifier

To prove Theorem 1, we need to find the Lipschitz constant of the smoothed classifier $\hat{f}$.

Let

$$f : \mathbb{R}^d \to [0, 1]$$
$$\hat{f}(x) = \mathbb{E}_{z \sim \mathcal{N}(0, I_d)}[f(x + \sigma z)]$$

It is well-known that $\hat{f}$ is Lipschitz (it has uniform bound on the Lipschitz constant). In practice, we can approximate $\hat{f}$ by empirical average

$$y^{(k)} = \sum_{i=1}^{k} f(x + \sigma z), \quad \text{where } z \sim \mathcal{N}(0, I_d)$$

It can be shown that if $k \to \infty$, $y^{(k)}$ almost surely converges to $\hat{f}$.

We have

$$\hat{f}(x) = \mathbb{E}_{z \sim \mathcal{N}(0, I_d)}[f(x + \sigma z)] = \frac{1}{(2\pi)^{d/2}} \int f(x + \sigma z) \exp\{-\frac{\|z\|_2^2}{2}\} dz$$

$\hat{f}(x)$ is the weighted average of $f(x)$ in the vicinity of x.

Lipschitz Constant of Randomized Smoothed Classifier

We had

$$\hat{f}(x) = \frac{1}{(2\pi)^{d/2}} \int f(x + \sigma z) \exp\{-\frac{\|z\|_2^2}{2}\} dz$$

Change of variable: $w = -\sigma z$.

Lipschitz Constant of Randomized Smoothed Classifier

Recall: change of variable

Double Integral.
Suppose that we want to integrate $f(x, y)$ over the region R. Under the transformation $x = g(u, v)$, $y = h(u, v)$ the region becomes S and the integral becomes

$$\iint_R f(x, y)\, dxdy = \iint_S f(g(u, v), h(u, v)) \left| \frac{\partial(x, y)}{\partial(u, v)} \right| dudv$$

where $\left| \frac{\partial(x,y)}{\partial(u,v)} \right| = \begin{vmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \end{vmatrix}$ is the absolute value of inverse Jacobian determinant of the transformation.

Vector Integral.
Suppose $x, y \in \mathbb{R}^d$ that we want to integrate $f(x)$ over the region R. Under the transformation $y = g(x)$ the region becomes S and the integral becomes

$$\int_R f(x)\, dx = \int_S f(g(x)) \left| \frac{\partial x}{\partial y} \right| dy$$

where $\left| \frac{\partial x}{\partial y} \right|$ is the absolute value of inverse Jacobian determinant of the transformation.

Lipschitz Constant of Randomized Smoothed Classifier

We had

$$\hat{f}(x) = \frac{1}{(2\pi)^d} \int f(x + \sigma z) \exp\{-\frac{\|z\|_2^2}{2}\} dz$$

Change of variable: $w = -\sigma z$. Hence, $dz = \frac{1}{\sigma^d} dw$. We get

$$\hat{f}(x) = \frac{1}{(2\pi)^{d/2}} \int f(x+\sigma z) \exp\{-\frac{\|z\|_2^2}{2}\} dz = \int f(x-w) \frac{1}{(2\pi)^{d/2}\sigma^d} \exp\{-\frac{\|w\|_2^2}{2\sigma^2}\} dw$$

Lipschitz Constant of Randomized Smoothed Classifier

We had

$$\hat{f}(x) = \frac{1}{(2\pi)^d} \int f(x + \sigma z) \exp\{-\frac{\|z\|_2^2}{2}\} dz$$

Change of variable: $w = -\sigma z$. Hence, $dz = \frac{1}{\sigma^d} dw$. We get

$$\hat{f}(x) = \frac{1}{(2\pi)^{d/2}} \int f(x+\sigma z) \exp\{-\frac{\|z\|_2^2}{2}\} dz = \int f(x-w) \underbrace{\frac{1}{(2\pi)^{d/2}\sigma^d} \exp\{-\frac{\|w\|_2^2}{2\sigma^2}\}}_{g_\sigma(w)} dw$$

Lipschitz Constant of Randomized Smoothed Classifier

We had

$$\hat{f}(x) = \frac{1}{(2\pi)^d} \int f(x + \sigma z) \exp\{-\frac{\|z\|_2^2}{2}\} dz$$

Change of variable: $w = -\sigma z$. Hence, $dz = \frac{1}{\sigma^d} dw$. We get

$$\hat{f}(x) = \frac{1}{(2\pi)^{d/2}} \int f(x+\sigma z) \exp\{-\frac{\|z\|_2^2}{2}\} dz = \int f(x-w) \underbrace{\frac{1}{(2\pi)^{d/2}\sigma^d} \exp\{-\frac{\|w\|_2^2}{2\sigma^2}\}}_{g_\sigma(w)} dw$$

Recall: Convolution

Convolution of two functions f and g over a infinite range $(-\infty, +\infty)$ is given by

$$f * g = \int_{-\infty}^{+\infty} f(\tau) g(t - \tau) d\tau = \int_{-\infty}^{+\infty} f(t - \tau) g(\tau) d\tau = g * f$$

where the symbol $[f * g](t)$ denotes convolution of f and g.

Lipschitz Constant of Randomized Smoothed Classifier

We had

$$\hat{f}(x) = \frac{1}{(2\pi)^d} \int f(x + \sigma z) \exp\{-\frac{\|z\|_2^2}{2}\} dz$$

Change of variable: $w = -\sigma z$. Hence, $dz = \frac{1}{\sigma^d} dw$. We get

$$\hat{f}(x) = \frac{1}{(2\pi)^{d/2}} \int f(x+\sigma z) \exp\{-\frac{\|z\|_2^2}{2}\} dz = \int f(x-w) \underbrace{\frac{1}{(2\pi)^{d/2}\sigma^d} \exp\{-\frac{\|w\|_2^2}{2\sigma^2}\}}_{g_\sigma(w)} dw$$

$$\hat{f}(x) = \int f(x-w) g_\sigma(w) dw = \int f(w) g_\sigma(x-w) dw = f * g_\sigma$$

Lipschitz Constant of Randomized Smoothed Classifier

We had

$$\hat{f}(x) = \frac{1}{(2\pi)^d} \int f(x + \sigma z) \exp\{-\frac{\|z\|_2^2}{2}\} dz$$

Change of variable: $w = -\sigma z$. Hence, $dz = \frac{1}{\sigma^d} dw$. We get

$$\hat{f}(x) = \frac{1}{(2\pi)^{d/2}} \int f(x+\sigma z) \exp\{-\frac{\|z\|_2^2}{2}\} dz = \int f(x-w) \underbrace{\frac{1}{(2\pi)^{d/2}\sigma^d} \exp\{-\frac{\|w\|_2^2}{2\sigma^2}\}}_{g_\sigma(w)} dw$$

$$\hat{f}(x) = \int f(x-w) g_\sigma(w) dw = \int f(w) g_\sigma(x-w) dw = f * g_\sigma$$

Recall

Let $f : I \to R$ be a continuous and differentiable function over some set $I \subseteq \mathbb{R}^d$, if we have $\|f'(x)\|_2 \leq m$ for all $x \in I$, then m is the upper Lipschitz constant of f ($L \leq m$).

Hence, We calculate $\|\nabla_x \hat{f}(x)\|_2$ in order to find the **upper bound on Lipschitz constant** of $\hat{f}$.

Lipschitz Constant of Randomized Smoothed Classifier

We had

$$\hat{f}(x) = f * g_\sigma$$

$$\nabla_x \hat{f}(x) = \nabla_x (f * g_\sigma)$$

Lipschitz Constant of Randomized Smoothed Classifier

We had

$$\hat{f}(x) = f * g_\sigma$$
$$\nabla_x \hat{f}(x) = \nabla_x (f * g_\sigma)$$

Generally speaking, since convolution is a linear operator and since only g depends on x, $\nabla_x (f * g_\sigma) = f * \nabla_x g_\sigma$. Therefore

$$\nabla_x \hat{f}(x) = \nabla_x (f * g_\sigma) = f * \nabla_x g_\sigma$$

Lipschitz Constant of Randomized Smoothed Classifier

We had

$$\hat{f}(x) = f * g_\sigma$$
$$\nabla_x \hat{f}(x) = \nabla_x (f * g_\sigma)$$

Generally speaking, since convolution is a linear operator and since only g depends on x, $\nabla_x (f * g_\sigma) = f * \nabla_x g_\sigma$. Therefore

$$\nabla_x \hat{f}(x) = \nabla_x (f * g_\sigma) = f * \nabla_x g_\sigma$$

Thus, we should compute $\nabla_x g_\sigma$

$$\nabla_x g_\sigma = \nabla_x \left(\frac{1}{(2\pi)^{d/2}\sigma^d} \exp\{-\frac{\|x-w\|_2^2}{2\sigma^2}\} \right) = \frac{1}{(2\pi)^{d/2}\sigma^d} \frac{-2(x-w)}{2\sigma^2} \exp\{-\frac{\|x-w\|_2^2}{2\sigma^2}\}$$
$$= \frac{-(x-w)}{\sigma^2} g_\sigma$$

Lipschitz Constant of Randomized Smoothed Classifier

We had

$$\hat{f}(x) = f * g_\sigma$$
$$\nabla_x \hat{f}(x) = \nabla_x (f * g_\sigma)$$

Generally speaking, since convolution is a linear operator and since only g depends on x, $\nabla_x (f * g_\sigma) = f * \nabla_x g_\sigma$. Therefore

$$\nabla_x \hat{f}(x) = \nabla_x (f * g_\sigma) = f * \nabla_x g_\sigma$$

Thus, we should compute $\nabla_x g_\sigma$

$$\nabla_x g_\sigma = \nabla_x \left(\frac{1}{(2\pi)^{d/2}\sigma^d} \exp\{-\frac{\|x-w\|_2^2}{2\sigma^2}\} \right) = \frac{1}{(2\pi)^{d/2}\sigma^d} \frac{-2(x-w)}{2\sigma^2} \exp\{-\frac{\|x-w\|_2^2}{2\sigma^2}\}$$
$$= \frac{-(x-w)}{\sigma^2} g_\sigma$$

We have

$$\nabla_x \hat{f}(x) = \int f(w) \frac{-(x-w)}{\sigma^2} g_\sigma(x-w) dw = \int f(x-w) \frac{-w}{\sigma^2} g_\sigma(w) dw$$

Lipschitz Constant of Randomized Smoothed Classifier

We had

$$\nabla_x \hat{f}(x) = \int f(x-w) \frac{-w}{\sigma^2} g_\sigma(w) dw$$

Lipschitz Constant of Randomized Smoothed Classifier

We had

$$\nabla_x \hat{f}(x) = \int f(x-w)\frac{-w}{\sigma^2} g_\sigma(w) dw$$

Change of variable: $w = -\sigma z$. We get

$$\nabla_x \hat{f}(x) = \int f(x+\sigma z)\frac{\sigma z}{\sigma^2} g_\sigma(-\sigma z)\sigma^d dz = \int f(x+\sigma z)\frac{z}{\sigma}\frac{\sigma^d}{(2\pi)^{d/2}\sigma^d}\exp\{-\frac{\|-\sigma z\|_2^2}{2\sigma^2}\}dz$$

$$= \int f(x+\sigma z)\frac{z}{\sigma}\underbrace{\frac{1}{(2\pi)^{d/2}}\exp\{-\frac{\|z\|_2^2}{2}\}}_{\mathcal{N}(0,I_d)} dz = \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[f(x+\sigma z)\frac{z}{\sigma}]$$

Lipschitz Constant of Randomized Smoothed Classifier

We had

$$\nabla_x \hat{f}(x) = \int f(x-w)\frac{-w}{\sigma^2} g_\sigma(w) dw$$

Change of variable: $w = -\sigma z$. We get

$$\nabla_x \hat{f}(x) = \int f(x+\sigma z)\frac{\sigma z}{\sigma^2} g_\sigma(-\sigma z)\sigma^d dz = \int f(x+\sigma z)\frac{z}{\sigma}\frac{\sigma^d}{(2\pi)^{d/2}\sigma^d}\exp\{-\frac{\|-\sigma z\|_2^2}{2\sigma^2}\}dz$$

$$= \int f(x+\sigma z)\frac{z}{\sigma}\underbrace{\frac{1}{(2\pi)^{d/2}}\exp\{-\frac{\|z\|_2^2}{2}\}}_{\mathcal{N}(0,I_d)} dz = \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[f(x+\sigma z)\frac{z}{\sigma}]$$

Therefore, for Lipschitz constant of $\hat{f}$, we have

$$L_{\hat{f}} \leq \|\nabla_x \hat{f}(x)\|_2 \Rightarrow L_{\hat{f}} \leq \|\mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[f(x+\sigma z)_{\in[0,1]}\frac{z}{\sigma}]\|_2$$

Lipschitz Constant of Randomized Smoothed Classifier

Recal: Triangel inequality

Triangle Inequality. Let $a_k \in \mathbb{R}^d$,

$$|\sum_{k=1}^{N} a_k| \leq \sum_{k=1}^{N} |a_k|.$$

Corollary: For random variable X, if X has a finite expectation, then

$$|\mathbb{E}[X]| \leq \mathbb{E}[|X|]$$

Proof sketch:

$$|\mathbb{E}[X]| = |\sum_{x} x P_X(x)| \underset{\text{Triangle Inequality}}{\leq} \sum_{x} |x| P_X(x) = \mathbb{E}[|X|]$$

Lipschitz Constant of Randomized Smoothed Classifier

We had

$$\nabla_x \hat{f}(x) = \int f(x-w)\frac{-w}{\sigma^2} g_\sigma(w) dw$$

Change of variable: $w = -\sigma z$. We get

$$\nabla_x \hat{f}(x) = \int f(x+\sigma z)\frac{\sigma z}{\sigma^2} g_\sigma(-\sigma z)\sigma^d dz = \int f(x+\sigma z)\frac{z}{\sigma}\frac{\sigma^d}{(2\pi)^{d/2}\sigma^d}\exp\{-\frac{\|-\sigma z\|_2^2}{2\sigma^2}\}dz$$

$$= \int f(x+\sigma z)\frac{z}{\sigma}\underbrace{\frac{1}{(2\pi)^{d/2}}\exp\{-\frac{\|z\|_2^2}{2}\}}_{\mathcal{N}(0,I_d)} dz = \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[f(x+\sigma z)\frac{z}{\sigma}]$$

Therefore, for Lipschitz constant of $\hat{f}$, we have

$$L_{\hat{f}} \leq \|\nabla_x \hat{f}(x)\|_2 \Rightarrow L_{\hat{f}} \leq \|\mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[f(x+\sigma z)\frac{z}{\sigma}]\|_2 \leq \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[\|f(x+\sigma z)\frac{z}{\sigma}\|_2]$$

Lipschitz Constant of Randomized Smoothed Classifier

We had

$$\nabla_x \hat{f}(x) = \int f(x-w)\frac{-w}{\sigma^2} g_\sigma(w) dw$$

Change of variable: $w = -\sigma z$. We get

$$\nabla_x \hat{f}(x) = \int f(x+\sigma z)\frac{\sigma z}{\sigma^2} g_\sigma(-\sigma z)\sigma^d dz = \int f(x+\sigma z)\frac{z}{\sigma}\frac{\sigma^d}{(2\pi)^{d/2}\sigma^d}\exp\{-\frac{\|-\sigma z\|_2^2}{2\sigma^2}\}dz$$

$$= \int f(x+\sigma z)\frac{z}{\sigma}\underbrace{\frac{1}{(2\pi)^{d/2}}\exp\{-\frac{\|z\|_2^2}{2}\}}_{\mathcal{N}(0,I_d)} dz = \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[f(x+\sigma z)\frac{z}{\sigma}]$$

Therefore, for Lipschitz constant of $\hat{f}$, we have

$$L_{\hat{f}} \leq \|\nabla_x \hat{f}(x)\|_2 \Rightarrow L_{\hat{f}} \leq \|\mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[f(x+\sigma z)\frac{z}{\sigma}]\|_2 \leq \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[\|\underbrace{f(x+\sigma z)}_{\in[0,1]}\frac{z}{\sigma}\|_2]$$

Lipschitz Constant of Randomized Smoothed Classifier

We had

$$\nabla_x \hat{f}(x) = \int f(x-w)\frac{-w}{\sigma^2} g_\sigma(w) dw$$

Change of variable: $w = -\sigma z$. We get

$$\nabla_x \hat{f}(x) = \int f(x+\sigma z)\frac{\sigma z}{\sigma^2} g_\sigma(-\sigma z)\sigma^d dz = \int f(x+\sigma z)\frac{z}{\sigma}\frac{\sigma^d}{(2\pi)^{d/2}\sigma^d}\exp\{-\frac{\|-\sigma z\|_2^2}{2\sigma^2}\}dz$$

$$= \int f(x+\sigma z)\frac{z}{\sigma}\underbrace{\frac{1}{(2\pi)^{d/2}}\exp\{-\frac{\|z\|_2^2}{2}\}}_{\mathcal{N}(0,I_d)} dz = \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[f(x+\sigma z)\frac{z}{\sigma}]$$

Therefore, for Lipschitz constant of $\hat{f}$, we have

$$L_{\hat{f}} \leq \|\nabla_x \hat{f}(x)\|_2 \Rightarrow L_{\hat{f}} \leq \|\mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[f(x+\sigma z)\frac{z}{\sigma}]\|_2 \leq \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[\|\underbrace{f(x+\sigma z)}_{\in[0,1]}\frac{z}{\sigma}\|_2]$$

$$\leq \frac{1}{\sigma}\mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[\|z\|_2]$$

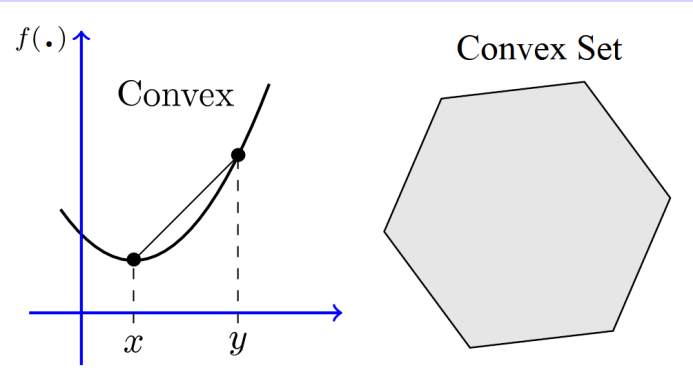
Lipschitz Constant of Randomized Smoothed Classifier

Recall: Convex Function and Jensen's Inequality

Convex set. A set C is convex if the line segment between any two points in C lies in C, i.e., if for any $x_1, x_2 \in C$ and any $\lambda \in [0, 1]$, we have: $\lambda x_1 + (1 - \lambda)x_2 \in C$

Convex Function. Consider a function $f : I \to \mathbb{R}$, where $I \subseteq \mathbb{R}$ is a convex set. We say that f is a convex function if, for any two points x and y in I and any $\lambda \in [0, 1]$, we have

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y).$$



Jensen's Inequality. If $f(x)$ is a convex function on I , and $\mathbb{E}[f(X)]$ and $f(\mathbb{E}[X])$ are finite, then

$$f(E[X]) \leq E[f(X)].$$

Lipschitz Constant of Randomized Smoothed Classifier

We had

$$\nabla_x \hat{f}(x) = \int f(x-w)\frac{-w}{\sigma^2} g_\sigma(w) dw$$

Change of variable: $w = -\sigma z$. We get

$$\nabla_x \hat{f}(x) = \int f(x+\sigma z)\frac{\sigma z}{\sigma^2} g_\sigma(-\sigma z)\sigma^d dz = \int f(x+\sigma z)\frac{z}{\sigma}\frac{\sigma^d}{(2\pi)^{d/2}\sigma^d}\exp\{-\frac{\|-\sigma z\|_2^2}{2\sigma^2}\}dz$$

$$= \int f(x+\sigma z)\frac{z}{\sigma}\underbrace{\frac{1}{(2\pi)^{d/2}}\exp\{-\frac{\|z\|_2^2}{2}\}}_{\mathcal{N}(0,I_d)} dz = \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[f(x+\sigma z)\frac{z}{\sigma}]$$

Therefore, for Lipschitz constant of $\hat{f}$, we have

$$L_{\hat{f}} \leq \|\nabla_x \hat{f}(x)\|_2 \Rightarrow L_{\hat{f}} \leq \|\mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[f(x+\sigma z)\frac{z}{\sigma}]\|_2 \leq \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[\|\underbrace{f(x+\sigma z)}_{\in[0,1]}\frac{z}{\sigma}\|_2]$$

$$\leq \frac{1}{\sigma}\mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[\|z\|_2] \leq \frac{1}{\sigma}\sqrt{\mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[\|z\|_2^2]}$$

Note: Since $f(x) = x^2$ is a convex function, for random variable X, we have

$$f(\mathbb{E}[g(X)]) \underset{\text{Jensen's Inequality}}{\leq} \mathbb{E}[f(g(X))] \Rightarrow \mathbb{E}^2[g(X)] \leq \mathbb{E}[g^2(X)] \Rightarrow \mathbb{E}[g(X)] \leq \sqrt{\mathbb{E}[g^2(X)]}$$

Lipschitz Constant of Randomized Smoothed Classifier

We had

$$L_{\hat{f}} \leq \frac{1}{\sigma}\sqrt{\mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[\|z\|_2^2]}$$

Lipschitz Constant of Randomized Smoothed Classifier

We had

$$L_{\hat{f}} \leq \frac{1}{\sigma}\sqrt{\mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[\|z\|_2^2]}$$

We know

$$\mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[\|z\|_2^2] = \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[z_1^2 + z_2^2 + ... + z_d^2]$$
$$= \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[z_1^2] + \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[z_2^2] + ... + \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[z_d^2] = 1 + 1 + ... + 1 = d$$

Recall: For random variable X, $var(X) = E(X^2) - E[X]^2$

Lipschitz Constant of Randomized Smoothed Classifier

We had

$$L_{\hat{f}} \leq \frac{1}{\sigma}\sqrt{\mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[\|z\|_2^2]}$$

We know

$$\mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[\|z\|_2^2] = \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[z_1^2 + z_2^2 + ... + z_d^2]$$
$$= \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[z_1^2] + \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[z_2^2] + ... + \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[z_d^2] = 1 + 1 + ... + 1 = d$$

Recall: For random variable X, $var(X) = E(X^2) - E[X]^2$

Finally, we have

$$L_{\hat{f}} \leq \frac{\sqrt{d}}{\sigma}$$

However, since the upper bound depends on the dimension of x, it increases as d grows. As $L_{\hat{f}}$ increases, the radios of our certify bound decreases.

Lipschitz Constant of Randomized Smoothed Classifier

We had

$$L_{\hat{f}} \leq \frac{1}{\sigma}\sqrt{\mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[\|z\|_2^2]}$$

We know

$$\mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[\|z\|_2^2] = \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[z_1^2 + z_2^2 + ... + z_d^2]$$
$$= \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[z_1^2] + \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[z_2^2] + ... + \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[z_d^2] = 1 + 1 + ... + 1 = d$$

Recall: For random variable X, $var(X) = E(X^2) - E[X]^2$

Finally, we have

$$L_{\hat{f}} \leq \frac{\sqrt{d}}{\sigma}$$

However, since the upper bound depends on the dimension of x, it increases as d grows. As $L_{\hat{f}}$ increases, the radios of our certify bound decreases.

The main issue is that we calculated the Lipschitz constant working for all x in the input space. We know that the robustness of various inputs is different. Hence, we should compute the **local Lipschitz constant** that depends on specific input x.

Local Lipschitz Constant

Lipschitz Constant

$$\exists L \ \ \forall x, y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$

Local Lipschitz Constant

Lipschitz Constant

$$\exists L \ \ \forall x, y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$

Local Lipschitz Constant

$$\forall x \ \ \exists L_x \ \ \forall y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$

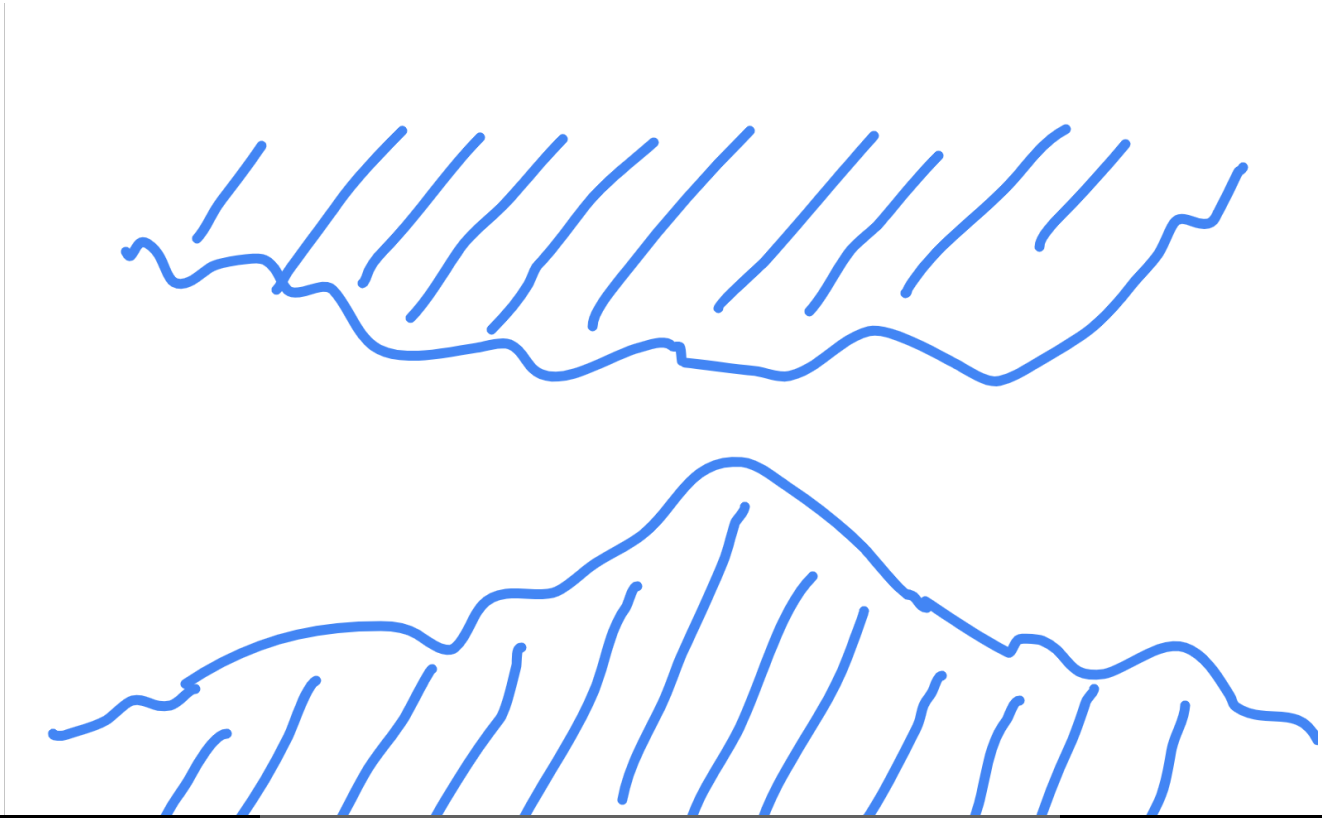
Local Lipschitz Constant

Lipschitz Constant

$$\exists L \ \ \forall x, y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$

Local Lipschitz Constant

$$\forall x \ \ \exists L_x \ \ \forall y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$



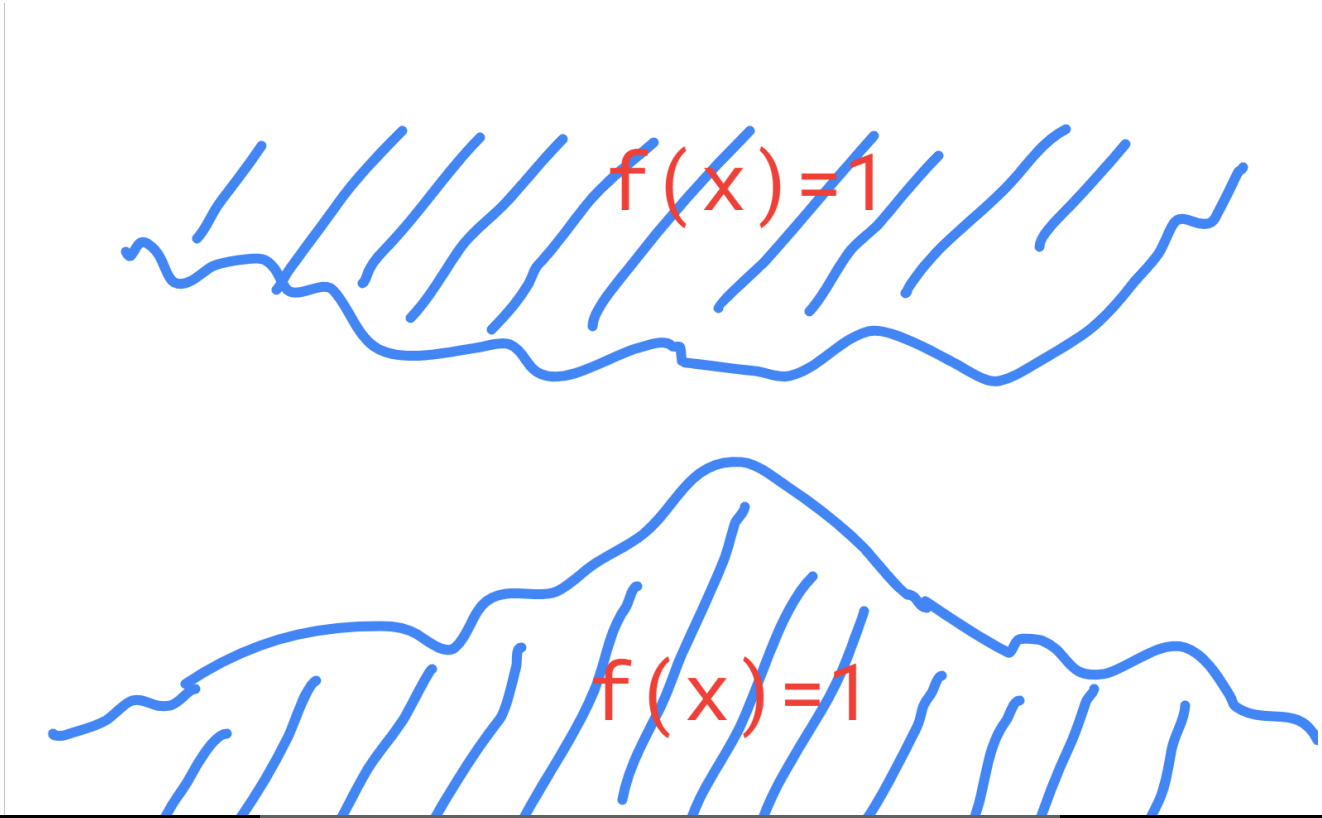
Local Lipschitz Constant

Lipschitz Constant

$$\exists L \ \ \forall x, y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$

Local Lipschitz Constant

$$\forall x \ \ \exists L_x \ \ \forall y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$



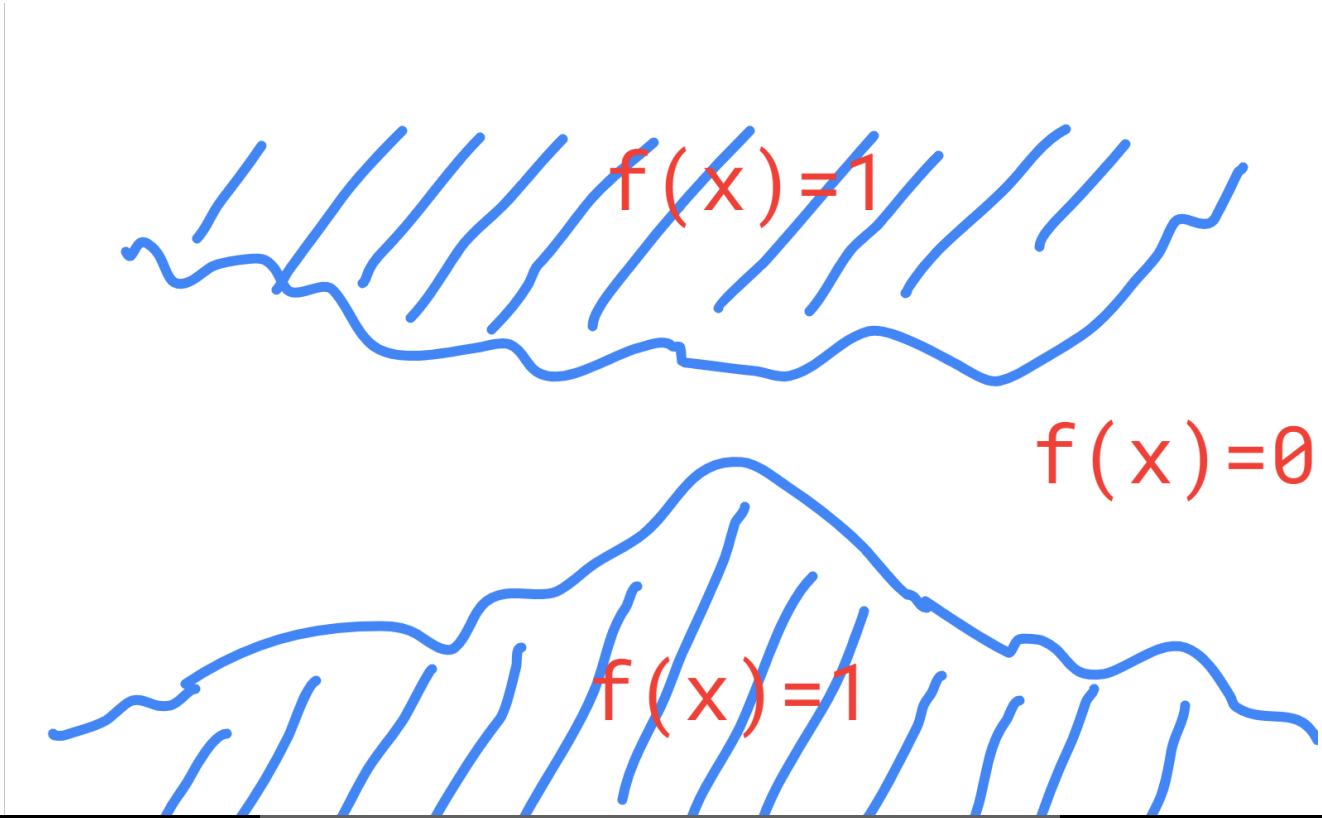
Local Lipschitz Constant

Lipschitz Constant

$$\exists L \ \ \forall x, y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$

Local Lipschitz Constant

$$\forall x \ \ \exists L_x \ \ \forall y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$



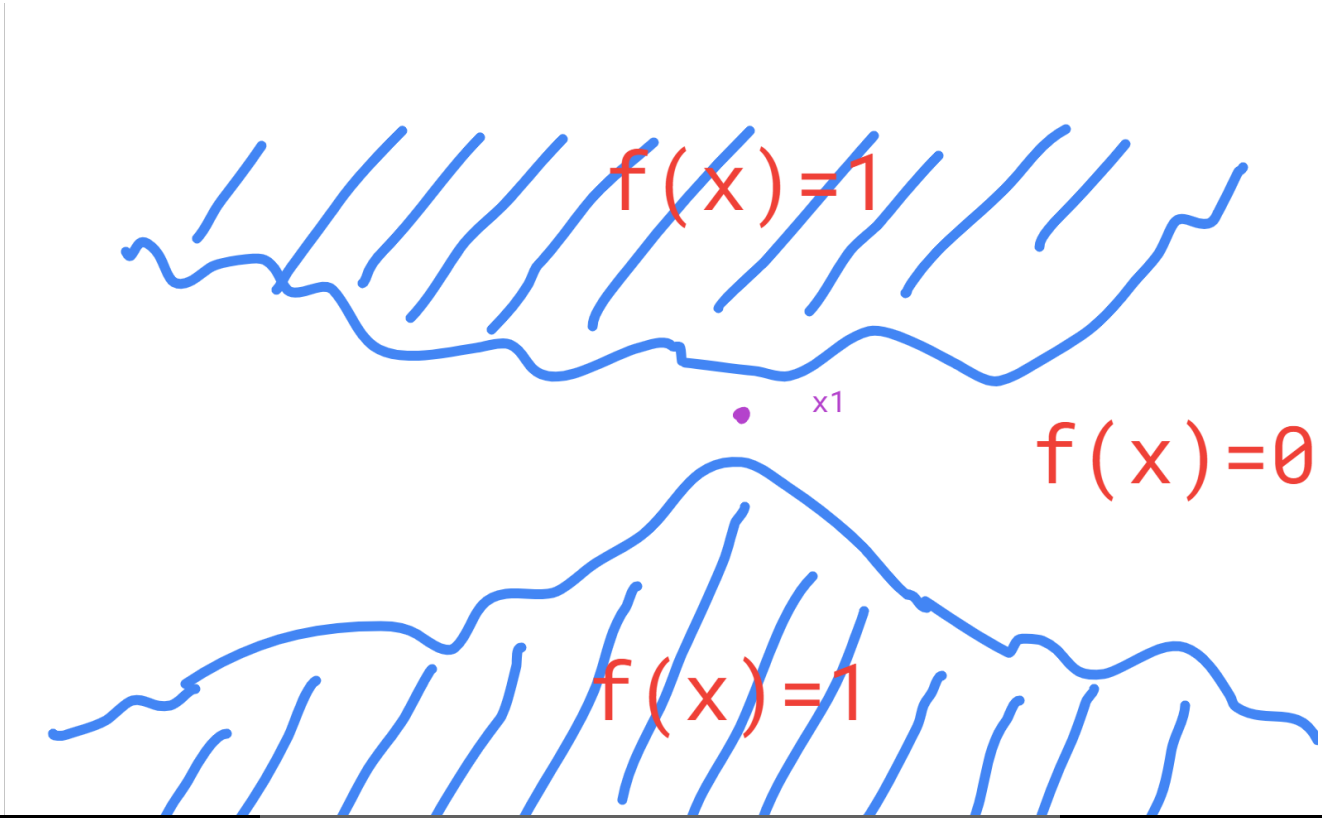
Local Lipschitz Constant

Lipschitz Constant

$$\exists L \ \ \forall x, y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$

Local Lipschitz Constant

$$\forall x \ \ \exists L_x \ \ \forall y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$



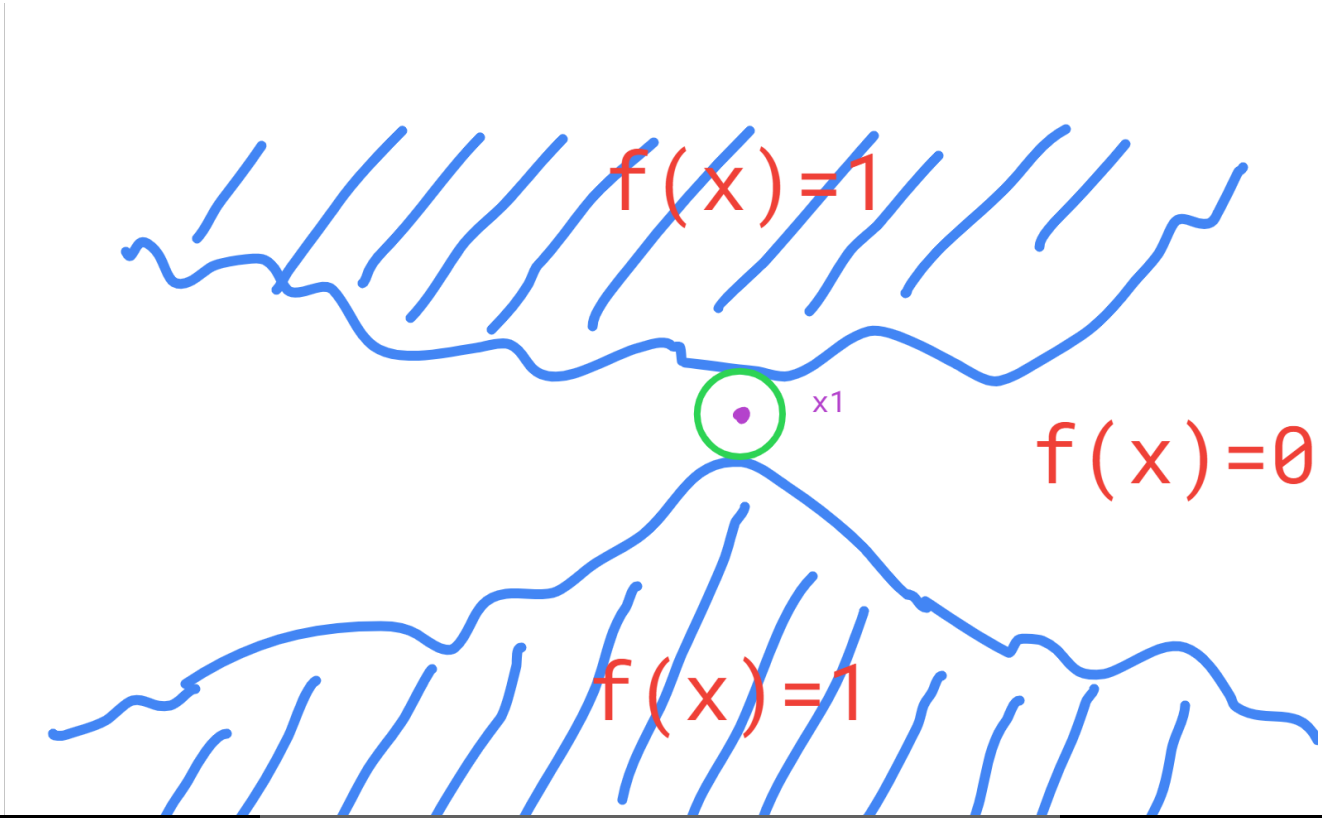
Local Lipschitz Constant

Lipschitz Constant

$$\exists L \ \ \forall x, y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$

Local Lipschitz Constant

$$\forall x \ \ \exists L_x \ \ \forall y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$



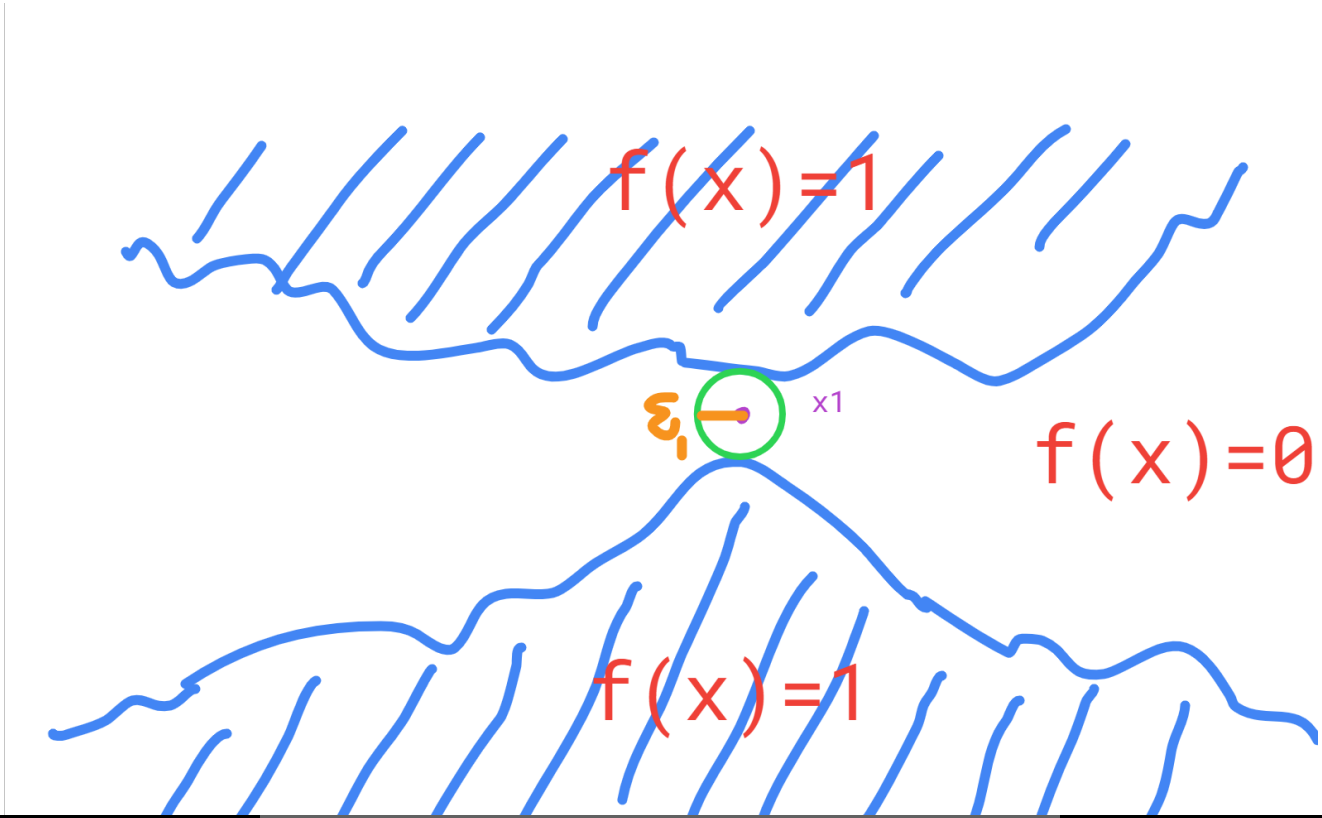
Local Lipschitz Constant

Lipschitz Constant

$$\exists L \ \ \forall x, y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$

Local Lipschitz Constant

$$\forall x \ \ \exists L_x \ \ \forall y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$



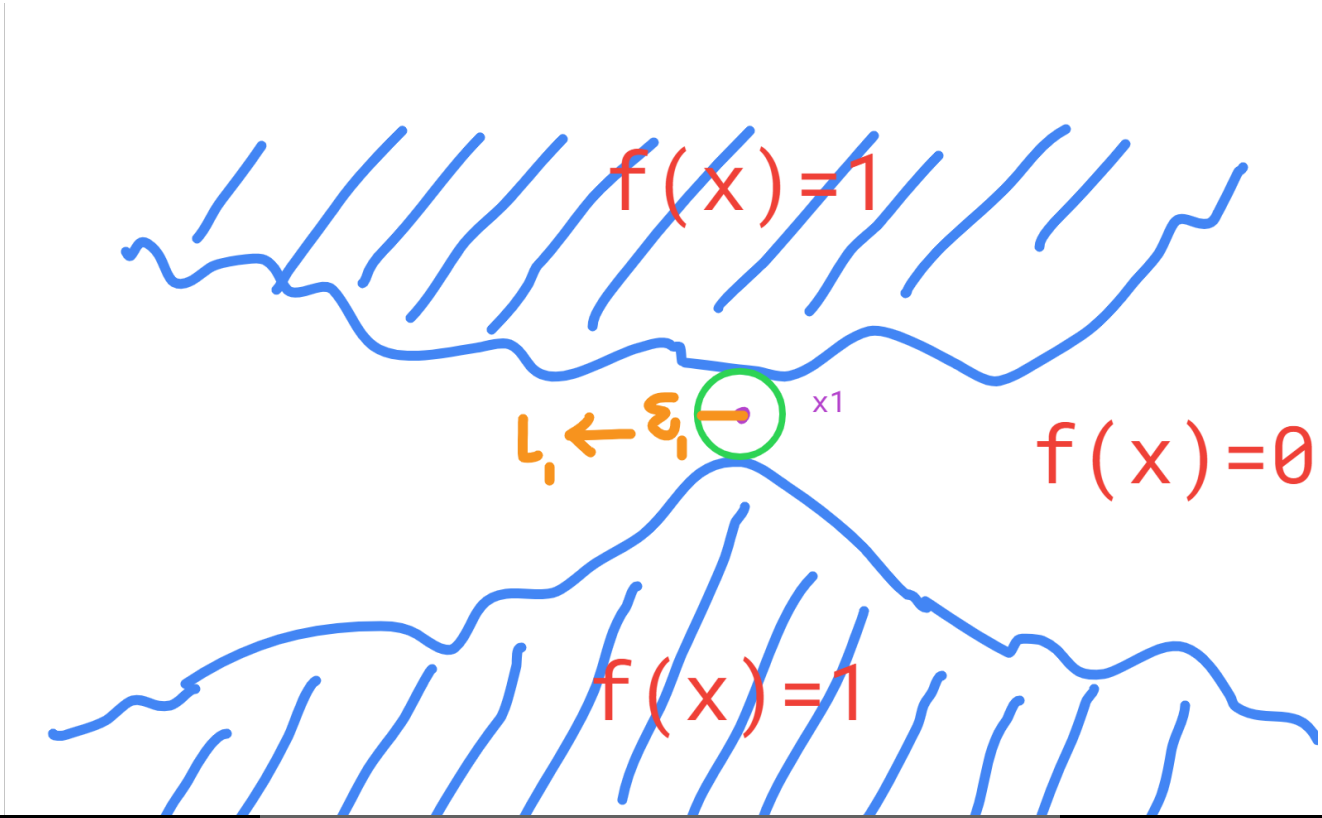
Local Lipschitz Constant

Lipschitz Constant

$$\exists L \ \ \forall x, y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$

Local Lipschitz Constant

$$\forall x \ \ \exists L_x \ \ \forall y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$



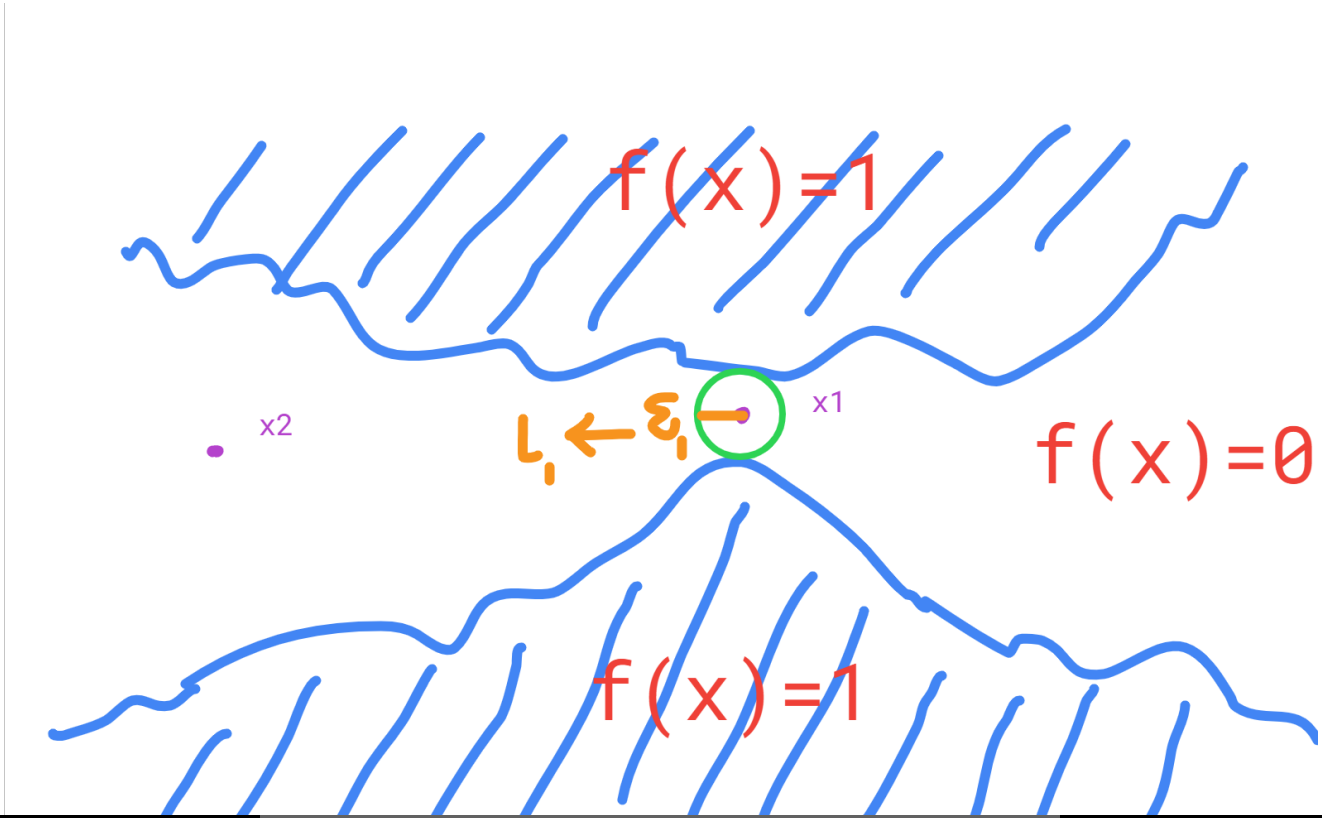
Local Lipschitz Constant

Lipschitz Constant

$$\exists L \ \ \forall x, y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$

Local Lipschitz Constant

$$\forall x \ \ \exists L_x \ \ \forall y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$



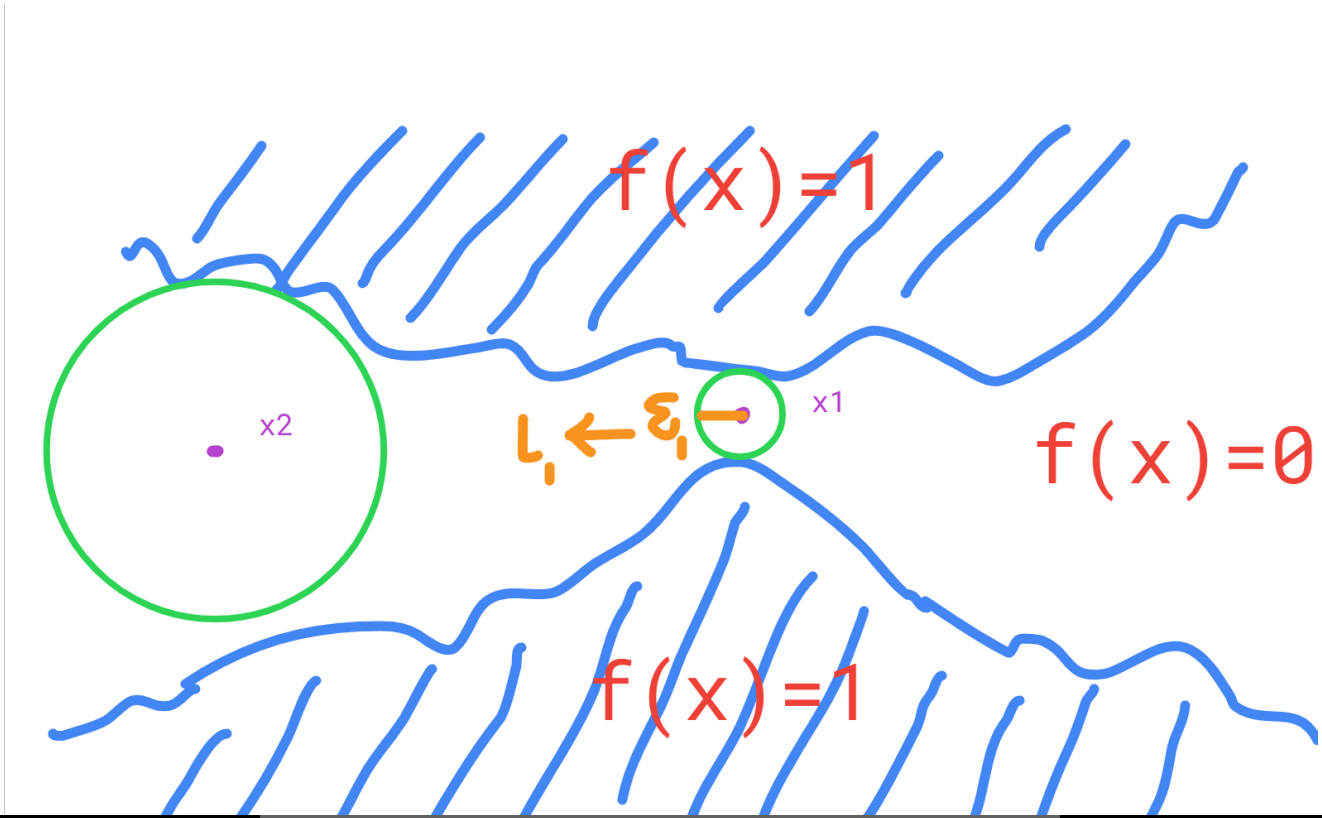
Local Lipschitz Constant

Lipschitz Constant

$$\exists L \ \ \forall x, y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$

Local Lipschitz Constant

$$\forall x \ \ \exists L_x \ \ \forall y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$



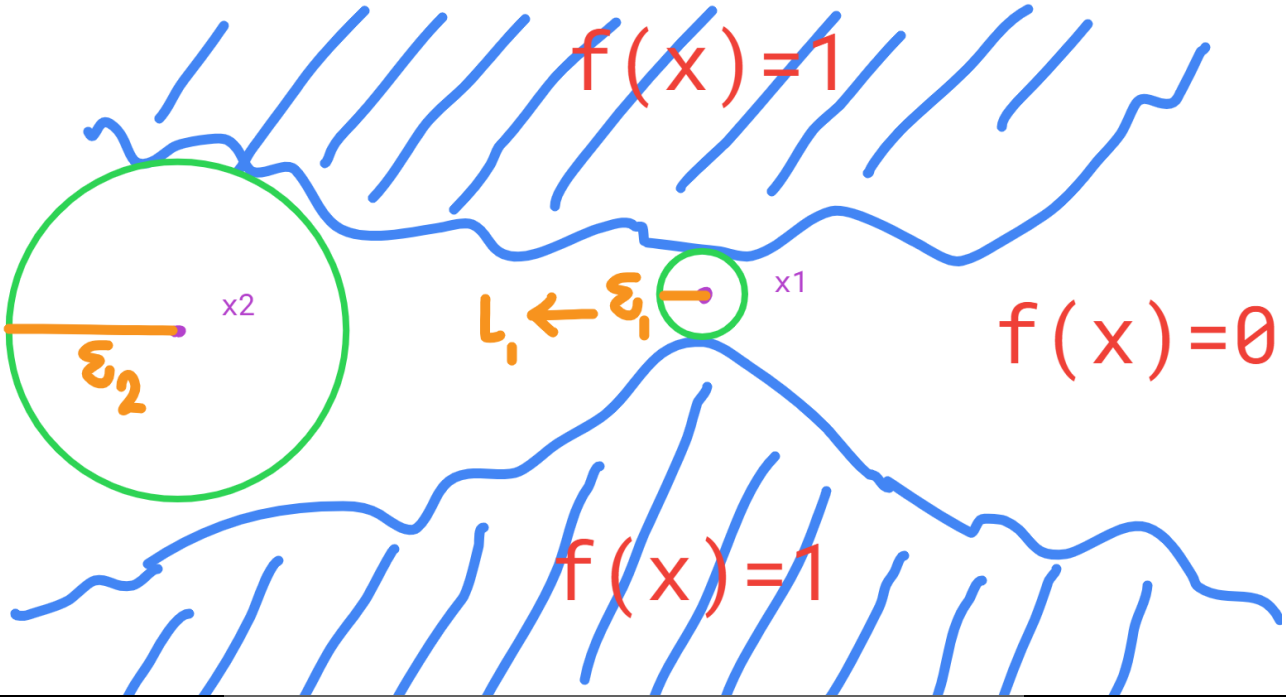
Local Lipschitz Constant

Lipschitz Constant

$$\exists L \ \ \forall x, y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$

Local Lipschitz Constant

$$\forall x \ \ \exists L_x \ \ \forall y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$



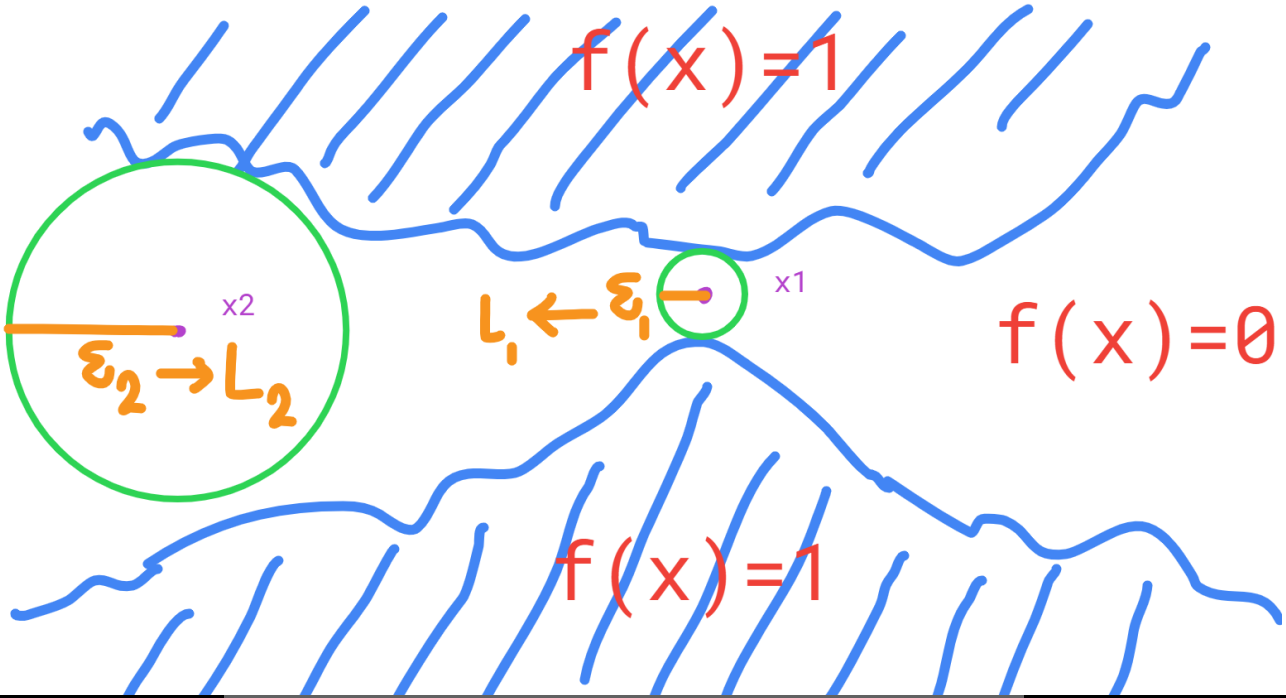
Local Lipschitz Constant

Lipschitz Constant

$$\exists L \ \ \forall x, y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$

Local Lipschitz Constant

$$\forall x \ \ \exists L_x \ \ \forall y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$



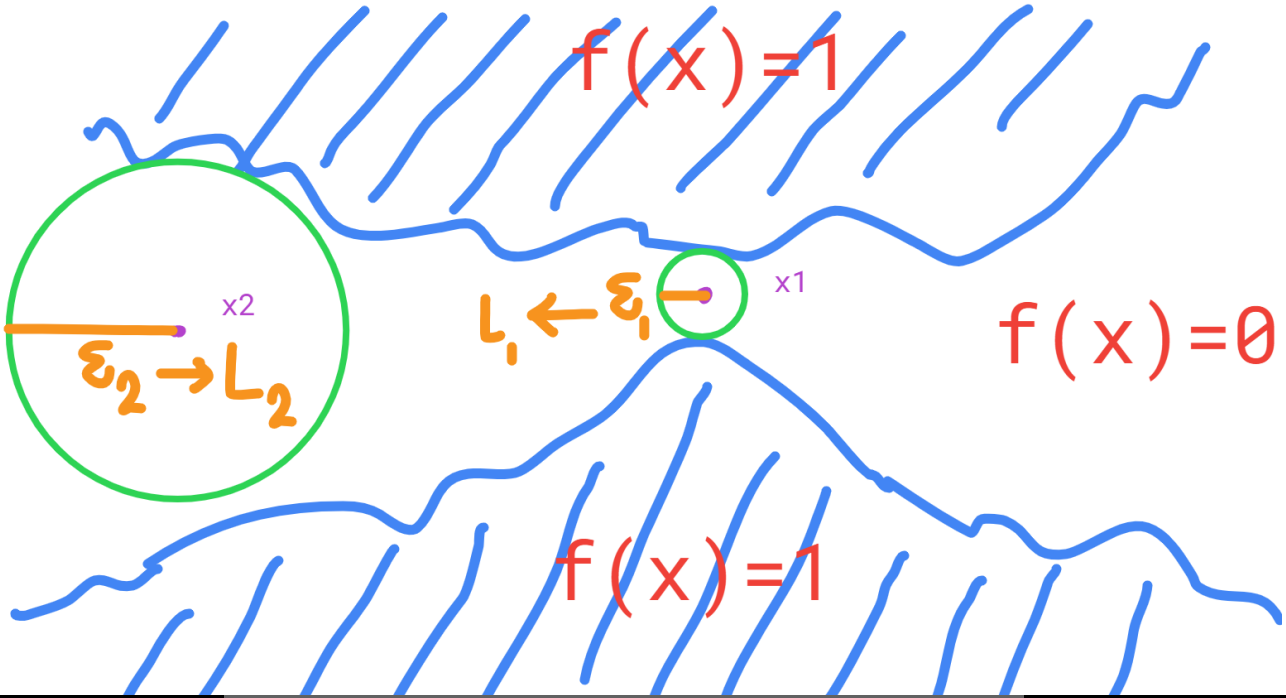
Local Lipschitz Constant

Lipschitz Constant

$$\exists L \ \ \forall x, y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$

Local Lipschitz Constant

$$\forall x \ \ \exists L_x \ \ \forall y: \quad \|f(y) - f(x)\|_2 \leq L_x \|y - x\|_2$$



Local Lipschitz Constant

We can write L_2 as the supremum of an inner product

$$\|\nabla_x \hat{f}\|_2 = \underset{\|u\|_2=1}{sup} \; u.\nabla_x \hat{f} \quad (\text{where } u^* = \frac{\nabla_x \hat{f}}{\|\nabla_x \hat{f}\|_2})$$

Local Lipschitz Constant

We can write L_2 as the supremum of an inner product

$$\|\nabla_x \hat{f}\|_2 = \underset{\|u\|_2=1}{sup}\ u.\nabla_x \hat{f} \quad (\text{where } u^* = \frac{\nabla_x \hat{f}}{\|\nabla_x \hat{f}\|_2})$$

We know $\nabla_x \hat{f}(x) = \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[\frac{z}{\sigma} f(x+\sigma z)]$. Therefor, we have

$$\|\nabla_x \hat{f}\|_2 = \underset{\|u\|_2=1}{sup}\ u.\mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[\frac{z}{\sigma} f(x+\sigma z)]$$

Local Lipschitz Constant

We can write L_2 as the supremum of an inner product

$$\|\nabla_x \hat{f}\|_2 = \underset{\|u\|_2=1}{sup}\ u.\nabla_x \hat{f} \quad (\text{where } u^* = \frac{\nabla_x \hat{f}}{\|\nabla_x \hat{f}\|_2})$$

We know $\nabla_x \hat{f}(x) = \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[\frac{z}{\sigma} f(x+\sigma z)]$. Therefor, we have

$$\|\nabla_x \hat{f}\|_2 = \underset{\|u\|_2=1}{sup}\ u.\mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[\frac{z}{\sigma} f(x+\sigma z)]$$

Since expectation and inner product are linear operators, we have

$$\|\nabla_x \hat{f}\|_2 = \underset{\|u\|_2=1}{sup}\ \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[\frac{z.u}{\sigma} f(x+\sigma z)]$$

Local Lipschitz Constant

We can write L_2 as the supremum of an inner product

$$\|\nabla_x \hat{f}\|_2 = \sup_{\|u\|_2=1} u.\nabla_x \hat{f} \quad (\text{where } u^* = \frac{\nabla_x \hat{f}}{\|\nabla_x \hat{f}\|_2})$$

We know $\nabla_x \hat{f}(x) = \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[\frac{z}{\sigma} f(x+\sigma z)]$. Therefor, we have

$$\|\nabla_x \hat{f}\|_2 = \sup_{\|u\|_2=1} u.\mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[\frac{z}{\sigma} f(x+\sigma z)]$$

Since expectation and inner product are linear operators, we have

$$\|\nabla_x \hat{f}\|_2 = \sup_{\|u\|_2=1} \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[\frac{z.u}{\sigma} f(x+\sigma z)]$$

We want to $\|\nabla_x \hat{f}\|_2$ becomes input-dependent. We assume $p_x = \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[f(x+\sigma z)]$. Thus, we have

$$\|\nabla_x \hat{f}\|_2 = \sup_{\|u\|_2=1} \mathbb{E}_{z\sim\mathcal{N}(0,I_d)}[\frac{z.u}{\sigma} f(x+\sigma z)]$$
$$\text{such that} \quad \mathbb{E}_z[f(x+\sigma z)] = p_x$$

(See **this video** for the rest of proof.)

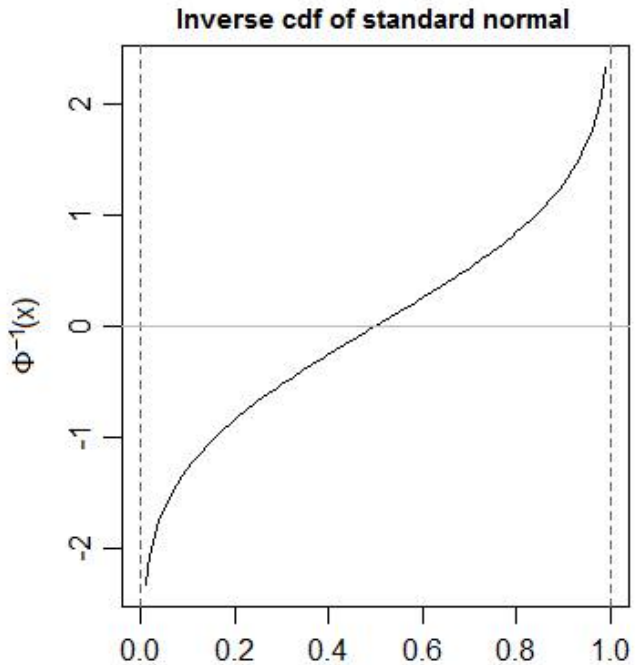
Local Lipschitz Constant

By taking supremum over u and f, the local Lipschitz constant L_x for input x is as follows

$$L_x \leq \frac{1}{\sqrt{2\pi}\sigma} e^{-\{\Phi^{-1}(p_x)\}^2/2}$$

Where Φ is the Cumulative Density Function (CDF) of standard Gaussian distribution $\mathcal{N}(0, I_d)$.

- The bound is input-dependent.
- The bound does not depend on the input dimension d.
- L_x decreases as p_x rises.
- L_x decreases as σ rises.



Local Lipschitz Constant

We introduce $\psi(t)$ so that the Lipschitz constant of $\psi(\hat{f}(x))$ is $\frac{1}{\sigma}$. We have

$$\nabla_x \psi(\hat{f}(x)) = \nabla_x \hat{f}(x).\nabla_{\hat{f}(x)} \psi(\hat{f}(x))$$
$$\|\nabla_x \psi(\hat{f}(x))\|_2 = \|\nabla_x \hat{f}(x)\|_2 |\nabla_{p_x} \psi(p_x)|$$

where $\nabla_{p_x} \psi(p_x)$ is a scalar. We know $\|\nabla_x \hat{f}(x)\|_2 \leq \frac{1}{\sqrt{2\pi}\sigma} e^{-\{\Phi^{-1}(p_x)\}^2/2}$. In order to the Lipschitz constant of $\psi(\hat{f}(x))$ be $\frac{1}{\sigma}$, $\nabla_{p_x} \psi(p_x)$ should be

$$\frac{1}{\frac{1}{\sqrt{2\pi}} e^{-\{\Phi^{-1}(p_x)\}^2/2}}$$

Therefore, we have

$$\psi'(p_x) = \frac{1}{\frac{1}{\sqrt{2\pi}} e^{-\{\Phi^{-1}(p_x)\}^2/2}}$$

Local Lipschitz Constant

Recall: The Derivative of the CDF of Standard Normal Distribution

The derivative of the CDF of standard normal distribution $\mathcal{N}(0, I)$

$$\Phi'(t) = \frac{d}{dt} \int_{-\infty}^{t} \frac{1}{\sqrt{2\pi}} e^{-s^2/2} ds = \frac{1}{\sqrt{2\pi}} e^{-t^2/2}$$

Recall: Fundamental Theorem of Calculus

If $f(x)$ is continuous over an interval $[a, b]$, and the function $F(x)$ is defined by

$$F(x) = \int_{a}^{x} f(t)dt,$$

then $F'(x) = f(x)$ over $[a, b]$.

Another way of stating the conclusion of the fundamental theorem of calculus is:

$$\frac{d}{dx} \int_{a}^{x} f(t)dt = f(x)$$

The conclusion of the fundamental theorem of calculus can be loosely expressed in words as: ”the derivative of an integral of a function is that original function”, or ”differentiation undoes the result of integration”.

Local Lipschitz Constant

We introduce $\psi(t)$ so that the Lipschitz constant of $\psi(\hat{f}(x))$ is $\frac{1}{\sigma}$. We have

$$\nabla_x \psi(\hat{f}(x)) = \nabla_x \hat{f}(x).\nabla_{\hat{f}(x)} \psi(\hat{f}(x))$$
$$\|\nabla_x \psi(\hat{f}(x))\|_2 = \|\nabla_x \hat{f}(x)\|_2 |\nabla_{p_x} \psi(p_x)|$$

where $\nabla_{p_x} \psi(p_x)$ is a scalar. We know $\|\nabla_x \hat{f}(x)\|_2 \leq \frac{1}{\sqrt{2\pi}\sigma} e^{-\{\Phi^{-1}(p_x)\}^2/2}$. In order to the Lipschitz constant of $\psi(\hat{f}(x))$ be $\frac{1}{\sigma}$, $\nabla_{p_x} \psi(p_x)$ should be

$$\frac{1}{\frac{1}{\sqrt{2\pi}} e^{-\{\Phi^{-1}(p_x)\}^2/2}}$$

Therefore, we have

$$\psi'(p_x) = \frac{1}{\frac{1}{\sqrt{2\pi}} e^{-\{\Phi^{-1}(p_x)\}^2/2}} = \left[\Phi'(\Phi^{-1}(p_x))\right]^{-1}$$

Local Lipschitz Constant

Recall:The Derivative of Inverse Function

Given an invertible function $y = f(x)$, the derivative of its inverse function $f^{-1}(y)$ is

$$\left[f^{-1}\right]'(y) = \frac{1}{f'(f^{-1}(y))} = \left[f'(f^{-1}(y))\right]^{-1}$$

Proof

To see why this is true, start with the function $x = f^{-1}(y)$. Write this as $y = f(x)$ and differentiate both sides implicitly with respect to y using the Chain Rule

$$1 = f'(x).\frac{dx}{dy}$$

Thus

$$\frac{dx}{dy} = \frac{1}{f'(x)}$$

but $x = f^{-1}(y)$. Thus,

$$\left[f^{-1}\right]'(y) = \frac{1}{f'\left[f^{-1}(y)\right]}$$

Local Lipschitz Constant

We introduce $\psi(t)$ so that the Lipschitz constant of $\psi(\hat{f}(x))$ is $\frac{1}{\sigma}$. We have

$$\nabla_x \psi(\hat{f}(x)) = \nabla_x \hat{f}(x).\nabla_{\hat{f}(x)} \psi(\hat{f}(x))$$
$$\|\nabla_x \psi(\hat{f}(x))\|_2 = \|\nabla_x \hat{f}(x)\|_2 |\nabla_{p_x} \psi(p_x)|$$

where $\nabla_{p_x} \psi(p_x)$ is a scalar. We know $\|\nabla_x \hat{f}(x)\|_2 \leq \frac{1}{\sqrt{2\pi}\sigma} e^{-\{\Phi^{-1}(p_x)\}^2/2}$. In order to the Lipschitz constant of $\psi(\hat{f}(x))$ be $\frac{1}{\sigma}$, $\nabla_{p_x} \psi(p_x)$ should be

$$\frac{1}{\frac{1}{\sqrt{2\pi}} e^{-\{\Phi^{-1}(p_x)\}^2/2}}$$

Therefore, we have

$$\psi'(p_x) = \frac{1}{\frac{1}{\sqrt{2\pi}} e^{-\{\Phi^{-1}(p_x)\}^2/2}} = \left[\Phi'(\Phi^{-1}(p_x))\right]^{-1} = \left[\Phi^{-1}\right]'(p_x)$$

Thus

$$\psi(p_x) = \Phi^{-1}(p_x) + c$$

where c is a constant (we set $c = 0$).

Certifiable Robustness for $1/\sigma$-Lipschitz smoothed classifier

Since, the Lipschitz constant of $\Phi^{-1}(\hat{f}(x))$ is $\frac{1}{\sigma}$, similar to **Theorem 0**, we can show that

$$\underset{i}{\text{argmax}}\ \hat{f}_i(x+\delta) = \underset{i}{\text{argmax}}\ \hat{f}_i(x)$$

for all $\|\delta\|_2 \leq R$, where

$$R = \frac{1}{2L}(\Phi^{-1}(P_A) - \Phi^{-1}(P_B)) = \frac{\sigma}{2}(\Phi^{-1}(P_A) - \Phi^{-1}(P_B)),$$

$P_A = \underset{i}{max}\ \hat{f}_i(x)$, $P_B = \underset{j\neq i}{max}\ \hat{f}_j(x)$, and $\hat{f}_k(x)$ is the k-th element of the probability vector $\hat{f}(x)$.

Training the Base Classifier

Theorem 1 holds regardless of how the base classifier f is trained.

- If the base classifier f is trained via standard supervised learning on the data distribution, it will **see no noisy images during training**, and hence **will not necessarily learn to classify** $\mathcal{N}(x, \sigma^2 I)$ with x's true label.
- Therefore, we **train the base classifier with Gaussian data augmentation** at variance σ^2.

Experiments

Evaluation metric is **certified test set accuracy** at radius r defined as

- The fraction of the test set which $\hat{f}$ classifies correctly with a prediction that is certifiably robust within an ℓ_2 ball of radius r.
- Thus, to compute **certified accuracy**, we pick a target radius and count the number of points in the test set whose certified radius $r \geq T$ and where the predicted c_A matches the test set label. Standard accuracy is instantiated with $T = 0$.

CIFAR10 model

- ResNet-110
- $n = 100000$ (use n samples from $f(x + \epsilon)$ to obtain some $\underline{P_A}$ and $\overline{P_B}$.)
- Certifying each example took 15 seconds on an NVIDIA RTX 2080 Ti.

ImageNet model

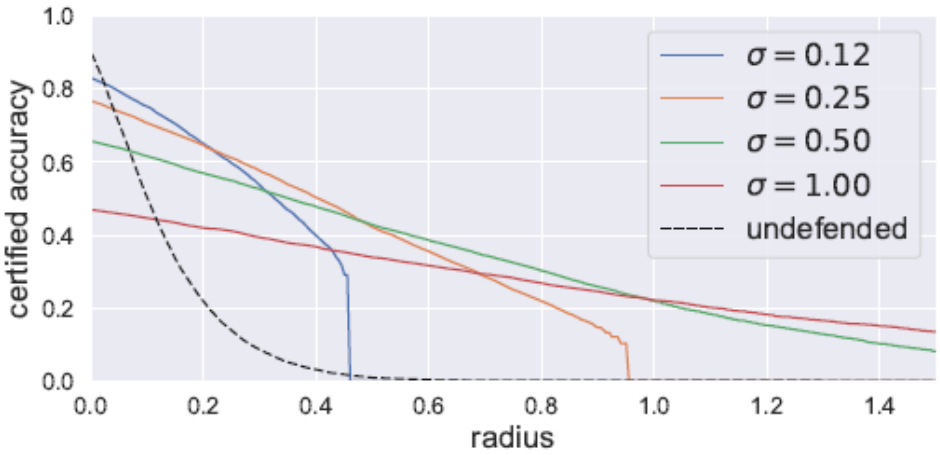
- ResNet-50
- $n = 100000$
- certifying each example took 110 seconds.

Experiments

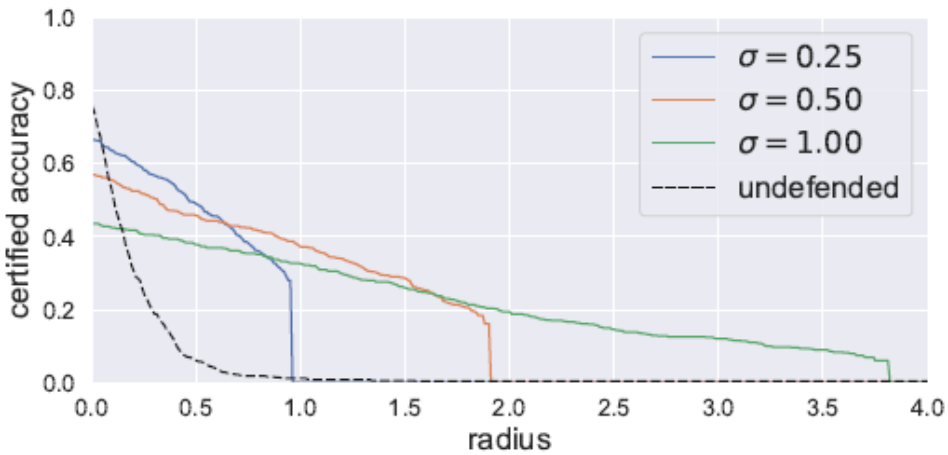
	$r = 0.0$	$r = 0.5$	$r = 1.0$	$r = 1.5$	$r = 2.0$	$r = 2.5$	$r = 3.0$
$\sigma = 0.25$	**0.67**	**0.49**	0.00	0.00	0.00	0.00	0.00
$\sigma = 0.50$	0.57	0.46	**0.37**	**0.29**	0.00	0.00	0.00
$\sigma = 1.00$	0.44	0.38	0.33	0.26	**0.19**	**0.15**	**0.12**

Table 2. Approximate certified test accuracy on ImageNet. Each row is a setting of the hyperparameter σ, each column is an ℓ_2 radius. The entry of the best σ for each radius is bolded. For comparison, random guessing would attain 0.001 accuracy.

Experiments



CIFAR10



ImageNet

Figure 6. Approximate certified accuracy attained by randomized smoothing on CIFAR-10 (left) and ImageNet (right). The hyper-parameter σ controls a robustness/accuracy tradeoff. The dashed black line is an upper bound on the empirical robust accuracy of an undefended classifier with the base classifier's architecture.

Gaussian Smoothing for L_P Attacks

If we use Gaussian smoothing against L_P attacks, for $p \geq 2$ we get

$$r_p = \frac{\sigma}{2d^{\frac{1}{2}-\frac{1}{p}}}(\Phi^{-1}(p_1(x)) - \Phi^{-1}(p_2(x)))$$

Curse of dimensionality: For L_P attacks where $p > 2$, the smoothing-based certificate upper bound decreases as d increases.

(See: Curse of Dimensionality on Randomized Smoothing for Certifiable Robustness, Kumar et al., 2020)

Black-box Adversarial Examples

Black-box Adversarial Examples

The only capability of the black-box adversary is to observe the **output** given by the target model to **chosen inputs**.

Black-box Adversarial Examples

The only capability of the black-box adversary is to observe the **output** given by the target model to **chosen inputs**.

- In this setting, back propagation for **gradient computation of the targeted model is prohibited**.

Black-box Adversarial Examples

The only capability of the black-box adversary is to observe the **output** given by the target model to **chosen inputs**.

- In this setting, back propagation for **gradient computation of the targeted model is prohibited**.
- Threat model
  - **Score-based** (the adversary has access to the target model scores)
  - **Decision-based** (the adversary has only access to the target model label)

Black-box Adversarial Examples

The only capability of the black-box adversary is to observe the **output** given by the target model to **chosen inputs**.

- In this setting, back propagation for **gradient computation of the targeted model is prohibited**.
- Threat model
  - **Score-based** (the adversary has access to the target model scores)
  - **Decision-based** (the adversary has only access to the target model label)
- Types
  - **Transfer-based** and **Query-based**

Black-box Adversarial Examples

The only capability of the black-box adversary is to observe the **output** given by the target model to **chosen inputs**.

- In this setting, back propagation for **gradient computation of the targeted model is prohibited**.
- Threat model
  - **Score-based** (the adversary has access to the target model scores)
  - **Decision-based** (the adversary has only access to the target model label)
- Types
  - **Transfer-based** and **Query-based**

Transfer-based

- Create a surrogate model with high **fidelity** to the taget model.
- Generate adversarial examples on the surrogate model using **white-box** attacks.
- Then, **transfer** pregenerated adversarial examples to the target model.

Black-box Adversarial Examples

The only capability of the black-box adversary is to observe the **output** given by the target model to **chosen inputs**.

- In this setting, back propagation for **gradient computation of the targeted model is prohibited**.
- Threat model
  - **Score-based** (the adversary has access to the target model scores)
  - **Decision-based** (the adversary has only access to the target model label)
- Types
  - **Transfer-based** and **Query-based**

Transfer-based

- Create a surrogate model with high **fidelity** to the taget model.
- Generate adversarial examples on the surrogate model using **white-box** attacks.
- Then, **transfer** pregenerated adversarial examples to the target model.

Query-based

- Based on the target model responses for consecutive queries
  - Gradient estimation
    - Based on zero-order (ZO) optimization algorithms
  - Search-based
    - Based on choosing a search strategy using a search distribution.

Jacobian-based Dataset Augmentation

Jacobian-based Dataset Augmentation

Practical Black-Box Attacks against Machine Learning

Nicolas Papernot
Pennsylvania State University
ngp5056@cse.psu.edu

Patrick McDaniel
Pennsylvania State University
mcdaniel@cse.psu.edu

Ian Goodfellow*
OpenAI
ian@openai.com

Somesh Jha
University of Wisconsin
jha@cs.wisc.edu

Z. Berkay Celik
Pennsylvania State University
zbc102@cse.psu.edu

Ananthram Swami
US Army Research Laboratory
ananthram.swami.civ@mail.mil

Abstract

The first black-box attack against DNN classifiers for real-world adversaries with no knowledge about the model.

- The only capability of the black-box adversary is to observe **labels** given by the DNN to chosen inputs.

The attack strategy

- **Training a local model** to substitute for the target DNN.
  - Using **inputs synthetically generated** by an adversary and labeled by the target DNN.
- The local substitute is used to craft adversarial examples, and find that they are misclassified by the targeted DNN.

To perform a real-world and properly-blinded evaluation, we attack a DNN hosted by **MetaMind**, an online deep learning API.

Threat Model

In the black-box setting, adversaries do not know internal details of a system to compromise it.

- The adversary has **no information about the structure or parameters** of the target DNN.
- The adversary has **no knowledge of the training data** used to learn the DNN's parameters.
- The adversary does **not have access to any large training dataset**.
- The adversary's only capability is to observe **labels** assigned by the DNN for chosen inputs.

Threat Model

In the black-box setting, adversaries do not know internal details of a system to compromise it.

- The adversary has **no information about the structure or parameters** of the target DNN.
- The adversary has **no knowledge of the training data** used to learn the DNN's parameters.
- The adversary does **not have access to any large training dataset**.
- The adversary's only capability is to observe **labels** assigned by the DNN for chosen inputs.

A data-limited adversary

- Many modern machine learning systems require large and expensive training sets for training.
  - This makes attacks based on training substitute model **unfeasible for adversaries without large labeled datasets**.
- To enable the adversary to train a substitute model without a real labeled dataset
  - The adversary **uses the target DNN as an oracle to construct a synthetic dataset**.

Adversarial Capabilities

The oracle O is the targeted DNN.

- Its name refers to the only capability of the adversary: accessing the label $\tilde{O}(x)$ for any input x by querying oracle O.

The output label $\tilde{O}(x)$ is the index of the class assigned the largest probability by the DNN

$$\tilde{O}(x) = \underset{j \in 0,\ldots,N-1}{argmax} \; O_j(x)$$

where $O_j(x)$ is the j-th component of the probability vector $O(x)$ output by DNN O, and N is the number of classes.

Accessing labels $\tilde{O}$ produced by the DNN O is the only capability assumed in our threat model.

Adversarial Goal

The adversary wants to produce a minimally altered version of any input x, named **adversarial example**, and denoted x^*, misclassified by oracle O

$$x^* = x + argmin\{z : \tilde{O}(x+z) \neq \tilde{O}(x)\} = x + \delta_x$$
$$\text{such that : } \tilde{O}(x^*) \neq \tilde{O}(x)$$

Black-box Attack

Black-box Attack Strategy

1. **Substitute Model Training**: the attacker queries the oracle with **synthetic inputs** selected by a Jacobian based heuristic to build a **model F approximating the oracle model O's decision boundaries**.
2. **Adversarial Sample Crafting**: the attacker uses substitute network F to craft adversarial samples, which are then misclassified by oracle O due to the transferability of adversarial samples.

Black-box Attack

Black-box Attack Strategy

1. **Substitute Model Training**: the attacker queries the oracle with **synthetic inputs** selected by a Jacobian based heuristic to build a **model F approximating the oracle model O's decision boundaries**.
2. **Adversarial Sample Crafting**: the attacker uses substitute network F to craft adversarial samples, which are then misclassified by oracle O due to the transferability of adversarial samples.

Challenges

- **Select an architecture** for our substitute without knowledge of the targeted oracle's architecture
- **Limit the number of queries** made to the oracle in order to ensure that the approach is tractable.

Generating a Synthetic Dataset

The heuristic used to generate **synthetic training inputs** is based on identifying **directions in which the model's output is varying**, around an initial set of training points.

Generating a Synthetic Dataset

The heuristic used to generate **synthetic training inputs** is based on identifying **directions in which the model's output is varying**, around an initial set of training points.

- These directions are identified with the substitute DNN's **Jacobian matrix** J_F , which is evaluated at several input points x.
- Precisely, the adversary evaluates the sign of the Jacobian matrix dimension corresponding to the **label assigned to input** x **by the oracle**

$$sign(J_F(x)[\tilde{O}(x)]) = sign(\nabla_x F(x)_{\tilde{O}(x)})$$

where $F(x)_i$ is the i-th element of the probability vector $F(x)$ output by substitute model F.

Generating a Synthetic Dataset

The heuristic used to generate **synthetic training inputs** is based on identifying **directions in which the model's output is varying**, around an initial set of training points.

- These directions are identified with the substitute DNN's **Jacobian matrix** J_F , which is evaluated at several input points x.
- Precisely, the adversary evaluates the sign of the Jacobian matrix dimension corresponding to the **label assigned to input x by the oracle**

$$sign(J_F(x)[\tilde{O}(x)]) = sign(\nabla_x F(x)_{\tilde{O}(x)})$$

where $F(x)_i$ is the i-th element of the probability vector $F(x)$ output by substitute model F.

The new synthetic training point x' is created as follows

$$x' = x + \lambda.sign(J_F(x)[\tilde{O}(x)]) = x + \lambda.sign(\nabla_x F(x)_{\tilde{O}(x)})$$

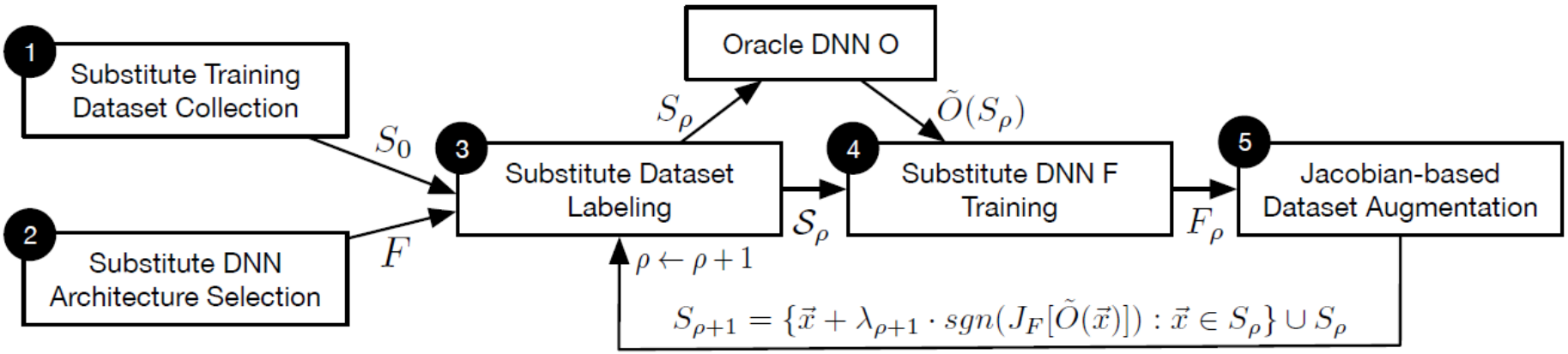
where x is already in the training set, and λ is a parameter of the augmentation.

This technique is called **Jacobian-based Dataset Augmentation**.

Jacobian-based Dataset Augmentation

- **Initial Collection**: The adversary collects a very small set S_0 of inputs representative of the input domain.
- **Architecture Selection**: The adversary selects an architecture to be trained as the substitute model F.
- **Substitute Training**: the adversary iteratively trains more accurate substitute model F_ρ by repeating the following for max_ρ rounds:
  1. **Labeling** S_ρ by oracle O
  2. **Training** F_ρ on S_ρ
  3. Create $S_{\rho+1}$ by **augmenting** S_ρ with more synthetic training points.

$$S_{\rho+1} = \{x + \lambda_{\rho+1}.sign(J_F(x)[\tilde{O}(x)]) : \ x \in S_\rho\} \cup S_\rho$$



Algorithm

Algorithm 1 - Substitute DNN Training: for oracle $\tilde{O}$, a maximum number max_ρ of substitute training epochs, a substitute architecture F, and an initial training set S_0.

Input: $\tilde{O}$, max_ρ, S_0, λ

Define architecture F
for $\rho \in 0 \,..\, max_\rho - 1$ **do**
    // Label the substitute training set
    $D \leftarrow \left\{(\vec{x}, \tilde{O}(\vec{x})) : \vec{x} \in S_\rho\right\}$
    // Train F on D to evaluate parameters θ_F
    $\theta_F \leftarrow \text{train}(F, D)$
    // Perform Jacobian-based dataset augmentation
    $S_{\rho+1} \leftarrow \{\vec{x} + \lambda \cdot \text{sgn}(J_F[\tilde{O}(\vec{x})]) : \vec{x} \in S_\rho\} \cup S_\rho$
end for
return θ_F

Next Papers

They introduce the *Jacobian-based Dataset Augmentation* (JbDA) technique for generating synthetic samples (row 13). It relies on computing the Jacobian matrices with the current F' evaluated on the already labeled samples in L. Each element $x \in L$ is modified by adding the sign of the Jacobian matrix $\nabla_x \mathcal{L}(F'(x, c_i))$ dimension corresponding to the label assigned to x by F, evaluated with regards to the classification loss $\mathcal{L}$. Thus, the set U is extended with $\{x + \lambda \cdot sign(\nabla_x \mathcal{L}(F'(x, c_i)))\}$, $\forall x \in L$. U has the same

PRADA, IEEE European S&P, 2018

```
# Get augmented inputs
X, Y = X.to(self.device), Y.to(self.device)
delta_i = self.fgsm_untargeted(model_adv, X, Y.argmax(dim=1), device=self.device, epsilon=step_size)
# Get correnponding outputs from blackbox
if self.aug_strategy == 'jbda':
    Y_i = self.blackbox(X + delta_i)
```

Prediction Poisoning, ICLR, 2020

2. *Jacobian Based Dataset Augmentation (JBDA)* (Papernot et al., 2017) uses synthetic data to query the target model. JBDA starts with a small set of in-distribution "seed" examples $\{x_i\}$. The attack iteratively trains the clone model by performing the following steps: (i) Obtain a labeled dataset $\{x_i, y_i\}$ by querying the target model (ii) Train the clone model on the labeled dataset (iii) Augment the dataset with synthetic examples x_i' by perturbing the original input x_i to change the prediction of the clone model by using the Jacobian of the loss function: $x_i' = x_i + \beta sign\left(\nabla_x \mathcal{L}\left(C\left(x_i; \theta_c\right), y_i\right)\right)$.

EDM, ICLR, 2021

generates a synthetic example x', by perturbing it using the jacobian of the clone model's loss function: $x' = x + \lambda sign\left(\nabla_x \mathcal{L}\left(f'\left(x; \theta'\right)\right)\right)$. These synthetic examples are labeled using the predictions of the defender's model $y' = f(x')$ and the labeled synthetic examples thus generated: $\mathcal{D}_{syn} = \{x', y'\}$, are used to augment the adversary's dataset: $\mathcal{D}_{seed} = \mathcal{D}_{seed} \cup \mathcal{D}_{syn}$ and retrain f'.

Adaptive Misinformation, CVPR, 2019

Attack on MetaMind MNIST Model

Initial Substitute Training Sets

- **MNIST subset**: This initial substitute training set is made of 150 samples from the MNIST test set.
- **Handcrafted set**: 100 samples by handwriting 10 digits for each class between 0 and 9 with a laptop trackpad.

Implementation details

- $max_\rho = 6$.
- During each of these 6 rounds, the model is trained for 10 epochs from scratch.
- $\lambda = 0.1$

The Accuracy of the Two Substitute Models

Substitute Epoch	Initial Substitute Training Set from MNIST test set	Handcrafted digits
0	24.86%	18.70%
1	41.37%	19.89%
2	65.38%	29.79%
3	74.86%	36.87%
4	80.36%	40.64%
5	79.18%	56.95%
6	81.20%	67.00%

Figure 4: **Substitute DNN Accuracies:** each column corresponds to an initial substitute training set: 150 MNIST test samples, and handcrafted digits. Accuracy is reported on the unused 9,850 MNIST test samples.

Evaluation

- MNIST test samples are used to generate adversarial examples using FGSM.
- The success rate is the proportion of adversarial samples misclassified by the substitute model.
- The transferability of adversarial samples refers to the oracle misclassification rate of adversarial samples crafted using the substitute DNN.

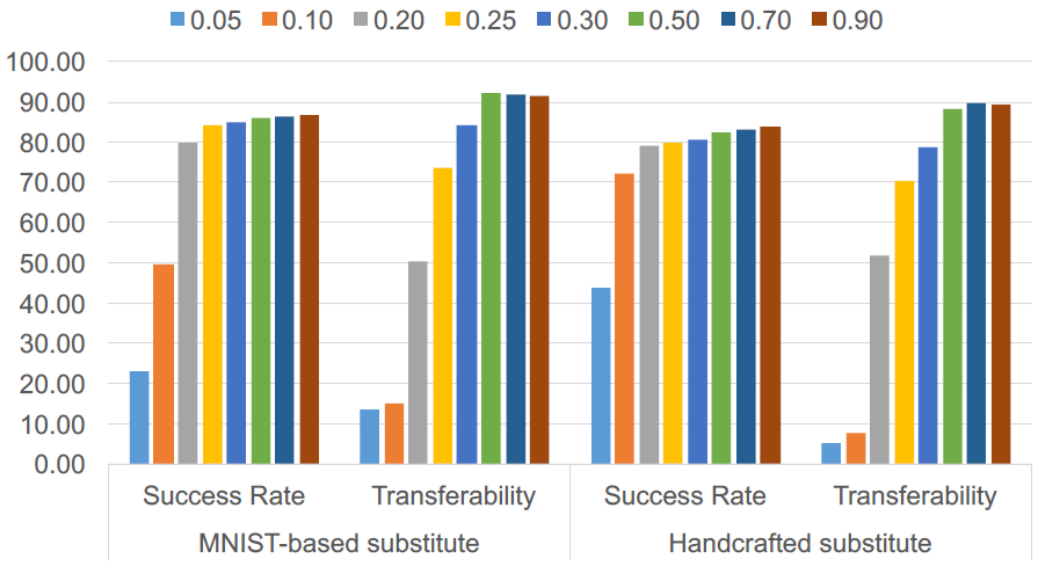


Figure 5: **Success Rate and Transferability of Adversarial Samples for the MetaMind attacks:** performed using MNIST-based and handcrafted substitutes: each bar corresponds to a different perturbation input variation.

Attack Algorithm Calibration

- The choice of **substitute DNN architecture** (number of layers, size, activation function, type) has a **limited impact** on adversarial sample transferability.

Attack Algorithm Calibration

- The choice of **substitute DNN architecture** (number of layers, size, activation function, type) has a **limited impact** on adversarial sample transferability.
- **Increasing the number of epochs**, after the substitute DNN has reached an asymptotic accuracy, does not improve adversarial sample transferability.

Attack Algorithm Calibration

- The choice of **substitute DNN architecture** (number of layers, size, activation function, type) has a **limited impact** on adversarial sample transferability.
- **Increasing the number of epochs**, after the substitute DNN has reached an asymptotic accuracy, does not improve adversarial sample transferability.
- Increasing **step size λ negatively impacts** adversarial sample transferability and does not modify the substitute accuracy by more than 3%.

Attack Algorithm Calibration

- The choice of **substitute DNN architecture** (number of layers, size, activation function, type) has a **limited impact** on adversarial sample transferability.
- **Increasing the number of epochs**, after the substitute DNN has reached an asymptotic accuracy, does not improve adversarial sample transferability.
- Increasing **step size λ negatively impacts** adversarial sample transferability and does not modify the substitute accuracy by more than 3%.
- Having the **step size periodically alternating between positive and negative values improves the quality** of the oracle approximation made by the substitute.

Attack Algorithm Calibration

- The choice of **substitute DNN architecture** (number of layers, size, activation function, type) has a **limited impact** on adversarial sample transferability.
- **Increasing the number of epochs**, after the substitute DNN has reached an asymptotic accuracy, does not improve adversarial sample transferability.
- Increasing **step size** λ **negatively impacts** adversarial sample transferability and does not modify the substitute accuracy by more than 3%.
- Having the **step size periodically alternating between positive and negative values improves the quality** of the oracle approximation made by the substitute.
- **Reducing Oracle Querying**: randomly select κ samples from a list of samples. The adversary after σ iterations selects κ new inputs for Jacobian-based dataset augmentation.