

CS480/640 Robustness

P. Michael Furlong

2024-11-26

All models are wrong, but some models are useful.

– G. E. Box, circa 1976.

All models are wrong, but some models are useful.

– G. E. Box, circa 1976.

But wrong models are still wrong, and can be wrong in pernicious ways....

What do we mean by robustness?

- ▶ In control theory: actions are completed successfully, despite small perturbations.
- ▶ The desired behaviour is that the system should work the way it is supposed to work, given some small perturbation.
- ▶ Similarly, we want learned models to be robust to small changes in the input.

What do we mean by robustness?

- ▶ In control theory: actions are completed successfully, despite small perturbations.
- ▶ The desired behaviour is that the system should work the way it is supposed to work, given some small perturbation.
- ▶ Similarly, we want learned models to be robust to small changes in the input.



Formally Stating the Problem Setting

We want to learn a function, $y = f(x)$, but f is unknown. We are given:

1. A training set, $(x_i, y_i) \in X \times Y$, where $y_i = f(x_i)$ and $X \subseteq \mathbb{R}^d$
2. A loss function, $\ell : Y \times Y \rightarrow \mathbb{R}^+$ (less loss is better)
3. A function approximator, $\hat{f}(x; \theta)$, that is parameterized by $\theta \in \mathbb{R}^p$

Formally Stating the Problem Setting

We want to learn a function, $y = f(x)$, but f is unknown. We are given:

1. A training set, $(x_i, y_i) \in X \times Y$, where $y_i = f(x_i)$ and $X \subseteq \mathbb{R}^d$
2. A loss function, $\ell : Y \times Y \rightarrow \mathbb{R}^+$ (less loss is better)
3. A function approximator, $\hat{f}(x; \theta)$, that is parameterized by $\theta \in \mathbb{R}^P$

Formally Stating the Problem Setting

We want to learn a function, $y = f(x)$, but f is unknown. We are given:

1. A training set, $(x_i, y_i) \in X \times Y$, where $y_i = f(x_i)$ and $X \subseteq \mathbb{R}^d$
2. A loss function, $\ell : Y \times Y \rightarrow \mathbb{R}^+$ (less loss is better)
3. A function approximator, $\hat{f}(x; \theta)$, that is parameterized by $\theta \in \mathbb{R}^P$

Formally Stating the Problem Setting

We want to learn a function, $y = f(x)$, but f is unknown. We are given:

1. A training set, $(x_i, y_i) \in X \times Y$, where $y_i = f(x_i)$ and $X \subseteq \mathbb{R}^d$
2. A loss function, $\ell : Y \times Y \rightarrow \mathbb{R}^+$ (less loss is better)
3. A function approximator, $\hat{f}(x; \theta)$, that is parameterized by $\theta \in \mathbb{R}^p$

Robustness is a problem of smoothness

We can talk about smooth functions in terms of the $\varepsilon - \delta$ definition of continuity:

$$\forall \delta > 0 \exists \varepsilon > 0 \text{ s.t. } 0 < d_X(x, x') < \varepsilon \implies d_Y(f(x), f(x')) < \delta.$$

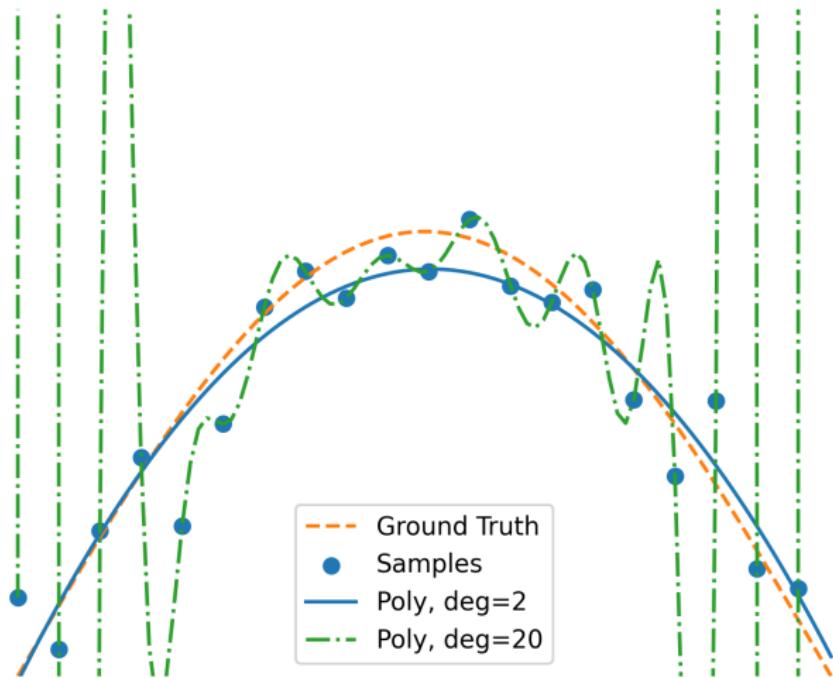
When the estimated function is not robust, we can find a $x' = x + \Delta x$ such that

$$0 < d_X(x, x') < \varepsilon \wedge d_Y(\hat{f}(x; \theta), \hat{f}(x'; \theta)) > \delta$$

or maybe more concisely put:

$$f(x) = f(x') \text{ but! } \hat{f}(x; \theta) \neq \hat{f}(x'; \theta)$$

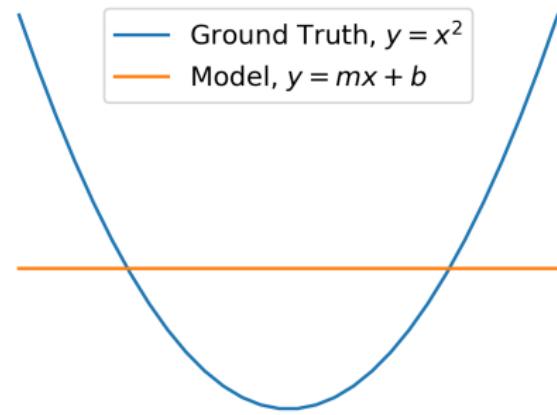
Example: Polynomial Fitting



Lack of Robustness Arise from Model Errors

Sources of error include:

1. Model Misspecification
2. Measurement Error
3. Dirty Data

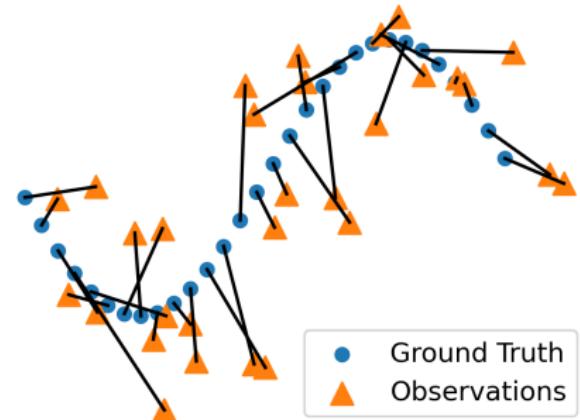


The model cannot capture the data generating distribution: the model class has the wrong inductive bias, the wrong inputs specified, or the wrong functional form specified.

Lack of Robustness Arise from Model Errors

Sources of error include:

1. Model Misspecification
2. Measurement Error
3. Dirty Data



$$\hat{x} = x + \eta_x$$

$$\hat{y} = f(x) + \eta_y$$

$$\eta_x \sim G_X, \eta_y \sim G_Y$$

Lack of Robustness Arise from Model Errors

Sources of error include:

1. Model Misspecification
2. Measurement Error
3. Dirty Data



“dog”

Dataset is not reflective of the generating distribution: inconsistent, mislabeled, or incomplete.

Lack of Robustness Arise from Model Errors

Sources of error include:

1. Model Misspecification
2. Measurement Error
3. Dirty Data

These problems can be exploited by
the *Adversary*.



“dog”

Dataset is not reflective of the generating distribution: inconsistent, mislabeled, or incomplete.

Formally Stating the Problem

Given a trained function estimator, $\hat{f}(x; \theta)$, $\exists \varepsilon \in \mathbb{R}^d$

1. A training set, $(x_i, y_i) \in X \times Y$, where $y_i = f(x_i)$
2. A loss function, $\ell : Y \times Y \rightarrow \mathbb{R}^+$ (less loss is better)
3. A function approximator, $\hat{f}(x; \theta)$, that is parameterized by $\theta \in \mathbb{R}^d$

Attacker's Goal: At test time, modify the data (x_i) to reduce test accuracy as much as possible.

Formally Stating the Problem

Given a trained function estimator, $\hat{f}(x; \theta)$, $\exists \varepsilon \in \mathbb{R}^d$

1. A training set, $(x_i, y_i) \in X \times Y$, where $y_i = f(x_i)$
2. A loss function, $\ell : Y \times Y \rightarrow \mathbb{R}^+$ (less loss is better)
3. A function approximator, $\hat{f}(x; \theta)$, that is parameterized by $\theta \in \mathbb{R}^d$

Attacker's Goal: At test time, modify the data (x_i) to reduce test accuracy as much as possible.

Formally Stating the Problem

Given a trained function estimator, $\hat{f}(x; \theta)$, $\exists \varepsilon \in \mathbb{R}^d$

1. A training set, $(x_i, y_i) \in X \times Y$, where $y_i = f(x_i)$
2. A loss function, $\ell : Y \times Y \rightarrow \mathbb{R}^+$ (less loss is better)
3. A function approximator, $\hat{f}(x; \theta)$, that is parameterized by $\theta \in \mathbb{R}^d$

Attacker's Goal: At test time, modify the data (x_i) to reduce test accuracy as much as possible.

Formally Stating the Problem

Given a trained function estimator, $\hat{f}(x; \theta)$, $\exists \varepsilon \in \mathbb{R}^d$

1. A training set, $(x_i, y_i) \in X \times Y$, where $y_i = f(x_i)$
2. A loss function, $\ell : Y \times Y \rightarrow \mathbb{R}^+$ (less loss is better)
3. A function approximator, $\hat{f}(x; \theta)$, that is parameterized by $\theta \in \mathbb{R}^d$

Attacker's Goal: At test time, modify the data (x_i) to reduce test accuracy as much as possible.

Attacker's Goal (Formally)

Given a trained model, $\hat{f}(x; \theta)$, and test example, x , and a perturbation budget, ε

Construct an adversarial example, $x' = x + \Delta x$, such that:

- ▶ $d_X(x, x') < \varepsilon$
- ▶ $f(x) = f(x')$
- ▶ $\hat{f}(x; \theta) \neq \hat{f}(x'; \theta)$ misclassifies the adversarial example.

Assuming:

- ▶ All parameters, θ , and structure of $\hat{f}(x; \theta)$ are visible, so we can compute gradient.
- ▶ The attack is untargeted, i.e., we just want the label to be different from the true label.

Attacker's Goal (Formally)

Given a trained model, $\hat{f}(x; \theta)$, and test example, x , and a perturbation budget, ε

Construct an adversarial example, $x' = x + \Delta x$, such that:

- ▶ $d_X(x, x') < \varepsilon$
- ▶ $f(x) = f(x')$
- ▶ $\hat{f}(x; \theta) \neq \hat{f}(x'; \theta)$ misclassifies the adversarial example.

Assuming:

- ▶ All parameters, θ , and structure of $\hat{f}(x; \theta)$ are visible, so we can compute gradient.
- ▶ The attack is untargeted, i.e., we just want the label to be different from the true label.

Attacker's Goal (Formally)

Given a trained model, $\hat{f}(x; \theta)$, and test example, x , and a perturbation budget, ε

Construct an adversarial example, $x' = x + \Delta x$, such that:

- ▶ $d_X(x, x') < \varepsilon \implies$ the difference between x and x' is small.
- ▶ $f(x) = f(x')$
- ▶ $\hat{f}(x; \theta) \neq \hat{f}(x'; \theta)$ misclassifies the adversarial example.

Assuming:

- ▶ All parameters, θ , and structure of $\hat{f}(x; \theta)$ are visible, so we can compute gradient.
- ▶ The attack is untargeted, i.e., we just want the label to be different from the true label.

Attacker's Goal (Formally)

Given a trained model, $\hat{f}(x; \theta)$, and test example, x , and a perturbation budget, ε

Construct an adversarial example, $x' = x + \Delta x$, such that:

- ▶ $d_X(x, x') < \varepsilon \implies$ the difference between x and x' is small.
- ▶ $f(x) = f(x')$
- ▶ $\hat{f}(x; \theta) \neq \hat{f}(x'; \theta)$ misclassifies the adversarial example.

Assuming:

- ▶ All parameters, θ , and structure of $\hat{f}(x; \theta)$ are visible, so we can compute gradient.
- ▶ The attack is untargeted, i.e., we just want the label to be different from the true label.

Attacker's Goal (Formally)

Given a trained model, $\hat{f}(x; \theta)$, and test example, x , and a perturbation budget, ε

Construct an adversarial example, $x' = x + \Delta x$, such that:

- ▶ $d_X(x, x') < \varepsilon \implies$ the difference between x and x' is small.
- ▶ $f(x) = f(x') \implies$ x and x' look the same
- ▶ $\hat{f}(x; \theta) \neq \hat{f}(x'; \theta)$ misclassifies the adversarial example.

Assuming:

- ▶ All parameters, θ , and structure of $\hat{f}(x; \theta)$ are visible, so we can compute gradient.
- ▶ The attack is untargeted, i.e., we just want the label to be different from the true label.

Attacker's Goal (Formally)

Given a trained model, $\hat{f}(x; \theta)$, and test example, x , and a perturbation budget, ε

Construct an adversarial example, $x' = x + \Delta x$, such that:

- ▶ $d_X(x, x') < \varepsilon \implies$ the difference between x and x' is small.
- ▶ $f(x) = f(x') \implies$ x and x' look the same
- ▶ $\hat{f}(x; \theta) \neq \hat{f}(x'; \theta)$ misclassifies the adversarial example.

Assuming:

- ▶ All parameters, θ , and structure of $\hat{f}(x; \theta)$ are visible, so we can compute gradient.
- ▶ The attack is untargeted, i.e., we just want the label to be different from the true label.

Attacker's Goal (Formally)

Given a trained model, $\hat{f}(x; \theta)$, and test example, x , and a perturbation budget, ε

Construct an adversarial example, $x' = x + \Delta x$, such that:

- ▶ $d_X(x, x') < \varepsilon \implies$ the difference between x and x' is small.
- ▶ $f(x) = f(x') \implies$ x and x' look the same
- ▶ $\hat{f}(x; \theta) \neq \hat{f}(x'; \theta) \implies$ the estimator misclassifies the adversarial example.

Assuming:

- ▶ All parameters, θ , and structure of $\hat{f}(x; \theta)$ are visible, so we can compute gradient.
- ▶ The attack is untargeted, i.e., we just want the label to be different from the true label.

Attacker's Goal (Formally)

Given a trained model, $\hat{f}(x; \theta)$, and test example, x , and a perturbation budget, ε

Construct an adversarial example, $x' = x + \Delta x$, such that:

- ▶ $d_X(x, x') < \varepsilon \implies$ the difference between x and x' is small.
- ▶ $f(x) = f(x') \implies$ x and x' look the same
- ▶ $\hat{f}(x; \theta) \neq \hat{f}(x'; \theta) \implies$ the estimator misclassifies the adversarial example.

Assuming:

- ▶ All parameters, θ , and structure of $\hat{f}(x; \theta)$ are visible, so we can compute gradient.
- ▶ The attack is untargeted, i.e., we just want the label to be different from the true label.

Attacker's Goal (Formally)

Given a trained model, $\hat{f}(x; \theta)$, and test example, x , and a perturbation budget, ε

Construct an adversarial example, $x' = x + \Delta x$, such that:

- ▶ $d_X(x, x') < \varepsilon \implies$ the difference between x and x' is small.
- ▶ $f(x) = f(x') \implies$ x and x' look the same
- ▶ $\hat{f}(x; \theta) \neq \hat{f}(x'; \theta) \implies$ the estimator misclassifies the adversarial example.

Assuming:

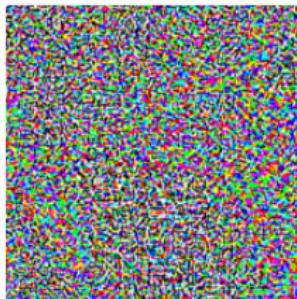
- ▶ All parameters, θ , and structure of $\hat{f}(x; \theta)$ are visible, so we can compute gradient.
- ▶ The attack is untargeted, *i.e.*, we just want the label to be different from the true label.

Example: Adversarial Example



x
“panda”
57.7% confidence

+ .007 ×



$\text{sign}(\nabla_x J(\theta, x, y))$
“nematode”
8.2% confidence

=



$x + \epsilon \text{sign}(\nabla_x J(\theta, x, y))$
“gibbon”
99.3 % confidence

Image source: Goodfellow IJ, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572. 2014 Dec 20.

How do we ensure $f(x) = f(x')$?

Where human judgement is the source of training data, $f(x)$ is a mathematical representation of an oracle that the attacker either does not (e.g., human-in-the-loop training too costly/slow) or should not (e.g., trying to act surreptitiously) have access to.

Instead we are going to use a metric over the data to try and find “small” perturbations. In this lecture we will use an L_p norm:

$$d_X(x, x') := \|x - x'\|_p = \left(\sum_{i=1}^d (x_i - x'_i)^p \right)^{\frac{1}{p}}$$

This is a proxy for how similar the oracle would judge the inputs.

How do we ensure $f(x) = f(x')$?

Where human judgement is the source of training data, $f(x)$ is a mathematical representation of an oracle that the attacker either does not (e.g., human-in-the-loop training too costly/slow) or should not (e.g., trying to act surreptitiously) have access to.

Instead we are going to use a metric over the data to try and find “small” perturbations. In this lecture we will use an L_p norm:

$$d_X(x, x') := \|x - x'\|_p = \left(\sum_{i=1}^d (x_i - x'_i)^p \right)^{\frac{1}{p}}$$

This is a proxy for how similar the oracle would judge the inputs.

How do we ensure $f(x) = f(x')$?

Where human judgement is the source of training data, $f(x)$ is a mathematical representation of an oracle that the attacker either does not (e.g., human-in-the-loop training too costly/slow) or should not (e.g., trying to act surreptitiously) have access to.

Instead we are going to use a metric over the data to try and find “small” perturbations. In this lecture we will use an L_p norm:

$$d_X(x, x') := \|x - x'\|_p = \left(\sum_{i=1}^d (x_i - x'_i)^p \right)^{\frac{1}{p}}$$

This is a proxy for how similar the oracle would judge the inputs.

How do we ensure $f(x) = f(x')$?

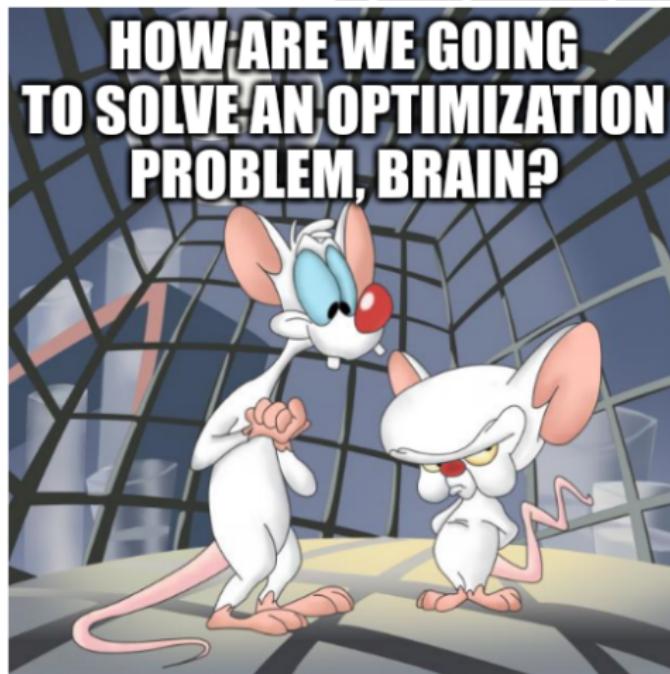
Where human judgement is the source of training data, $f(x)$ is a mathematical representation of an oracle that the attacker either does not (e.g., human-in-the-loop training too costly/slow) or should not (e.g., trying to act surreptitiously) have access to.

Instead we are going to use a metric over the data to try and find “small” perturbations. In this lecture we will use an L_p norm:

$$d_X(x, x') := \|x - x'\|_p = \left(\sum_{i=1}^d (x_i - x'_i)^p \right)^{\frac{1}{p}}$$

This is a proxy for how similar the oracle would judge the inputs.

How to Find the Perturbation, Δx ?



**HOW ARE WE GOING
TO SOLVE AN OPTIMIZATION
PROBLEM, BRAIN?**

**THE SAME WAY WE SOLVE
EVERY* OPTIMIZATION PROBLEM
PINKY, WITH GRADIENT BASED METHODS!**

How to Find the Perturbation, Δx ?

Solving the model fitting problem

$$\arg \min_{\theta \in \Theta} \sum_{i=1}^N \ell(\hat{f}(x_i; \theta), y_i)$$

with gradient *descent* on θ

$$\theta_{t+1} = \theta_t - \gamma \sum_{i=1}^N \nabla_{\theta} \ell(\hat{f}(x_i; \theta_t), y_i)$$

How to Find the Perturbation, Δx ?

Solving the model fitting problem

$$\arg \min_{\theta \in \Theta} \sum_{i=1}^N \ell(\hat{f}(x_i; \theta), y_i)$$

with gradient *descent* on θ

$$\theta_{t+1} = \theta_t - \gamma \sum_{i=1}^N \nabla_{\theta} \ell(\hat{f}(x_i; \theta_t), y_i)$$

Solve the untargeted adversarial problem

$$\arg \max_{\Delta x \in \mathbb{R}^d} \ell(\hat{f}(x_i + \Delta x; \theta), y_i)$$

$$\text{s.t. } d_X(x_i + \Delta x, x_i) < \varepsilon$$

with gradient *ascent* on Δx

$$\Delta x_{t+1} = \Delta x_t + \nabla_{\Delta x} \ell(\hat{f}(x_i + \Delta x_t; \theta), y_i)$$

constrained by the perturbation budget, ε

How to Find the Perturbation, Δx ?

Solving the model fitting problem

$$\arg \min_{\theta \in \Theta} \sum_{i=1}^N \ell(\hat{f}(x_i; \theta), y_i)$$

with gradient *descent* on θ

$$\theta_{t+1} = \theta_t - \gamma \sum_{i=1}^N \nabla_{\theta} \ell(\hat{f}(x_i; \theta_t), y_i)$$

We will choose our distance metric carefully to make the constraint satisfaction easy.

Solve the untargeted adversarial problem

$$\arg \max_{\Delta x \in \mathbb{R}^d} \ell(\hat{f}(x_i + \Delta x; \theta), y_i)$$

$$\text{s.t. } d_X(x_i + \Delta x, x_i) < \varepsilon$$

with gradient *ascent* on Δx

$$\Delta x_{t+1} = \Delta x_t + \nabla_{\Delta x} \ell(\hat{f}(x_i + \Delta x_t; \theta), y_i)$$

constrained by the perturbation budget, ε

What Kinds of Norms Are There?

- ▶ $L_0 : \|x - x'\|_0 = \sum_{i=1}^d \mathbb{1}((x_i - x'_i) \neq 0) \implies$ the attacker can change ε elements arbitrarily
- ▶ $L_2 : \|x - x'\|_2 = \sqrt{\sum_{i=1}^d (x_i - x'_i)^2} \implies$ the adversarial example must live in a ball of radius ε around the test point
- ▶ $L_\infty : \|x - x'\|_\infty = \max\{|x_i - x'_i|\} \implies$ the attacker can change any (and every) element of x by at most ε

There are other distance metrics we can use, e.g., Cosine distance, Earth Mover's (Wasserstein) Distance, Geometric Transforms, but we won't today.

The best distance metric is the one that maximizes your felicity.

What Kinds of Norms Are There?

- ▶ $L_0 : \|x - x'\|_0 = \sum_{i=1}^d \mathbb{1}((x_i - x'_i) \neq 0) \implies$ the attacker can change ε elements arbitrarily
- ▶ $L_2 : \|x - x'\|_2 = \sqrt{\sum_{i=1}^d (x_i - x'_i)^2} \implies$ the adversarial example must live in a ball of radius ε around the test point
- ▶ $L_\infty : \|x - x'\|_\infty = \max\{|x_i - x'_i|\} \implies$ the attacker can change any (and every) element of x by at most ε

There are other distance metrics we can use, e.g., Cosine distance, Earth Mover's (Wasserstein) Distance, Geometric Transforms, but we won't today.

The best distance metric is the one that maximizes your felicity.

What Kinds of Norms Are There?

- ▶ $L_0 : \|x - x'\|_0 = \sum_{i=1}^d \mathbb{1}((x_i - x'_i) \neq 0) \implies$ the attacker can change ε elements arbitrarily
- ▶ $L_2 : \|x - x'\|_2 = \sqrt{\sum_{i=1}^d (x_i - x'_i)^2} \implies$ the adversarial example must live in a ball of radius ε around the test point
- ▶ $L_\infty : \|x - x'\|_\infty = \max\{|x_i - x'_i|\} \implies$ the attacker can change any (and every) element of x by at most ε

There are other distance metrics we can use, e.g., Cosine distance, Earth Mover's (Wasserstein) Distance, Geometric Transforms, but we won't today.

The best distance metric is the one that maximizes your felicity.

What Kinds of Norms Are There?

- ▶ $L_0 : \|x - x'\|_0 = \sum_{i=1}^d \mathbb{1}((x_i - x'_i) \neq 0) \implies$ the attacker can change ε elements arbitrarily
- ▶ $L_2 : \|x - x'\|_2 = \sqrt{\sum_{i=1}^d (x_i - x'_i)^2} \implies$ the adversarial example must live in a ball of radius ε around the test point
- ▶ $L_\infty : \|x - x'\|_\infty = \max\{|x_i - x'_i|\} \implies$ the attacker can change any (and every) element of x by at most ε

There are other distance metrics we can use, e.g., Cosine distance, Earth Mover's (Wasserstein) Distance, Geometric Transforms, but we won't today.

The best distance metric is the one that maximizes your felicity.

Fast Gradient Sign Method

Algorithm Fast Gradient Sign Method (FGSM)

```
1: procedure FGSM(model  $\theta$ ; data  $x$ ; label  $y$ )
2:    $x_0 \leftarrow x$ 
3:   for  $t = 1, \dots, T$  do
4:      $x_t \leftarrow x_{t-1} + \varepsilon \cdot \text{sign}(\nabla_x f(x_{t-1}, y; \theta))$ 
5:      $x_t \leftarrow P(x_t)$ 
```

- ▶ Goal: Optimize inputs to *maximize* loss (*i.e.*, gradient ascent)
- ▶ Observe: Because we are using L_∞ norm, can change all elements by ε .
- ▶ $P(x)$ projects x back into the ε -box centred at x

A effective algorithm, but simple, and does not permit the gradient to control the size of the steps taken along any axis.

Fast Gradient Sign Method

Algorithm Fast Gradient Sign Method (FGSM)

```
1: procedure FGSM(model  $\theta$ ; data  $x$ ; label  $y$ )
2:    $x_0 \leftarrow x$ 
3:   for  $t = 1, \dots, T$  do
4:      $x_t \leftarrow x_{t-1} + \varepsilon \cdot \text{sign}(\nabla_x f(x_{t-1}, y; \theta))$ 
5:      $x_t \leftarrow P(x_t)$ 
```

- ▶ Goal: Optimize inputs to *maximize* loss (*i.e.*, gradient ascent)
- ▶ Observe: Because we are using L_∞ norm, can change all elements by ε .
- ▶ $P(x)$ projects x back into the ε -box centred at x

A effective algorithm, but simple, and does not permit the gradient to control the size of the steps taken along any axis.

Projected Gradient Descent

Algorithm Projected Gradient Descent (PGD)

```
1: procedure PGD(model  $\theta$ ; data  $x$ ; label  $y$ ;  
   learning rate,  $\eta$ )  
2:    $x_0 \leftarrow x$   
3:   for  $t = 1, \dots, T$  do  
4:      $x_t \leftarrow x_{t-1} + \eta \nabla_x f(x_{t-1}, y; \theta)$   
5:      $x_t \leftarrow P(x_t)$ 
```

- ▶ Goal: Optimize inputs to *maximize* loss (*i.e.*, gradient ascent)
- ▶ Observe: Don't threshold the gradient, allowing for smaller steps.

Projected Gradient Descent

Algorithm Projected Gradient Descent (PGD)

```
1: procedure PGD(model  $\theta$ ; data  $x$ ; label  $y$ ;  
   learning rate,  $\eta$ )  
2:    $x_0 \leftarrow x$   
3:   for  $t = 1, \dots, T$  do  
4:      $x_t \leftarrow x_{t-1} + \eta \nabla_x f(x_{t-1}, y; \theta)$   
5:      $x_t \leftarrow P(x_t)$ 
```

- ▶ Goal: Optimize inputs to *maximize* loss (*i.e.*, gradient ascent)
- ▶ Observe: Don't threshold the gradient, allowing for smaller steps.

A more powerful algorithm that leads to better attacks, *i.e.*, attacks requiring a smaller perturbation budget

Turning *Untargeted Attacks* into *Targeted Attacks*

If you don't want to just mess up the classifier, but lead it astray in a particular way we start with the untargeted optimization problem:

$$\arg \max_{\Delta x \in \mathbb{R}^d} \ell(\hat{f}(x + \Delta x; \theta), y) \text{ s.t } d_X(x + \Delta x, x) < \varepsilon.$$

then we augment the loss function with a term that drives classification towards the desired label, c :

$$\arg \max_{\Delta x \in \mathbb{R}^d} \ell(\hat{f}(x + \Delta x; \theta), y) - \ell(\hat{f}(x + \Delta x; \theta), c) \text{ s.t } d_X(x + \Delta x, x) < \varepsilon.$$

Turning *Untargeted Attacks* into *Targeted Attacks*

If you don't want to just mess up the classifier, but lead it astray in a particular way we start with the untargeted optimization problem:

$$\arg \max_{\Delta x \in \mathbb{R}^d} \ell(\hat{f}(x + \Delta x; \theta), y) \text{ s.t } d_X(x + \Delta x, x) < \varepsilon.$$

then we augment the loss function with a term that drives classification towards the desired label, c :

$$\arg \max_{\Delta x \in \mathbb{R}^d} \ell(\hat{f}(x + \Delta x; \theta), y) - \ell(\hat{f}(x + \Delta x; \theta), c) \text{ s.t } d_X(x + \Delta x, x) < \varepsilon.$$

Turning *Untargeted Attacks* into *Targeted Attacks*

If you don't want to just mess up the classifier, but lead it astray in a particular way we start with the untargeted optimization problem:

$$\arg \max_{\Delta x \in \mathbb{R}^d} \ell(\hat{f}(x + \Delta x; \theta), y) \text{ s.t. } d_X(x + \Delta x, x) < \varepsilon.$$

then we augment the loss function with a term that drives classification towards the desired label, c :

$$\arg \max_{\Delta x \in \mathbb{R}^d} \ell(\hat{f}(x + \Delta x; \theta), y) - \ell(\hat{f}(x + \Delta x; \theta), c) \text{ s.t. } d_X(x + \Delta x, x) < \varepsilon.$$

Defence Against Attacks: Adversarial Training

Usual goal: $\min_{\theta \in \Theta} \sum_{i=1}^N \ell(\hat{f}(x_i; \theta), y_i)$

Robust setting: $\min_{\theta \in \Theta} \sum_{i=1}^N \left(\max_{\Delta x_i: \|\Delta x\|_\infty < \varepsilon} \ell(\hat{f}(x_i + \Delta x_i; \theta), y_i) \right)$

To solve for θ , we will:

1. Draw minibatch, B_t

2. For each $(x_i, y_i) \in B_t$ compute

$$\Delta x_i^* = \arg \max_{\Delta x_i: \|\Delta x_i\| < \varepsilon} \ell(\hat{f}(x_i + \Delta x_i; \theta_t), y_i)$$

3. Update $\theta_t \leftarrow \theta_{t-1} - \gamma \frac{1}{\|B_t\|} \sum_{(x_i, y_i) \in B_t} \nabla_{\theta} \ell(\hat{f}(x_i + \Delta x_i^*; \theta_{t-1}), y_i)$
4. Repeat until stopping criterion

Defence Against Attacks: Adversarial Training

Usual goal: $\min_{\theta \in \Theta} \sum_{i=1}^N \ell(\hat{f}(x_i; \theta), y_i)$

Robust setting: $\min_{\theta \in \Theta} \sum_{i=1}^N \left(\max_{\Delta x_i: \|\Delta x\|_\infty < \varepsilon} \ell(\hat{f}(x_i + \Delta x_i; \theta), y_i) \right)$

To solve for θ , we will:

1. Draw minibatch, B_t

2. For each $(x_i, y_i) \in B_t$ compute

$$\Delta x_i^* = \arg \max_{\Delta x_i: \|\Delta x_i\| < \varepsilon} \ell(\hat{f}(x_i + \Delta x_i; \theta_t), y_i)$$

3. Update $\theta_t \leftarrow \theta_{t-1} - \gamma \frac{1}{\|B_t\|} \sum_{(x_i, y_i) \in B_t} \nabla_{\theta} \ell(\hat{f}(x_i + \Delta x_i^*; \theta_{t-1}), y_i)$

4. Repeat until stopping criterion

Defence Against Attacks: Adversarial Training

Usual goal: $\min_{\theta \in \Theta} \sum_{i=1}^N \ell(\hat{f}(x_i; \theta), y_i)$

Robust setting: $\min_{\theta \in \Theta} \sum_{i=1}^N \left(\max_{\Delta x_i: \|\Delta x\|_\infty < \varepsilon} \ell(\hat{f}(x_i + \Delta x_i; \theta), y_i) \right)$

To solve for θ , we will:

1. Draw minibatch, B_t

2. For each $(x_i, y_i) \in B_t$ compute

$$\Delta x_i^* = \arg \max_{\Delta x_i: \|\Delta x_i\| < \varepsilon} \ell(\hat{f}(x_i + \Delta x_i; \theta_t), y_i)$$

3. Update $\theta_t \leftarrow \theta_{t-1} - \gamma \frac{1}{\|B_t\|} \sum_{(x_i, y_i) \in B_t} \nabla_{\theta} \ell(\hat{f}(x_i + \Delta x_i^*; \theta_{t-1}), y_i)$
4. Repeat until stopping criterion

Attacks on Defenses: Obfuscated Gradients

Adversarial Attacks work on some methods designed to resist,

Defense	Dataset	Distance	Accuracy
Buckman et al. (2018)	CIFAR	0.031 (ℓ_∞)	0%*
Ma et al. (2018)	CIFAR	0.031 (ℓ_∞)	5%
Guo et al. (2018)	ImageNet	0.005 (ℓ_2)	0%*
Dhillon et al. (2018)	CIFAR	0.031 (ℓ_∞)	0%
Xie et al. (2018)	ImageNet	0.031 (ℓ_∞)	0%*
Song et al. (2018)	CIFAR	0.031 (ℓ_∞)	9%*
Samangouei et al. (2018)	MNIST	0.005 (ℓ_2)	55%**
Madry et al. (2018)	CIFAR	0.031 (ℓ_∞)	47%
Na et al. (2018)	CIFAR	0.015 (ℓ_∞)	15%

Table source: Athalye A, Carlini N, Wagner D. "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples." In International conference on machine learning 2018 Jul 3 (pp. 274-283). PMLR.

Backdoor Attacks

Adversary can inject *triggers* in outsourced training datasets.



Liu K, Dolan-Gavitt B, Garg S. "Fine-pruning: Defending against backdooring attacks on deep neural networks." In Intl symp on research in attacks, intrusions, and defenses 2018 Sep 7 (pp. 273-294).

But what do you do about this?



Granny Smith	85.6%
iPod	0.4%
library	0.0%
pizza	0.0%
toaster	0.0%
dough	0.1%



Granny Smith	0.1%
iPod	99.7%
library	0.0%
pizza	0.0%
toaster	0.0%
dough	0.0%

Or this?

GSM-NoOp

Oliver picks 44 kiwis on Friday. Then he picks 58 kiwis on Saturday. On Sunday, he picks double the number of kiwis he did on Friday, but five of them were a bit smaller than average. How many kiwis does Oliver have?

o1-mini: Oliver picks kiwis over three days as follows:

- Friday: 44 kiwis
- Saturday: 58 kiwis
- Sunday: Double the number he picked on Friday, which is $2 \times 44 = 88$ kiwis

However, on Sunday, 5 of these kiwis were smaller than average. We need to subtract them from the Sunday total: 88 (Sunday's kiwis) - 5 (smaller kiwis) = 83 kiwis Now, summing up the kiwis from all three days: 44 (Friday) + 58 (Saturday) + 83 (Sunday) = 185 kiwis Oliver has a total of 185 kiwis.

Llama3-8B: Let's break it down step by step:

Oliver picks 44 kiwis on Friday. He picks 58 kiwis on Saturday. On Sunday, he picks double the number of kiwis he did on Friday, which is $2 \times 44 = 88$ kiwis.

Five of the kiwis picked on Sunday are a bit smaller than average, so we subtract 5 from the total number of kiwis picked on Sunday: $88 - 5 = 83$ kiwis. Now, let's add up the total number of kiwis Oliver has:

$$44 \text{ (Friday)} + 58 \text{ (Saturday)} + 83 \text{ (Sunday)} = 185 \text{ kiwis}$$

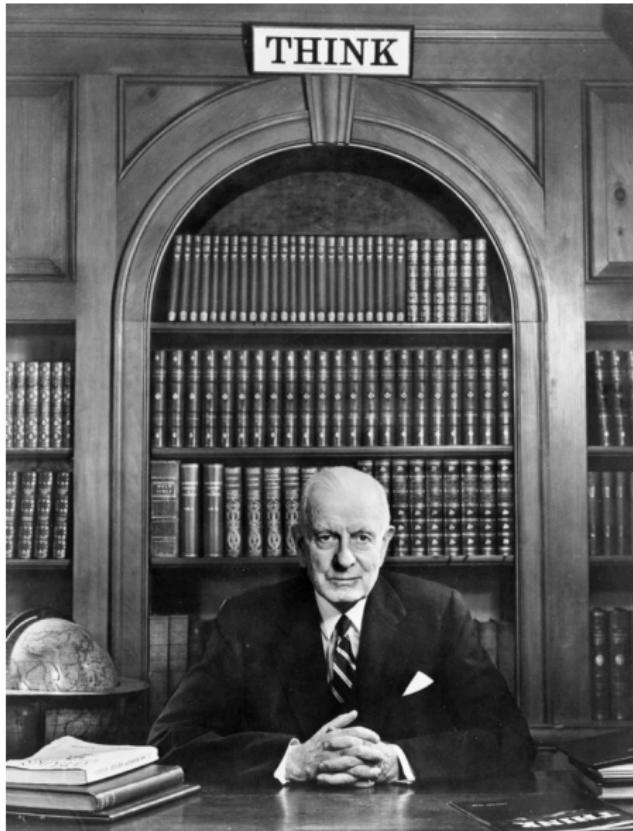
So, Oliver has 185 kiwis in total.

Figure source: Mirzadeh I, Alizadeh K, Shahrokhi H, Tuzel O, Bengio S, Farajtabar M. "GSM-symbolic: Understanding the limitations of mathematical reasoning in large language models." arXiv preprint arXiv:2410.05229. 2024 Oct 7. [PDF]

**JUST ONE MORE PETABYTE BRO
ONE MORE PETABYTE WILL FIX IT**



Alternatively:



Resources

- Wiyatno, Rey Reza, et al. “Adversarial examples in modern machine learning: A review.” arXiv preprint arXiv:1911.05268 (2019). [PDF]