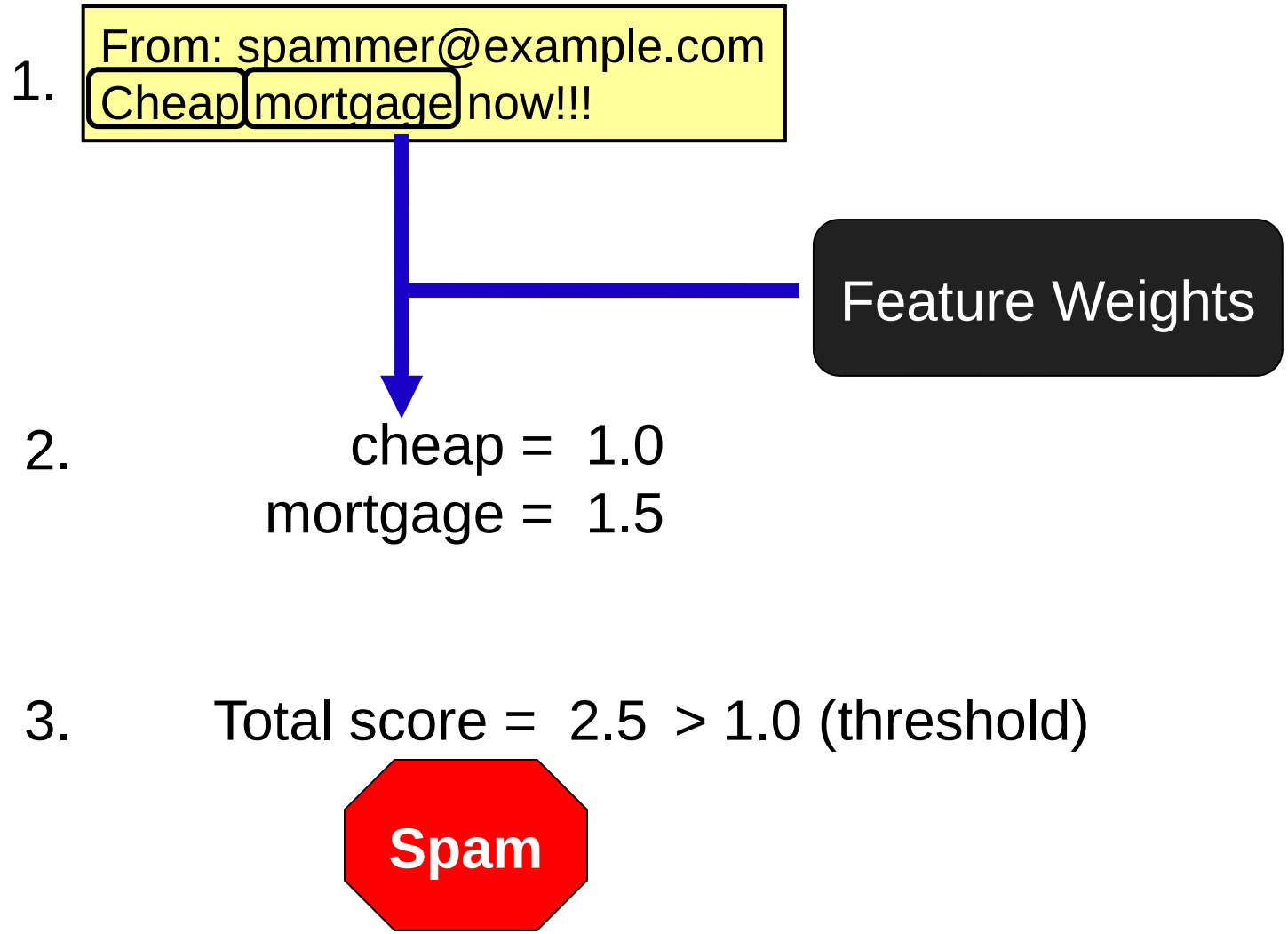
# Adversarial Machine Learning

Quan Zhang, Ph.D.

# A motivating example

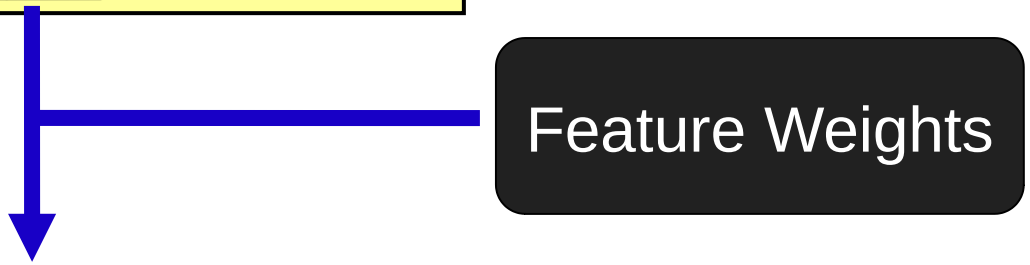
- Many adversarial problems
  - Spam filtering
  - Malware detection
  - Evolutionary pressure
  - New virus every year
- Want general-purpose solutions
- We can gain much insight by modeling adversarial situations mathematically

# Example: Spam Filtering



# Example: Spammers Adapt

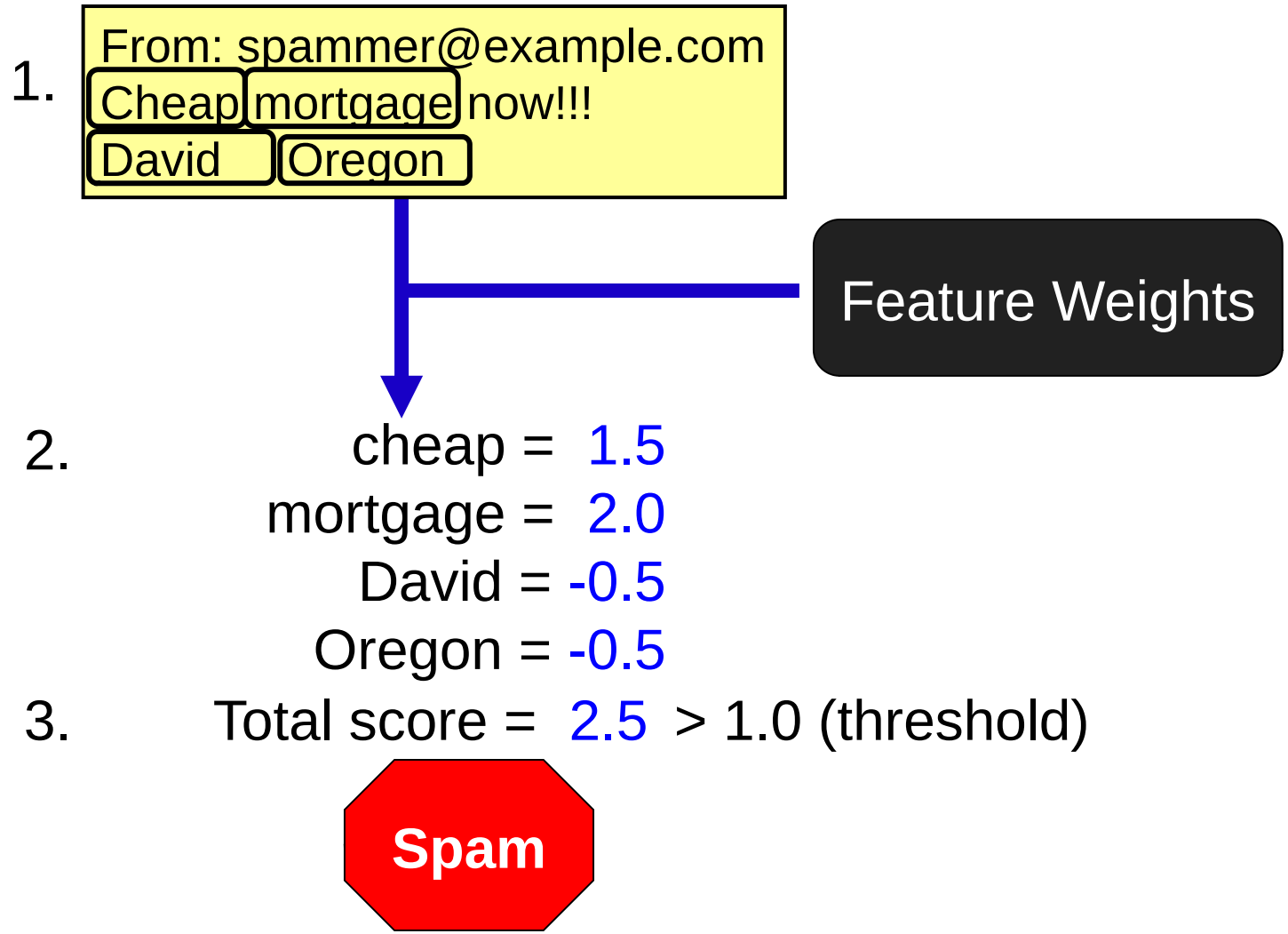
1. From: spammer@example.com
   Cheap mortgage now!!!
   David Oregon



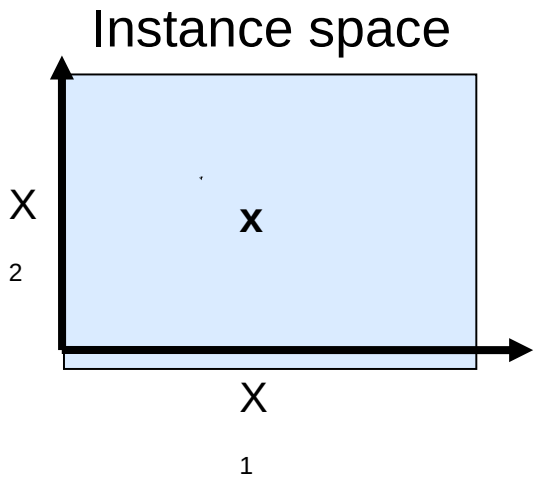

2. cheap = 1.0
   mortgage = 1.5
   David = -1.0
   Oregon = -1.0
3. Total score = 0.5 < 1.0 (threshold)

**OK**

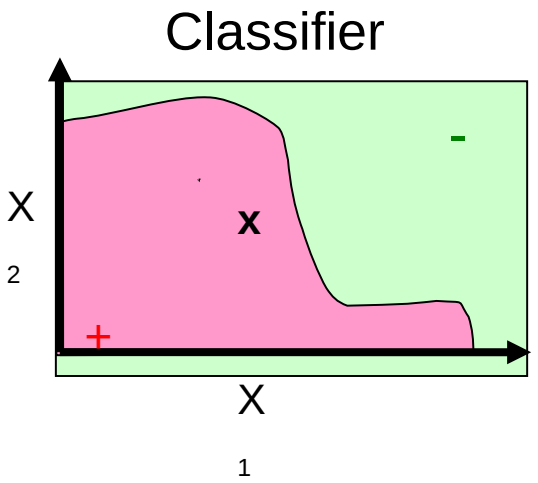
# Example: Classifier Adapts



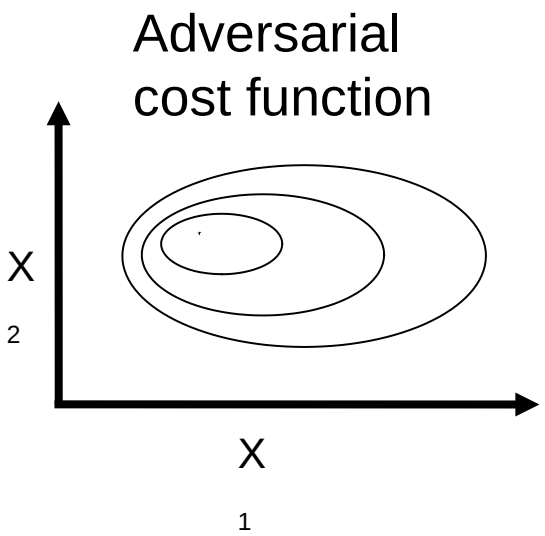
# Definitions




$X = \{X_1, X_2, \ldots, X_n\}$
Each $X_i$ is a feature
(e.g., a word)
Instances, **x** ∈ X
(e.g., emails)

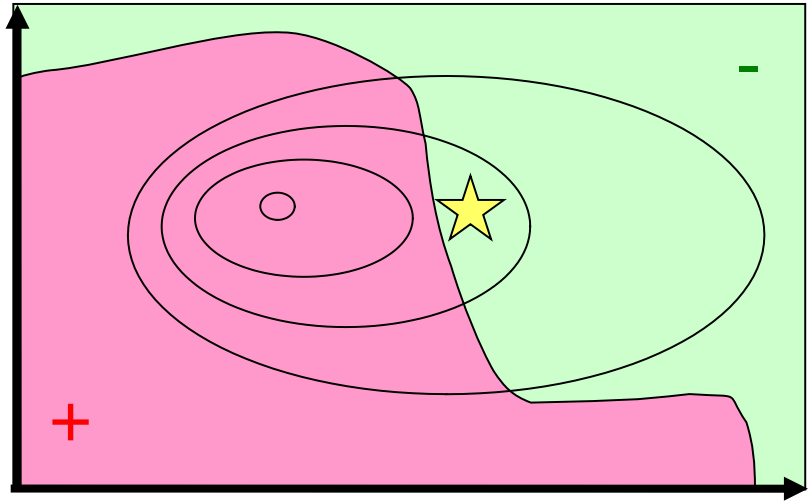


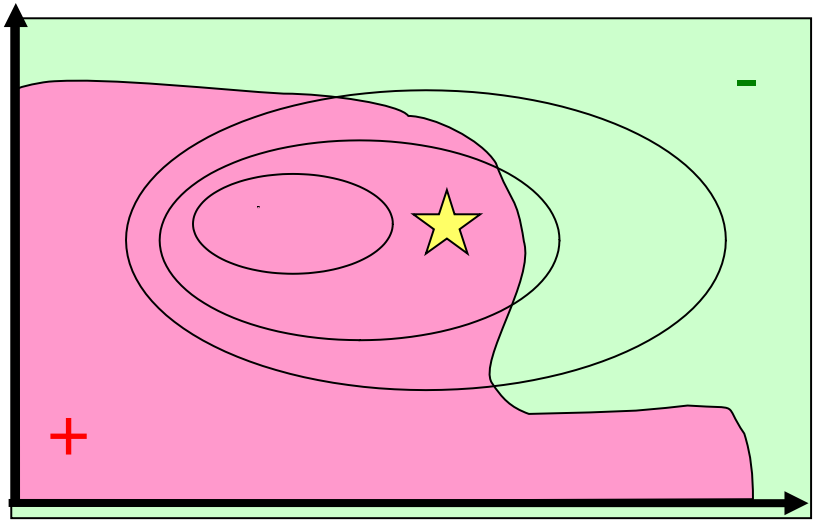

$c(\mathbf{x})$: $X \rightarrow \{+,-\}$
$c \in C$, concept class
(e.g., linear classifier)




$a(\mathbf{x})$: $X \rightarrow R$
$a \in A$
(e.g., more legible
spam is better)

# Adversarial scenario




**Adversary's Task:**
Choose **x** to minimize
$a(\mathbf{x})$ subject to $c(\mathbf{x}) = -$

**Defender's Task:**
Choose new $c'(\mathbf{x})$ minimize
(cost-sensitive) error

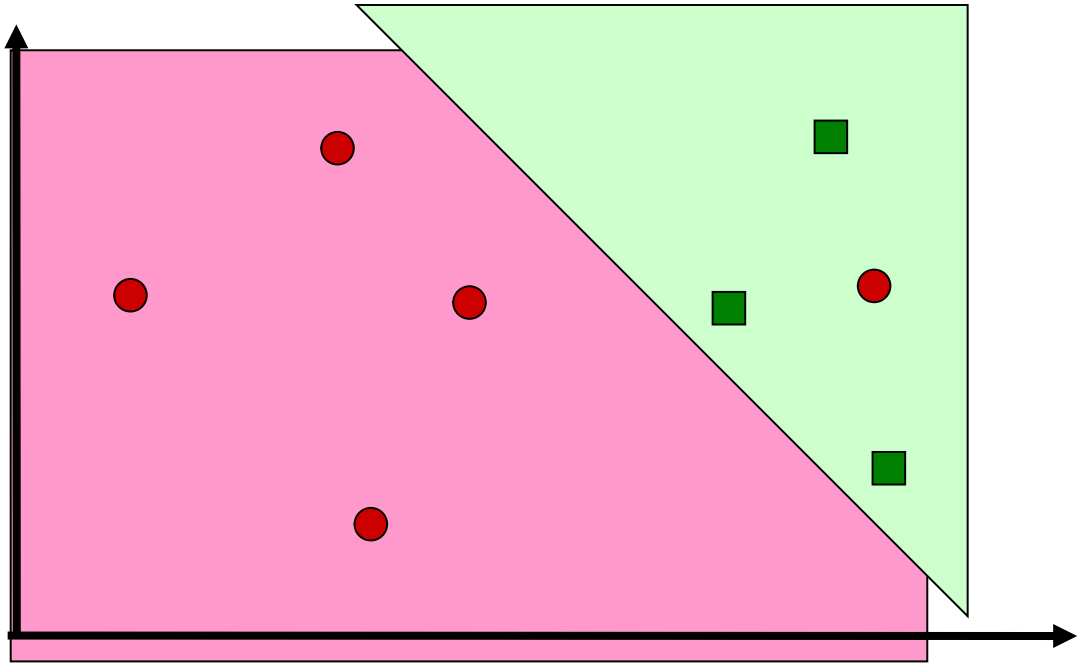
# This is a game!

- Adversary's actions: {$\mathbf{x} \in X$}
- Defender's actions: {$c \in C$}
- Assume perfect information
- Finding a Nash equilibrium is triply exponential (at best)!
- Instead, we'll look at optimal myopic strategies: <u>Best action assuming nothing else changes</u>

# Initial classifier

Learned weights:

cheap = 1.0
mortgage = 1.5
David = -1.0
Oregon = -1.0

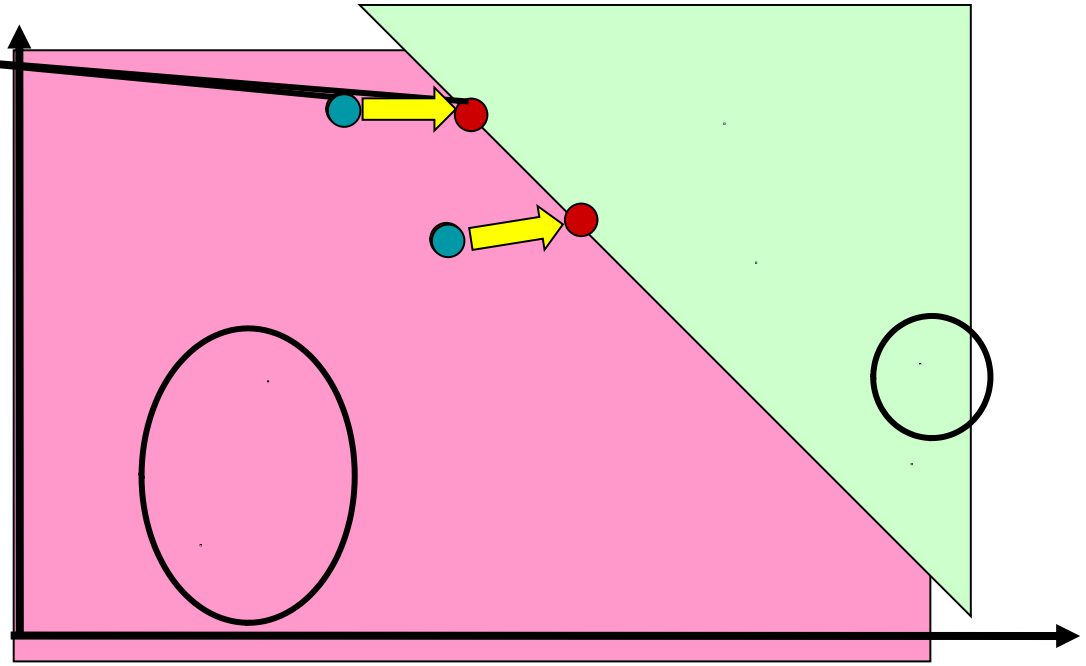


- Set weights using cost-sensitive naïve Bayes
- Assume: training data is untainted

# Adversary's strategy

From: spammer@
example.com
Cheap mortgage now!!!
David Oregon

**affordable** = -1.0
mortgage = 1.5
David = -1.0
Oregon = -1.0

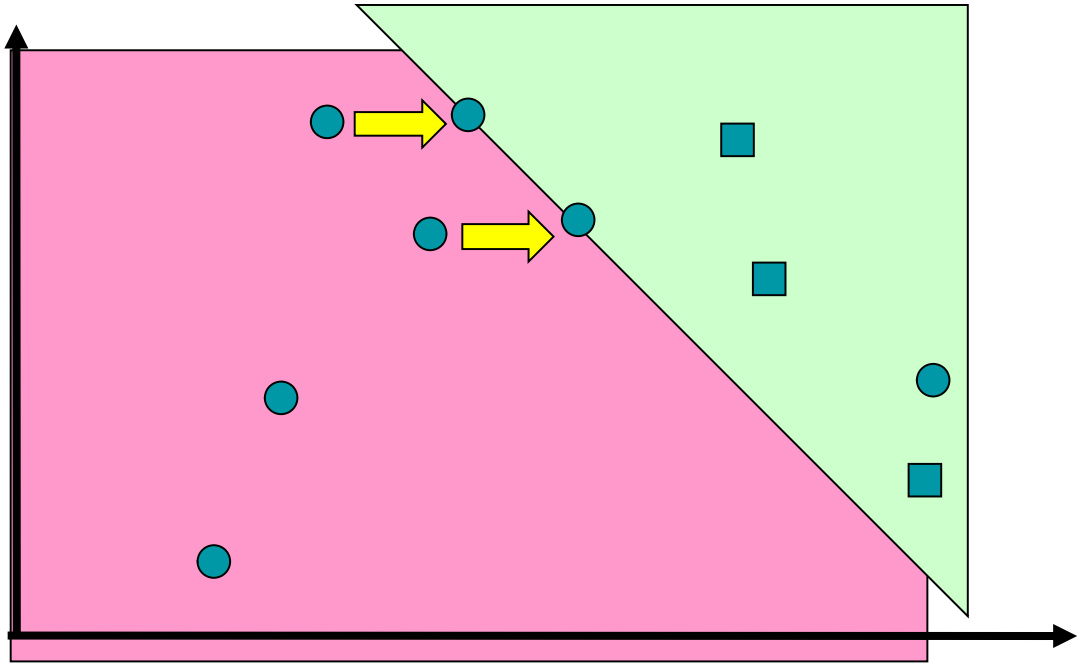


- Cost: $a(\mathbf{x}) = \Sigma_i\, w(x_i, b_i)$
- Solve by dynamic programming
- Assume: that the classifier will not modify $c(\mathbf{x})$

# Classifier's strategy

Learned weights:

cheap = 1.0
mortgage = 1.5
David = -1.0
Oregon = -1.0

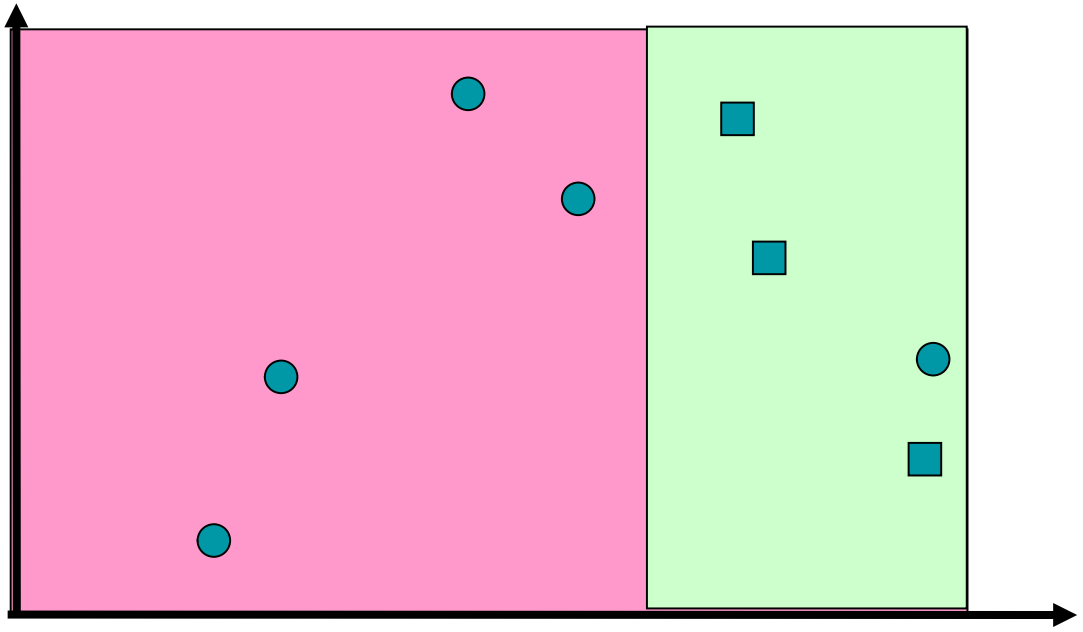


- For given **x**, compute probability it was modified by adversary
- Assume: the adversary is using the optimal strategy

# Classifier's strategy

Learned weights:

affordable = 1.5
mortgage = 2.0
David = -0.5
Oregon = -0.5



- For given **x**, compute probability it was modified by adversary
- Assume: the adversary is using the optimal strategy

# Repeated game

Adversary responds to new classifier;
classifier predicts adversary's revised response

Oscillations occur as adversaries switch strategies back and forth.

# Adversarial learning

- Examine whether the system’s adversarial robustness and reliability
- Develop classifiers that are robust to perturbations (test time) of their inputs, by an adversary intent on fooling the classifier?

# Adversarial attack on Images

- Self-driving car

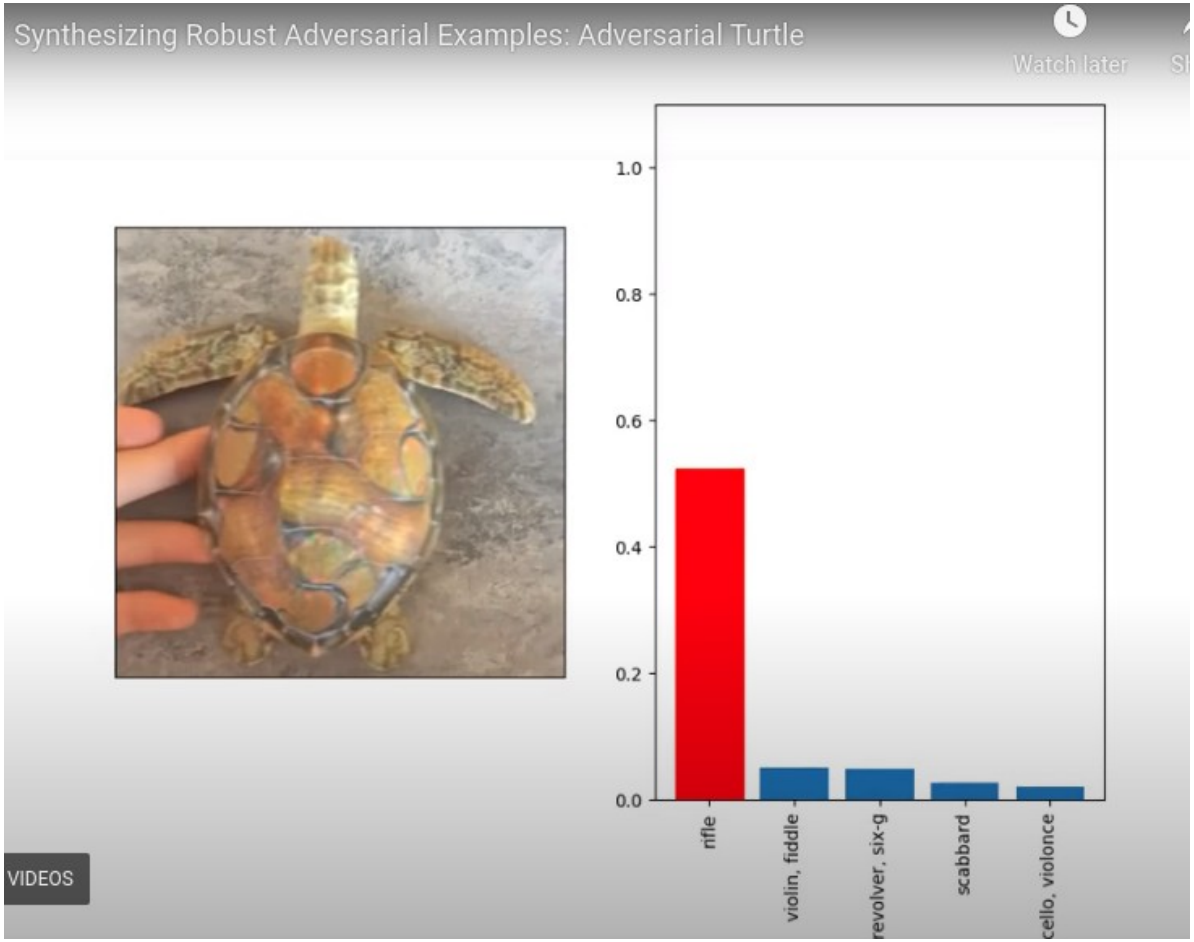
**Lab (Stationary) Test**

Physical road signs with adversarial perturbation under different conditions


**Field (Drive-By) Test**

Video sequences taken under different driving speeds

# Adversarial attack on Images




https://www.youtube.com/watch?v=YXy6oX1iNoA

# Adversarial attack on Images

Simen Thys, Wiebe Van Ranst, Toon Goedemé, 2019

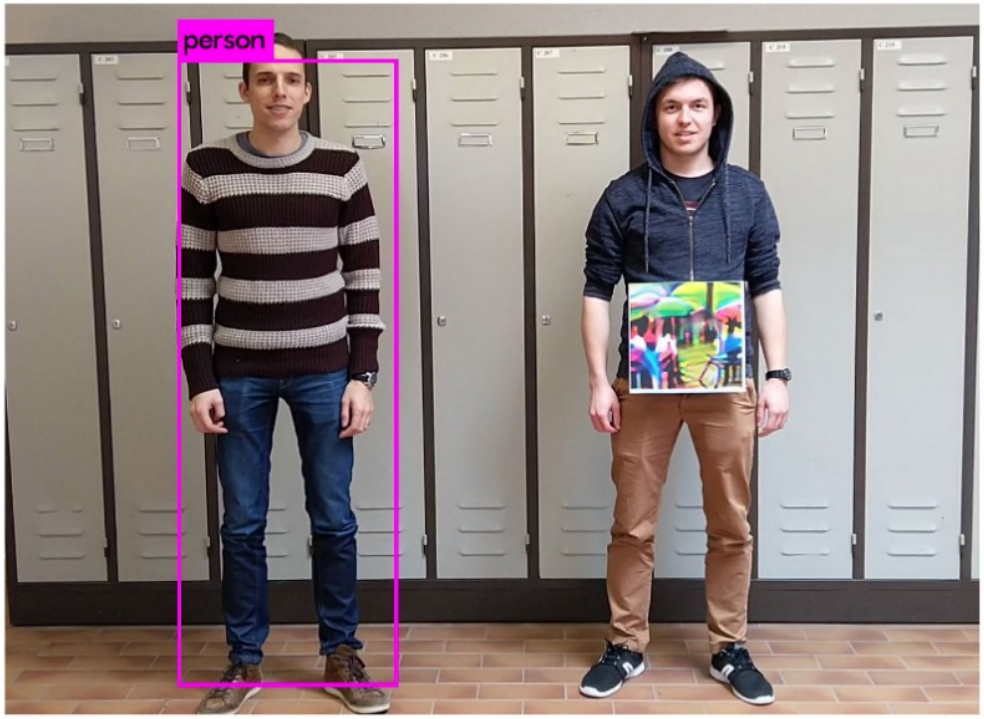



Figure 1: We create an adversarial patch that is successfully able to hide persons from a person detector. Left: The person without a patch is successfully detected. Right: The person holding the patch is ignored.

# Adversarial attack on Images

- A panda or a gibbon?

v.s.

# Image classification

$\{x_i \in X, y_i \in Z\}$

$$\underset{\theta}{\text{minimize}} \frac{1}{m} \sum_{i=1}^{m} \ell(h_\theta(x_i), y_i)$$

$$\theta := \theta - \frac{\alpha}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_\theta \ell(h_\theta(x_i), y_i)$$

The gradient computes how a small adjustment to parameters $\theta$ will affect the loss function

$\boldsymbol{x}$

"panda"
57.7% confidence

# Continuous data

How about the gradient of the loss with respect to the input $x$ itself?

# Continuous data

How about the gradient of the loss with respect to the input $x$ itself?

This quantity will tell us how small changes to the image itself affect the loss function.

# Continuous data

How about the gradient of the loss with respect to the input $x$ itself?

This quantity will tell us how small changes to the image itself affect the loss function.

If the loss is increased, the classification may be wrong!

# Adversarial attack

We’re going to adjust the image to maximize the loss

$$\underset{\hat{x}}{\text{maximize}}\, \ell(h_\theta(\hat{x}), y)$$

where $\hat{x}$ denotes our adversarial example that is attempting to maximize the loss

# Adversarial attack

We’re going to adjust the image to maximize the loss

$$\underset{\hat{x}}{\text{maximize}}\, \ell(h_\theta(\hat{x}), y)$$

where $\hat{x}$ denotes our adversarial example that is attempting to maximize the loss

Any problems?

# Adversarial attack

We’re going to adjust the image to maximize the loss

$$\underset{\hat{x}}{\text{maximize}}\, \ell(h_\theta(\hat{x}), y)$$

where $\hat{x}$ denotes our adversarial example that is attempting to maximize the loss

Any problems?

need to ensure that $\hat{x}$ is close to our original input $x$

# Adversarial attack

Maximize the loss function over the perturbation to data

$$\underset{\delta \in \Delta}{\text{maximize}}\, \ell(h_\theta(x+\delta), y)$$

where $\Delta$ represents an allowable set of perturbations

# Adversarial attack

Maximize the loss function over the perturbation to data

$$\underset{\delta \in \Delta}{\text{maximize}}\, \ell(h_\theta(x+\delta), y)$$

where $\Delta$ represents an allowable set of perturbations

We would like $\Delta$ to capture anything that humans visually feel to be the “same” as the original image:

slight amounts of noise, rotating, scaling, completely changing the image in the “non-panda” locations...

# Adversarial attack

A common practice: small noise

$$\underset{\delta \in \Delta}{\text{maximize}}\, \ell(h_\theta(x+\delta), y)$$

$$\Delta = \{\delta : \|\delta\|_\infty \leq \epsilon\}$$

i.e., we allow the perturbation to have magnitude between $[-\epsilon, \epsilon]$ in each of the pixels

# Adversarial attack

A common practice: small noise

$$\underset{\delta \in \Delta}{\text{maximize}}\, \ell(h_\theta(x+\delta), y)$$

$$\Delta = \{\delta : \|\delta\|_\infty \leq \epsilon\}$$

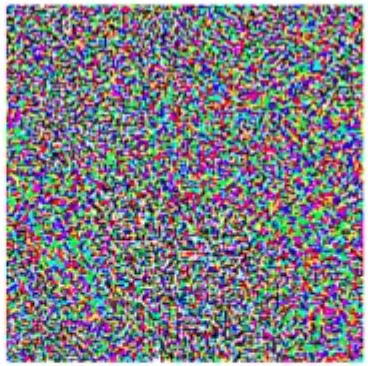
Create perturbations which add such a small component to each pixel in the image that they are visually indistinguishable from the original image

# Add small noise

$\boldsymbol{x}$

"panda"
57.7% confidence

$+ .007 \times$



$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

$=$

$\boldsymbol{x} +$
$\epsilon \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

# Targeted attacks

**Lab (Stationary) Test**

Physical road signs with adversarial perturbation under different conditions


**Field (Drive-By) Test**

Video sequences taken under different driving speeds


Not necessary a speed limit sign; may be a red light

# Targeted attacks

**Lab (Stationary) Test**

Physical road signs with adversarial perturbation under different conditions


**Field (Drive-By) Test**

Video sequences taken under different driving speeds


Not necessary a speed limit sign; may be a red light

Targeted attacks: Make the image classified as any class we desire

# Targeted attacks

We maximize the loss of the correct class while also minimizing the loss of the target class

$$\underset{\delta \in \Delta}{\text{maximize}} (\ell(h_\theta(x+\delta), y) - \ell(h_\theta(x+\delta), y_{\text{target}}))$$

# Adversarial robustness and training

We know how to attack

But how to defend?

Makes classifiers more resistant to attacks

# Risk of a classifier

The risk of a classifier is it's expected loss under the true distribution of samples

$$R(h_\theta) = \mathbf{E}_{(x,y)\sim\mathcal{D}}[\ell(h_\theta(x)), y)]$$

where $\mathcal{D}$ denotes the true distribution over samples.

In most cases, we don't know this true distribution.

Instead, $D = \{(x_i, y_i) \sim \mathcal{D}\}, i = 1, \ldots, m$

Empirical risk: $\hat{R}(h_\theta, D) = \frac{1}{|D|} \sum_{(x,y)\in D} \ell(h_\theta(x)), y)$

# Adversarial risk

Instead of suffering the loss on each sample point, we suffer the worst case loss in some region around the sample point

$$R_{\text{adv}}(h_\theta) = \mathbf{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\delta\in\Delta(x)} \ell(h_\theta(x+\delta)), y)\right]$$

Analogously, the worst-case empirical loss of the classifier:

$$\hat{R}_{\text{adv}}(h_\theta, D) = \frac{1}{|D|}\sum_{(x,y)\in D} \max_{\delta\in\Delta(x)} \ell(h_\theta(x+\delta)), y)$$

# Adversarial training

Minimize worst-case empirical loss of the classifier with respect to θ:

$$\underset{\theta}{\text{minimize}} \frac{1}{|D_{\text{train}}|} \sum_{(x,y)\in D_{\text{train}}} \max_{\delta\in\Delta(x)} \ell(h_\theta(x+\delta)), y)$$

A mini-max or robust optimization formulation of adversarial learning

# Adversarial training

$$\underset{\theta}{\text{minimize}} \frac{1}{|D_{\text{train}}|} \sum_{(x,y)\in D_{\text{train}}} \max_{\delta\in\Delta(x)} \ell(h_\theta(x+\delta)), y)$$

1. For each $x, y \in B$, solve the inner maximization problem (i.e., compute an adversarial example)

$$\delta^\star(s) = \underset{\delta\in\Delta(x)}{\text{argmax}}\, \ell(h_\theta(x+\delta)), y)$$

1. Compute the gradient of the empirical adversarial risk, and update $\theta$

$$\theta := \theta - \frac{\alpha}{|B|} \sum_{(x,y)\in B} \nabla_\theta \ell(h_\theta(x+\delta^\star(x))), y).$$

# Adversarial training

$$\underset{\theta}{\text{minimize}} \frac{1}{|D_{\text{train}}|} \sum_{(x,y) \in D_{\text{train}}} \max_{\delta \in \Delta(x)} \ell(h_\theta(x+\delta)), y)$$

Wh

- adversarial environment:
  - spam classification, malware detection, network intrusion detection
- understand “worst case” performance of the classifier:
  - self-driving car
- Make the system robust

# Summary

- Adversarial attack and defense are a method for approximately solving the inner maximization and/or outer minimization problem respectively
- A mini-max game:
  - “One” loss function
  - One player minimizes it, the other maximizes it
- In practice, adversarial training is complicated: unstable, local minimum/maximum, easy to collapse