

Security and Privacy of Machine Learning

Bogdan Kulynych

July 2019

Table of contents

1. Security and Privacy of ML?
2. Modern Security Principles
3. Adversary Models for Security and Privacy of ML
4. Active Attacks at Test Time
5. Dealing with Black Boxes
6. Attacks at Training Time
7. Privacy and Machine Learning
8. Privacy and Security against Machine Learning

Security and Privacy of ML?

The Holy Grail

$$\theta^* = \arg \min_{\theta} \mathbb{E}_{(x,y) \sim \text{i.i.d. } D} L(x, y; \theta)$$



Image Credit: Sharif et al. [2016]

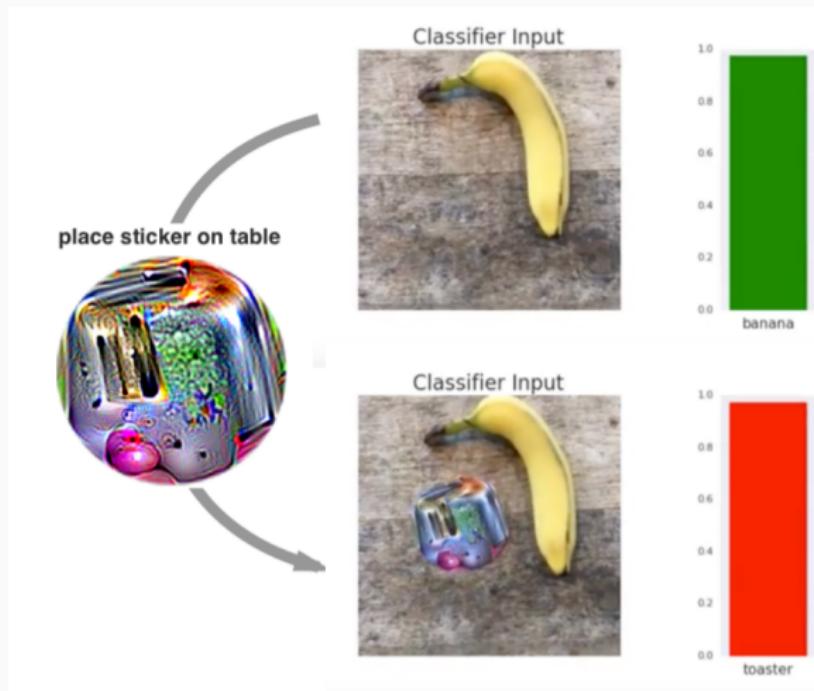


Image Credit: Brown et al. [2017]



Image Credit: Song et al. [2018]



Image Credit: Gu et al. [2019]

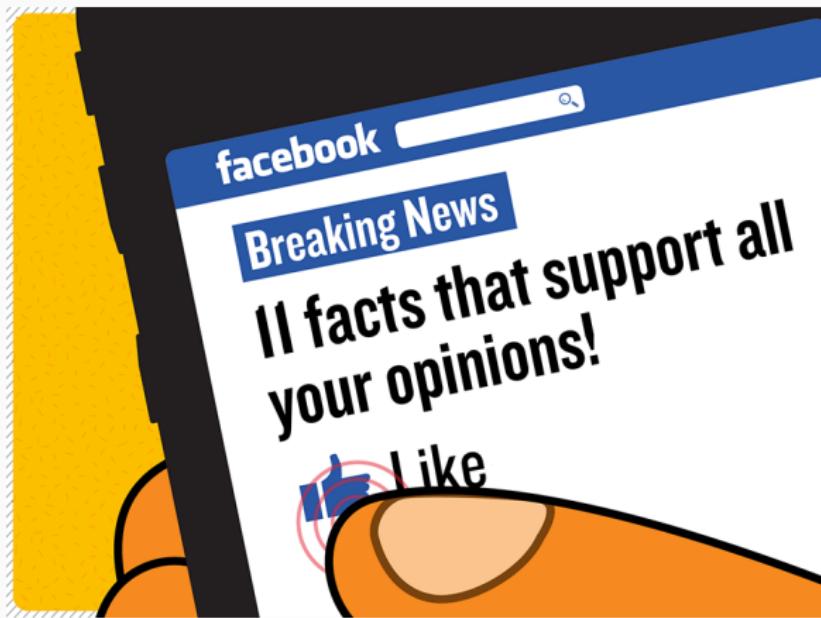




Image Credit: Fredrikson et al. [2015]



Modern Security Principles

Image Perturbations

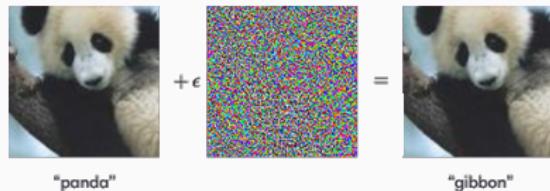


Image Credit: Goodfellow et al. [2014]

Imagine an algorithm $\text{Magic}(f, x, \tilde{y}) \mapsto \tilde{x}$:

- Takes any computer vision classifier $f(x)$.
- Takes any image x and target label \tilde{y} .
- Magically perturbs the image in an imperceptible way: \tilde{x} .
- Resulting image is missclassified by f as anything you want (\tilde{y}).

Image Perturbations

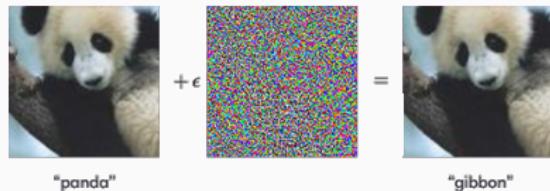


Image Credit: Goodfellow et al. [2014]

Imagine an algorithm $\text{Magic}(f, x, \tilde{y}) \mapsto \tilde{x}$:

- Takes any computer vision classifier $f(x)$.
- **Takes any image x and target label \tilde{y} .**
- Magically perturbs the image in an imperceptible way: \tilde{x} .
- Resulting image is missclassified by f as anything you want (\tilde{y}).

Image Perturbations

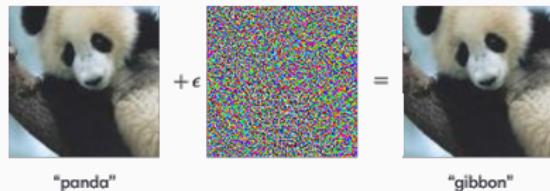


Image Credit: Goodfellow et al. [2014]

Imagine an algorithm $\text{Magic}(f, x, \tilde{y}) \mapsto \tilde{x}$:

- Takes any computer vision classifier $f(x)$.
- Takes any image x and target label \tilde{y} .
- Magically perturbs the image in an imperceptible way: \tilde{x} .
- Resulting image is missclassified by f as anything you want (\tilde{y}).

Image Perturbations

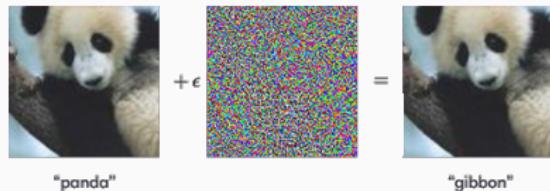


Image Credit: Goodfellow et al. [2014]

Imagine an algorithm $\text{Magic}(f, x, \tilde{y}) \mapsto \tilde{x}$:

- Takes any computer vision classifier $f(x)$.
- Takes any image x and target label \tilde{y} .
- Magically perturbs the image in an imperceptible way: \tilde{x} .
- Resulting image is missclassified by f as anything you want (\tilde{y}).

Image Perturbations

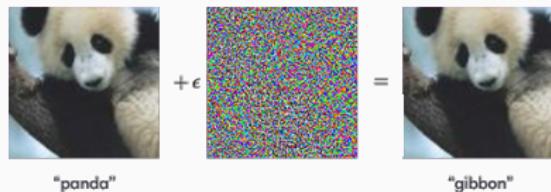


Image Credit: Goodfellow et al. [2014]

Imagine an algorithm $\text{Magic}(f, x, \tilde{y}) \mapsto \tilde{x}$:

- Takes any computer vision classifier $f(x)$.
- Takes any image x and target label \tilde{y} .
- Magically perturbs the image in an imperceptible way: \tilde{x} .
- Resulting image is missclassified by f as anything you want (\tilde{y}).

Group discussion

Is this an attack?

Context Matters

- Who performs the attack? What settings?
- What can the attacker do? What are the costs?
- What does the attacker know about the target?
- If the attack succeeds, why does it matter?
- In general: need *context*.
- Many methodologies to formalize these.

Context Matters

- Who performs the attack? What settings?
- What can the attacker do? What are the costs?
- What does the attacker know about the target?
- If the attack succeeds, why does it matter?
- In general: need *context*.
- Many methodologies to formalize these.

Context Matters

- Who performs the attack? What settings?
- What can the attacker do? What are the costs?
- **What does the attacker know about the target?**
- If the attack succeeds, why does it matter?
- In general: need *context*.
- Many methodologies to formalize these.

Context Matters

- Who performs the attack? What settings?
- What can the attacker do? What are the costs?
- What does the attacker know about the target?
- If the attack succeeds, why does it matter?
- In general: need *context*.
- Many methodologies to formalize these.

Context Matters

- Who performs the attack? What settings?
- What can the attacker do? What are the costs?
- What does the attacker know about the target?
- If the attack succeeds, why does it matter?
- In general: need *context*.
- Many methodologies to formalize these.

Context Matters

- Who performs the attack? What settings?
- What can the attacker do? What are the costs?
- What does the attacker know about the target?
- If the attack succeeds, why does it matter?
- In general: need *context*.
- Many methodologies to formalize these.

Generic High-Level Method: Threat Model

Threat Model

- **Threat:** *What* are the bad things that can happen
 - Spammers can get past your spam filter
- **Risk:** *How much* the exploitation of the threat will cost you
 - People stop using your email service
- **Attack:** *How* can the threat be exploited
- This approach is useful for assessing risks, and prioritizing threats.

Generic High-Level Method: Threat Model

Threat Model

- **Threat:** *What* are the bad things that can happen
 - Spammers can get past your spam filter
- **Risk:** *How much* the exploitation of the threat will cost you
 - People stop using your email service
- **Attack:** *How* can the threat be exploited
- This approach is useful for assessing risks, and prioritizing threats.

Generic High-Level Method: Threat Model

Threat Model

- **Threat:** *What* are the bad things that can happen
 - Spammers can get past your spam filter
- **Risk:** *How much* the exploitation of the threat will cost you
 - People stop using your email service
- **Attack:** *How* can the threat be exploited
- This approach is useful for assessing risks, and prioritizing threats.

Generic High-Level Method: Threat Model

Threat Model

- **Threat:** *What* are the bad things that can happen
 - Spammers can get past your spam filter
- **Risk:** *How much* the exploitation of the threat will cost you
 - People stop using your email service
- **Attack:** *How* can the threat be exploited
- This approach is useful for assessing risks, and prioritizing threats.

Generic High-Level Method: Threat Model

Threat Model

- **Threat:** *What* are the bad things that can happen
 - Spammers can get past your spam filter
 - **Risk:** *How much* the exploitation of the threat will cost you
 - People stop using your email service
 - **Attack:** *How* can the threat be exploited
-
- This approach is useful for assessing risks, and prioritizing threats.

Generic High-Level Method: Threat Model

Threat Model

- **Threat:** *What* are the bad things that can happen
 - Spammers can get past your spam filter
 - **Risk:** *How much* the exploitation of the threat will cost you
 - People stop using your email service
 - **Attack:** *How* can the threat be exploited
-
- This approach is useful for assessing risks, and prioritizing threats.

Working Method: Adversary Model

Adversary Model

- “Know the enemy and know yourself”
 - Sun Tzu, “Art of War”, 500 BC
- **Adversarial Goals:** What does the adversary *want to achieve*
- **Adversarial Capabilities:** What does the adversary *can do*
- **Adversarial Knowledge:** What does the *adversary know about you and your system*
- More detailed than the previous model
- Enables comparisons of attacks, and principled approach to *defenses*
- Phrase “threat model” and “adversary model” are often used interchangeably. Normally people mean “adversary model”
- Impact (risks) of the attack is usually implicit, but we must keep them in mind.

Working Method: Adversary Model

Adversary Model

- “Know the enemy and know yourself”
 - Sun Tzu, “Art of War”, 500 BC
 - **Adversarial Goals:** What does the adversary *want to achieve*
 - **Adversarial Capabilities:** What does the adversary *can do*
 - **Adversarial Knowledge:** What does the adversary *know* about you and your system
-
- More detailed than the previous model
 - Enables comparisons of attacks, and principled approach to *defenses*
 - Phrase “threat model” and “adversary model” are often used interchangeably. Normally people mean “adversary model”
 - Impact (risks) of the attack is usually implicit, but we must keep them in mind.

Working Method: Adversary Model

Adversary Model

- “Know the enemy and know yourself”
 - Sun Tzu, “Art of War”, 500 BC
 - **Adversarial Goals:** What does the adversary *want to achieve*
 - **Adversarial Capabilities:** What does the adversary *can do*
 - **Adversarial Knowledge:** What does the adversary *know* about you and your system
-
- More detailed than the previous model
 - Enables comparisons of attacks, and principled approach to *defenses*
 - Phrase “threat model” and “adversary model” are often used interchangeably. Normally people mean “adversary model”
 - Impact (risks) of the attack is usually implicit, but we must keep them in mind.

Working Method: Adversary Model

Adversary Model

- “Know the enemy and know yourself”
 - Sun Tzu, “Art of War”, 500 BC
 - **Adversarial Goals:** What does the adversary *want to achieve*
 - **Adversarial Capabilities:** What does the adversary *can do*
 - **Adversarial Knowledge:** What does the *adversary know about you and your system*
-
- More detailed than the previous model
 - Enables comparisons of attacks, and principled approach to *defenses*
 - Phrase “threat model” and “adversary model” are often used interchangeably. Normally people mean “adversary model”
 - Impact (risks) of the attack is usually implicit, but we must keep them in mind.

Working Method: Adversary Model

Adversary Model

- “Know the enemy and know yourself”
 - Sun Tzu, “Art of War”, 500 BC
 - **Adversarial Goals:** What does the adversary *want to achieve*
 - **Adversarial Capabilities:** What does the adversary *can do*
 - **Adversarial Knowledge:** What does the *adversary know* about you and your system
-
- More detailed than the previous model
 - Enables comparisons of attacks, and principled approach to *defenses*
 - Phrase “threat model” and “adversary model” are often used interchangeably. Normally people mean “adversary model”
 - Impact (risks) of the attack is usually implicit, but we must keep them in mind.

Working Method: Adversary Model

Adversary Model

- “Know the enemy and know yourself”
 - Sun Tzu, “Art of War”, 500 BC
 - **Adversarial Goals:** What does the adversary *want to achieve*
 - **Adversarial Capabilities:** What does the adversary *can do*
 - **Adversarial Knowledge:** What does the *adversary know* about you and your system
-
- More detailed than the previous model
 - Enables comparisons of attacks, and principled approach to *defenses*
 - Phrase “threat model” and “adversary model” are often used interchangeably. Normally people mean “adversary model”
 - Impact (risks) of the attack is usually implicit, but we must keep them in mind.

Working Method: Adversary Model

Adversary Model

- “Know the enemy and know yourself”
 - Sun Tzu, “Art of War”, 500 BC
 - **Adversarial Goals:** What does the adversary *want to achieve*
 - **Adversarial Capabilities:** What does the adversary *can do*
 - **Adversarial Knowledge:** What does the adversary *know* about you and your system
-
- More detailed than the previous model
 - Enables comparisons of attacks, and principled approach to *defenses*
 - Phrase “threat model” and “adversary model” are often used interchangeably. Normally people mean “adversary model”
 - Impact (risks) of the attack is usually implicit, but we must keep them in mind.

Working Method: Adversary Model

Adversary Model

- “Know the enemy and know yourself”
 - Sun Tzu, “Art of War”, 500 BC
 - **Adversarial Goals:** What does the adversary *want to achieve*
 - **Adversarial Capabilities:** What does the adversary *can do*
 - **Adversarial Knowledge:** What does the adversary *know* about you and your system
-
- More detailed than the previous model
 - Enables comparisons of attacks, and principled approach to *defenses*
 - Phrase “threat model” and “adversary model” are often used interchangeably. Normally people mean “adversary model”
 - Impact (risks) of the attack is usually implicit, but we must keep them in mind.

Adversary Model Example: Spam Email

- **Adversarial Goals:** Get an email past the spam detector.
- **Adversarial Capabilities:**
 - Modify email content as long as some meaning is still preserved
 - Send emails from different addresses
- **Adversarial Knowledge:** Does not know how detector works, but observes if the email was classified as spam or not spam

Adversary Model Example: Spam Email

- **Adversarial Goals:** Get an email past the spam detector.
- **Adversarial Capabilities:**
 - Modify email content as long as some meaning is still preserved
 - Send emails from different addresses
- **Adversarial Knowledge:** Does not know how detector works, but observes if the email was classified as spam or not spam

Adversary Model Example: Spam Email

- **Adversarial Goals:** Get an email past the spam detector.
- **Adversarial Capabilities:**
 - Modify email content as long as some meaning is still preserved
 - Send emails from different addresses
- **Adversarial Knowledge:** Does not know how detector works, but observes if the email was classified as spam or not spam

Adversary Model Example: Spam Email

- **Adversarial Goals:** Get an email past the spam detector.
- **Adversarial Capabilities:**
 - Modify email content as long as some meaning is still preserved
 - Send emails from different addresses
- **Adversarial Knowledge:** Does not know how detector works, but observes if the email was classified as spam or not spam

Adversary Model Example: Spam Email

- **Adversarial Goals:** Get an email past the spam detector.
- **Adversarial Capabilities:**
 - Modify email content as long as some meaning is still preserved
 - Send emails from different addresses
- **Adversarial Knowledge:** Does not know how detector works, but observes if the email was classified as spam or not spam

Recall: Image Perturbations

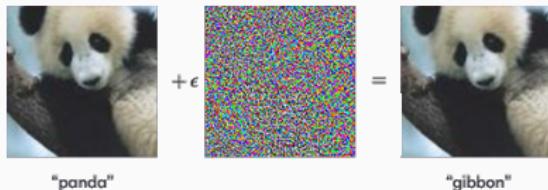


Image Credit: Goodfellow et al. [2014]

Imagine an algorithm $\text{Magic}(f, x, \tilde{y}) \mapsto \tilde{x}$:

- Takes a computer vision classifier and its parameters $f(x)$.
- Takes any image x and target label \tilde{y} .
- Magically perturbs the image in an imperceptible way: \tilde{x} .
- Resulting image is missclassified by f as anything you want (\tilde{y}).

Group discussion:

- Who is the adversary? Who is the *defender*?
- What are the possible settings and scenarios?
- What is the adversary model? What is the threat model?

Recall: Image Perturbations

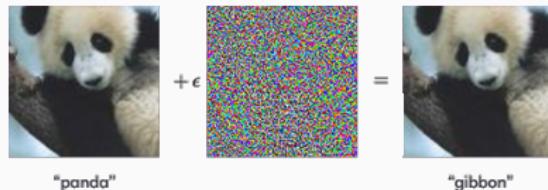


Image Credit: Goodfellow et al. [2014]

Imagine an algorithm $\text{Magic}(f, x, \tilde{y}) \mapsto \tilde{x}$:

- Takes a computer vision classifier and its parameters $f(x)$.
- Takes any image x and target label \tilde{y} .
- Magically perturbs the image in an imperceptible way: \tilde{x} .
- Resulting image is missclassified by f as anything you want (\tilde{y}).

Group discussion:

- Who is the adversary? Who is the *defender*?
- What are the possible settings and scenarios?
- What is the adversary model? What is the threat model?

Recall: Image Perturbations

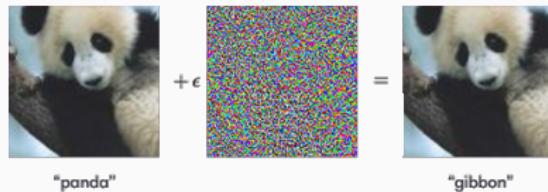


Image Credit: Goodfellow et al. [2014]

Imagine an algorithm $\text{Magic}(f, x, \tilde{y}) \mapsto \tilde{x}$:

- Takes a computer vision classifier and its parameters $f(x)$.
- Takes any image x and target label \tilde{y} .
- Magically perturbs the image in an imperceptible way: \tilde{x} .
- Resulting image is missclassified by f as anything you want (\tilde{y}).

Group discussion:

- Who is the adversary? Who is the *defender*?
- What are the possible settings and scenarios?
- **What is the adversary model? What is the threat model?**

Adversary Model Example: Image Perturbations

Classifier Instagram's detector of porn and nudity

Adversary People that want to distribute porn on Instagram anyway

Threat Users see nudes

Risk This might breach laws and you get fined

Goals Wants the detector to not detect

Capabilities Modify the image as long as the semantics is preserved

Knowledge Knows the exact classifier and its parameters

Adversary Model Example: Image Perturbations

Classifier Instagram's detector of porn and nudity

Adversary People that want to distribute porn on Instagram anyway

Threat Users see nudes

Risk This might breach laws and you get fined

Goals Wants the detector to not detect

Capabilities Modify the image as long as the semantics is preserved

Knowledge Knows the exact classifier and its parameters

Adversary Model Example: Image Perturbations

Classifier Instagram's detector of porn and nudity

Adversary People that want to distribute porn on Instagram anyway

Threat Users see nudes

Risk This might breach laws and you get fined

Goals Wants the detector to not detect

Capabilities Modify the image as long as the semantics is preserved

Knowledge Knows the exact classifier and its parameters

Adversary Model Example: Image Perturbations

Classifier Instagram's detector of porn and nudity

Adversary People that want to distribute porn on Instagram anyway

Threat Users see nudes

Risk This might breach laws and you get fined

Goals Wants the detector to not detect

Capabilities Modify the image as long as the semantics is preserved

Knowledge Knows the exact classifier and its parameters

Adversary Model Example: Image Perturbations

Classifier Instagram's detector of porn and nudity

Adversary People that want to distribute porn on Instagram anyway

Threat Users see nudes

Risk This might breach laws and you get fined

Goals Wants the detector to not detect

Capabilities Modify the image as long as the semantics is preserved

Knowledge Knows the exact classifier and its parameters

Adversary Model Example: Image Perturbations

Classifier Instagram's detector of porn and nudity

Adversary People that want to distribute porn on Instagram anyway

Threat Users see nudes

Risk This might breach laws and you get fined

Goals Wants the detector to not detect

Capabilities Modify the image as long as the semantics is preserved

Knowledge Knows the exact classifier and its parameters

Adversary Model Example: Image Perturbations

Classifier Instagram's detector of porn and nudity

Adversary People that want to distribute porn on Instagram anyway

Threat Users see nudes

Risk This might breach laws and you get fined

Goals Wants the detector to not detect

Capabilities Modify the image as long as the semantics is preserved

Knowledge Knows the exact classifier and its parameters

Defending Against the Attack

Goals Wants the detector to not detect the image

Capabilities Modify the image as long as the semantics is preserved

Knowledge Knows the exact classifier and its parameters

Questions

1. Is this adversarial model realistic? Yes/No
2. What is the best defense strategy?

• How can we defend against adversarial attacks?

• What are the trade-offs between security and performance?

• How can we evaluate the robustness of a machine learning model?

• What are the challenges in developing effective defense strategies?

• How can we improve the efficiency of defense mechanisms?

• What are the future directions for research in this field?

Defending Against the Attack

Goals Wants the detector to not detect the image

Capabilities Modify the image as long as the semantics is preserved

Knowledge Knows the exact classifier and its parameters

Questions

1. Is this adversarial model realistic? Yes/No
2. What is the best defense strategy?
 - 2.1 Enforce strict access control inside Instagram to ensure model parameters cannot leak.
 - 2.2 Ensure that the Magic algorithm does not become publicly known.
Use legal means (e.g., sue people or platforms that publicize it)
 - 2.3 Try to find kinds of classifiers that Magic algorithm does not work with.

Defending Against the Attack

Goals Wants the detector to not detect the image

Capabilities Modify the image as long as the semantics is preserved

Knowledge Knows the exact classifier and its parameters

Questions

1. Is this adversarial model realistic? Yes/No
2. What is the best defense strategy?
 - 2.1 Enforce strict access control inside Instagram to ensure model parameters cannot leak.
 - 2.2 Ensure that the Magic algorithm does not become publicly known.
Use legal means (e.g., sue people or platforms that publicize it)
 - 2.3 Try to find kinds of classifiers that Magic algorithm does not work with.

Defending Against the Attack

Goals Wants the detector to not detect the image

Capabilities Modify the image as long as the semantics is preserved

Knowledge Knows the exact classifier and its parameters

Questions

1. Is this adversarial model realistic? Yes/No
2. What is the best defense strategy?
 - 2.1 Enforce strict access control inside Instagram to ensure model parameters cannot leak.
 - 2.2 Ensure that the Magic algorithm does not become publicly known.
Use legal means (e.g., sue people or platforms that publicize it)
 - 2.3 Try to find kinds of classifiers that Magic algorithm does not work with.

Defending Against the Attack

Goals Wants the detector to not detect the image

Capabilities Modify the image as long as the semantics is preserved

Knowledge Knows the exact classifier and its parameters

Questions

1. Is this adversarial model realistic? Yes/No
2. What is the best defense strategy?
 - 2.1 Enforce strict access control inside Instagram to ensure model parameters cannot leak.
 - 2.2 Ensure that the Magic algorithm does not become publicly known.
Use legal means (e.g., sue people or platforms that publicize it)
 - 2.3 Try to find kinds of classifiers that Magic algorithm does not work with.

Defending Against the Attack

Goals Wants the detector to not detect the image

Capabilities Modify the image as long as the semantics is preserved

Knowledge Knows the exact classifier and its parameters

Questions

1. Is this adversarial model realistic? Yes/No
2. What is the best defense strategy?
 - 2.1 Enforce strict access control inside Instagram to ensure model parameters cannot leak.
 - 2.2 Ensure that the Magic algorithm does not become publicly known.
Use legal means (e.g., sue people or platforms that publicize it)
 - 2.3 Try to find kinds of classifiers that Magic algorithm does not work with.

Need to Attack in order to Defend

Principle

To defend, need to know how the adversary would attack.

- This is why adversary models are important
- Need to work on coming up with new attacks, and incentivize white-hat hackers to *responsibly disclose* the attacks to you (e.g., via rewards).

Need to Attack in order to Defend

Principle

To defend, need to know how the adversary would attack.

- This is why adversary models are important
- Need to work on coming up with new attacks, and incentivize white-hat hackers to *responsibly disclose* the attacks to you (e.g., via rewards).

Need to Attack in order to Defend

Principle

To defend, need to know how the adversary would attack.

- This is why adversary models are important
- Need to work on coming up with new attacks, and incentivize white-hat hackers to *responsibly disclose* the attacks to you (e.g., via rewards).

Worst-Case Security Principle

Principle

Do not underestimate your adversary. Strive to defend against the most powerful adversary, even if some aspects of the model might seem unrealistic.

- Kerckhoffs' Principle. As a defender, assume the attacker that *knows everything about your system*.
- In general, more powerful attacker \implies more knowledge, more capabilities.
- You should defend against an attacker that is as powerful as possible.

Worst-Case Security Principle

Principle

Do not underestimate your adversary. Strive to defend against the most powerful adversary, even if some aspects of the model might seem unrealistic.

- Kerckhoffs' Principle. As a defender, assume the attacker that *knows everything about your system*.
- In general, more powerful attacker \implies more knowledge, more capabilities.
- You should defend against an attacker that is as powerful as possible.

Worst-Case Security Principle

Principle

Do not underestimate your adversary. Strive to defend against the most powerful adversary, even if some aspects of the model might seem unrealistic.

- Kerckhoffs' Principle. As a defender, assume the attacker that *knows everything about your system*.
- In general, more powerful attacker \implies more knowledge, more capabilities.
- You should defend against an attacker that is as powerful as possible.

Worst-Case Security Principle

Principle

Do not underestimate your adversary. Strive to defend against the most powerful adversary, even if some aspects of the model might seem unrealistic.

- Kerckhoffs' Principle. As a defender, assume the attacker that *knows everything about your system*.
- In general, more powerful attacker \implies more knowledge, more capabilities.
- You should defend against an attacker that is as powerful as possible.

Worst-Case Security Principle (flipped)

Principle

When designing an attack, strive to achieve most with the least power.

- A good attack means the attacker achieves the goals with as little capabilities and knowledge as possible.

Worst-Case Security Principle (flipped)

Principle

When designing an attack, strive to achieve most with the least power.

- A good attack means the attacker achieves the goals with as little capabilities and knowledge as possible.

How to Defend?

Goals Wants the detector to not detect the image

Capabilities Modify the image as long as the semantics is preserved

Knowledge Knows the exact classifier and its parameters

Question

What is the best defense strategy?

1. Enforce strict access control inside Instagram to ensure model parameters cannot leak.
2. Ensure that the Magic algorithm does not become publicly known.
Use legal means (e.g., sue people or platforms that publicize it)
3. Try to find kinds of classifiers that Magic algorithm does not work with.

How to Defend? (Answers)

1. Enforce strict access control inside Instagram to ensure model parameters cannot leak.
 - 1.1 Worst-case security principle breached: *Security by obscurity*
 - 1.2 What if they leak: get stolen or reverse engineered?
2. Ensure that the Magic algorithm does not become publicly known. Use legal means (e.g., sue people or platforms that publicize it)
 - 2.1 Worst-case security principle breached: *Security by obscurity*
 - 2.2 Eventually the algorithm will be known: spread or sold.
 - 2.3 Know-the-attacks principle breached.
 - 2.4 Punishing researchers that find security vulnerability is a huge no-no.
They need to be encouraged to disclose vulnerabilities to you, not punished.
3. Try to find kinds of classifiers that Magic algorithm does not work with.
 - 3.1 Worst-case adversary: knows the parameters of the model

How to Defend? (Answers)

1. Enforce strict access control inside Instagram to ensure model parameters cannot leak.
 - 1.1 Worst-case security principle breached: *Security by obscurity*
 - 1.2 What if they leak: get stolen or reverse engineered?
2. Ensure that the Magic algorithm does not become publicly known. Use legal means (e.g., sue people or platforms that publicize it)
 - 2.1 Worst-case security principle breached: *Security by obscurity*
 - 2.2 Eventually the algorithm will be known: spread or sold.
 - 2.3 Know-the-attacks principle breached.
 - 2.4 Punishing researchers that find security vulnerability is a huge no-no.
They need to be encouraged to disclose vulnerabilities to you, not punished.
3. Try to find kinds of classifiers that Magic algorithm does not work with.
 - 3.1 Worst-case adversary: knows the parameters of the model

How to Defend? (Answers)

1. Enforce strict access control inside Instagram to ensure model parameters cannot leak.
 - 1.1 Worst-case security principle breached: *Security by obscurity*
 - 1.2 What if they leak: get stolen or reverse engineered?
2. Ensure that the Magic algorithm does not become publicly known. Use legal means (e.g., sue people or platforms that publicize it)
 - 2.1 Worst-case security principle breached: *Security by obscurity*
 - 2.2 Eventually the algorithm will be known: spread or sold.
 - 2.3 Know-the-attacks principle breached.
 - 2.4 Punishing researchers that find security vulnerability is a huge no-no.
They need to be encouraged to disclose vulnerabilities to you, not punished.
3. Try to find kinds of classifiers that Magic algorithm does not work with.
 - 3.1 Worst-case adversary: knows the parameters of the model

How to Defend? (Answers)

1. Enforce strict access control inside Instagram to ensure model parameters cannot leak.
 - 1.1 Worst-case security principle breached: *Security by obscurity*
 - 1.2 What if they leak: get stolen or reverse engineered?
2. Ensure that the Magic algorithm does not become publicly known.
Use legal means (e.g., sue people or platforms that publicize it)
 - 2.1 Worst-case security principle breached: *Security by obscurity*
 - 2.2 Eventually the algorithm will be known: spread or sold.
 - 2.3 Know-the-attacks principle breached.
 - 2.4 Punishing researchers that find security vulnerability is a huge no-no.
They need to be encouraged to disclose vulnerabilities to you, not punished.
3. Try to find kinds of classifiers that Magic algorithm does not work with.
 - 3.1 Worst-case adversary: knows the parameters of the model

How to Defend? (Answers)

1. Enforce strict access control inside Instagram to ensure model parameters cannot leak.
 - 1.1 Worst-case security principle breached: *Security by obscurity*
 - 1.2 What if they leak: get stolen or reverse engineered?
2. Ensure that the Magic algorithm does not become publicly known. Use legal means (e.g., sue people or platforms that publicize it)
 - 2.1 Worst-case security principle breached: *Security by obscurity*
 - 2.2 Eventually the algorithm will be known: spread or sold.
 - 2.3 Know-the-attacks principle breached.
 - 2.4 Punishing researchers that find security vulnerability is a huge no-no. They need to be encouraged to disclose vulnerabilities to you, not punished.
3. Try to find kinds of classifiers that Magic algorithm does not work with.
 - 3.1 Worst-case adversary: knows the parameters of the model

How to Defend? (Answers)

1. Enforce strict access control inside Instagram to ensure model parameters cannot leak.
 - 1.1 Worst-case security principle breached: *Security by obscurity*
 - 1.2 What if they leak: get stolen or reverse engineered?
2. Ensure that the Magic algorithm does not become publicly known. Use legal means (e.g., sue people or platforms that publicize it)
 - 2.1 Worst-case security principle breached: *Security by obscurity*
 - 2.2 Eventually the algorithm will be known: spread or sold.
 - 2.3 Know-the-attacks principle breached.
 - 2.4 Punishing researchers that find security vulnerability is a huge no-no. They need to be encouraged to disclose vulnerabilities to you, not punished.
3. Try to find kinds of classifiers that Magic algorithm does not work with.
 - 3.1 Worst-case adversary: knows the parameters of the model

How to Defend? (Answers)

1. Enforce strict access control inside Instagram to ensure model parameters cannot leak.
 - 1.1 Worst-case security principle breached: *Security by obscurity*
 - 1.2 What if they leak: get stolen or reverse engineered?
2. Ensure that the Magic algorithm does not become publicly known. Use legal means (e.g., sue people or platforms that publicize it)
 - 2.1 Worst-case security principle breached: *Security by obscurity*
 - 2.2 Eventually the algorithm will be known: spread or sold.
 - 2.3 **Know-the-attacks principle breached.**
 - 2.4 Punishing researchers that find security vulnerability is a huge no-no. They need to be encouraged to disclose vulnerabilities to you, not punished.
3. Try to find kinds of classifiers that Magic algorithm does not work with.
 - 3.1 Worst-case adversary: knows the parameters of the model

How to Defend? (Answers)

1. Enforce strict access control inside Instagram to ensure model parameters cannot leak.
 - 1.1 Worst-case security principle breached: *Security by obscurity*
 - 1.2 What if they leak: get stolen or reverse engineered?
2. Ensure that the Magic algorithm does not become publicly known. Use legal means (e.g., sue people or platforms that publicize it)
 - 2.1 Worst-case security principle breached: *Security by obscurity*
 - 2.2 Eventually the algorithm will be known: spread or sold.
 - 2.3 Know-the-attacks principle breached.
 - 2.4 Punishing researchers that find security vulnerability is a huge no-no.
They need to be encouraged to disclose vulnerabilities to you, not punished.
3. Try to find kinds of classifiers that Magic algorithm does not work with.
 - 3.1 Worst-case adversary: knows the parameters of the model

How to Defend? (Answers)

1. Enforce strict access control inside Instagram to ensure model parameters cannot leak.
 - 1.1 Worst-case security principle breached: *Security by obscurity*
 - 1.2 What if they leak: get stolen or reverse engineered?
2. Ensure that the Magic algorithm does not become publicly known.
Use legal means (e.g., sue people or platforms that publicize it)
 - 2.1 Worst-case security principle breached: *Security by obscurity*
 - 2.2 Eventually the algorithm will be known: spread or sold.
 - 2.3 Know-the-attacks principle breached.
 - 2.4 Punishing researchers that find security vulnerability is a huge no-no.
They need to be encouraged to disclose vulnerabilities to you, not punished.
3. Try to find kinds of classifiers that Magic algorithm does not work with.
 - 3.1 Worst-case adversary: knows the parameters of the model

How to Defend? (Answers)

1. Enforce strict access control inside Instagram to ensure model parameters cannot leak.
 - 1.1 Worst-case security principle breached: *Security by obscurity*
 - 1.2 What if they leak: get stolen or reverse engineered?
2. Ensure that the Magic algorithm does not become publicly known.
Use legal means (e.g., sue people or platforms that publicize it)
 - 2.1 Worst-case security principle breached: *Security by obscurity*
 - 2.2 Eventually the algorithm will be known: spread or sold.
 - 2.3 Know-the-attacks principle breached.
 - 2.4 Punishing researchers that find security vulnerability is a huge no-no.
They need to be encouraged to disclose vulnerabilities to you, not punished.
3. Try to find kinds of classifiers that Magic algorithm does not work with.
 - 3.1 Worst-case adversary: knows the parameters of the model

Takeaways

- Know your adversary and know yourself: specify adversary's goals, capabilities, and knowledge.
- Need to find and encourage the discovery *and responsible disclosure* of attacks – to fix them before they are exploited
- When defending, defend against even the strongest possible adversary model and attacks
- When attacking, try to achieve adversarial goals with the least power

Takeaways

- Know your adversary and know yourself: specify adversary's goals, capabilities, and knowledge.
- Need to find and encourage the discovery *and responsible disclosure* of attacks – to fix them before they are exploited
- When defending, defend against even the strongest possible adversary model and attacks
- When attacking, try to achieve adversarial goals with the least power

Takeaways

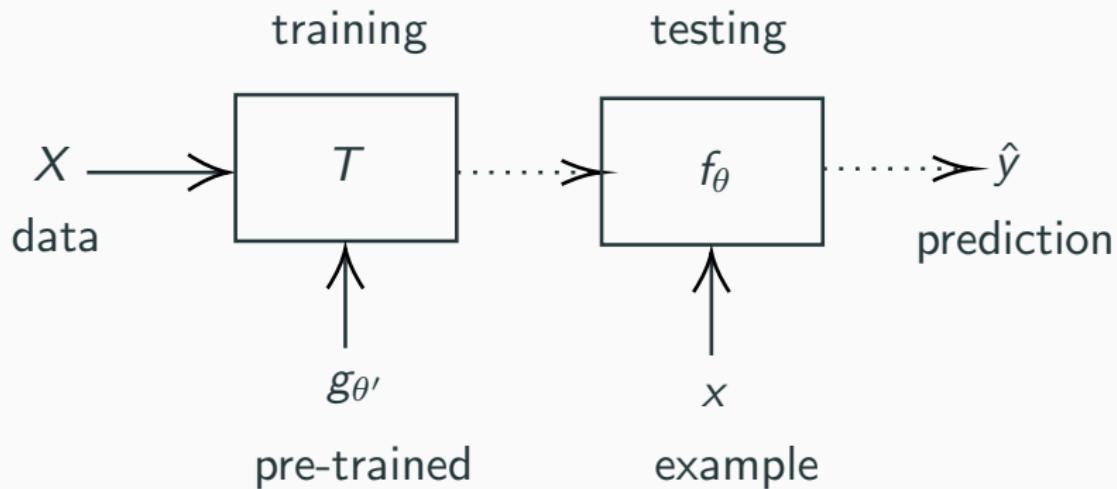
- Know your adversary and know yourself: specify adversary's goals, capabilities, and knowledge.
- Need to find and encourage the discovery *and responsible disclosure* of attacks – to fix them before they are exploited
- When defending, defend against even the strongest possible adversary model and attacks
- When attacking, try to achieve adversarial goals with the least power

Takeaways

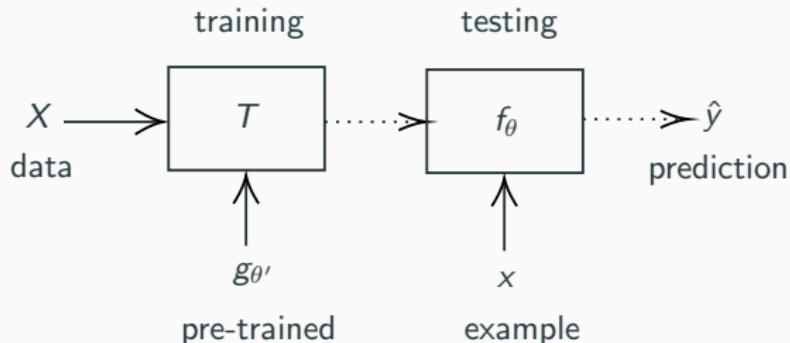
- Know your adversary and know yourself: specify adversary's goals, capabilities, and knowledge.
- Need to find and encourage the discovery *and responsible disclosure* of attacks – to fix them before they are exploited
- When defending, defend against even the strongest possible adversary model and attacks
- When attacking, try to achieve adversarial goals with the least power

Adversary Models for Security and Privacy of ML

Machine Learning Pipeline



Adversary's Capabilities

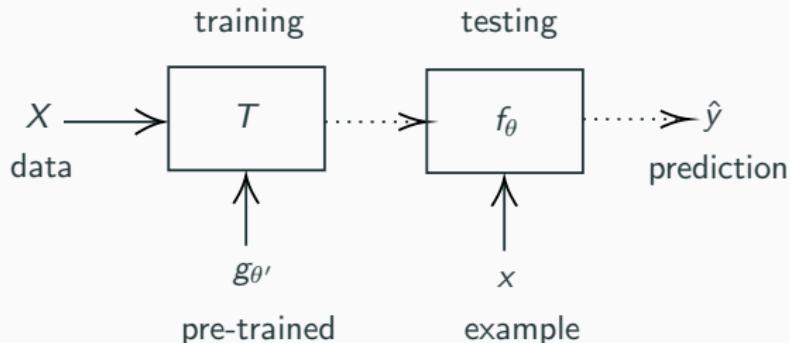


Question

What could be the adversary's capabilities?

- Modify or supply new testing examples x
- Modify or supply new training data X
- Modify or supply new pre-trained components $g_{\theta'}$
- Observe outputs (passive adversary)

Adversary's Capabilities

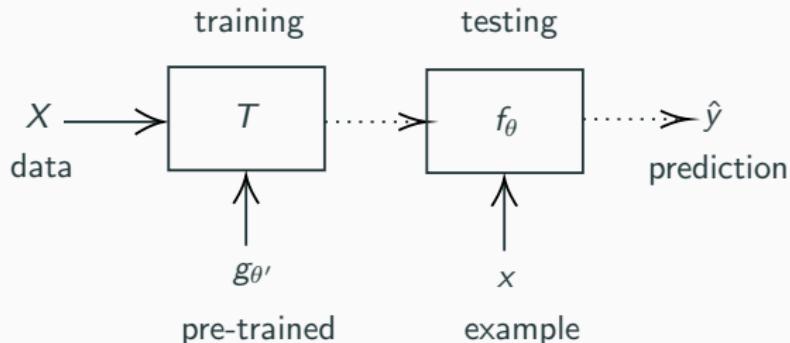


Question

What could be the adversary's capabilities?

- Modify or supply new testing examples x
- Modify or supply new training data X
- Modify or supply new pre-trained components $g_{\theta'}$
- Observe outputs (passive adversary)

Adversary's Capabilities

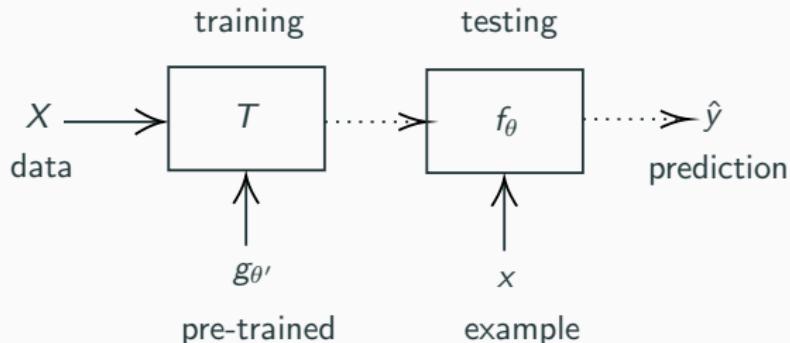


Question

What could be the adversary's capabilities?

- Modify or supply new testing examples x
- Modify or supply new training data X
- Modify or supply new pre-trained components $g_{\theta'}$
- Observe outputs (passive adversary)

Adversary's Capabilities

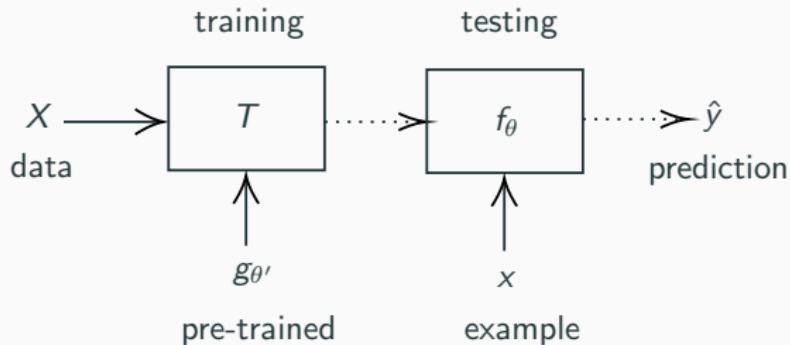


Question

What could be the adversary's capabilities?

- Modify or supply new testing examples x
- **Modify or supply new training data X**
- Modify or supply new pre-trained components $g_{\theta'}$
- Observe outputs (passive adversary)

Adversary's Capabilities

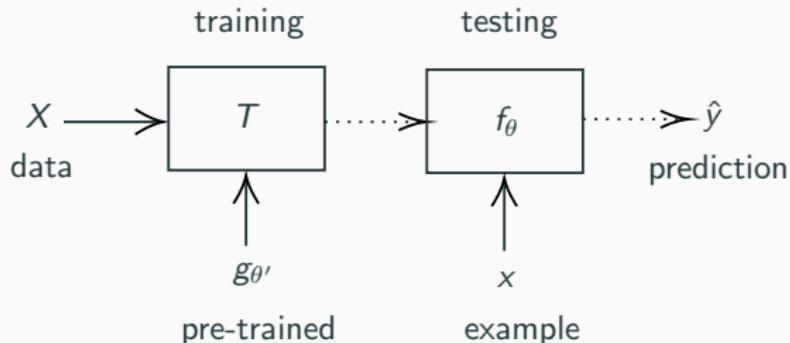


Question

What could be the adversary's capabilities?

- Modify or supply new testing examples x
- Modify or supply new training data X
- Modify or supply new pre-trained components $g_{\theta'}$
- Observe outputs (passive adversary)

Adversary's Capabilities

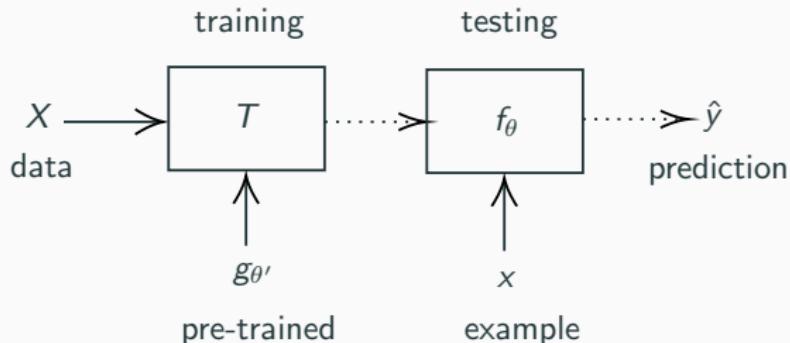


Question

What could be the adversary's capabilities?

- Modify or supply new testing examples x
 - Modify or supply new training data X
 - Modify or supply new pre-trained components $g_{\theta'}$
- » Observe outputs (passive adversary)

Adversary's Capabilities



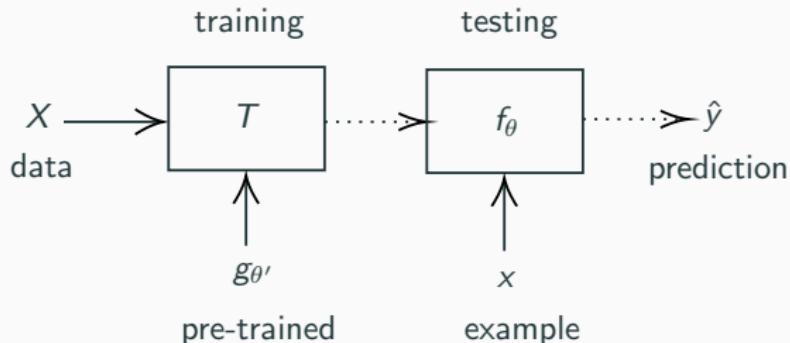
Question

What could be the adversary's capabilities?

- Modify or supply new testing examples x
- Modify or supply new training data X
- Modify or supply new pre-trained components $g_{\theta'}$

↳ Observe outputs (passive adversary)

Adversary's Capabilities

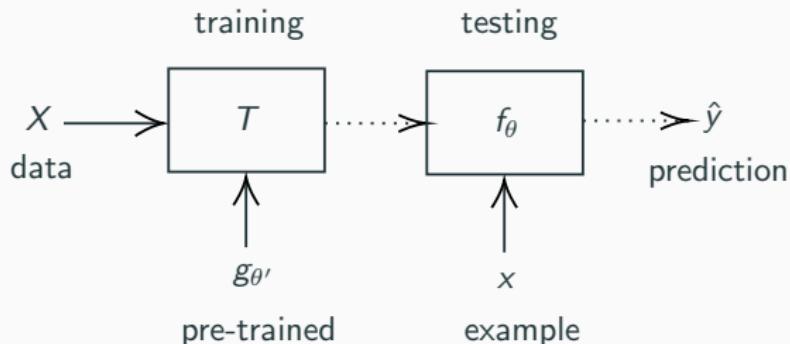


Question

What could be the adversary's capabilities?

- Modify or supply new testing examples x
- Modify or supply new training data X
- Modify or supply new pre-trained components $g_{\theta'}$
- Observe outputs (passive adversary)

Adversary's Knowledge

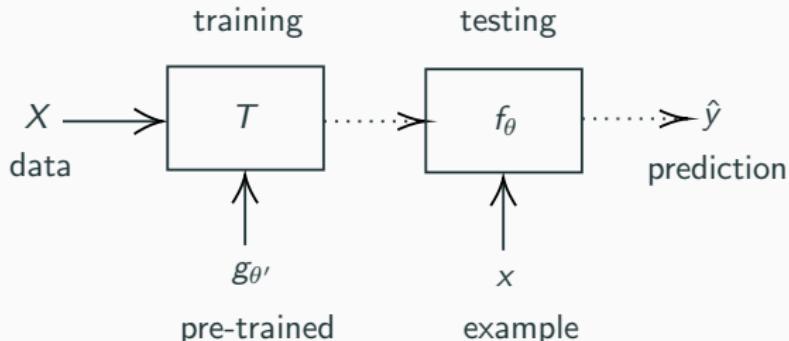


Question

What could be the options for adversary's knowledge?

- No 100% consistent terms
- White-box: usually architecture f , parameters θ , procedure T
- Black-box: often knows architecture f but no parameters, and has query access to f or T
- Agnostic black-box: only query access to f or T

Adversary's Knowledge

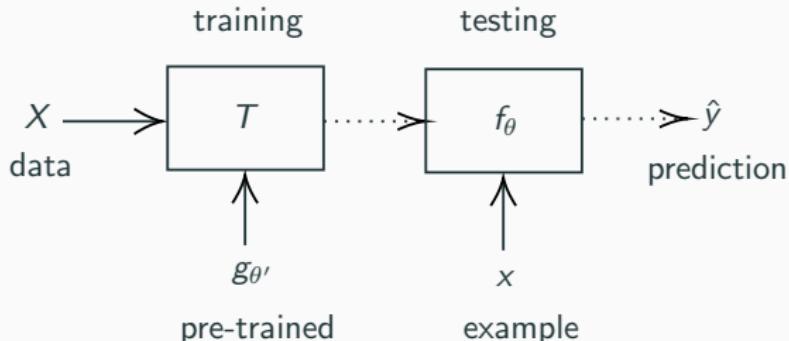


Question

What could be the options for adversary's knowledge?

- No 100% consistent terms
 - White-box: usually architecture f , parameters θ , procedure T
 - Black-box: often knows architecture f but no parameters, and has query access to f or T
 - Agnostic black-box: only query access to f or T

Adversary's Knowledge

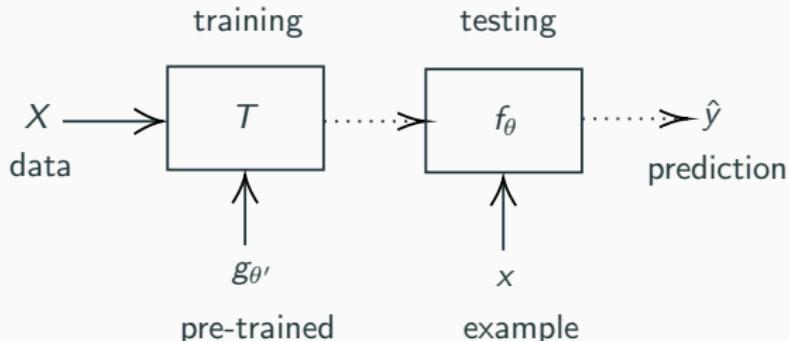


Question

What could be the options for adversary's knowledge?

- No 100% consistent terms
 - White-box: usually architecture f , parameters θ , procedure T
 - Black-box: often knows architecture f but no parameters, and has query access to f or T
 - Agnostic black-box: only query access to f or T

Adversary's Knowledge

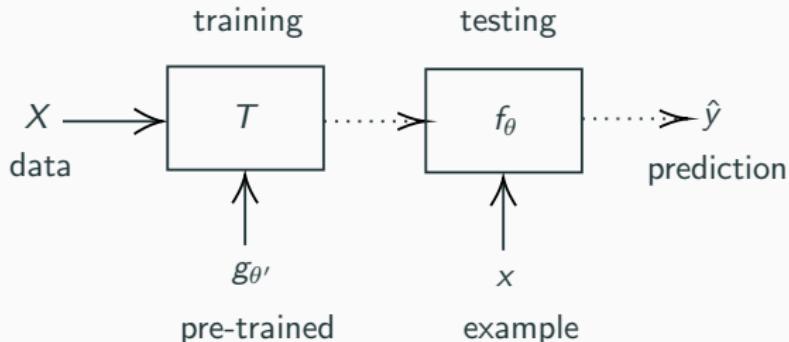


Question

What could be the options for adversary's knowledge?

- No 100% consistent terms
- White-box: usually architecture f , parameters θ , procedure T
- Black-box: often knows architecture f but no parameters, and has query access to f or T
- Agnostic black-box: only query access to f or T

Adversary's Knowledge

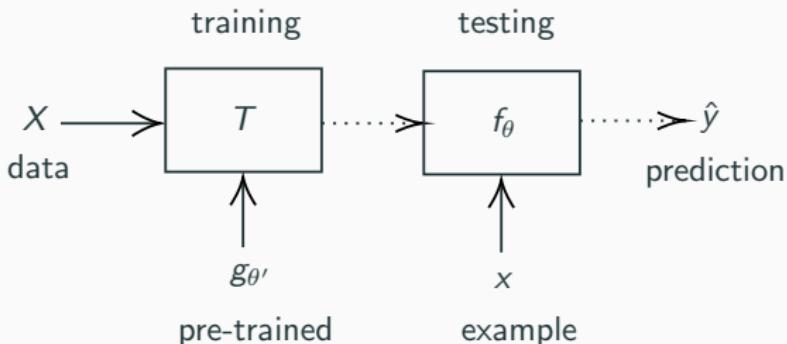


Question

What could be the options for adversary's knowledge?

- No 100% consistent terms
- White-box: usually architecture f , parameters θ , procedure T
- Black-box: often knows architecture f but no parameters, and has query access to f or T
- Agnostic black-box: only query access to f or T

Adversary's Knowledge

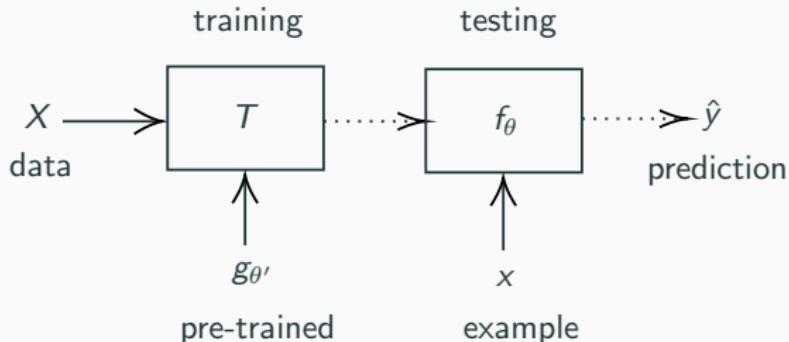


Question

What could be the options for adversary's knowledge?

- No 100% consistent terms
 - White-box: usually architecture f , parameters θ , procedure T
 - Black-box: often knows architecture f but no parameters, and has query access to f or T
 - Agnostic black-box: only query access to f or T

Adversary's Knowledge

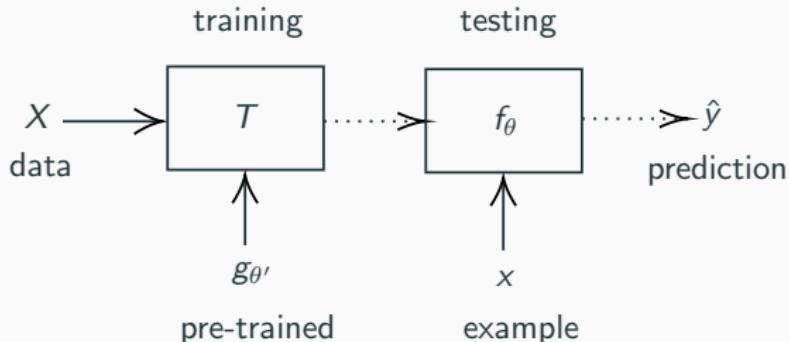


Question

What could be the options for adversary's knowledge?

- No 100% consistent terms
- White-box: usually architecture f , parameters θ , procedure T
- Black-box: often knows architecture f but no parameters, and has query access to f or T
- Agnostic black-box: only query access to f or T

Adversary's Knowledge

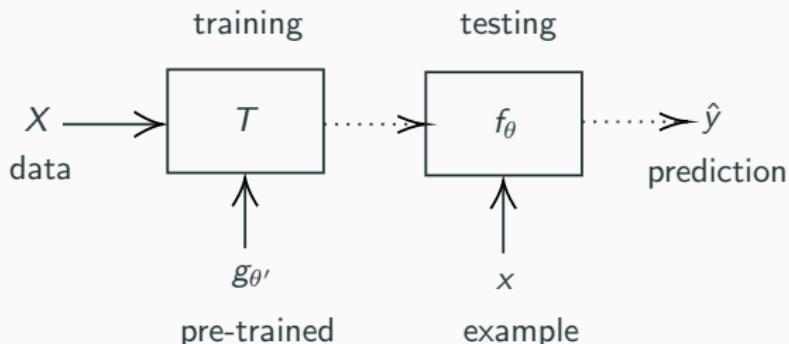


Question

What could be the options for adversary's knowledge?

- No 100% consistent terms
- White-box: usually architecture f , parameters θ , procedure T
- Black-box: often knows architecture f but no parameters, and has query access to f or T
- Agnostic black-box: only query access to f or T

Adversary's Goals

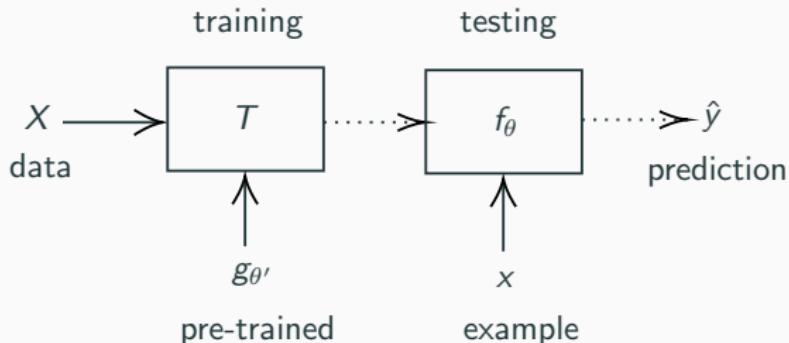


Question

What could be adversary's goals?

- *Evasion*: the classifier at test time: provide an *adversarial example* x that causes a mistake
- Increase errors or make the classifier behave in a specific way
- Steal the model weights θ
- Infer information about training data X .

Adversary's Goals

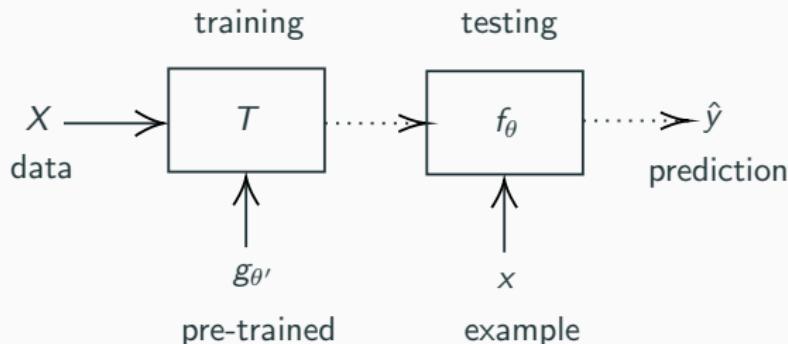


Question

What could be adversary's goals?

- *Evide the classifier at test time:* provide an *adversarial example* x that causes a mistake
- Increase errors or make the classifier behave in a specific way
- Infer information about training data X .
- Steal the model weights θ

Adversary's Goals

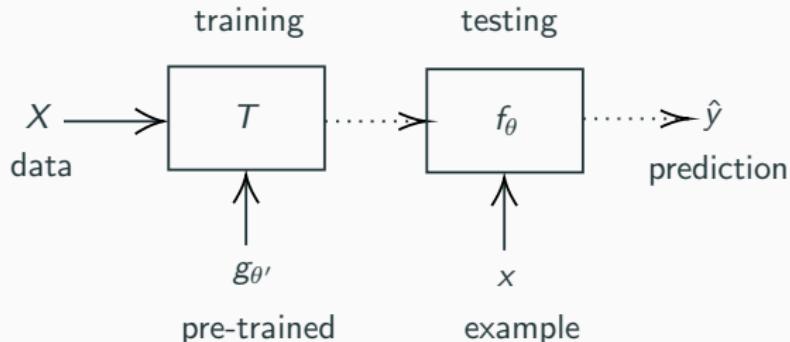


Question

What could be adversary's goals?

- *Evide the classifier* at test time: provide an *adversarial example* x that causes a mistake
- Increase errors or make the classifier behave in a specific way
- Infer information about training data X .
- Steal the model weights θ

Adversary's Goals

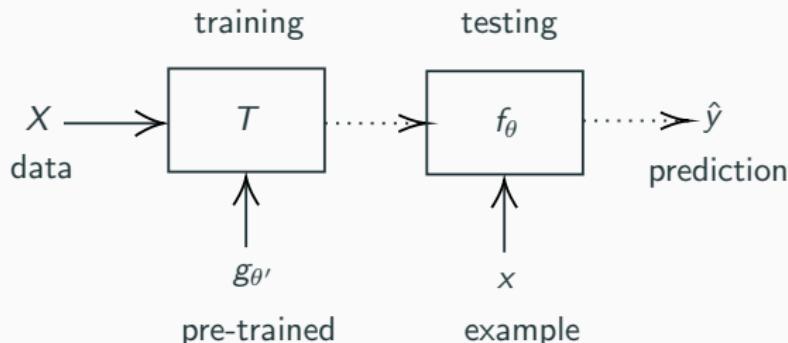


Question

What could be adversary's goals?

- *Evide the classifier* at test time: provide an *adversarial example* x that causes a mistake
- **Increase errors or make the classifier behave in a specific way**
 - by *poisoning* the training data X
 - by *poisoning* the pre-trained model $g_{\theta'}$
- *Steal the model weights* θ
- *Infer* information about training data X .

Adversary's Goals

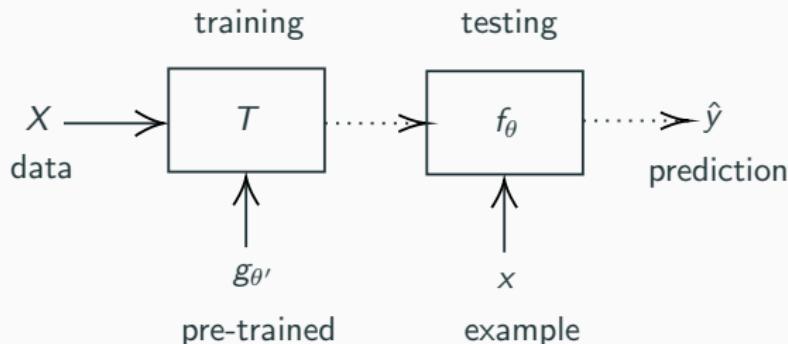


Question

What could be adversary's goals?

- *Evide the classifier* at test time: provide an *adversarial example* x that causes a mistake
- Increase errors or make the classifier behave in a specific way
 - by *poisoning* the training data X
 - by *poisoning* the pre-trained model $g_{\theta'}$
- Steal the model weights θ
- Infer information about training data X .

Adversary's Goals

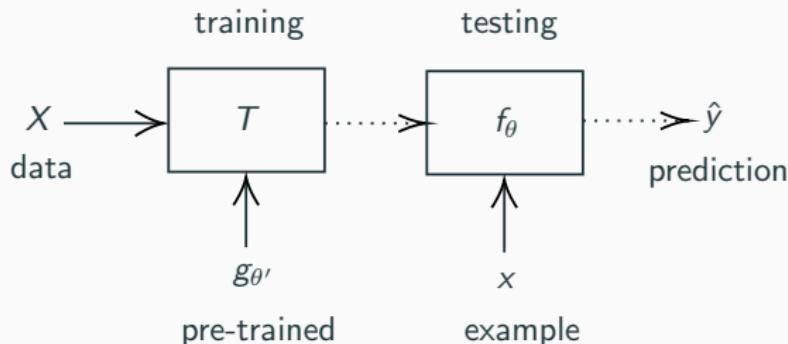


Question

What could be adversary's goals?

- *Evide the classifier* at test time: provide an *adversarial example* x that causes a mistake
- Increase errors or make the classifier behave in a specific way
 - by *poisoning* the training data X
 - by *poisoning* the pre-trained model $g_{\theta'}$
- Steal the model weights θ
- Infer information about training data X .

Adversary's Goals

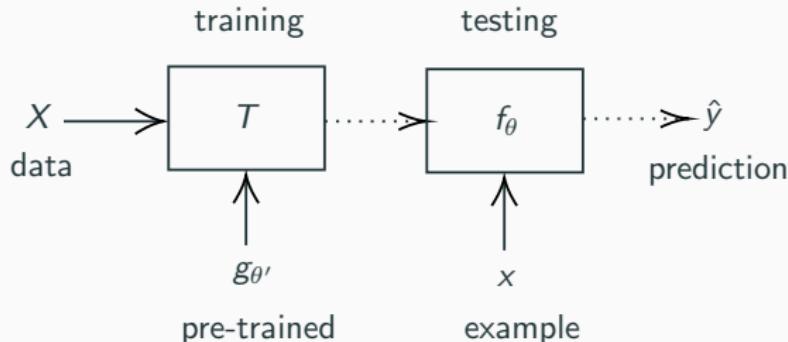


Question

What could be adversary's goals?

- *Evide the classifier* at test time: provide an *adversarial example* x that causes a mistake
- Increase errors or make the classifier behave in a specific way
 - by *poisoning* the training data X
 - by *poisoning* the pre-trained model $g_{\theta'}$
- *Steal the model weights* θ
- *Infer* information about training data X .

Adversary's Goals

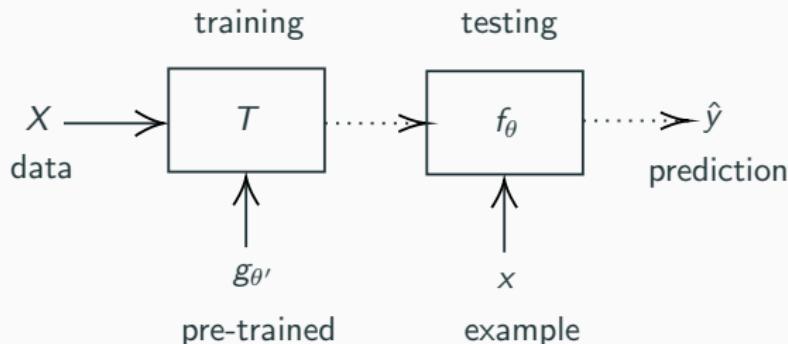


Question

What could be adversary's goals?

- *Evide the classifier* at test time: provide an *adversarial example* x that causes a mistake
- Increase errors or make the classifier behave in a specific way
 - by *poisoning* the training data X
 - by *poisoning* the pre-trained model $g_{\theta'}$
- *Steal the model weights* θ
 - *Infer information about training data* X .

Adversary's Goals

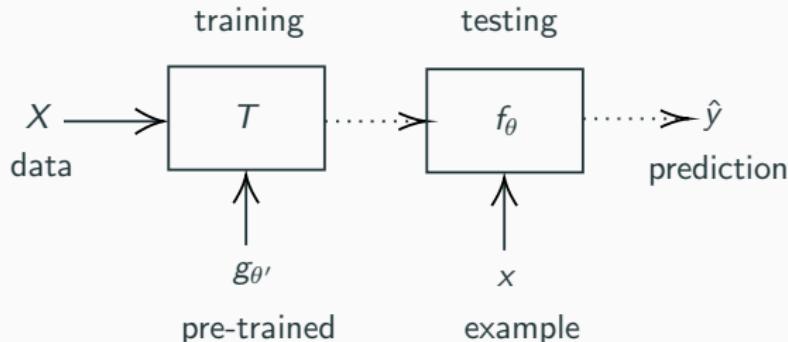


Question

What could be adversary's goals?

- *Evide the classifier* at test time: provide an *adversarial example* x that causes a mistake
- Increase errors or make the classifier behave in a specific way
 - by *poisoning* the training data X
 - by *poisoning* the pre-trained model $g_{\theta'}$
- *Steal* the model weights θ
 - *Infer* information about training data X .

Adversary's Goals



Question

What could be adversary's goals?

- *Evide the classifier* at test time: provide an *adversarial example* x that causes a mistake
- Increase errors or make the classifier behave in a specific way
 - by *poisoning* the training data X
 - by *poisoning* the pre-trained model $g_{\theta'}$
- *Steal* the model weights θ
- *Infer* information about training data X .

Active Attacks at Test Time

Evasion Attacks



Image Credit: Biggio and Roli [2018]

- A goal of an *evasion attack* is to evade detection, e.g., spam detection, fraud detection
- Goes back to 2004 in the context of Naïve Bayes spam classification (Dalvi et al. [2004], Lowd and Meek [2005])
- Evasion attacks are done with **adversarial examples**:
Examples constructed by an adversary to deliberately cause a classification mistake

Evasion Attacks



Image Credit: Biggio and Roli [2018]

- A goal of an *evasion attack* is to evade detection, e.g., spam detection, fraud detection
- Goes back to 2004 in the context of Naïve Bayes spam classification (Dalvi et al. [2004], Lowd and Meek [2005])
- Evasion attacks are done with **adversarial examples**:
Examples constructed by an adversary to deliberately cause a classification mistake

Evasion Attacks



Image Credit: Biggio and Roli [2018]

- A goal of an *evasion attack* is to evade detection, e.g., spam detection, fraud detection
- Goes back to 2004 in the context of Naïve Bayes spam classification (Dalvi et al. [2004], Lowd and Meek [2005])
- Evasion attacks are done with **adversarial examples**:

Examples constructed by an adversary to deliberately cause a classification mistake

Evasion Attacks



Image Credit: Biggio and Roli [2018]

- A goal of an *evasion attack* is to evade detection, e.g., spam detection, fraud detection
- Goes back to 2004 in the context of Naïve Bayes spam classification (Dalvi et al. [2004], Lowd and Meek [2005])
- Evasion attacks are done with **adversarial examples**:
Examples constructed by an adversary to deliberately cause a classification mistake

Discovery of Non-Robustness of Neural Nets

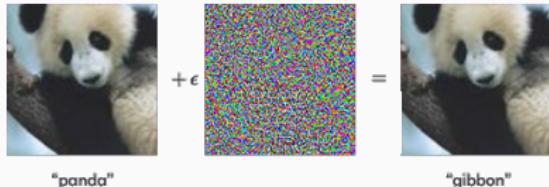


Image Credit: Goodfellow et al. [2014]

- Szegedy et al. [2013] discover that outputs of neural nets are not robust to specially crafted imperceptible patterns. Authors pose it as a question of security.
- This attracted great interest in ML and Security communities with a long line of similar attack algorithms and defences
- Attacks work for any ML models, not only neural nets
- Because this line of work is very known, when people mention *adversarial attacks*, *adversarial robustness*, and *adversarial examples* in ML, they usually talk about “imperceptible perturbations that change classification outputs”

Discovery of Non-Robustness of Neural Nets

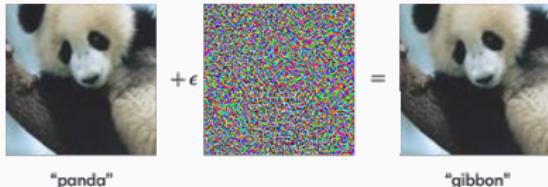


Image Credit: Goodfellow et al. [2014]

- Szegedy et al. [2013] discover that outputs of neural nets are not robust to specially crafted imperceptible patterns. Authors pose it as a question of security.
- This attracted great interest in ML and Security communities with a long line of similar attack algorithms and defences
- Attacks work for any ML models, not only neural nets
- Because this line of work is very known, when people mention *adversarial attacks*, *adversarial robustness*, and *adversarial examples* in ML, they usually talk about “imperceptible perturbations that change classification outputs”

Discovery of Non-Robustness of Neural Nets

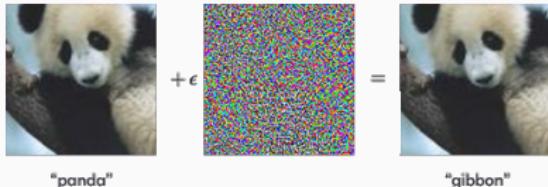


Image Credit: Goodfellow et al. [2014]

- Szegedy et al. [2013] discover that outputs of neural nets are not robust to specially crafted imperceptible patterns. Authors pose it as a question of security.
- This attracted great interest in ML and Security communities with a long line of similar attack algorithms and defences
- Attacks work for any ML models, not only neural nets
- Because this line of work is very known, when people mention *adversarial attacks*, *adversarial robustness*, and *adversarial examples* in ML, they usually talk about “imperceptible perturbations that change classification outputs”

Discovery of Non-Robustness of Neural Nets

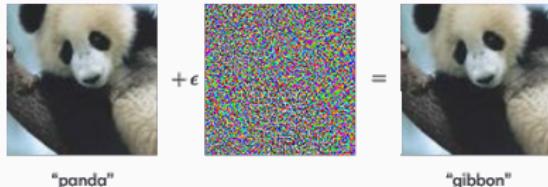


Image Credit: Goodfellow et al. [2014]

- Szegedy et al. [2013] discover that outputs of neural nets are not robust to specially crafted imperceptible patterns. Authors pose it as a question of security.
- This attracted great interest in ML and Security communities with a long line of similar attack algorithms and defences
- Attacks work for any ML models, not only neural nets
- Because this line of work is very known, when people mention *adversarial attacks*, *adversarial robustness*, and *adversarial examples* in ML, they usually talk about “imperceptible perturbations that change classification outputs”

Image Perturbations

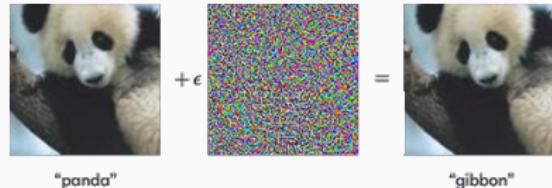


Image Credit: Goodfellow et al. [2014]

Can we build the image perturbation algorithm?

- Goal: missclassify x, y as something else
- Initial example x, y
- Perturbation δ
- Perturbed image $x + \delta$
- Small perturbation: $\Delta = \{\delta \mid \|\delta\| < \epsilon\}$
- Model parameters θ
- Loss of the model $L(x, y; \theta)$

Image Perturbations

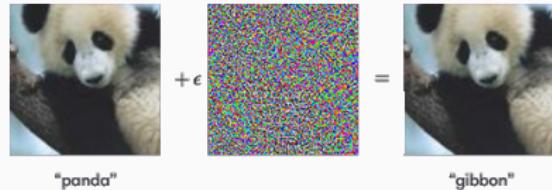


Image Credit: Goodfellow et al. [2014]

Can we build the image perturbation algorithm?

- Goal: missclassify x, y as something else
- Initial example x, y
- Perturbation δ
- Perturbed image $x + \delta$
- Small perturbation: $\Delta = \{\delta \mid \|\delta\| < \varepsilon\}$
- Model parameters θ
- Loss of the model $L(x, y; \theta)$

Image Perturbations

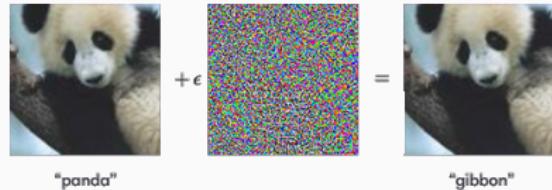


Image Credit: Goodfellow et al. [2014]

Can we build the image perturbation algorithm?

- Goal: missclassify x, y as something else
- Initial example x, y
- **Perturbation δ**
- Perturbed image $x + \delta$
- Small perturbation: $\Delta = \{\delta \mid \|\delta\| < \varepsilon\}$
- Model parameters θ
- Loss of the model $L(x, y; \theta)$

Image Perturbations

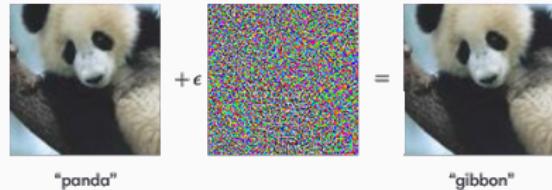


Image Credit: Goodfellow et al. [2014]

Can we build the image perturbation algorithm?

- Goal: missclassify x, y as something else
- Initial example x, y
- Perturbation δ
- **Perturbed image $x + \delta$**
- Small perturbation: $\Delta = \{\delta \mid \|\delta\| < \epsilon\}$
- Model parameters θ
- Loss of the model $L(x, y; \theta)$

Image Perturbations

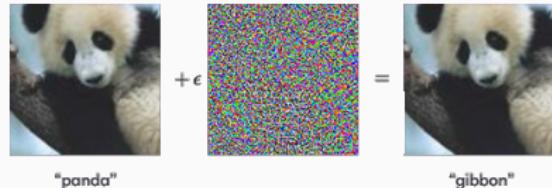


Image Credit: Goodfellow et al. [2014]

Can we build the image perturbation algorithm?

- Goal: missclassify x, y as something else
- Initial example x, y
- Perturbation δ
- Perturbed image $x + \delta$
- Small perturbation: $\Delta = \{\delta \mid \|\delta\| < \varepsilon\}$
- Model parameters θ
- Loss of the model $L(x, y; \theta)$

Image Perturbations

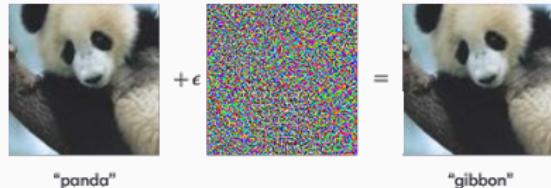


Image Credit: Goodfellow et al. [2014]

Can we build the image perturbation algorithm?

- Goal: missclassify x, y as something else
- Initial example x, y
- Perturbation δ
- Perturbed image $x + \delta$
- Small perturbation: $\Delta = \{\delta \mid \|\delta\| < \varepsilon\}$
- **Model parameters θ**
- Loss of the model $L(x, y; \theta)$

Image Perturbations

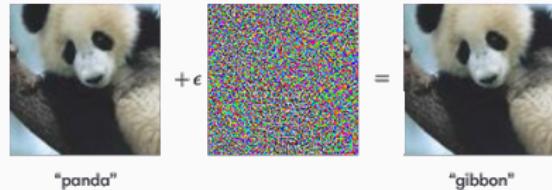


Image Credit: Goodfellow et al. [2014]

Can we build the image perturbation algorithm?

- Goal: missclassify x, y as something else
- Initial example x, y
- Perturbation δ
- Perturbed image $x + \delta$
- Small perturbation: $\Delta = \{\delta \mid \|\delta\| < \varepsilon\}$
- Model parameters θ
- Loss of the model $L(x, y; \theta)$

Image Perturbations

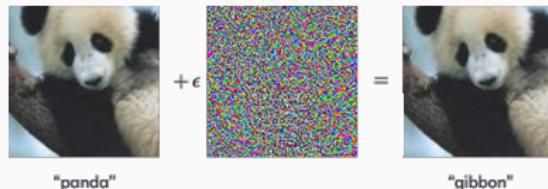


Image Credit: Goodfellow et al. [2014]

Can we build the image perturbation algorithm?

- Goal: missclassify x, y as something else
- Initial example x, y
- Perturbation δ
- Perturbed image $x + \delta$
- Small perturbation: $\Delta = \{\delta \mid \|\delta\| < \varepsilon\}$
- Model parameters θ
- Loss of the model $L(x, y; \theta)$

Group discussion:

How can we do it? What is the optimization problem?

Image Perturbations

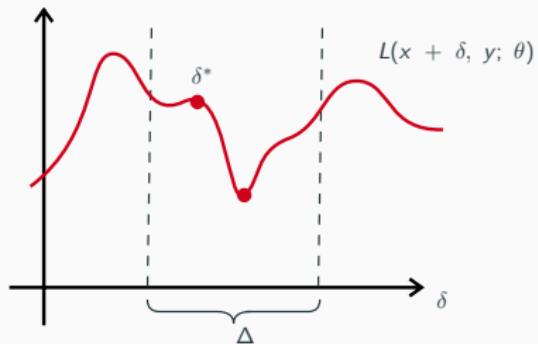
Optimization problem

$$\delta^* = \arg \max_{\delta} L(x + \delta, y; \theta)$$

$$\text{s.t. } \| \delta \| \leq \varepsilon$$

Question:

How do we solve it?



Gradient Descent

Optimization problem

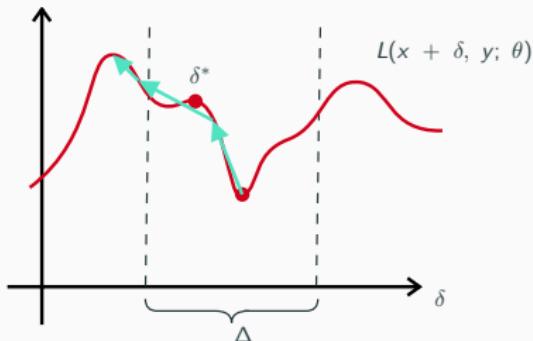
$$\delta^* = \arg \max_{\delta} L(x + \delta, y; \theta)$$

$$\text{s.t. } \|\delta\| \leq \varepsilon$$

- Basic gradient descent
(technically, ascent):

$$\delta := \delta + \alpha \nabla_{\delta} L(x + \delta, y; \theta)$$

- How to meet the constraint?



Gradient Descent

Optimization problem

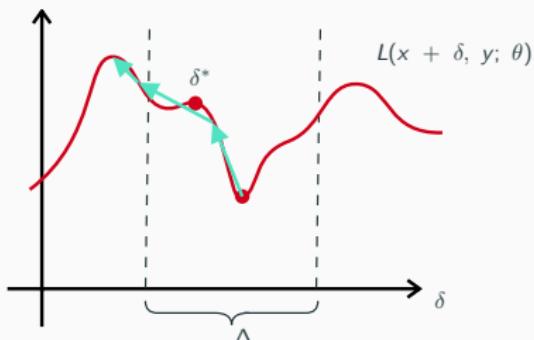
$$\delta^* = \arg \max_{\delta} L(x + \delta, y; \theta)$$

$$\text{s.t. } \|\delta\| \leq \varepsilon$$

- Basic gradient descent
(technically, ascent):

$$\delta := \delta + \alpha \nabla_{\delta} L(x + \delta, y; \theta)$$

- How to meet the constraint?



Descent with Constraints

- Step, then project:

$$\delta := \mathcal{P}_\Delta[\delta + \alpha \nabla_\delta L(x + \delta, y)]$$

$$\alpha \nabla_\delta \text{Loss}(x + \delta, y; \theta)$$

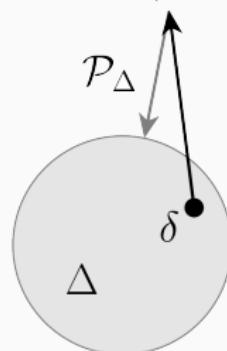


Image: Zico Kolter

Descent with Constraints

- Step, then project:

$$\delta := \mathcal{P}_\Delta[\delta + \alpha \nabla_\delta L(x + \delta, y)]$$

Question

For x that were in training,
 $\nabla_x L(x, y; \theta)$ is small. What is the
problem with small gradient values?

$$\alpha \nabla_\delta L(x + \delta, y; \theta)$$

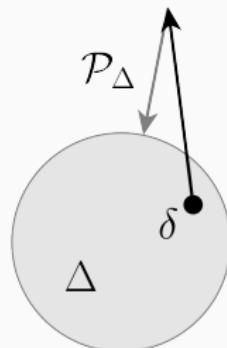


Image: Zico Kolter

Projected Gradient Descent (PGD)

- Final version proposed by Madry et al. [2017]:

$$\delta := \mathcal{P}_\Delta [\delta + \alpha \text{sgn}(\nabla_\delta L(x + \delta, y))]$$

- Run the algorithm for many (thousands of) steps until convergence
- Random restarts with different initial $\delta \in \Delta$

Projected Gradient Descent (PGD)

- Final version proposed by Madry et al. [2017]:

$$\delta := \mathcal{P}_\Delta [\delta + \alpha \text{sgn}(\nabla_\delta L(x + \delta, y))]$$

- Run the algorithm for many (thousands of) steps until convergence
- Random restarts with different initial $\delta \in \Delta$

Projected Gradient Descent (PGD)

- Final version proposed by Madry et al. [2017]:

$$\delta := \mathcal{P}_{\Delta} [\delta + \alpha \text{sgn}(\nabla_{\delta} L(x + \delta, y))]$$

- Run the algorithm for many (thousands of) steps until convergence
- Random restarts with different initial $\delta \in \Delta$

Projected Gradient Descent for ℓ_∞

Question

Let $\Delta = \{\delta \mid \|\delta\|_\infty \leq \varepsilon\}$. How do we project back to Δ ?

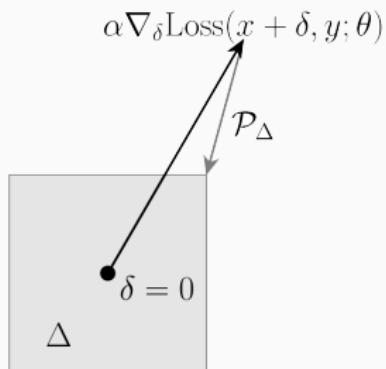


Image: Zico Kolter

Projected Gradient Descent for ℓ_∞

Question

Let $\Delta = \{\delta \mid \|\delta\|_\infty \leq \varepsilon\}$. How do we project back to Δ ?

- Projection:

$$\mathcal{P}_\Delta(\delta) = \text{Clip}_\varepsilon[\delta] = (\max\{\delta_i, \varepsilon\})_i$$

- PGD update:

$$\delta := \text{Clip}_\varepsilon [\delta + \alpha \text{sgn}(\nabla_\delta L(x + \delta, y))]$$

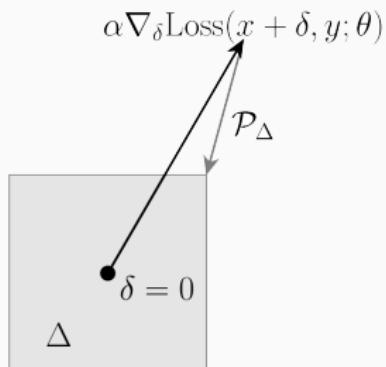


Image: Zico Kolter

Projected Gradient Descent for ℓ_∞

Question

Let $\Delta = \{\delta \mid \|\delta\|_\infty \leq \varepsilon\}$. How do we project back to Δ ?

- Projection:

$$\mathcal{P}_\Delta(\delta) = \text{Clip}_\varepsilon[\delta] = (\max\{\delta_i, \varepsilon\})_i$$

- PGD update:

$$\delta := \text{Clip}_\varepsilon [\delta + \alpha \text{sgn}(\nabla_\delta L(x + \delta, y))]$$

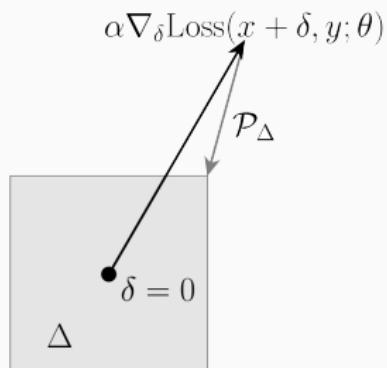


Image: Zico Kolter

Evasion Attacks with PGD

Group Discussion

What is the adversary model for an attack with PGD? Is this a powerful attacker?

Evasion Attacks with PGD

Group Discussion

What is the adversary model for an attack with PGD? Is this a powerful attacker?

Goals Cause a classification mistake

Capabilities Adversary can construct modifications of existing examples of the following form:

$$\tilde{x} = x + \delta, \quad \|\delta\| \leq \varepsilon$$

We assume this means “imperceptible”.

Knowledge White-box, need loss $L(\cdot)$ and parameters of the model θ

Evasion Attacks with PGD

Group Discussion

What is the adversary model for an attack with PGD? Is this a powerful attacker?

Goals Cause a classification mistake

Capabilities Adversary can construct modifications of existing examples of the following form:

$$\tilde{x} = x + \delta, \quad \|\delta\| \leq \varepsilon$$

We assume this means “imperceptible”.

Knowledge White-box, need loss $L(\cdot)$ and parameters of the model θ

Evasion Attacks with PGD

Group Discussion

What is the adversary model for an attack with PGD? Is this a powerful attacker?

Goals Cause a classification mistake

Capabilities Adversary can construct modifications of existing examples of the following form:

$$\tilde{x} = x + \delta, \quad \|\delta\| \leq \varepsilon$$

We assume this means “imperceptible”.

Knowledge White-box, need loss $L(\cdot)$ and parameters of the model θ

Properties of Adversarial Examples

- Let \tilde{x} be an adversarial example for model $f_\theta(\cdot)$
- **Transferability:** It is likely to be missclassified by another model f' trained on the same dataset
- **Black-box:** You can create a surrogate of the target model f' , and the adversarial example against f' is likely to be missclassified by the actual target.
- **Resilience:** You can distort it (e.g., print) and it is likely to still be missclassified by f_θ .

Properties of Adversarial Examples

- Let \tilde{x} be an adversarial example for model $f_\theta(\cdot)$
- **Transferability:** It is likely to be missclassified by another model f' trained on the same dataset
- **Black-box:** You can create a surrogate of the target model f' , and the adversarial example against f' is likely to be missclassified by the actual target.
- **Resilience:** You can distort it (e.g., print) and it is likely to still be missclassified by f_θ .

Properties of Adversarial Examples

- Let \tilde{x} be an adversarial example for model $f_\theta(\cdot)$
- **Transferability:** It is likely to be missclassified by another model f' trained on the same dataset
- **Black-box:** You can create a surrogate of the target model f' , and the adversarial example against f' is likely to be missclassified by the actual target.
- **Resilience:** You can distort it (e.g., print) and it is likely to still be missclassified by f_θ .

Properties of Adversarial Examples

- Let \tilde{x} be an adversarial example for model $f_\theta(\cdot)$
- **Transferability:** It is likely to be missclassified by another model f' trained on the same dataset
- **Black-box:** You can create a surrogate of the target model f' , and the adversarial example against f' is likely to be missclassified by the actual target.
- **Resilience:** You can distort it (e.g., print) and it is likely to still be missclassified by f_θ .

How to Defend?

$$\delta := \mathcal{P}_\Delta [\delta + \alpha \operatorname{sgn}(\nabla_\delta L(x + \delta, y))]$$

Question

What is the best strategy to defend against adversarial examples like these?

1. Ensure that the attacker cannot obtain parameters of the model θ through careful access controls.
2. Ensure that all or overwhelming majority of examples $x \in \Delta$ are classified the same.
3. During training, simulate the adversary and penalize its wins. In other words, generate adversarial examples and train on them with correct labels.
4. Train the model in such a way that the gradient does not point in the best direction
5. Preprocess the inputs to ensure small perturbations are blurred

How to Defend?

$$\delta := \mathcal{P}_\Delta [\delta + \alpha \operatorname{sgn}(\nabla_\delta L(x + \delta, y))]$$

Question

What is the best strategy to defend against adversarial examples like these?

1. Ensure that the attacker cannot obtain parameters of the model θ through careful access controls.
2. Ensure that all or overwhelming majority of examples $x \in \Delta$ are classified the same.
3. During training, simulate the adversary and penalize its wins. In other words, generate adversarial examples and train on them with correct labels.
4. Train the model in such a way that the gradient does not point in the best direction
5. Preprocess the inputs to ensure small perturbations are blurred

How to Defend?

$$\delta := \mathcal{P}_\Delta [\delta + \alpha \operatorname{sgn}(\nabla_\delta L(x + \delta, y))]$$

Question

What is the best strategy to defend against adversarial examples like these?

1. Ensure that the attacker cannot obtain parameters of the model θ through careful access controls.
2. Ensure that all or overwhelming majority of examples $x \in \Delta$ are classified the same.
3. During training, simulate the adversary and penalize its wins. In other words, generate adversarial examples and train on them with correct labels.
4. Train the model in such a way that the gradient does not point in the best direction
5. Preprocess the inputs to ensure small perturbations are blurred

How to Defend?

$$\delta := \mathcal{P}_\Delta [\delta + \alpha \operatorname{sgn}(\nabla_\delta L(x + \delta, y))]$$

Question

What is the best strategy to defend against adversarial examples like these?

1. Ensure that the attacker cannot obtain parameters of the model θ through careful access controls.
2. Ensure that all or overwhelming majority of examples $x \in \Delta$ are classified the same.
3. During training, simulate the adversary and penalize its wins. In other words, generate adversarial examples and train on them with correct labels.
4. Train the model in such a way that the gradient does not point in the best direction
5. Preprocess the inputs to ensure small perturbations are blurred

How to Defend?

$$\delta := \mathcal{P}_\Delta [\delta + \alpha \operatorname{sgn}(\nabla_\delta L(x + \delta, y))]$$

Question

What is the best strategy to defend against adversarial examples like these?

1. Ensure that the attacker cannot obtain parameters of the model θ through careful access controls.
2. Ensure that all or overwhelming majority of examples $x \in \Delta$ are classified the same.
3. During training, simulate the adversary and penalize its wins. In other words, generate adversarial examples and train on them with correct labels.
4. Train the model in such a way that the gradient does not point in the best direction
5. Preprocess the inputs to ensure small perturbations are blurred

How to Defend? (Answers)

- Ensure that the attacker cannot obtain parameters of the model θ through careful access controls.
 - Worst-case security principle breached: *Security by obscurity*
- Preprocess the inputs to ensure small perturbations are blurred
 - Relies on the fact that the adversary does not know what is the preprocessing
 - If adversary knew what was the preprocessing, they could adapt adversarial examples accordingly
 - Hence, worst-case security principle breached: *Security by obscurity*
- Train the model in such a way that the gradient does not point in the best direction
 - This is similar to *Security by obscurity*
 - There are other attacks that do not rely on the gradient.

How to Defend? (Answers)

- Ensure that the attacker cannot obtain parameters of the model θ through careful access controls.
 - Worst-case security principle breached: *Security by obscurity*
- Preprocess the inputs to ensure small perturbations are blurred
 - Relies on the fact that the adversary does not know what is the preprocessing
 - If adversary knew what was the preprocessing, they could adapt adversarial examples accordingly
 - Hence, worst-case security principle breached: *Security by obscurity*
- Train the model in such a way that the gradient does not point in the best direction
 - This is similar to *Security by obscurity*
 - There are other attacks that do not rely on the gradient.

How to Defend? (Answers)

- Ensure that the attacker cannot obtain parameters of the model θ through careful access controls.
 - Worst-case security principle breached: *Security by obscurity*
- Preprocess the inputs to ensure small perturbations are blurred
 - Relies on the fact that the adversary does not know what is the preprocessing
 - If adversary knew what was the preprocessing, they could adapt adversarial examples accordingly
 - Hence, worst-case security principle breached: *Security by obscurity*
- Train the model in such a way that the gradient does not point in the best direction
 - This is similar to *Security by obscurity*
 - There are other attacks that do not rely on the gradient.

How to Defend? (Answers)

- Ensure that the attacker cannot obtain parameters of the model θ through careful access controls.
 - Worst-case security principle breached: *Security by obscurity*
- Preprocess the inputs to ensure small perturbations are blurred
 - Relies on the fact that the adversary does not know what is the preprocessing
 - If adversary knew what was the preprocessing, they could adapt adversarial examples accordingly
 - Hence, worst-case security principle breached: *Security by obscurity*
- Train the model in such a way that the gradient does not point in the best direction
 - This is similar to *Security by obscurity*
 - There are other attacks that do not rely on the gradient.

How to Defend? (Answers)

- Ensure that the attacker cannot obtain parameters of the model θ through careful access controls.
 - Worst-case security principle breached: *Security by obscurity*
- Preprocess the inputs to ensure small perturbations are blurred
 - Relies on the fact that the adversary does not know what is the preprocessing
 - If adversary knew what was the preprocessing, they could adapt adversarial examples accordingly
 - Hence, worst-case security principle breached: *Security by obscurity*
- Train the model in such a way that the gradient does not point in the best direction
 - This is similar to *Security by obscurity*
 - There are other attacks that do not rely on the gradient.

How to Defend? (Answers)

- Ensure that the attacker cannot obtain parameters of the model θ through careful access controls.
 - Worst-case security principle breached: *Security by obscurity*
- Preprocess the inputs to ensure small perturbations are blurred
 - Relies on the fact that the adversary does not know what is the preprocessing
 - If adversary knew what was the preprocessing, they could adapt adversarial examples accordingly
 - Hence, worst-case security principle breached: *Security by obscurity*
- Train the model in such a way that the gradient does not point in the best direction
 - This is similar to *Security by obscurity*
 - There are other attacks that do not rely on the gradient.

How to Defend? (Answers)

- Ensure that the attacker cannot obtain parameters of the model θ through careful access controls.
 - Worst-case security principle breached: *Security by obscurity*
- Preprocess the inputs to ensure small perturbations are blurred
 - Relies on the fact that the adversary does not know what is the preprocessing
 - If adversary knew what was the preprocessing, they could adapt adversarial examples accordingly
 - Hence, worst-case security principle breached: *Security by obscurity*
- Train the model in such a way that the gradient does not point in the best direction
 - This is similar to *Security by obscurity*
 - There are other attacks that do not rely on the gradient.

How to Defend? (Answers)

- Ensure that the attacker cannot obtain parameters of the model θ through careful access controls.
 - Worst-case security principle breached: *Security by obscurity*
- Preprocess the inputs to ensure small perturbations are blurred
 - Relies on the fact that the adversary does not know what is the preprocessing
 - If adversary knew what was the preprocessing, they could adapt adversarial examples accordingly
 - Hence, worst-case security principle breached: *Security by obscurity*
- Train the model in such a way that the gradient does not point in the best direction
 - This is similar to *Security by obscurity*
 - There are other attacks that do not rely on the gradient.

How to Defend? (Answers)

- Ensure that the attacker cannot obtain parameters of the model θ through careful access controls.
 - Worst-case security principle breached: *Security by obscurity*
- Preprocess the inputs to ensure small perturbations are blurred
 - Relies on the fact that the adversary does not know what is the preprocessing
 - If adversary knew what was the preprocessing, they could adapt adversarial examples accordingly
 - Hence, worst-case security principle breached: *Security by obscurity*
- Train the model in such a way that the gradient does not point in the best direction
 - This is similar to *Security by obscurity*
 - There are other attacks that do not rely on the gradient.

How to Defend? (Answers)

- Ensure that all or overwhelming majority of examples $x \in \Delta$ are classified the same.
 - This is the ideal case.
 - Impossible in practice, but this is what people strive for.
- During training, simulate the adversary and penalize its wins. In other words, generate adversarial examples and train on them with correct labels.
 - This is called *adversarial training*
 - In practice, this defence only approximately protects against a particular attack, but it works.
 - This is your best bet at defending.

How to Defend? (Answers)

- Ensure that all or overwhelming majority of examples $x \in \Delta$ are classified the same.
 - This is the ideal case.
 - Impossible in practice, but this is what people strive for.
- During training, simulate the adversary and penalize its wins. In other words, generate adversarial examples and train on them with correct labels.
 - This is called *adversarial training*
 - In practice, this defence only approximately protects against a particular attack, but it works.
 - This is your best bet at defending.

How to Defend? (Answers)

- Ensure that all or overwhelming majority of examples $x \in \Delta$ are classified the same.
 - This is the ideal case.
 - Impossible in practice, but this is what people strive for.
- During training, simulate the adversary and penalize its wins. In other words, generate adversarial examples and train on them with correct labels.
 - This is called *adversarial training*
 - In practice, this defence only approximately protects against a particular attack, but it works.
 - This is your best bet at defending.

How to Defend? (Answers)

- Ensure that all or overwhelming majority of examples $x \in \Delta$ are classified the same.
 - This is the ideal case.
 - Impossible in practice, but this is what people strive for.
- During training, simulate the adversary and penalize its wins. In other words, generate adversarial examples and train on them with correct labels.
 - This is called *adversarial training*
 - In practice, this defence only approximately protects against a particular attack, but it works.
 - This is your best bet at defending.

How to Defend? (Answers)

- Ensure that all or overwhelming majority of examples $x \in \Delta$ are classified the same.
 - This is the ideal case.
 - Impossible in practice, but this is what people strive for.
- During training, simulate the adversary and penalize its wins. In other words, generate adversarial examples and train on them with correct labels.
 - This is called *adversarial training*
 - In practice, this defence only approximately protects against a particular attack, but it works.
 - This is your best bet at defending.

How to Defend? (Answers)

- Ensure that all or overwhelming majority of examples $x \in \Delta$ are classified the same.
 - This is the ideal case.
 - Impossible in practice, but this is what people strive for.
- During training, simulate the adversary and penalize its wins. In other words, generate adversarial examples and train on them with correct labels.
 - This is called *adversarial training*
 - In practice, this defence only approximately protects against a particular attack, but it works.
 - This is your best bet at defending.

How to Defend? (Answers)

- Ensure that all or overwhelming majority of examples $x \in \Delta$ are classified the same.
 - This is the ideal case.
 - Impossible in practice, but this is what people strive for.
- During training, simulate the adversary and penalize its wins. In other words, generate adversarial examples and train on them with correct labels.
 - This is called *adversarial training*
 - In practice, this defence only approximately protects against a particular attack, but it works.
 - This is your best bet at defending.

Adversarial Training

- Regular loss minimization objective:

$$\theta^* = \arg \min_{\theta} \sum_{x,y \in X} L(x, y; \theta)$$

- Recall: When defending, defend against even the strongest possible adversary model and attacks
- Adversarial loss minimization objective:

$$\theta^* = \arg \min_{\theta} \sum_{x,y \in X} \underbrace{\max_{\delta \in \Delta} L(x + \delta, y; \theta)}_{\text{Adversarial example}}$$

- Best we can do: solve the inner maximization exactly. Even this does not guarantee that there exist no adversarial examples in Δ
- PGD solves the inner maximization approximately \implies approximate security, but works better than most other proposed defenses in practice
- In fact, instead of finding adversarial examples, you can just randomly sample many examples from Δ (Cohen et al. [2019])

Adversarial Training

- Regular loss minimization objective:

$$\theta^* = \arg \min_{\theta} \sum_{x,y \in X} L(x, y; \theta)$$

- Recall: When defending, defend against even the strongest possible adversary model and attacks
- Adversarial loss minimization objective:

$$\theta^* = \arg \min_{\theta} \sum_{x,y \in X} \underbrace{\max_{\delta \in \Delta} L(x + \delta, y; \theta)}_{\text{Adversarial example}}$$

- Best we can do: solve the inner maximization exactly. Even this does not guarantee that there exist no adversarial examples in Δ
- PGD solves the inner maximization approximately \implies approximate security, but works better than most other proposed defenses in practice
- In fact, instead of finding adversarial examples, you can just randomly sample many examples from Δ (Cohen et al. [2019])

Adversarial Training

- Regular loss minimization objective:

$$\theta^* = \arg \min_{\theta} \sum_{x,y \in X} L(x, y; \theta)$$

- Recall: When defending, defend against even the strongest possible adversary model and attacks
- Adversarial loss minimization objective:

$$\theta^* = \arg \min_{\theta} \sum_{x,y \in X} \underbrace{\max_{\delta \in \Delta} L(x + \delta, y; \theta)}_{\text{Adversarial example}}$$

- Best we can do: solve the inner maximization exactly. Even this does not guarantee that there exist no adversarial examples in Δ
- PGD solves the inner maximization approximately \implies approximate security, but works better than most other proposed defenses in practice
- In fact, instead of finding adversarial examples, you can just randomly sample many examples from Δ (Cohen et al. [2019])

Adversarial Training

- Regular loss minimization objective:

$$\theta^* = \arg \min_{\theta} \sum_{x,y \in X} L(x, y; \theta)$$

- Recall: When defending, defend against even the strongest possible adversary model and attacks
- Adversarial loss minimization objective:

$$\theta^* = \arg \min_{\theta} \sum_{x,y \in X} \underbrace{\max_{\delta \in \Delta} L(x + \delta, y; \theta)}_{\text{Adversarial example}}$$

- Best we can do: solve the inner maximization exactly. Even this does not guarantee that there exist no adversarial examples in Δ
- PGD solves the inner maximization approximately \implies approximate security, but works better than most other proposed defenses in practice
- In fact, instead of finding adversarial examples, you can just randomly sample many examples from Δ (Cohen et al. [2019])

Adversarial Training

- Regular loss minimization objective:

$$\theta^* = \arg \min_{\theta} \sum_{x,y \in X} L(x, y; \theta)$$

- Recall: When defending, defend against even the strongest possible adversary model and attacks
- Adversarial loss minimization objective:

$$\theta^* = \arg \min_{\theta} \sum_{x,y \in X} \underbrace{\max_{\delta \in \Delta} L(x + \delta, y; \theta)}_{\text{Adversarial example}}$$

- Best we can do: solve the inner maximization exactly. Even this does not guarantee that there exist no adversarial examples in Δ
- PGD solves the inner maximization approximately \implies approximate security, but works better than most other proposed defenses in practice
- In fact, instead of finding adversarial examples, you can just randomly sample many examples from Δ (Cohen et al. [2019])

Adversarial Training

- Regular loss minimization objective:

$$\theta^* = \arg \min_{\theta} \sum_{x,y \in X} L(x, y; \theta)$$

- Recall: When defending, defend against even the strongest possible adversary model and attacks
- Adversarial loss minimization objective:

$$\theta^* = \arg \min_{\theta} \sum_{x,y \in X} \underbrace{\max_{\delta \in \Delta} L(x + \delta, y; \theta)}_{\text{Adversarial example}}$$

- Best we can do: solve the inner maximization exactly. Even this does not guarantee that there exist no adversarial examples in Δ
- PGD solves the inner maximization approximately \implies approximate security, but works better than most other proposed defenses in practice
- In fact, instead of finding adversarial examples, you can just randomly sample many examples from Δ (Cohen et al. [2019])

Big Picture: Security Implications of Non-Robust Features

- Recent research suggests: ML is not robust to imperceptible perturbations, because models pick up *non-robust features* specific to the dataset, that are not interpretable by humans (Ilyas et al. [2019])
- We have some idea on how to reduce the impact of this problem in $\Delta = \{\delta \mid \|\delta\| \leq \varepsilon\}$

Big Picture: Security Implications of Non-Robust Features

- Recent research suggests: ML is not robust to imperceptible perturbations, because models pick up *non-robust features* specific to the dataset, that are not interpretable by humans (Ilyas et al. [2019])
- We have some idea on how to reduce the impact of this problem in $\Delta = \{\delta \mid \| \delta \| \leq \varepsilon\}$

Big Picture: Security Implications of Non-Robust Features

- Recent research suggests: ML is not robust to imperceptible perturbations, because models pick up *non-robust features* specific to the dataset, that are not interpretable by humans (Ilyas et al. [2019])
- We have some idea on how to reduce the impact of this problem in $\Delta = \{\delta \mid \| \delta \| \leq \varepsilon\}$

Big Picture: Security Implications of Non-Robust Features

- Recent research suggests: ML is not robust to imperceptible perturbations, because models pick up *non-robust features* specific to the dataset, that are not interpretable by humans (Ilyas et al. [2019])
- We have some idea on how to reduce the impact of this problem in $\Delta = \{\delta \mid \|\delta\| \leq \varepsilon\}$

Question

Do the defenses before (fixing non-robust features) solve the security of ML against evasion attacks?

- The threat model of $\{\delta \mid \|\delta\| \leq \varepsilon\}$ covers only one kind of attack.
- Opinion: ML community focuses on this adversarial robustness because of the mathematically elegant form
- Opinion: Robustness to imperceptible perturbations is more important to ML theory than ML security

Big Picture: Security Implications of Non-Robust Features

- Recent research suggests: ML is not robust to imperceptible perturbations, because models pick up *non-robust features* specific to the dataset, that are not interpretable by humans (Ilyas et al. [2019])
- We have some idea on how to reduce the impact of this problem in $\Delta = \{\delta \mid \|\delta\| \leq \varepsilon\}$

Question

Do the defenses before (fixing non-robust features) solve the security of ML against evasion attacks?

- The threat model of $\{\delta \mid \|\delta\| \leq \varepsilon\}$ covers only one kind of attack.
- Opinion: ML community focuses on this adversarial robustness because of the mathematically elegant form
- Opinion: Robustness to imperceptible perturbations is more important to ML theory than ML security

Big Picture: Security Implications of Non-Robust Features

- Recent research suggests: ML is not robust to imperceptible perturbations, because models pick up *non-robust features* specific to the dataset, that are not interpretable by humans (Ilyas et al. [2019])
- We have some idea on how to reduce the impact of this problem in $\Delta = \{\delta \mid \|\delta\| \leq \varepsilon\}$

Question

Do the defenses before (fixing non-robust features) solve the security of ML against evasion attacks?

- The threat model of $\{\delta \mid \|\delta\| \leq \varepsilon\}$ covers only one kind of attack.
- Opinion: ML community focuses on this adversarial robustness because of the mathematically elegant form
- Opinion: Robustness to imperceptible perturbations is more important to ML theory than ML security

Big Picture: Security Implications of Non-Robust Features

- Recent research suggests: ML is not robust to imperceptible perturbations, because models pick up *non-robust features* specific to the dataset, that are not interpretable by humans (Ilyas et al. [2019])
- We have some idea on how to reduce the impact of this problem in $\Delta = \{\delta \mid \|\delta\| \leq \varepsilon\}$

Question

Do the defenses before (fixing non-robust features) solve the security of ML against evasion attacks?

- The threat model of $\{\delta \mid \|\delta\| \leq \varepsilon\}$ covers only one kind of attack.
- Opinion: ML community focuses on this adversarial robustness because of the mathematically elegant form
- Opinion: Robustness to imperceptible perturbations is more important to ML theory than ML security

Beyond Imperceptible Perturbations



(a)

(b)

(c)

(b) and (c) have the same $\|\delta\|$

Image Credit: Jacobsen et al. [2019]

Beyond Imperceptible Perturbations

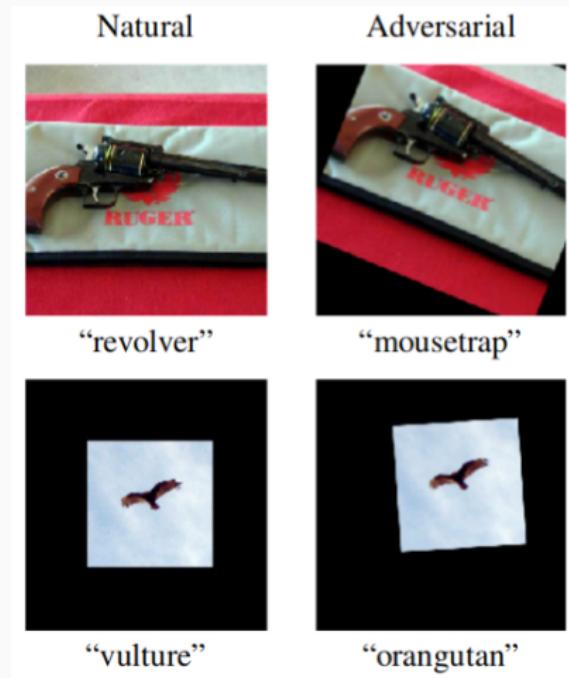


Image Credit: Engstrom et al. [2017]

Beyond Imperceptible Perturbations



These are not even imperceptible!
Image Credit: Sharif et al. [2016]

Beyond Imperceptible Perturbations



What about discrete domains?

See Kulynych et al. [2018]

Takeaways

- Currently, discussion about adversarial robustness of ML at test time puts robustness to imperceptible perturbations in the spotlight
- This problem is of interest both to ML Theory and ML Security
- However, this likely is not the main security concern for your application, unless you have a security-critical image classifier
- There are many other attack vectors against classifiers at test time
- Beyond imperceptible perturbation for images, the field seems very much in its infancy, many open problems.
- Adversarial training approach likely provides some remedy from different kind of attacks – but without any formal guarantees

Takeaways

- Currently, discussion about adversarial robustness of ML at test time puts robustness to imperceptible perturbations in the spotlight
- This problem is of interest both to ML Theory and ML Security
- However, this likely is not the main security concern for your application, unless you have a security-critical image classifier
- There are many other attack vectors against classifiers at test time
- Beyond imperceptible perturbation for images, the field seems very much in its infancy, many open problems.
- Adversarial training approach likely provides some remedy from different kind of attacks – but without any formal guarantees

Takeaways

- Currently, discussion about adversarial robustness of ML at test time puts robustness to imperceptible perturbations in the spotlight
- This problem is of interest both to ML Theory and ML Security
- However, this likely is not the main security concern for your application, unless you have a security-critical image classifier
- There are many other attack vectors against classifiers at test time
- Beyond imperceptible perturbation for images, the field seems very much in its infancy, many open problems.
- Adversarial training approach likely provides some remedy from different kind of attacks – but without any formal guarantees

Takeaways

- Currently, discussion about adversarial robustness of ML at test time puts robustness to imperceptible perturbations in the spotlight
- This problem is of interest both to ML Theory and ML Security
- However, this likely is not the main security concern for your application, unless you have a security-critical image classifier
- There are many other attack vectors against classifiers at test time
- Beyond imperceptible perturbation for images, the field seems very much in its infancy, many open problems.
- Adversarial training approach likely provides some remedy from different kind of attacks – but without any formal guarantees

Takeaways

- Currently, discussion about adversarial robustness of ML at test time puts robustness to imperceptible perturbations in the spotlight
- This problem is of interest both to ML Theory and ML Security
- However, this likely is not the main security concern for your application, unless you have a security-critical image classifier
- There are many other attack vectors against classifiers at test time
- Beyond imperceptible perturbation for images, the field seems very much in its infancy, many open problems.
- Adversarial training approach likely provides some remedy from different kind of attacks – but without any formal guarantees

Takeaways

- Currently, discussion about adversarial robustness of ML at test time puts robustness to imperceptible perturbations in the spotlight
- This problem is of interest both to ML Theory and ML Security
- However, this likely is not the main security concern for your application, unless you have a security-critical image classifier
- There are many other attack vectors against classifiers at test time
- Beyond imperceptible perturbation for images, the field seems very much in its infancy, many open problems.
- Adversarial training approach likely provides some remedy from different kind of attacks – but without any formal guarantees

Quick Recap: Security Principles

Are any security principles violated here?

- You use a custom random-forest-based network-intrusion detection system (not open-source). You are not afraid of hackers evading it because they don't know how it works.
- A researcher has emailed you about a vulnerability in the computer vision system of your automated drones. You sue them.
- Your traffic-sign recognition system is provably robust to $\epsilon = 0.65$ ℓ_1 -perturbations of all examples in your training set. This is state-of-the-art ϵ for this task.
- You have gone through all existing evasion attacks and have constructed a detector for each. It works 100% of the time. In deployment, your model first runs the detector, and rejects the input if it looks like an adversarial example.

Quick Recap: Security Principles

Are any security principles violated here?

- You use a custom random-forest-based network-intrusion detection system (not open-source). You are not afraid of hackers evading it because they don't know how it works.
- A researcher has emailed you about a vulnerability in the computer vision system of your automated drones. You sue them.
- Your traffic-sign recognition system is provably robust to $\epsilon = 0.65$ ℓ_1 -perturbations of all examples in your training set. This is state-of-the-art ϵ for this task.
- You have gone through all existing evasion attacks and have constructed a detector for each. It works 100% of the time. In deployment, your model first runs the detector, and rejects the input if it looks like an adversarial example.

Quick Recap: Security Principles

Are any security principles violated here?

- You use a custom random-forest-based network-intrusion detection system (not open-source). You are not afraid of hackers evading it because they don't know how it works.
- A researcher has emailed you about a vulnerability in the computer vision system of your automated drones. You sue them.
- Your traffic-sign recognition system is provably robust to $\epsilon = 0.65$ ℓ_1 -perturbations of all examples in your training set. This is state-of-the-art ϵ for this task.
- You have gone through all existing evasion attacks and have constructed a detector for each. It works 100% of the time. In deployment, your model first runs the detector, and rejects the input if it looks like an adversarial example.

Quick Recap: Security Principles

Are any security principles violated here?

- You use a custom random-forest-based network-intrusion detection system (not open-source). You are not afraid of hackers evading it because they don't know how it works.
- A researcher has emailed you about a vulnerability in the computer vision system of your automated drones. You sue them.
- Your traffic-sign recognition system is provably robust to $\epsilon = 0.65$ ℓ_1 -perturbations of all examples in your training set. This is state-of-the-art ϵ for this task.
- You have gone through all existing evasion attacks and have constructed a detector for each. It works 100% of the time. In deployment, your model first runs the detector, and rejects the input if it looks like an adversarial example.

Dealing with Black Boxes

Recall: Adversarial Knowledge

- No 100% consistent terms
- White-box: usually architecture f , parameters θ , procedure T
- Black-box: often knows architecture f but no parameters, and has query access to f or T
- Agnostic black-box: only query access to f or T

Recall: Adversarial Knowledge

- No 100% consistent terms
- White-box: usually architecture f , parameters θ , procedure T
- Black-box: often knows architecture f but no parameters, and has query access to f or T
- Agnostic black-box: only query access to f or T

Recall: Adversarial Knowledge

- No 100% consistent terms
- White-box: usually architecture f , parameters θ , procedure T
- Black-box: often knows architecture f but no parameters, and has query access to f or T
- Agnostic black-box: only query access to f or T

Recall: Adversarial Knowledge

- No 100% consistent terms
- White-box: usually architecture f , parameters θ , procedure T
- Black-box: often knows architecture f but no parameters, and has query access to f or T
- Agnostic black-box: only query access to f or T

Black-Box Access

- The PGD attack only works in the white-box setting: need the gradient ∇L :

$$\delta := \mathcal{P}_\Delta [\delta + \alpha \operatorname{sgn}(\nabla_\delta L(x + \delta, y))]$$

- The PGD attack only works in the white-box setting: need the gradient ∇L :

$$\delta := \mathcal{P}_\Delta [\delta + \alpha \operatorname{sgn}(\nabla_\delta L(x + \delta, y))]$$

Group Discussion

Can you attack if a model has non-differentiable components?

- The PGD attack only works in the white-box setting: need the gradient ∇L :

$$\delta := \mathcal{P}_\Delta [\delta + \alpha \operatorname{sgn}(\nabla_\delta L(x + \delta, y))]$$

Group Discussion

Can you attack if a model has non-differentiable components?

Group Discussion

Can you attack if you only have query access to $f(\cdot)$?

Black-Box Attacks at Test Time

Classical approach (Lowd and Meek [2005])

1. Reverse engineer the model
2. Run a white-box attack

Smarter approach (Chen et al. [2017])

1. Run a black-box optimization algorithm that only needs query access to $f(\cdot)$

Black-Box Attacks at Test Time

Classical approach (Lowd and Meek [2005])

1. Reverse engineer the model
2. Run a white-box attack

Smarter approach (Chen et al. [2017])

1. Run a black-box optimization algorithm that only needs query access to $f(\cdot)$

Black-Box Attacks at Test Time

Classical approach (Lowd and Meek [2005])

1. Reverse engineer the model
2. Run a white-box attack

Smarter approach (Chen et al. [2017])

1. Run a black-box optimization algorithm that only needs query access to $f(\cdot)$

Black-Box Attacks at Test Time

Classical approach (Lowd and Meek [2005])

1. Reverse engineer the model
2. Run a white-box attack

Smarter approach (Chen et al. [2017])

1. Run a black-box optimization algorithm that only needs query access to $f(\cdot)$

Estimating the Gradient

- Can estimate the gradient in a black-box manner from two evaluations:

$$\frac{d}{dx_i} f(x) = \frac{f(x + \delta e_i) - f(x - \delta e_i)}{2\delta}$$

for small δ , and basis vectors e_i .

- Attack principle: run PGD but estimate gradient in every iteration (Chen et al. [2017])

Estimating the Gradient

- Can estimate the gradient in a black-box manner from two evaluations:

$$\frac{d}{dx_i} f(x) = \frac{f(x + \delta e_i) - f(x - \delta e_i)}{2\delta}$$

for small δ , and basis vectors e_i .

- Attack principle: run PGD but estimate gradient in every iteration (Chen et al. [2017])

Estimating the Gradient

- Can estimate the gradient in a black-box manner from two evaluations:

$$\frac{d}{dx_i} f(x) = \frac{f(x + \delta e_i) - f(x - \delta e_i)}{2\delta}$$

for small δ , and basis vectors e_i .

- Attack principle: run PGD but estimate gradient in every iteration (Chen et al. [2017])

Estimating the Gradient

- Can estimate the gradient in a black-box manner from two evaluations:

$$\frac{d}{dx_i} f(x) = \frac{f(x + \delta e_i) - f(x - \delta e_i)}{2\delta}$$

for small δ , and basis vectors e_i .

- Attack principle: run PGD but estimate gradient in every iteration (Chen et al. [2017])

You already know this from worst-case security but...

Adding non-differentiable components is not a valid defense.

Model Stealing

- We can also steal the model parameters by querying it many times
 - Construct a large dataset $X' = \{(x, f(x)), \dots\}$
 - Train $f_{\theta'}$.
-
- Need $d + 1$ to steal a linear model using equation solving
 - But, need a lot more queries to steal a neural net
 - For a net with 2K parameters, need 11K queries to get 99.9% similarity (Tramèr et al. [2016])

Model Stealing

- We can also steal the model parameters by querying it many times
 - Construct a large dataset $X' = \{(x, f(x)), \dots\}$
 - Train $f_{\theta'}$.
-
- Need $d + 1$ to steal a linear model using equation solving
 - But, need a lot more queries to steal a neural net
 - For a net with 2K parameters, need 11K queries to get 99.9% similarity (Tramèr et al. [2016])

Model Stealing

- We can also steal the model parameters by querying it many times
 - Construct a large dataset $X' = \{(x, f(x)), \dots\}$
 - Train $f_{\theta'}$.
-
- Need $d + 1$ to steal a linear model using equation solving
 - But, need a lot more queries to steal a neural net
 - For a net with $2K$ parameters, need $11K$ queries to get 99.9% similarity (Tramèr et al. [2016])

Model Stealing

- We can also steal the model parameters by querying it many times
 - Construct a large dataset $X' = \{(x, f(x)), \dots\}$
 - Train $f_{\theta'}$.
-
- Need $d + 1$ to steal a linear model using equation solving
 - But, need a lot more queries to steal a neural net
 - For a net with $2K$ parameters, need $11K$ queries to get 99.9% similarity (Tramèr et al. [2016])

Model Stealing

- We can also steal the model parameters by querying it many times
- Construct a large dataset $X' = \{(x, f(x)), \dots\}$
- Train $f_{\theta'}$.

Question

Let $f(x)$ be a linear model. Its decision boundary is $h(x) = w \cdot x + b$. w is a 750-dimensional vector. How many queries are needed to steal $f(x)$?

- Need $d+1$ to steal a linear model using equation solving
- But, need a lot more queries to steal a neural net
- For a net with 2K parameters, need 11K queries to get 99.9% similarity (Tramèr et al. [2016])

Model Stealing

- We can also steal the model parameters by querying it many times
- Construct a large dataset $X' = \{(x, f(x)), \dots\}$
- Train $f_{\theta'}$.

Question

Let $f(x)$ be a linear model. Its decision boundary is $h(x) = w \cdot x + b$. w is a 750-dimensional vector. How many queries are needed to steal $f(x)$?

- Need $d+1$ to steal a linear model using equation solving
- But, need a lot more queries to steal a neural net
- For a net with 2K parameters, need 11K queries to get 99.9% similarity (Tramèr et al. [2016])

Model Stealing

- We can also steal the model parameters by querying it many times
- Construct a large dataset $X' = \{(x, f(x)), \dots\}$
- Train $f_{\theta'}$.

Question

Let $f(x)$ be a linear model. Its decision boundary is $h(x) = w \cdot x + b$. w is a 750-dimensional vector. How many queries are needed to steal $f(x)$?

- Need $d + 1$ to steal a linear model using equation solving
- But, need a lot more queries to steal a neural net
- For a net with 2K parameters, need 11K queries to get 99.9% similarity (Tramèr et al. [2016])

Model Stealing

- We can also steal the model parameters by querying it many times
- Construct a large dataset $X' = \{(x, f(x)), \dots\}$
- Train $f_{\theta'}$.

Question

Let $f(x)$ be a linear model. Its decision boundary is $h(x) = w \cdot x + b$. w is a 750-dimensional vector. How many queries are needed to steal $f(x)$?

- Need $d + 1$ to steal a linear model using equation solving
- But, need a lot more queries to steal a neural net
- For a net with 2K parameters, need 11K queries to get 99.9% similarity (Tramèr et al. [2016])

Model Stealing

- We can also steal the model parameters by querying it many times
- Construct a large dataset $X' = \{(x, f(x)), \dots\}$
- Train $f_{\theta'}$.

Question

Let $f(x)$ be a linear model. Its decision boundary is $h(x) = w \cdot x + b$. w is a 750-dimensional vector. How many queries are needed to steal $f(x)$?

- Need $d + 1$ to steal a linear model using equation solving
- But, need a lot more queries to steal a neural net
- For a net with 2K parameters, need 11K queries to get 99.9% similarity (Tramèr et al. [2016])

Takeaways

- If your system is queriable by adversaries, it can be reverse-engineered (model stealing)
- Evasion attacks can be mounted even without reverse engineering: black-box attacks.
- You don't need the gradient to attack: non-differentiable classifiers can be attacked using black-box attacks

Takeaways

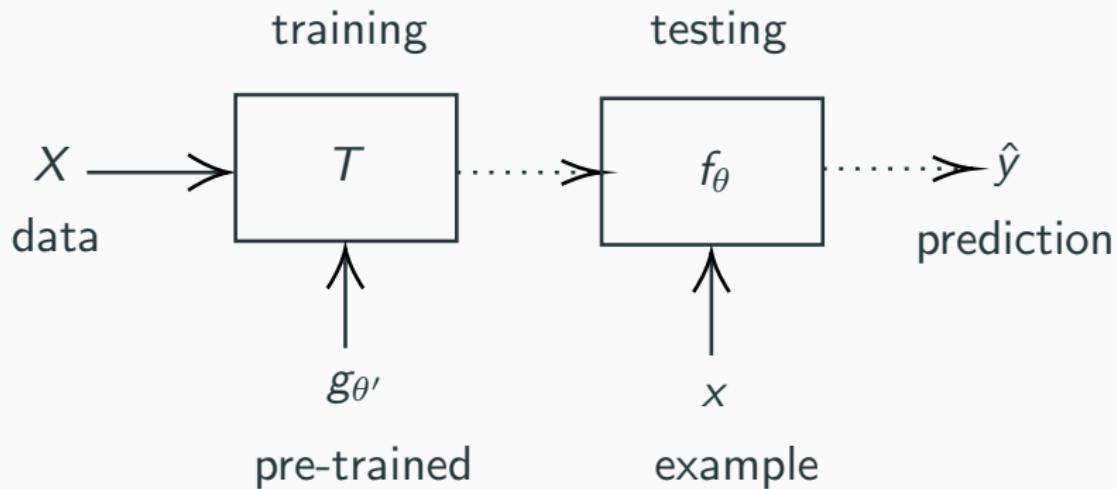
- If your system is queriable by adversaries, it can be reverse-engineered (model stealing)
- Evasion attacks can be mounted even without reverse engineering: black-box attacks.
- You don't need the gradient to attack: non-differentiable classifiers can be attacked using black-box attacks

Takeaways

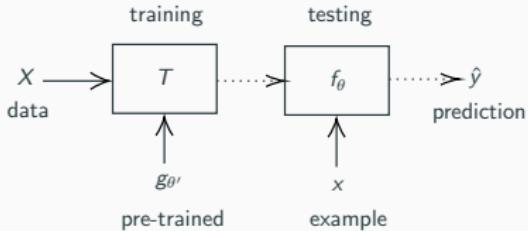
- If your system is queriable by adversaries, it can be reverse-engineered (model stealing)
- Evasion attacks can be mounted even without reverse engineering: black-box attacks.
- You don't need the gradient to attack: non-differentiable classifiers can be attacked using black-box attacks

Attacks at Training Time

Recall: Machine Learning Pipeline



Poisoning Attacks

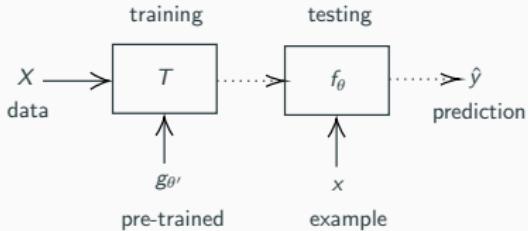


- Adversarial Goals: Reduce performance of the model or modify the task
- Adversarial Capabilities: Inject or modify training examples X , or pre-trained components $g_{\theta'}$
- Adversarial Knowledge: Knows the algorithm T

Question

Is this a realistic adversary model? Are there any relevant settings?

Poisoning Attacks

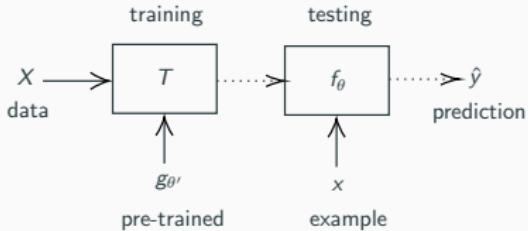


- Adversarial Goals: Reduce performance of the model or modify the task
- Adversarial Capabilities: Inject or modify training examples X , or pre-trained components $g_{\theta'}$
- Adversarial Knowledge: Knows the algorithm T

Question

Is this a realistic adversary model? Are there any relevant settings?

Poisoning Attacks



- Adversarial Goals: Reduce performance of the model or modify the task
- Adversarial Capabilities: Inject or modify training examples X , or pre-trained components $g_{\theta'}$
- Adversarial Knowledge: Knows the algorithm T

Question

Is this a realistic adversary model? Are there any relevant settings?

Poisoning Attacks with Poisoned Examples

Naïve approach:

- Mess up with data (e.g., modify labels)
 - Retrain and check if helped
 - Adapt and repeat
-
- Does not scale
 - Retraining is a bottleneck
 - Need a tool to guide what features or examples are important

Poisoning Attacks with Poisoned Examples

Naïve approach:

- Mess up with data (e.g., modify labels)
- Retrain and check if helped
- Adapt and repeat
- Does not scale
- Retraining is a bottleneck
- Need a tool to guide what features or examples are important

Poisoning Attacks with Poisoned Examples

Naïve approach:

- Mess up with data (e.g., modify labels)
 - Retrain and check if helped
 - **Adapt and repeat**
-
- Does not scale
 - Retraining is a bottleneck
 - Need a tool to guide what features or examples are important

Poisoning Attacks with Poisoned Examples

Naïve approach:

- Mess up with data (e.g., modify labels)
 - Retrain and check if helped
 - Adapt and repeat
-
- Does not scale
 - Retraining is a bottleneck
 - Need a tool to guide what features or examples are important

Poisoning Attacks with Poisoned Examples

Naïve approach:

- Mess up with data (e.g., modify labels)
- Retrain and check if helped
- Adapt and repeat
- Does not scale
- Retraining is a bottleneck
- Need a tool to guide what features or examples are important

Poisoning Attacks with Poisoned Examples

Naïve approach:

- Mess up with data (e.g., modify labels)
- Retrain and check if helped
- Adapt and repeat
- Does not scale
- **Retraining is a bottleneck**
- Need a tool to guide what features or examples are important

Poisoning Attacks with Poisoned Examples

Naïve approach:

- Mess up with data (e.g., modify labels)
- Retrain and check if helped
- Adapt and repeat
- Does not scale
- Retraining is a bottleneck
- Need a tool to guide what features or examples are important

In Search of a Closed-Form Expression

- Recall the empirical loss minimization problem:

$$\theta^* = \arg \min_{\theta} \sum_{x,y \in X} L(x, y; \theta)$$

- What if an example $z = (x, y)$ was upweighted by ε ?

$$\theta_{z,\varepsilon}^* = \arg \min_{\theta} \left[\sum_{x',y' \in X} L(x', y'; \theta) + \varepsilon L(z; \theta) \right]$$

- These correspond to two different classifiers
- Let's call the difference between their weights Δ :

$$\Delta_{z,\varepsilon} := \theta_{z,\varepsilon}^* - \theta^*$$

In Search of a Closed-Form Expression

- Recall the empirical loss minimization problem:

$$\theta^* = \arg \min_{\theta} \sum_{x,y \in X} L(x, y; \theta)$$

- What if an example $z = (x, y)$ was upweighted by ε ?

$$\theta_{z,\varepsilon}^* = \arg \min_{\theta} \left[\sum_{x',y' \in X} L(x', y'; \theta) + \varepsilon L(z; \theta) \right]$$

- These correspond to two different classifiers
- Let's call the difference between their weights Δ :

$$\Delta_{z,\varepsilon} := \theta_{z,\varepsilon}^* - \theta^*$$

In Search of a Closed-Form Expression

- Recall the empirical loss minimization problem:

$$\theta^* = \arg \min_{\theta} \sum_{x,y \in X} L(x, y; \theta)$$

- What if an example $z = (x, y)$ was upweighted by ε ?

$$\theta_{z,\varepsilon}^* = \arg \min_{\theta} \left[\sum_{x',y' \in X} L(x', y'; \theta) + \varepsilon L(z; \theta) \right]$$

- These correspond to two different classifiers
- Let's call the difference between their weights Δ :

$$\Delta_{z,\varepsilon} := \theta_{z,\varepsilon}^* - \theta^*$$

In Search of a Closed-Form Expression

- Recall the empirical loss minimization problem:

$$\theta^* = \arg \min_{\theta} \sum_{x,y \in X} L(x, y; \theta)$$

- What if an example $z = (x, y)$ was upweighted by ε ?

$$\theta_{z,\varepsilon}^* = \arg \min_{\theta} \left[\sum_{x',y' \in X} L(x', y'; \theta) + \varepsilon L(z; \theta) \right]$$

- These correspond to two different classifiers
- Let's call the difference between their weights Δ :

$$\Delta_{z,\varepsilon} := \theta_{z,\varepsilon}^* - \theta^*$$

Strategy: First-Order Approximation

1. Recall the first-order Taylor approximation:

$$f(\mathbf{x}) \approx f(\mathbf{x}') + \nabla f(\mathbf{x}') \cdot (\mathbf{x} - \mathbf{x}')$$

2. Recall the classifier with upweighted z :

$$\theta_{z,\varepsilon}^* = \arg \min_{\theta} \underbrace{\sum_{x',y' \in X} L(x', y'; \theta)}_{R(\theta)} + \varepsilon L(z; \theta)$$

3. Recall that a global optimizer $\theta_{z,\varepsilon}^*$ satisfies:

$$\nabla R(\theta_{z,\varepsilon}^*) + \varepsilon \nabla L(z; \theta_{z,\varepsilon}^*) = 0$$

Derivation

How can we get a first-order approximation of $\frac{d}{d\varepsilon} \Delta_{z,\varepsilon} = \frac{d}{d\varepsilon} [\theta_{z,\varepsilon}^* - \theta^*]$?

Strategy: First-Order Approximation

1. Recall the first-order Taylor approximation:

$$f(\mathbf{x}) \approx f(\mathbf{x}') + \nabla f(\mathbf{x}') \cdot (\mathbf{x} - \mathbf{x}')$$

2. Recall the classifier with upweighted z :

$$\theta_{z,\varepsilon}^* = \arg \min_{\theta} \underbrace{\sum_{x',y' \in X} L(x', y'; \theta)}_{R(\theta)} + \varepsilon L(z; \theta)$$

3. Recall that a global optimizer $\theta_{z,\varepsilon}^*$ satisfies:

$$\nabla R(\theta_{z,\varepsilon}^*) + \varepsilon \nabla L(z; \theta_{z,\varepsilon}^*) = 0$$

Derivation

How can we get a first-order approximation of $\frac{d}{d\varepsilon} \Delta_{z,\varepsilon} = \frac{d}{d\varepsilon} [\theta_{z,\varepsilon}^* - \theta^*]$?

Strategy: First-Order Approximation

1. Recall the first-order Taylor approximation:

$$f(\mathbf{x}) \approx f(\mathbf{x}') + \nabla f(\mathbf{x}') \cdot (\mathbf{x} - \mathbf{x}')$$

2. Recall the classifier with upweighted z :

$$\theta_{z,\varepsilon}^* = \arg \min_{\theta} \underbrace{\sum_{x',y' \in X} L(x', y'; \theta)}_{R(\theta)} + \varepsilon L(z; \theta)$$

3. Recall that a global optimizer $\theta_{z,\varepsilon}^*$ satisfies:

$$\nabla R(\theta_{z,\varepsilon}^*) + \varepsilon \nabla L(z; \theta_{z,\varepsilon}^*) = 0$$

Derivation

How can we get a first-order approximation of $\frac{d}{d\varepsilon} \Delta_{z,\varepsilon} = \frac{d}{d\varepsilon} [\theta_{z,\varepsilon}^* - \theta^*]$?

Strategy: First-Order Approximation

1. Recall the first-order Taylor approximation:

$$f(\mathbf{x}) \approx f(\mathbf{x}') + \nabla f(\mathbf{x}') \cdot (\mathbf{x} - \mathbf{x}')$$

2. Recall the classifier with upweighted z :

$$\theta_{z,\varepsilon}^* = \arg \min_{\theta} \underbrace{\sum_{x',y' \in X} L(x', y'; \theta)}_{R(\theta)} + \varepsilon L(z; \theta)$$

3. Recall that a global optimizer $\theta_{z,\varepsilon}^*$ satisfies:

$$\nabla R(\theta_{z,\varepsilon}^*) + \varepsilon \nabla L(z; \theta_{z,\varepsilon}^*) = 0$$

Derivation

How can we get a first-order approximation of $\frac{d}{d\varepsilon} \Delta_{z,\varepsilon} = \frac{d}{d\varepsilon} [\theta_{z,\varepsilon}^* - \theta^*]$?

Strategy: First-Order Approximation

1. Recall the first-order Taylor approximation:

$$f(\mathbf{x}) \approx f(\mathbf{x}') + \nabla f(\mathbf{x}') \cdot (\mathbf{x} - \mathbf{x}')$$

2. Recall the classifier with upweighted z :

$$\theta_{z,\varepsilon}^* = \arg \min_{\theta} \underbrace{\sum_{x',y' \in X} L(x', y'; \theta)}_{R(\theta)} + \varepsilon L(z; \theta)$$

3. Recall that a global optimizer $\theta_{z,\varepsilon}^*$ satisfies:

$$\nabla R(\theta_{z,\varepsilon}^*) + \varepsilon \nabla L(z; \theta_{z,\varepsilon}^*) = 0$$

Derivation

How can we get a first-order approximation of $\frac{d}{d\varepsilon} \Delta_{z,\varepsilon} = \frac{d}{d\varepsilon} [\theta_{z,\varepsilon}^* - \theta^*]$?

Deriving the Influence Function

1. Rewrite (3) in terms of θ^* :

$$0 \approx [\nabla R(\theta^*) + \varepsilon \nabla L(z; \theta^*) + [\mathcal{H}R(\theta^*) + \varepsilon \mathcal{H}L(z; \theta^*)] \Delta_{z, \varepsilon}],$$

where $\mathcal{H}f(x)$ is a Hessian matrix.

2. Solve for $\Delta_{z, \varepsilon}$:

$$\Delta_{z, \varepsilon} \approx -[\nabla R(\theta^*) + \varepsilon \nabla L(z; \theta^*)] [\mathcal{H}R(\theta^*) + \varepsilon \mathcal{H}L(z; \theta^*)]^{-1}$$

3. Because $\nabla R(\theta^*) = 0$ and assuming ε is small (Koh and Liang [2017]):

$$\Delta_{z, \varepsilon} \approx -\varepsilon \nabla L(z; \theta^*) [\mathcal{H}R(\theta^*)]^{-1}$$

Deriving the Influence Function

1. Rewrite (3) in terms of θ^* :

$$0 \approx [\nabla R(\theta^*) + \varepsilon \nabla L(z; \theta^*)] + \\ [\mathcal{H}R(\theta^*) + \varepsilon \mathcal{H}L(z; \theta^*)] \Delta_{z, \varepsilon},$$

where $\mathcal{H}f(x)$ is a Hessian matrix.

2. Solve for $\Delta_{z, \varepsilon}$:

$$\Delta_{z, \varepsilon} \approx -[\nabla R(\theta^*) + \varepsilon \nabla L(z; \theta^*)] \\ [\mathcal{H}R(\theta^*) + \varepsilon \mathcal{H}L(z; \theta^*)]^{-1}$$

3. Because $\nabla R(\theta^*) = 0$ and assuming ε is small (Koh and Liang [2017]):

$$\Delta_{z, \varepsilon} \approx -\varepsilon \nabla L(z; \theta^*) [\mathcal{H}R(\theta^*)]^{-1}$$

Deriving the Influence Function

1. Rewrite (3) in terms of θ^* :

$$0 \approx [\nabla R(\theta^*) + \varepsilon \nabla L(z; \theta^*)] + \\ [\mathcal{H}R(\theta^*) + \varepsilon \mathcal{H}L(z; \theta^*)] \Delta_{z, \varepsilon},$$

where $\mathcal{H}f(x)$ is a Hessian matrix.

2. Solve for $\Delta_{z, \varepsilon}$:

$$\Delta_{z, \varepsilon} \approx -[\nabla R(\theta^*) + \varepsilon \nabla L(z; \theta^*)] \\ [\mathcal{H}R(\theta^*) + \varepsilon \mathcal{H}L(z; \theta^*)]^{-1}$$

3. Because $\nabla R(\theta^*) = 0$ and assuming ε is small (Koh and Liang [2017]):

$$\Delta_{z, \varepsilon} \approx -\varepsilon \nabla L(z; \theta^*) [\mathcal{H}R(\theta^*)]^{-1}$$

Influence Function

1. We call the rate of change of $\Delta_{z,\varepsilon}$ *the influence function*:

$$\frac{d}{d\varepsilon} \Delta_{z,\varepsilon} = \nabla L(z; \theta^*) [\mathcal{H}R(\theta^*)]^{-1}$$

2. What about poisoning examples? Let $z_\delta = (x + \delta, y)$
3. Modify the upweighted model:

$$\theta_{z,z_\delta,\varepsilon}^* = \arg \min_{\theta} [R(\theta) - \varepsilon L(z, \theta) + \varepsilon L(z_\delta, \theta)]$$

Influence Function

1. We call the rate of change of $\Delta_{z,\varepsilon}$ *the influence function*:

$$\frac{d}{d\varepsilon} \Delta_{z,\varepsilon} = \nabla L(z; \theta^*) [\mathcal{H}R(\theta^*)]^{-1}$$

2. What about poisoning examples? Let $z_\delta = (x + \delta, y)$
3. Modify the upweighted model:

$$\theta_{z, z_\delta, \varepsilon}^* = \arg \min_{\theta} [R(\theta) - \varepsilon L(z, \theta) + \varepsilon L(z_\delta, \theta)]$$

Influence Function

1. We call the rate of change of $\Delta_{z,\varepsilon}$ *the influence function*:

$$\frac{d}{d\varepsilon} \Delta_{z,\varepsilon} = \nabla L(z; \theta^*) [\mathcal{H}R(\theta^*)]^{-1}$$

2. What about poisoning examples? Let $z_\delta = (x + \delta, y)$
3. Modify the upweighted model:

$$\theta_{z,z_\delta,\varepsilon}^* = \arg \min_{\theta} [R(\theta) - \varepsilon L(z, \theta) + \varepsilon L(z_\delta, \theta)]$$

Influence Function

1. We call the rate of change of $\Delta_{z,\varepsilon}$ *the influence function*:

$$\frac{d}{d\varepsilon} \Delta_{z,\varepsilon} = \nabla L(z; \theta^*) [\mathcal{H}R(\theta^*)]^{-1}$$

2. What about poisoning examples? Let $z_\delta = (x + \delta, y)$
3. Modify the upweighted model:

$$\theta_{z,z_\delta,\varepsilon}^* = \arg \min_{\theta} [R(\theta) - \varepsilon L(z, \theta) + \varepsilon L(z_\delta, \theta)]$$

Influence Function

1. We call the rate of change of $\Delta_{z,\varepsilon}$ *the influence function*:

$$\frac{d}{d\varepsilon} \Delta_{z,\varepsilon} = \nabla L(z; \theta^*) [\mathcal{H}R(\theta^*)]^{-1}$$

2. What about poisoning examples? Let $z_\delta = (x + \delta, y)$
3. Modify the upweighted model:

$$\theta_{z,z_\delta,\varepsilon}^* = \arg \min_{\theta} [R(\theta) - \varepsilon L(z, \theta) + \varepsilon L(z_\delta, \theta)]$$

Derivation

What is the expression for $\frac{d}{d\varepsilon} \Delta_{z,z_\delta,\varepsilon} = \frac{d}{d\varepsilon} [\theta_{z,z_\delta,\varepsilon}^* - \theta^*]$?

Influence Functions for Poisoning

1. Same strategy as before:

$$\Delta_{z, z_\delta, \varepsilon} \approx [\nabla R(\theta^*) + \varepsilon \nabla L(z_\delta; \theta^*) - \varepsilon \nabla L(z; \theta^*)] + \\ [\mathcal{H}R(\theta^*) + \varepsilon \mathcal{H}L(z; \theta^*) - \varepsilon \mathcal{H}L(z; \theta^*)]^{-1}$$

2. You can see that:

$$\frac{d}{d\varepsilon} \Delta_{z, z_\delta, \varepsilon} = \left[\frac{d}{d\varepsilon} \Delta_{z_\delta, \varepsilon} - \frac{d}{d\varepsilon} \Delta_{z, \varepsilon} \right]$$

3. Final step: Influence on some target example t :

$$\nabla \frac{d}{d\varepsilon} L(t; \theta_{z, z_\delta, \varepsilon}^*) \approx \nabla L(t; \theta^*) \cdot \frac{d}{d\varepsilon} \Delta_{z, z_\delta, \varepsilon}$$

Influence Functions for Poisoning

1. Same strategy as before:

$$\Delta_{z, z_\delta, \varepsilon} \approx [\nabla R(\theta^*) + \varepsilon \nabla L(z_\delta; \theta^*) - \varepsilon \nabla L(z; \theta^*)] + \\ [\mathcal{H}R(\theta^*) + \varepsilon \mathcal{H}L(z; \theta^*) - \varepsilon \mathcal{H}L(z; \theta^*)]^{-1}$$

2. You can see that:

$$\frac{d}{d\varepsilon} \Delta_{z, z_\delta, \varepsilon} = \left[\frac{d}{d\varepsilon} \Delta_{z_\delta, \varepsilon} - \frac{d}{d\varepsilon} \Delta_{z, \varepsilon} \right]$$

3. Final step: Influence on some target example t :

$$\nabla \frac{d}{d\varepsilon} L(t; \theta_{z, z_\delta, \varepsilon}^*) \approx \nabla L(t; \theta^*) \cdot \frac{d}{d\varepsilon} \Delta_{z, z_\delta, \varepsilon}$$

Influence Functions for Poisoning

1. Same strategy as before:

$$\Delta_{z, z_\delta, \varepsilon} \approx [\nabla R(\theta^*) + \varepsilon \nabla L(z_\delta; \theta^*) - \varepsilon \nabla L(z; \theta^*)] + \\ [\mathcal{H}R(\theta^*) + \varepsilon \mathcal{H}L(z; \theta^*) - \varepsilon \mathcal{H}L(z; \theta^*)]^{-1}$$

2. You can see that:

$$\frac{d}{d\varepsilon} \Delta_{z, z_\delta, \varepsilon} = \left[\frac{d}{d\varepsilon} \Delta_{z_\delta, \varepsilon} - \frac{d}{d\varepsilon} \Delta_{z, \varepsilon} \right]$$

3. Final step: Influence on some target example t :

$$\nabla \frac{d}{d\varepsilon} L(t; \theta_{z, z_\delta, \varepsilon}^*) \approx \nabla L(t; \theta^*) \cdot \frac{d}{d\varepsilon} \Delta_{z, z_\delta, \varepsilon}$$

Influence Functions for Poisoning

1. Same strategy as before:

$$\Delta_{z, z_\delta, \varepsilon} \approx [\nabla R(\theta^*) + \varepsilon \nabla L(z_\delta; \theta^*) - \varepsilon \nabla L(z; \theta^*)] + \\ [\mathcal{H}R(\theta^*) + \varepsilon \mathcal{H}L(z; \theta^*) - \varepsilon \mathcal{H}L(z; \theta^*)]^{-1}$$

2. You can see that:

$$\frac{d}{d\varepsilon} \Delta_{z, z_\delta, \varepsilon} = \left[\frac{d}{d\varepsilon} \Delta_{z_\delta, \varepsilon} - \frac{d}{d\varepsilon} \Delta_{z, \varepsilon} \right]$$

3. Final step: Influence on some target example t :

$$\nabla \frac{d}{d\varepsilon} L(t; \theta_{z, z_\delta, \varepsilon}^*) \approx \nabla L(t; \theta^*) \cdot \frac{d}{d\varepsilon} \Delta_{z, z_\delta, \varepsilon}$$

Influence Functions for Poisoning

1. Same strategy as before:

$$\Delta_{z, z_\delta, \varepsilon} \approx [\nabla R(\theta^*) + \varepsilon \nabla L(z_\delta; \theta^*) - \varepsilon \nabla L(z; \theta^*)] + \\ [\mathcal{H}R(\theta^*) + \varepsilon \mathcal{H}L(z; \theta^*) - \varepsilon \mathcal{H}L(z; \theta^*)]^{-1}$$

2. You can see that:

$$\frac{d}{d\varepsilon} \Delta_{z, z_\delta, \varepsilon} = \left[\frac{d}{d\varepsilon} \Delta_{z_\delta, \varepsilon} - \frac{d}{d\varepsilon} \Delta_{z, \varepsilon} \right]$$

3. Final step: Influence on some target example t :

$$\nabla \frac{d}{d\varepsilon} L(t; \theta_{z, z_\delta, \varepsilon}^*) \approx \nabla L(t; \theta^*) \cdot \frac{d}{d\varepsilon} \Delta_{z, z_\delta, \varepsilon}$$

Question

How do we use this to build a poisoning attack to force a model to make a mistake on t ?

Influence Functions for Poisoning

- Reparameterize:

$$Q(x, \delta; \theta) = L(t; \theta^*) \cdot \frac{d}{d\varepsilon} \Delta_{z, z_\delta, \varepsilon}$$

- Poisoning optimization problem:

$$\delta^* = \arg \max_{\delta} Q(x, \delta; \theta) \text{ s.t. } \| \delta \| \leq \epsilon$$

- How to solve it?
- PGD for poisoning:

$$\delta := \mathcal{P}[\delta + \alpha \nabla_{\delta} Q(x, \delta; \theta)]$$

Influence Functions for Poisoning

- Reparameterize:

$$Q(x, \delta; \theta) = L(t; \theta^*) \cdot \frac{d}{d\varepsilon} \Delta_{z, z_\delta, \varepsilon}$$

- Poisoning optimization problem:

$$\delta^* = \arg \max_{\delta} Q(x, \delta; \theta) \text{ s.t. } \| \delta \| \leq \epsilon$$

- How to solve it?
- PGD for poisoning:

$$\delta := \mathcal{P}[\delta + \alpha \nabla_{\delta} Q(x, \delta; \theta)]$$

Influence Functions for Poisoning

- Reparameterize:

$$Q(x, \delta; \theta) = L(t; \theta^*) \cdot \frac{d}{d\varepsilon} \Delta_{z, z_\delta, \varepsilon}$$

- Poisoning optimization problem:

$$\delta^* = \arg \max_{\delta} Q(x, \delta; \theta) \text{ s.t. } \| \delta \| \leq \epsilon$$

- How to solve it?

- PGD for poisoning:

$$\delta := P[\delta + \alpha \nabla_{\delta} Q(x, \delta; \theta)]$$

Influence Functions for Poisoning

- Reparameterize:

$$Q(x, \delta; \theta) = L(t; \theta^*) \cdot \frac{d}{d\varepsilon} \Delta_{z, z_\delta, \varepsilon}$$

- Poisoning optimization problem:

$$\delta^* = \arg \max_{\delta} Q(x, \delta; \theta) \text{ s.t. } \| \delta \| \leq \epsilon$$

- How to solve it?
 - PGD for poisoning:

$$\delta := P[\delta + \alpha \nabla_{\delta} Q(x, \delta; \theta)]$$

Influence Functions for Poisoning

- Reparameterize:

$$Q(x, \delta; \theta) = L(t; \theta^*) \cdot \frac{d}{d\varepsilon} \Delta_{z, z_\delta, \varepsilon}$$

- Poisoning optimization problem:

$$\delta^* = \arg \max_{\delta} Q(x, \delta; \theta) \text{ s.t. } \| \delta \| \leq \epsilon$$

- How to solve it?
- PGD for poisoning:

$$\delta := \mathcal{P}[\delta + \alpha \nabla_{\delta} Q(x, \delta; \theta)]$$

Poisoning Models

A small perturbation to one **training** example:

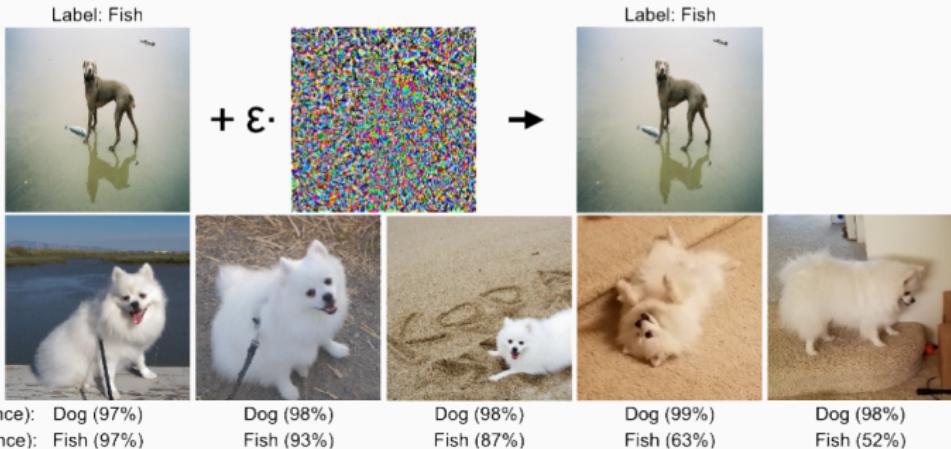


Image Credit: Koh and Liang [2017]

Many Settings of Poisoning

Possible goals:

- Increase error
- Prevent convergence
- Increase error only on certain inputs (backdooring)
- Replace the model with the model of your choosing
- Modify a property of the model (e.g., fairness)
- Increase error
- Not discussing poisoning pre-trained models, but it is doable
(Bagdasaryan et al. [2018])
- There is hope for defences, but it is not developed well yet
(Steinhardt et al. [2017])

Many Settings of Poisoning

Possible goals:

- Increase error
- Prevent convergence
- Increase error only on certain inputs (backdooring)
- Replace the model with the model of your choosing
- Modify a property of the model (e.g., fairness)
- Increase error
- Not discussing poisoning pre-trained models, but it is doable
(Bagdasaryan et al. [2018])
- There is hope for defences, but it is not developed well yet
(Steinhardt et al. [2017])

Many Settings of Poisoning

Possible goals:

- Increase error
 - Prevent convergence
 - Increase error only on certain inputs (backdooring)
 - Replace the model with the model of your choosing
 - Modify a property of the model (e.g., fairness)
 - Increase error
-
- Not discussing poisoning pre-trained models, but it is doable (Bagdasaryan et al. [2018])
 - There is hope for defences, but it is not developed well yet (Steinhardt et al. [2017])

Many Settings of Poisoning

Possible goals:

- Increase error
 - Prevent convergence
 - Increase error only on certain inputs (backdooring)
 - Replace the model with the model of your choosing
 - Modify a property of the model (e.g., fairness)
 - Increase error
-
- Not discussing poisoning pre-trained models, but it is doable (Bagdasaryan et al. [2018])
 - There is hope for defences, but it is not developed well yet (Steinhardt et al. [2017])

Many Settings of Poisoning

Possible goals:

- Increase error
 - Prevent convergence
 - Increase error only on certain inputs (backdooring)
 - Replace the model with the model of your choosing
 - **Modify a property of the model (e.g., fairness)**
 - Increase error
-
- Not discussing poisoning pre-trained models, but it is doable (Bagdasaryan et al. [2018])
 - There is hope for defences, but it is not developed well yet (Steinhardt et al. [2017])

Many Settings of Poisoning

Possible goals:

- Increase error
 - Prevent convergence
 - Increase error only on certain inputs (backdooring)
 - Replace the model with the model of your choosing
 - Modify a property of the model (e.g., fairness)
 - Increase error
-
- Not discussing poisoning pre-trained models, but it is doable (Bagdasaryan et al. [2018])
 - There is hope for defences, but it is not developed well yet (Steinhardt et al. [2017])

Many Settings of Poisoning

Possible goals:

- Increase error
 - Prevent convergence
 - Increase error only on certain inputs (backdooring)
 - Replace the model with the model of your choosing
 - Modify a property of the model (e.g., fairness)
 - Increase error
-
- Not discussing poisoning pre-trained models, but it is doable
(Bagdasaryan et al. [2018])
 - There is hope for defences, but it is not developed well yet
(Steinhardt et al. [2017])

Many Settings of Poisoning

Possible goals:

- Increase error
 - Prevent convergence
 - Increase error only on certain inputs (backdooring)
 - Replace the model with the model of your choosing
 - Modify a property of the model (e.g., fairness)
 - Increase error
-
- Not discussing poisoning pre-trained models, but it is doable
(Bagdasaryan et al. [2018])
 - There is hope for defences, but it is not developed well yet
(Steinhardt et al. [2017])

Takeaways



Takeaways

- Whenever the data is not trusted (e.g., federated learning), the trained model can be manipulated
- Adversaries might be able to modify the behaviour of models even if injecting a single example
- Universal defences are not there yet

Takeaways

- Whenever the data is not trusted (e.g., federated learning), the trained model can be manipulated
- Adversaries might be able to modify the behaviour of models even if injecting a single example
- Universal defences are not there yet

Takeaways

- Whenever the data is not trusted (e.g., federated learning), the trained model can be manipulated
- Adversaries might be able to modify the behaviour of models even if injecting a single example
- Universal defences are not there yet

Privacy and Machine Learning

Wait, what is privacy?

Adrián @UdeaEdu · Jul 3
how about some privacy?

David Lieb @dfrieb · Jul 2
Hi Twitter! It's no-meetings week at @googlephotos and I've got a couple hours free. Tell me what you want to see next from Google Photos! New features, bug fixes, performance improvements, you name it. (No promises but very open minds!)

Show this thread

1 4

David Lieb @dfrieb

Replies to @UdeaEdu

Would love to learn more about what you mean exactly.
Google Photos is extremely private.

5:20 PM · Jul 3, 2019 · Twitter Web Client

18 Likes

Group discussion
What is happening here?

Concepts of Privacy (One of Many Approaches)

- Institutional privacy
 - Your data is private because Google promises so in their Privacy Policy
 - Need to trust Google
- Social privacy
 - Your data is private because Google gives you fine-grained controls on when to share what photos and with whom
 - Need to trust Google
- Hard privacy (“surveillance privacy”)
 - Your data is private because Google *provably cannot see and access any of it*
 - Often enforced through *cryptography* and *privacy-enhancing technologies*
 - Ideally need to trust Math
- Privacy here will mean *hard privacy*, not access controls or privacy policies
- Question: What was happening with Google Photos?

Concepts of Privacy (One of Many Approaches)

- Institutional privacy
 - Your data is private because Google promises so in their Privacy Policy
 - Need to trust Google
- Social privacy
 - Your data is private because Google gives you fine-grained controls on when to share what photos and with whom
 - Need to trust Google
- Hard privacy (“surveillance privacy”)
 - Your data is private because Google *provably cannot see and access any of it*
 - Often enforced through *cryptography* and *privacy-enhancing technologies*
 - Ideally need to trust Math
- Privacy here will mean *hard privacy*, not access controls or privacy policies
- Question: What was happening with Google Photos?

Concepts of Privacy (One of Many Approaches)

- Institutional privacy
 - Your data is private because Google promises so in their Privacy Policy
 - Need to trust Google
- Social privacy
 - Your data is private because Google gives you fine-grained controls on when to share what photos and with whom
 - Need to trust Google
- Hard privacy (“surveillance privacy”)
 - Your data is private because Google *provably cannot see and access any of it*
 - Often enforced through *cryptography* and *privacy-enhancing technologies*
 - Ideally need to trust Math
- Privacy here will mean *hard privacy*, not access controls or privacy policies
- Question: What was happening with Google Photos?

Concepts of Privacy (One of Many Approaches)

- Institutional privacy
 - Your data is private because Google promises so in their Privacy Policy
 - Need to **trust Google**
- Social privacy
 - Your data is private because Google gives you fine-grained controls on when to share what photos and with whom
 - Need to **trust Google**
- Hard privacy (“surveillance privacy”)
 - Your data is private because Google *provably cannot see and access any of it*
 - Often enforced through *cryptography* and *privacy-enhancing technologies*
 - Ideally need to **trust Math**
- Privacy here will mean *hard privacy*, not access controls or privacy policies
- **Question:** What was happening with Google Photos?

Concepts of Privacy (One of Many Approaches)

- Institutional privacy
 - Your data is private because Google promises so in their Privacy Policy
 - Need to **trust Google**
- Social privacy
 - Your data is private because Google gives you fine-grained controls on when to share what photos and with whom
 - Need to **trust Google**
- Hard privacy (“surveillance privacy”)
 - Your data is private because Google *provably cannot see and access any of it*
 - Often enforced through *cryptography* and *privacy-enhancing technologies*
 - Ideally need to **trust Math**
- Privacy here will mean *hard privacy*, not access controls or privacy policies
- **Question:** What was happening with Google Photos?

Concepts of Privacy (One of Many Approaches)

- Institutional privacy
 - Your data is private because Google promises so in their Privacy Policy
 - Need to **trust Google**
- Social privacy
 - Your data is private because Google gives you fine-grained controls on when to share what photos and with whom
 - **Need to trust Google**
- Hard privacy (“surveillance privacy”)
 - Your data is private because Google *provably cannot see and access any of it*
 - Often enforced through *cryptography* and *privacy-enhancing technologies*
 - Ideally need to **trust Math**
- Privacy here will mean *hard privacy*, not access controls or privacy policies
- **Question:** What was happening with Google Photos?

Concepts of Privacy (One of Many Approaches)

- Institutional privacy
 - Your data is private because Google promises so in their Privacy Policy
 - Need to **trust Google**
- Social privacy
 - Your data is private because Google gives you fine-grained controls on when to share what photos and with whom
 - Need to **trust Google**
- Hard privacy (“surveillance privacy”)
 - Your data is private because Google *provably cannot see and access any of it*
 - Often enforced through *cryptography* and privacy-enhancing technologies
 - Ideally need to **trust Math**
- Privacy here will mean *hard privacy*, not access controls or privacy policies
- Question: What was happening with Google Photos?

Concepts of Privacy (One of Many Approaches)

- Institutional privacy
 - Your data is private because Google promises so in their Privacy Policy
 - Need to **trust Google**
- Social privacy
 - Your data is private because Google gives you fine-grained controls on when to share what photos and with whom
 - Need to **trust Google**
- Hard privacy ("surveillance privacy")
 - Your data is private because Google *provably cannot see and access any of it*
 - Often enforced through *cryptography* and privacy-enhancing technologies
 - Ideally need to **trust Math**
- Privacy here will mean *hard privacy*, not access controls or privacy policies
- Question: What was happening with Google Photos?

Concepts of Privacy (One of Many Approaches)

- Institutional privacy
 - Your data is private because Google promises so in their Privacy Policy
 - Need to **trust Google**
- Social privacy
 - Your data is private because Google gives you fine-grained controls on when to share what photos and with whom
 - Need to **trust Google**
- Hard privacy ("surveillance privacy")
 - Your data is private because Google *provably cannot see and access any of it*
 - Often enforced through *cryptography* and privacy-enhancing technologies
 - Ideally need to **trust Math**
- Privacy here will mean *hard privacy*, not access controls or privacy policies
- Question: What was happening with Google Photos?

Concepts of Privacy (One of Many Approaches)

- Institutional privacy
 - Your data is private because Google promises so in their Privacy Policy
 - Need to **trust Google**
- Social privacy
 - Your data is private because Google gives you fine-grained controls on when to share what photos and with whom
 - Need to **trust Google**
- Hard privacy ("surveillance privacy")
 - Your data is private because Google *provably cannot see and access any of it*
 - Often enforced through *cryptography* and privacy-enhancing technologies
 - **Ideally need to trust Math**
- Privacy here will mean *hard privacy*, not access controls or privacy policies
- **Question:** What was happening with Google Photos?

Concepts of Privacy (One of Many Approaches)

- Institutional privacy
 - Your data is private because Google promises so in their Privacy Policy
 - Need to **trust Google**
- Social privacy
 - Your data is private because Google gives you fine-grained controls on when to share what photos and with whom
 - Need to **trust Google**
- Hard privacy ("surveillance privacy")
 - Your data is private because Google *provably cannot see and access any of it*
 - Often enforced through *cryptography* and privacy-enhancing technologies
 - Ideally need to **trust Math**
- Privacy here will mean *hard privacy*, not access controls or privacy policies
- Question: What was happening with Google Photos?

Concepts of Privacy (One of Many Approaches)

- Institutional privacy
 - Your data is private because Google promises so in their Privacy Policy
 - Need to **trust Google**
- Social privacy
 - Your data is private because Google gives you fine-grained controls on when to share what photos and with whom
 - Need to **trust Google**
- Hard privacy ("surveillance privacy")
 - Your data is private because Google *provably cannot see and access any of it*
 - Often enforced through *cryptography* and privacy-enhancing technologies
 - Ideally need to **trust Math**
- Privacy here will mean *hard privacy*, not access controls or privacy policies
- Question: What was happening with Google Photos?

Leaky Models

Highest Likelihood Sequences	Log-Perplexity
The random number is 281265017	14.63
The random number is 281265117	18.56
The random number is 281265011	19.01
The random number is 286265117	20.65
The random number is 528126501	20.88
The random number is 281266511	20.99
The random number is 287265017	20.99
The random number is 281265111	21.16
The random number is 281265010	21.36

Table 1: Possible sequences sorted by Log-Perplexity. The inserted canary— 281265017—has the lowest log-perplexity. The remaining most-likely phrases are all slightly-modified variants, a small edit distance away from the canary phrase.

- High-capacity ML models memorize data (Carlini et al. [2018])
- Models behave differently on examples that were in training and that were not

Leaky Models

Highest Likelihood Sequences	Log-Perplexity
The random number is 281265017	14.63
The random number is 281265117	18.56
The random number is 281265011	19.01
The random number is 286265117	20.65
The random number is 528126501	20.88
The random number is 281266511	20.99
The random number is 287265017	20.99
The random number is 281265111	21.16
The random number is 281265010	21.36

Table 1: Possible sequences sorted by Log-Perplexity. The inserted canary— 281265017—has the lowest log-perplexity. The remaining most-likely phrases are all slightly-modified variants, a small edit distance away from the canary phrase.

- High-capacity ML models memorize data (Carlini et al. [2018])
- Models behave differently on examples that were in training and that were not

The Goals of the Privacy Adversary

Question

What are the possible goals of the privacy adversary?

- **Membership Inference:** Learn if an example x was in the training dataset X or not
- **Attribute Inference:** Knowing a part of an example \tilde{x} , learn the other part
- **Model Inversion:** Given a model, extract training data points

The Goals of the Privacy Adversary

Question

What are the possible goals of the privacy adversary?

- **Membership Inference:** Learn if an example x was in the training dataset X or not
- **Attribute Inference:** Knowing a part of an example \tilde{x} , learn the other part
- **Model Inversion:** Given a model, extract training data points

The Goals of the Privacy Adversary

Question

What are the possible goals of the privacy adversary?

- **Membership Inference:** Learn if an example x was in the training dataset X or not
- **Attribute Inference:** Knowing a part of an example \tilde{x} , learn the other part
- **Model Inversion:** Given a model, extract training data points

The Goals of the Privacy Adversary

Question

What are the possible goals of the privacy adversary?

- **Membership Inference:** Learn if an example x was in the training dataset X or not
- **Attribute Inference:** Knowing a part of an example \tilde{x} , learn the other part
- **Model Inversion:** Given a model, extract training data points

The Goals of the Privacy Adversary

Question

What are the possible goals of the privacy adversary?

- **Membership Inference:** Learn if an example x was in the training dataset X or not
- **Attribute Inference:** Knowing a part of an example \tilde{x} , learn the other part
- **Model Inversion:** Given a model, extract training data points

Question

Assuming one attribute is unknown, and it takes two different values (e.g., HIV status), how to mount an attribute inference attack using a membership inference attack?

Membership Inference

Question

What should be the capabilities of the adversary to perform membership inference?

Membership Inference

Question

What should be the capabilities of the adversary to perform membership inference?

Surprisingly, agnostic black-box might be enough

Ad-Hoc: Confidence Distributions

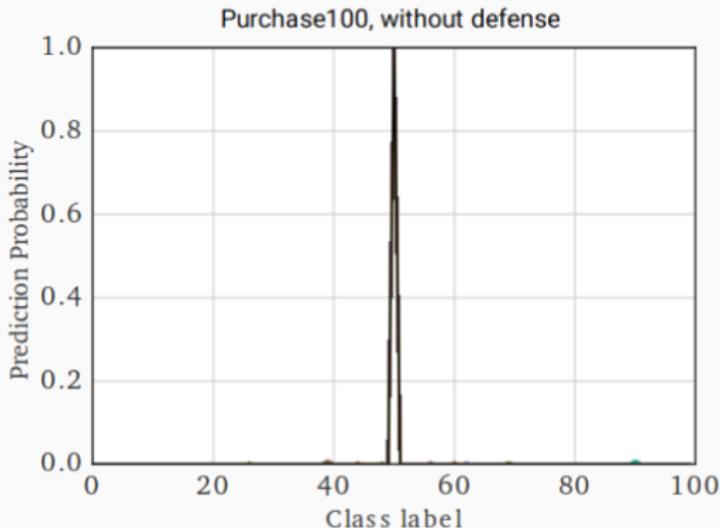
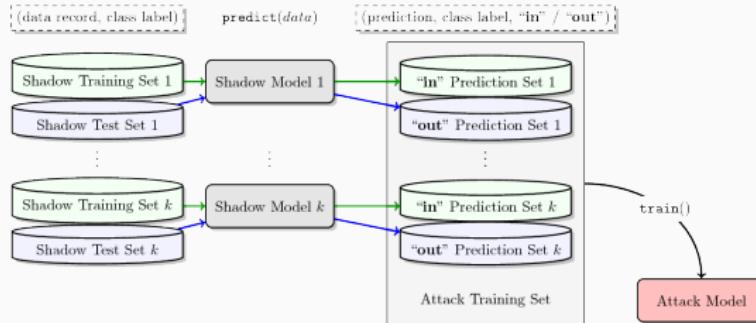


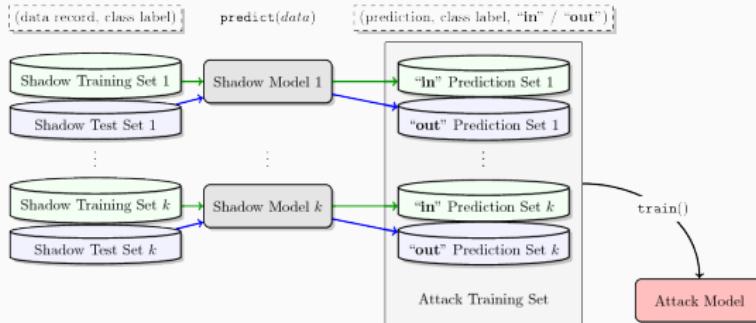
Image Credit: Nasr et al. [2018]

Shadow Model Attack



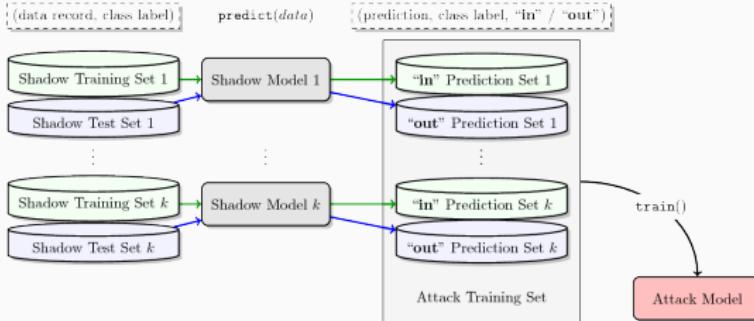
- Pioneering membership inference attack (Shokri et al. [2016])
- Adversary's knowledge: dataset X_A coming from the same distribution as X , model architecture f , query access to f_θ
- Adversary's goal: learn if x was in training X
- Train a bunch of *shadow models* on subsets of X_A
- Train an attacker learn how this architecture behaves on examples that were in training, and out.

Shadow Model Attack



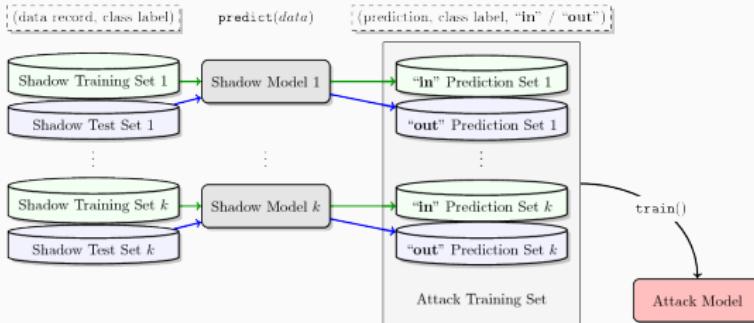
- Pioneering membership inference attack (Shokri et al. [2016])
- Adversary's knowledge: dataset X_A coming from the same distribution as X , model architecture f , query access to f_θ
- Adversary's goal: learn if x was in training X
- Train a bunch of *shadow models* on subsets of X_A
- Train an attacker learn how this architecture behaves on examples that were in training, and out.

Shadow Model Attack



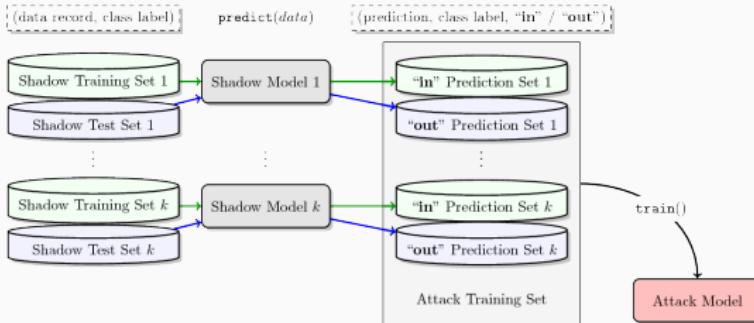
- Pioneering membership inference attack (Shokri et al. [2016])
- Adversary's knowledge: dataset X_A coming from the same distribution as X , model architecture f , query access to f_θ
- **Adversary's goal: learn if x was in training X**
- Train a bunch of *shadow models* on subsets of X_A
- Train an attacker learn how this architecture behaves on examples that were in training, and out.

Shadow Model Attack



- Pioneering membership inference attack (Shokri et al. [2016])
- Adversary's knowledge: dataset X_A coming from the same distribution as X , model architecture f , query access to f_θ
- Adversary's goal: learn if x was in training X
- Train a bunch of *shadow models* on subsets of X_A
- Train an attacker learn how this architecture behaves on examples that were in training, and out.

Shadow Model Attack



- Pioneering membership inference attack (Shokri et al. [2016])
- Adversary's knowledge: dataset X_A coming from the same distribution as X , model architecture f , query access to f_θ
- Adversary's goal: learn if x was in training X
- Train a bunch of *shadow models* on subsets of X_A
- **Train an attacker learn how this architecture behaves on examples that were in training, and out.**

Defending Against Membership Inference

Question

What is the best defence against Membership Inference?

1. Simulate this attack during training, and penalize the attacker's successes
2. Ensure that model predictions do not differ on examples that were in training, and out of training
3. Do not output confidence scores

Defending Against Membership Inference

Question

What is the best defence against Membership Inference?

1. Simulate this attack during training, and penalize the attacker's successes
2. Ensure that model predictions do not differ on examples that were in training, and out of training
3. Do not output confidence scores

Defending Against Membership Inference

Question

What is the best defence against Membership Inference?

1. Simulate this attack during training, and penalize the attacker's successes
2. Ensure that model predictions do not differ on examples that were in training, and out of training
3. Do not output confidence scores

Defending Against Membership Inference (Answers)

- Do not output confidence scores
 - Security by Obscurity
 - Also, the shadow model attack works well without confidence scores.
- Simulate this attack during training, and penalize the attacker's successes
 - This one is OK, but, unlike PGD, there's no clear view on how close an attacker is to being optimal
 - Hence, this violates the Worst-Case Security Principle
- Ensure that model predictions do not differ on examples that were in training, and out of training
 - This is the ideal case
 - Hard to achieve, and in practice requires sacrifices in model's performance
 - There are good methods for some models

Defending Against Membership Inference (Answers)

- Do not output confidence scores
 - Security by Obscurity
 - Also, the shadow model attack works well without confidence scores.
- Simulate this attack during training, and penalize the attacker's successes
 - This one is OK, but, unlike PGD, there's no clear view on how close an attacker is to being optimal
 - Hence, this violates the Worst-Case Security Principle
- Ensure that model predictions do not differ on examples that were in training, and out of training
 - This is the ideal case
 - Hard to achieve, and in practice requires sacrifices in model's performance
 - There are good methods for some models

Defending Against Membership Inference (Answers)

- Do not output confidence scores
 - Security by Obscurity
 - Also, the shadow model attack works well without confidence scores.
- Simulate this attack during training, and penalize the attacker's successes
 - This one is OK, but, unlike PGD, there's no clear view on how close an attacker is to being optimal
 - Hence, this violates the Worst-Case Security Principle
- Ensure that model predictions do not differ on examples that were in training, and out of training
 - This is the ideal case
 - Hard to achieve, and in practice requires sacrifices in model's performance
 - There are good methods for some models

Defending Against Membership Inference (Answers)

- Do not output confidence scores
 - Security by Obscurity
 - Also, the shadow model attack works well without confidence scores.
- Simulate this attack during training, and penalize the attacker's successes
 - This one is OK, but, unlike PGD, there's no clear view on how close an attacker is to being optimal
 - Hence, this violates the Worst-Case Security Principle
- Ensure that model predictions do not differ on examples that were in training, and out of training
 - This is the ideal case
 - Hard to achieve, and in practice requires sacrifices in model's performance
 - There are good methods for some models

Defending Against Membership Inference (Answers)

- Do not output confidence scores
 - Security by Obscurity
 - Also, the shadow model attack works well without confidence scores.
- Simulate this attack during training, and penalize the attacker's successes
 - This one is OK, but, unlike PGD, there's no clear view on how close an attacker is to being optimal
 - Hence, this violates the Worst-Case Security Principle
- Ensure that model predictions do not differ on examples that were in training, and out of training
 - This is the ideal case
 - Hard to achieve, and in practice requires sacrifices in model's performance
 - There are good methods for some models

Defending Against Membership Inference (Answers)

- Do not output confidence scores
 - Security by Obscurity
 - Also, the shadow model attack works well without confidence scores.
- Simulate this attack during training, and penalize the attacker's successes
 - This one is OK, but, unlike PGD, there's no clear view on how close an attacker is to being optimal
 - Hence, this violates the Worst-Case Security Principle
- Ensure that model predictions do not differ on examples that were in training, and out of training
 - This is the ideal case
 - Hard to achieve, and in practice requires sacrifices in model's performance
 - There are good methods for some models

Defending Against Membership Inference (Answers)

- Do not output confidence scores
 - Security by Obscurity
 - Also, the shadow model attack works well without confidence scores.
- Simulate this attack during training, and penalize the attacker's successes
 - This one is OK, but, unlike PGD, there's no clear view on how close an attacker is to being optimal
 - Hence, this violates the Worst-Case Security Principle
- Ensure that model predictions do not differ on examples that were in training, and out of training
 - This is the ideal case
 - Hard to achieve, and in practice requires sacrifices in model's performance
 - There are good methods for some models

Defending Against Membership Inference (Answers)

- Do not output confidence scores
 - Security by Obscurity
 - Also, the shadow model attack works well without confidence scores.
- Simulate this attack during training, and penalize the attacker's successes
 - This one is OK, but, unlike PGD, there's no clear view on how close an attacker is to being optimal
 - Hence, this violates the Worst-Case Security Principle
- Ensure that model predictions do not differ on examples that were in training, and out of training
 - **This is the ideal case**
 - Hard to achieve, and in practice requires sacrifices in model's performance
 - There are good methods for some models

Defending Against Membership Inference (Answers)

- Do not output confidence scores
 - Security by Obscurity
 - Also, the shadow model attack works well without confidence scores.
- Simulate this attack during training, and penalize the attacker's successes
 - This one is OK, but, unlike PGD, there's no clear view on how close an attacker is to being optimal
 - Hence, this violates the Worst-Case Security Principle
- Ensure that model predictions do not differ on examples that were in training, and out of training
 - This is the ideal case
 - Hard to achieve, and in practice requires sacrifices in model's performance
 - There are good methods for some models

Defending Against Membership Inference (Answers)

- Do not output confidence scores
 - Security by Obscurity
 - Also, the shadow model attack works well without confidence scores.
- Simulate this attack during training, and penalize the attacker's successes
 - This one is OK, but, unlike PGD, there's no clear view on how close an attacker is to being optimal
 - Hence, this violates the Worst-Case Security Principle
- Ensure that model predictions do not differ on examples that were in training, and out of training
 - This is the ideal case
 - Hard to achieve, and in practice requires sacrifices in model's performance
 - There are good methods for some models

Adversarial Training

- Similar to GAN
- Minimax optimization problem:

$$\min_{\theta} \left[R(\theta) + \lambda \max_{\theta'} \underbrace{A(\theta, \theta')}_{\text{Attacker's expected success}} \right]$$

θ' is attacker model's parameters.

Adversarial Training

- Similar to GAN
- Minimax optimization problem:

$$\min_{\theta} \left[R(\theta) + \lambda \max_{\theta'} \underbrace{A(\theta, \theta')}_{\text{Attacker's expected success}} \right]$$

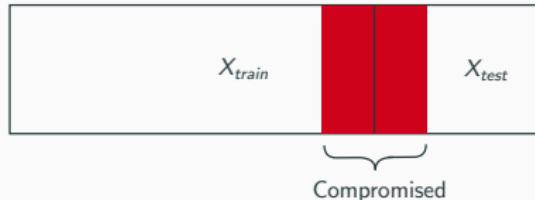
θ' is attacker model's parameters.

A Stronger Attacker

Question

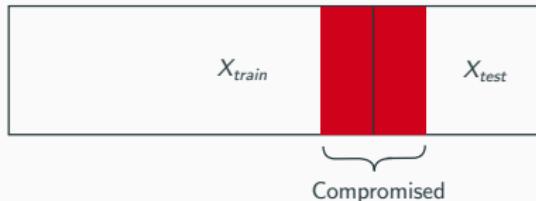
Can we imagine a stronger attack than shadow models? Stronger \implies more knowledge, more capabilities

A Stronger Attacker



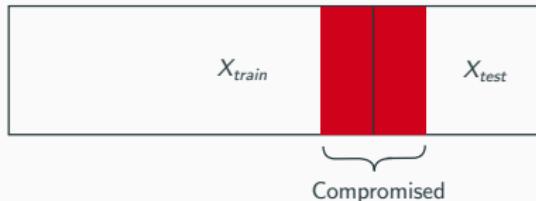
1. Take a stronger attacker than shadow models: knows parts of training data
2. Observe the current target model's predictions on the compromised dataset
3. Train the attacker to distinguish between in and out based on predictions

A Stronger Attacker



1. Take a stronger attacker than shadow models: knows parts of training data
2. Observe the current target model's predictions on the compromised dataset
3. Train the attacker to distinguish between in and out based on predictions

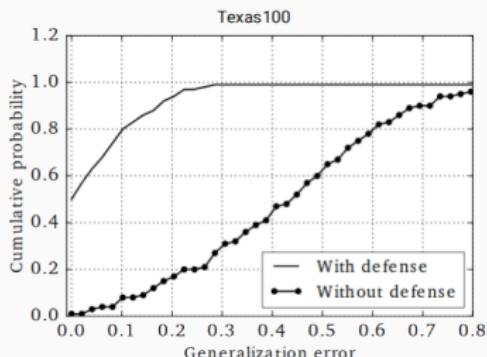
A Stronger Attacker



1. Take a stronger attacker than shadow models: knows parts of training data
2. Observe the current target model's predictions on the compromised dataset
3. Train the attacker to distinguish between in and out based on predictions

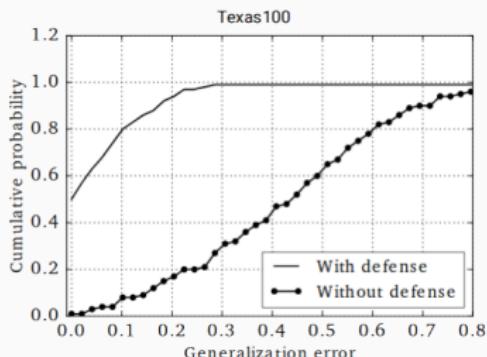
Adversarial Training Properties

- No optimality guarantees
- Unstable training process (as with GANs)
- Ideally, privacy and generalization performance are friends. In practice, can act as a good regularizer (Nasr et al. [2018])



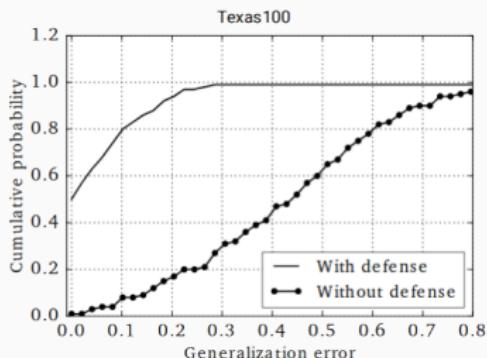
Adversarial Training Properties

- No optimality guarantees
- Unstable training process (as with GANs)
- Ideally, privacy and generalization performance are friends. In practice, can act as a good regularizer (Nasr et al. [2018])



Adversarial Training Properties

- No optimality guarantees
- Unstable training process (as with GANs)
- Ideally, privacy and generalization performance are friends. In practice, can act as a good regularizer (Nasr et al. [2018])



Provable Method: Training with Differential Privacy

- *Differential Privacy* is a property of a probabilistic training procedure $T(X)$. For any X, X' differing by a single example, corresponding classifiers $\theta = T(X), \theta' = T(X')$ are ϵ -similar.
- Formally, for any two datasets X, X' differing by one example:

$$D(T(X) \parallel T(X')) \leq \epsilon,$$

where D is distance between distributions. Usually ∞ -Rényi.

- For example, clipping and randomizing gradient updates in gradient descent results in some differential privacy:

$$\theta := \text{Clip} [\theta + \alpha \nabla L(x, y; \theta) + \text{noise}]$$

- The lower ϵ the higher privacy.
- There are efficient differential private training procedures that work well with $\epsilon = 0.1$ (Iyengar et al. [2019]). For comparison, Apple uses $\epsilon = 15$).

Provable Method: Training with Differential Privacy

- *Differential Privacy* is a property of a probabilistic training procedure $T(X)$. For any X, X' differing by a single example, corresponding classifiers $\theta = T(X), \theta' = T(X')$ are ε -similar.
- Formally, for any two datasets X, X' differing by one example:

$$D(T(X) \parallel T(X')) \leq \varepsilon,$$

where D is distance between distributions. Usually ∞ -Rényi.

- For example, clipping and randomizing gradient updates in gradient descent results in some differential privacy:

$$\theta := \text{Clip} [\theta + \alpha \nabla L(x, y; \theta) + \text{noise}]$$

- The lower ε the higher privacy.
- There are efficient differential private training procedures that work well with $\varepsilon = 0.1$ (Iyengar et al. [2019]). For comparison, Apple uses $\varepsilon = 15$).

Provable Method: Training with Differential Privacy

- *Differential Privacy* is a property of a probabilistic training procedure $T(X)$. For any X, X' differing by a single example, corresponding classifiers $\theta = T(X), \theta' = T(X')$ are ε -similar.
- Formally, for any two datasets X, X' differing by one example:

$$D(T(X) \parallel T(X')) \leq \varepsilon,$$

where D is distance between distributions. Usually ∞ -Rényi.

- For example, clipping and randomizing gradient updates in gradient descent results in some differential privacy:

$$\theta := \text{Clip} [\theta + \alpha \nabla L(x, y; \theta) + \text{noise}]$$

- The lower ε the higher privacy.
- There are efficient differential private training procedures that work well with $\varepsilon = 0.1$ (Iyengar et al. [2019]). For comparison, Apple uses $\varepsilon = 15$).

Provable Method: Training with Differential Privacy

- *Differential Privacy* is a property of a probabilistic training procedure $T(X)$. For any X, X' differing by a single example, corresponding classifiers $\theta = T(X), \theta' = T(X')$ are ε -similar.
- Formally, for any two datasets X, X' differing by one example:

$$D(T(X) \parallel T(X')) \leq \varepsilon,$$

where D is distance between distributions. Usually ∞ -Rényi.

- For example, clipping and randomizing gradient updates in gradient descent results in some differential privacy:

$$\theta := \text{Clip} [\theta + \alpha \nabla L(x, y; \theta) + \text{noise}]$$

- The lower ε the higher privacy.
- There are efficient differential private training procedures that work well with $\varepsilon = 0.1$ (Iyengar et al. [2019]). For comparison, Apple uses $\varepsilon = 15$).

Provable Method: Training with Differential Privacy

- *Differential Privacy* is a property of a probabilistic training procedure $T(X)$. For any X, X' differing by a single example, corresponding classifiers $\theta = T(X), \theta' = T(X')$ are ε -similar.
- Formally, for any two datasets X, X' differing by one example:

$$D(T(X) \parallel T(X')) \leq \varepsilon,$$

where D is distance between distributions. Usually ∞ -Rényi.

- For example, clipping and randomizing gradient updates in gradient descent results in some differential privacy:

$$\theta := \text{Clip} [\theta + \alpha \nabla L(x, y; \theta) + \text{noise}]$$

- The lower ε the higher privacy.
- There are efficient differential private training procedures that work well with $\varepsilon = 0.1$ (Iyengar et al. [2019]). For comparison, Apple uses $\varepsilon = 15$.

Takeaways

- ML models leak the data left and right: they overfit and memorize
- Adversaries can often infer if the data points were present in the training data, and learn unknown parts of these data points
- As with adversarial robustness, solving this issue could benefit both generalization performance and privacy
- Unlike other problems in security, there exist cryptographically strong defences (differential privacy), that you should use wherever possible.

Takeaways

- ML models leak the data left and right: they overfit and memorize
- Adversaries can often infer if the data points were present in the training data, and learn unknown parts of these data points
- As with adversarial robustness, solving this issue could benefit both generalization performance and privacy
- Unlike other problems in security, there exist cryptographically strong defences (differential privacy), that you should use wherever possible.

Takeaways

- ML models leak the data left and right: they overfit and memorize
- Adversaries can often infer if the data points were present in the training data, and learn unknown parts of these data points
- As with adversarial robustness, solving this issue could benefit both generalization performance and privacy
- Unlike other problems in security, there exist cryptographically strong defences (differential privacy), that you should use wherever possible.

Takeaways

- ML models leak the data left and right: they overfit and memorize
- Adversaries can often infer if the data points were present in the training data, and learn unknown parts of these data points
- As with adversarial robustness, solving this issue could benefit both generalization performance and privacy
- Unlike other problems in security, there exist cryptographically strong defences (differential privacy), that you should use wherever possible.

Privacy and Security against Machine Learning

Pitfalls of High Costs of Errors

- Threats from adversaries
- ML often finds correlation, not causation
- Societal biases learned from data
- Produce errors due to distributional shift
- Base-rate fallacy. Need extremely high recall in highly class-unbalanced problems
- Reward hacking: get good score without fulfilling the purpose
- Distribute errors unfairly
- In general: result in antisocial and negative *environmental outcomes*

Pitfalls of High Costs of Errors

- Threats from adversaries
- ML often finds correlation, not causation
- Societal biases learned from data
- Produce errors due to distributional shift
- Base-rate fallacy. Need extremely high recall in highly class-unbalanced problems
- Reward hacking: get good score without fulfilling the purpose
- Distribute errors unfairly
- In general: result in antisocial and negative *environmental outcomes*

Pitfalls of High Costs of Errors

- Threats from adversaries
- ML often finds correlation, not causation
- **Societal biases learned from data**
- Produce errors due to distributional shift
- Base-rate fallacy. Need extremely high recall in highly class-unbalanced problems
- Reward hacking: get good score without fulfilling the purpose
- Distribute errors unfairly
- In general: result in antisocial and negative *environmental outcomes*

Pitfalls of High Costs of Errors

- Threats from adversaries
- ML often finds correlation, not causation
- Societal biases learned from data
- Produce errors due to distributional shift
- Base-rate fallacy. Need extremely high recall in highly class-unbalanced problems
- Reward hacking: get good score without fulfilling the purpose
- Distribute errors unfairly
- In general: result in antisocial and negative *environmental outcomes*

Pitfalls of High Costs of Errors

- Threats from adversaries
- ML often finds correlation, not causation
- Societal biases learned from data
- Produce errors due to distributional shift
- **Base-rate fallacy. Need extremely high recall in highly class-unbalanced problems**
- Reward hacking: get good score without fulfilling the purpose
- Distribute errors unfairly
- In general: result in antisocial and negative *environmental outcomes*

Pitfalls of High Costs of Errors

- Threats from adversaries
- ML often finds correlation, not causation
- Societal biases learned from data
- Produce errors due to distributional shift
- Base-rate fallacy. Need extremely high recall in highly class-unbalanced problems
- **Reward hacking: get good score without fulfilling the purpose**
- Distribute errors unfairly
- In general: result in antisocial and negative *environmental* outcomes

Pitfalls of High Costs of Errors

- Threats from adversaries
- ML often finds correlation, not causation
- Societal biases learned from data
- Produce errors due to distributional shift
- Base-rate fallacy. Need extremely high recall in highly class-unbalanced problems
- Reward hacking: get good score without fulfilling the purpose
- **Distribute errors unfairly**
- In general: result in antisocial and negative *environmental* outcomes

Pitfalls of High Costs of Errors

- Threats from adversaries
- ML often finds correlation, not causation
- Societal biases learned from data
- Produce errors due to distributional shift
- Base-rate fallacy. Need extremely high recall in highly class-unbalanced problems
- Reward hacking: get good score without fulfilling the purpose
- Distribute errors unfairly
- In general: result in antisocial and negative *environmental outcomes*

Example: Accidental Disparate Impact

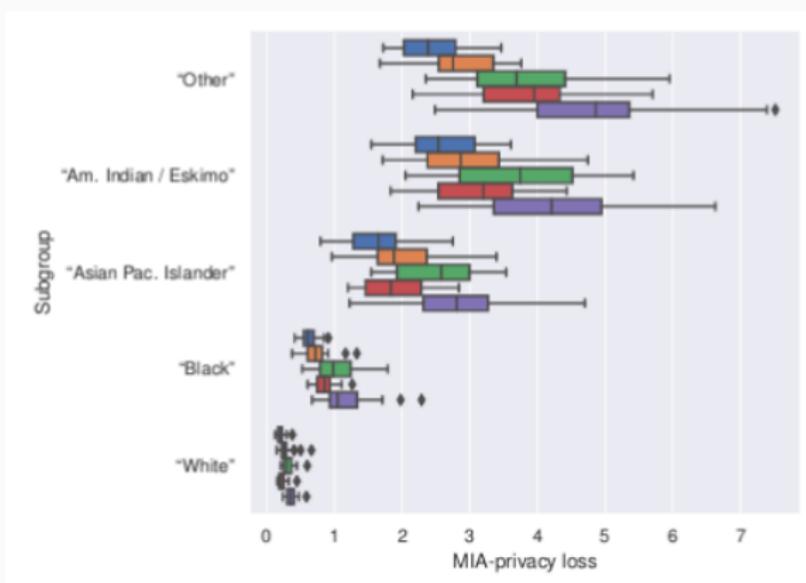


Image Credit: Yaghini et al. 2019

Illegitimate uses: Manipulation



Illegitimate uses: Private Attribute Inference

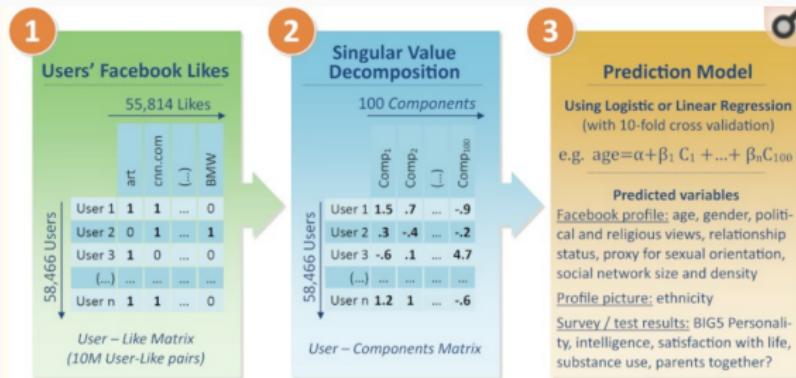


Image Credit: Kosinski et al. 2019

Open Problems

- How do we protect people against these?
- We can use adversarial ML attacks *as defences* (Overdorf et al. [2018])
- These threat models are not considered at all
- Context and impact of models matters

Open Problems

- How do we protect people against these?
- We can use adversarial ML attacks *as defences* (Overdorf et al. [2018])
- These threat models are not considered at all
- Context and impact of models matters

Open Problems

- How do we protect people against these?
- We can use adversarial ML attacks *as defences* (Overdorf et al. [2018])
- These threat models are not considered at all
- Context and impact of models matters

Open Problems

- How do we protect people against these?
- We can use adversarial ML attacks *as defences* (Overdorf et al. [2018])
- These threat models are not considered at all
- Context and impact of models matters

References

- Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter.
Accessorize to a crime: Real and stealthy attacks on state-of-the-art
face recognition. In *ACM SIGSAC*, 2016.
- Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin
Gilmer. Adversarial patch. *CoRR*, abs/1712.09665, 2017.
- Dawn Song, Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li,
Amir Rahmati, Florian Tramèr, Atul Prakash, and Tadayoshi Kohno.
Physical adversarial examples for object detectors. In *12th USENIX
Workshop on Offensive Technologies, WOOT*, 2018.

References ii

- Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg.
Badnets: Evaluating backdooring attacks on deep neural networks.
IEEE Access, pages 47230–47244, 2019.
- Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures.
In *ACM SIGSAC*, 2015.
- Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples.
CoRR, abs/1412.6572, 2014.
- Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning.
Pattern Recognition, 2018.
- Nilesh N. Dalvi, Pedro M. Domingos, Mausam, Sumit K. Sanghai, and Deepak Verma. Adversarial classification.
In *ACM SIGKDD*, 2004.
- Daniel Lowd and Christopher Meek. Adversarial learning.
In *SIGKDD*, 2005.

References iii

- Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.
- Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *CoRR*, abs/1706.06083, 2017.
- Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. In *ICML*, 2019.
- Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *CoRR*, abs/1905.02175, 2019.
- Jörn-Henrik Jacobsen, Jens Behrmann, Nicholas Carlini, Florian Tramèr, and Nicolas Papernot. Exploiting excessive invariance caused by norm-bounded adversarial robustness. *CoRR*, abs/1903.10484, 2019.

- Logan Engstrom, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. A rotation and a translation suffice: Fooling cnns with simple transformations. *CoRR*, abs/1712.02779, 2017.
- Bogdan Kulynych, Jamie Hayes, Nikita Samarin, and Carmela Troncoso. Evading classifiers in discrete domains with provable optimality guarantees. *CoRR*, abs/1810.10939, 2018.
- Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. ZOO: zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *AISec@CCS*, 2017.
- Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *USENIX Security Symposium*, 2016.
- Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *ICML*, 2017.

- Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. *CoRR*, 2018.
- Jacob Steinhardt, Pang Wei Koh, and Percy S. Liang. Certified defenses for data poisoning attacks. In *NIPS*, 2017.
- Nicholas Carlini, Chang Liu, Jernej Kos, Úlfar Erlingsson, and Dawn Song. The secret sharer: Measuring unintended neural network memorization & extracting secrets. *CoRR*, abs/1802.08232, 2018.
- Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine learning with membership privacy using adversarial regularization. In *ACM SIGSAC*, 2018.
- Reza Shokri, Marco Stronati, and Vitaly Shmatikov. Membership inference attacks against machine learning models. *CoRR*, abs/1610.05820, 2016. URL <http://arxiv.org/abs/1610.05820>.

- Roger Iyengar, Joseph P Near, Dawn Song, Om Thakkar, Abhradeep Thakurta, and Lun Wang. Towards practical differentially private convex optimization. In *Towards Practical Differentially Private Convex Optimization*, page 0. IEEE, 2019.
- Rebekah Overdorf, Bogdan Kulynych, Ero Balsa, Carmela Troncoso, and Seda Gürses. POTs: Protective optimization technologies. *CoRR*, abs/1806.02711, 2018.