Week 9: Adversarial Machine Learning – Vulnerabilities (Part I)

COMP90073
Security Analytics

Yi Han, CIS

Semester 2, 2021

Overview

- Week 9: Adversarial Machine Learning – Vulnerabilities
  - Definition + examples
  - Classification
  - Evasion attacks
    - Gradient-descent based approaches
    - Automatic differentiation
    - Real-world example
  - Poisoning attacks
  - Transferability

Overview

- Week 9: Adversarial Machine Learning – Vulnerabilities
  - Definition + examples
  - Classification
  - Evasion attacks
    - Gradient-descent based approaches
    - Automatic differentiation
    - Real-world example
  - Poisoning attacks
  - Transferability

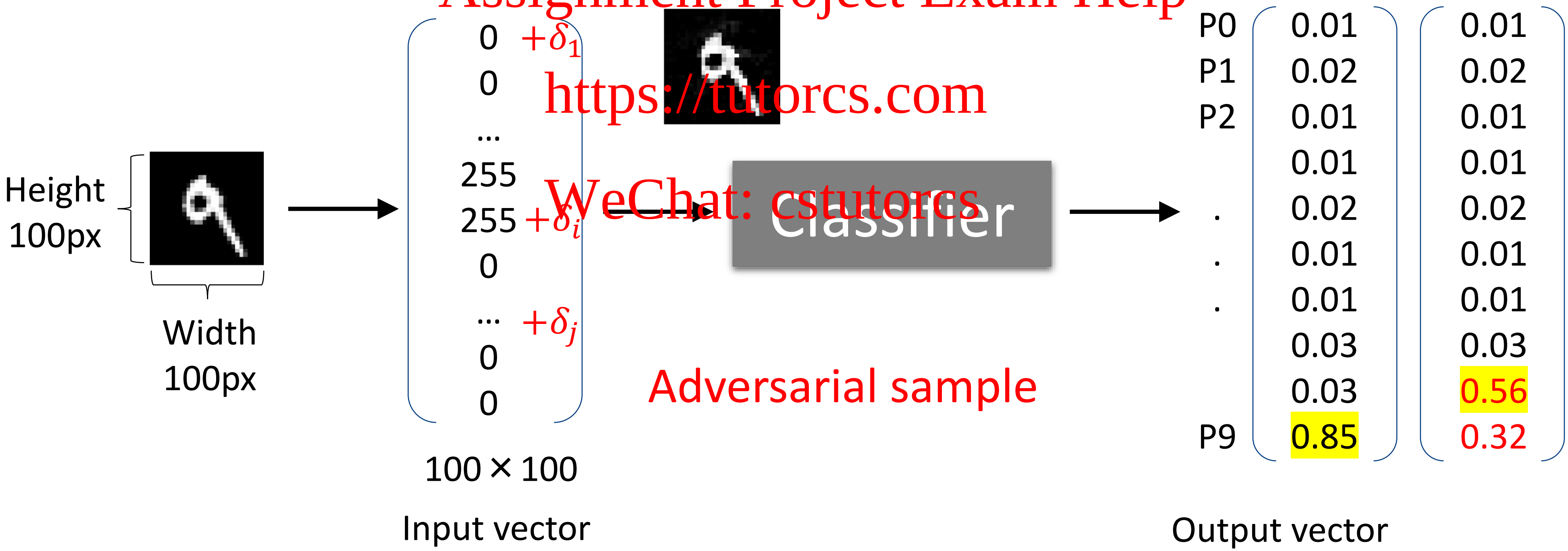
Definition

What is Adversarial Machine Learning (AML)?

"Adversarial machine learning is a technique employed in the field of machine learning which attempts to fool models through malicious input." – Wikipedia

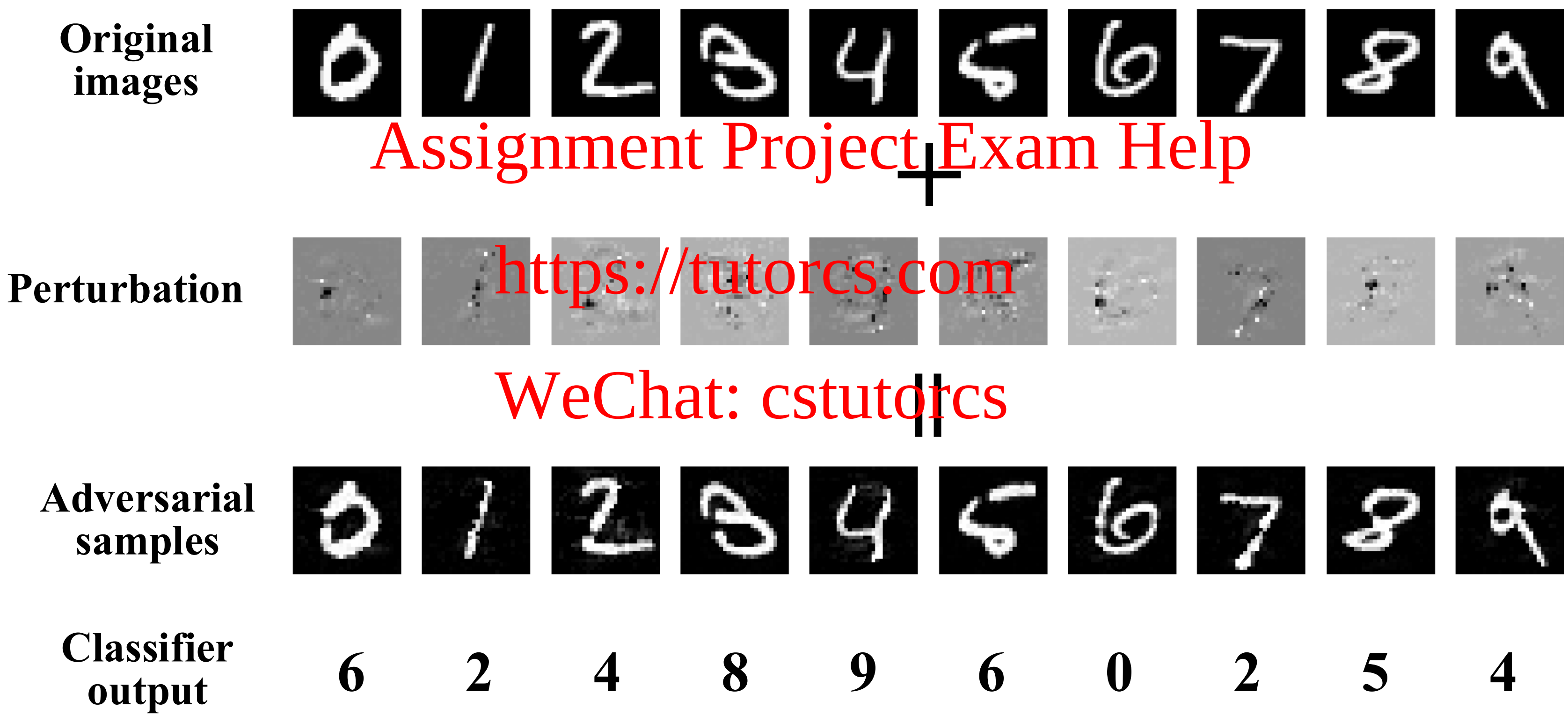
Examples

- Test-time attack
  - Image classifier C: input images $X \rightarrow \{0, 1, 2, \ldots, 9\}$,



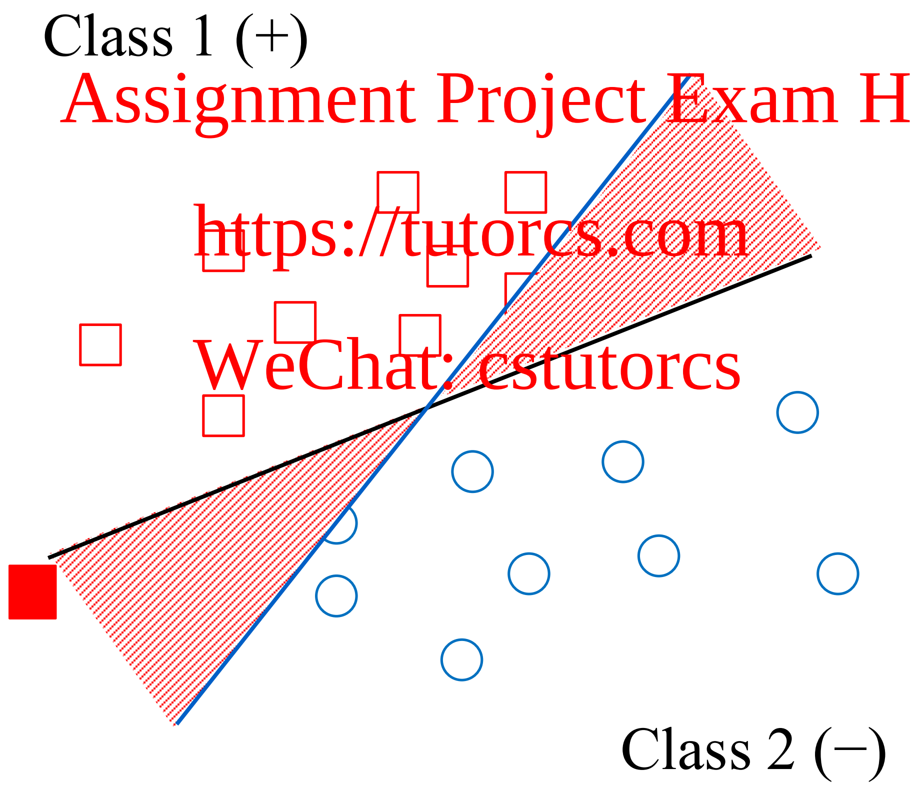
Examples

- Test-time attack



Examples

- Training-time attack
  - Insert extra training points to maximise the loss



Examples

- Huge amount of attention
  - Mission-critical tasks

Overview

- Week 9: Adversarial Machine Learning – Vulnerabilities
  - Definition + examples
  - Classification
  - Evasion attacks
    - Gradient-descent based approaches
    - Automatic differentiation
    - Real-world example
  - Poisoning attacks
  - Transferability

Classification

- Classification [1]
  - Exploratory vs. Causative – influence
    - Exploratory/evasion: test time
    - Causative/poisoning: training time
  - Integrity vs. Availability – security violation
    - Integrity: harmful instances to pass filters
    - Availability: denial of service, benign instances to be filtered
  - Targeted vs. Indiscriminate/Untargeted – specificity
    - Targeted: misclassified as a specific class
    - Indiscriminate/untargeted: misclassified as any other class
  - White-box vs. Black-box – attacker information
    - White-box: full knowledge of the victim model
    - Black-box: no/minimum knowledge of the model

Overview

- Week 9: Adversarial Machine Learning – Vulnerabilities
  - Definition + examples
  - Classification
  - Evasion attacks
    - Gradient-descent based approaches
    - Automatic differentiation
    - Real-world example
  - Poisoning attacks
  - Transferability

Evasion attacks (definition)

- Evasion attack
  - Aim: minimum perturbation δ to the input x, in order to cause model *C* to misclassify

$$x \to x + \delta \quad (x, \delta \in [0,1]^d)$$

such that (*s.t.*)

$$C(x + \delta) \neq C(x)$$ Indiscriminate

OR $$C(x + \delta) = l_{target}$$ Targeted

Evasion attacks (definition)

- Evasion attack
  - Formulated as an optimisation problem

$$\arg\min_{\delta\in[0,1]^d} \|\delta\| \qquad (1)$$

$$\left.\begin{array}{l} \text{s.t. } C(x+\delta) \neq C(x) \\ \text{OR} \quad C(x+\delta) = l_{target} \end{array}\right\} \text{Highly non-linear}$$

  - p-norm: $\|\delta\|_p = \left(\sum_{i=1}^{d} |\delta_i|^p\right)^{1/p}$
  - $\|\delta\|_1 = \sum_{i=1}^{d} |\delta_i|$, $\|\delta\|_2 = \sqrt{\sum_{i=1}^{d} |\delta_i|^2}$, $\|\delta\|_\infty = \max_i |\delta_i|$
    - E.g., $\delta = \langle 1, 2, 3, -4 \rangle$, $\|\delta\|_1 = 10$, $\|\delta\|_2 = \sqrt{30}$, $\|\delta\|_\infty = 4$

Evasion attacks (definition)

Transform (1) to the following problem [2]:

$$\arg\min_{\delta \in [0,1]^d} \|\delta\| - c \cdot f_{true}(x+\delta)$$

Indiscriminate

$$\arg\min_{\delta \in [0,1]^d} \|\delta\| + c \cdot f_{target}(x+\delta)$$

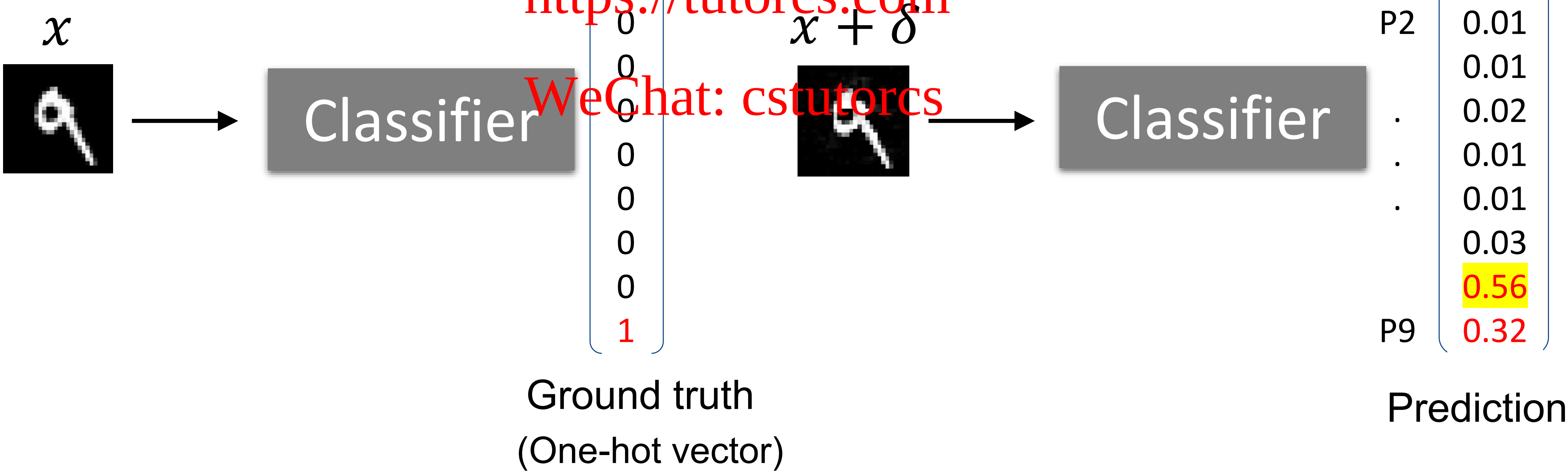
Targeted

Objective function *f*: how close the prediction and the target are, e.g., the cross entropy loss function

Evasion attacks (definition)

- Indiscriminate attack: $\arg\min_{\delta\in[0,1]^d} \|\delta\| - c \cdot f_{true}(x+\delta)$
  - Prediction as different from the truth as possible

Misclassified as any class other than "9"



Evasion attacks (definition)

- Targeted attack: $\arg\min_{\delta\in[0,1]^d} \|\delta\| + c \cdot f_{target}(x+\delta)$
  - Prediction as close to the target as possible

	Prediction	Target (One-hot vector)
P0	0.01	0
P1	0.02	0
P2	0.01	0
	0.01	0
.	0.02	0
.	0.01	0
.	0.01	0
	0.03	0
	0.56	1
P9	0.32	0

Misclassified as “8” (target class)

Evasion attacks (definition)

Transform (1) to the following problem [2]:

$$\arg\min_{\delta \in [0,1]^d} \|\delta\| - c \cdot f_{true}(x+\delta)$$ Indiscriminate

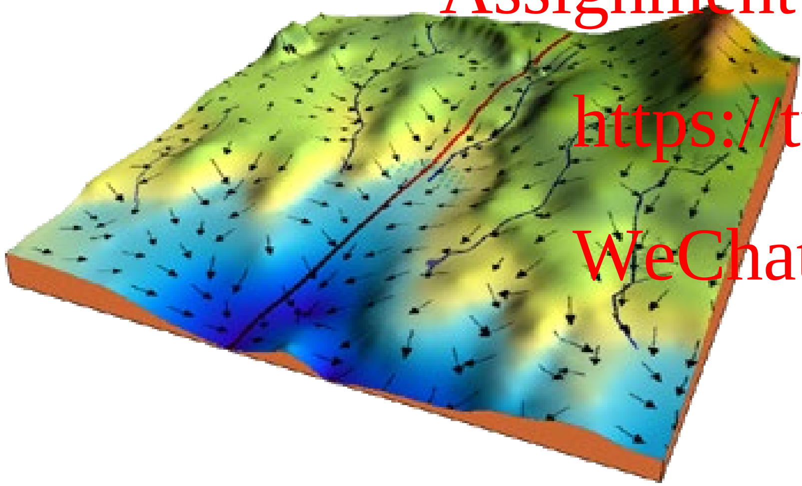
$$\arg\min_{\delta \in [0,1]^d} \|\delta\| + c \cdot f_{target}(x+\delta)$$ Targeted

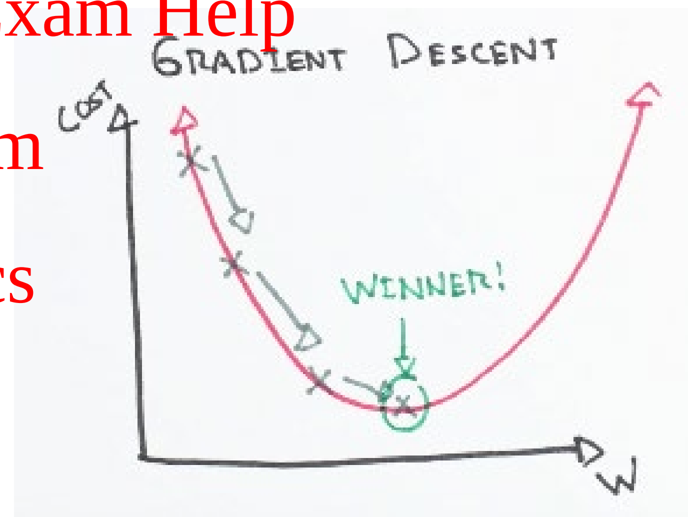
How to find the minimum perturbation δ?

Evasion attacks (gradient descent)

Gradient descent

- Gradient: a vector that points in the direction of greatest increase of a function



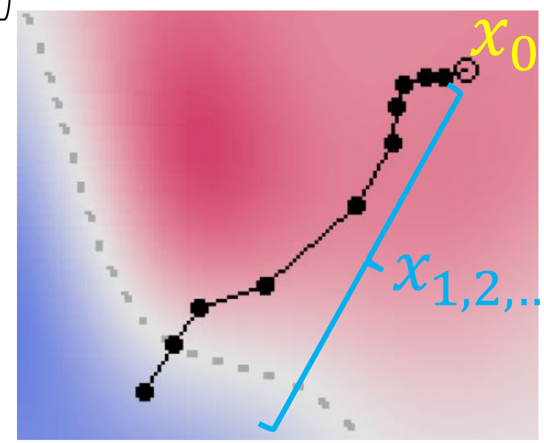


https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html

Evasion attacks (gradient descent)

$$\arg\min_{\delta\in[0,1]^d} \|\delta\| - c\cdot f_{true}(x+\delta)$$ Indiscriminate

$$\arg\min_{\delta\in[0,1]^d} \underbrace{\|\delta\| + c\cdot f_{target}(x+\delta)}_{J}$$ Targeted



- Start with the initial input x_0
- Repeat $x_i \leftarrow x_{i-1} - \alpha \boxed{\frac{\partial J}{\partial x_{i-1}}}, i > 0$
- Until (1) $C(x_i) \neq C(x_0)$ (or $C(x_i) = l_{target}$), or → success

  (2) $\|\delta\| = \|x_i - x_0\| > \epsilon$, or

  (3) $i \geq i_{max}$, or → failure

  (4) $|J(x_i) - J(x_{i-1})| \leq \Delta$

Evasion attacks (gradient descent)

$$\arg\min_{\delta \in [0,1]^d} \|\delta\| - c \cdot f_{true}(x+\delta)$$ Indiscriminate

$$\arg\min_{\delta \in [0,1]^d} \underbrace{\|\delta\| + c \cdot f_{target}(x+\delta)}_{J}$$ Targeted

- …
- Repeat $x_i \leftarrow x_{i-1} - \alpha \boxed{\frac{\partial J}{\partial x_{i-1}}}, i > 0$
- …

How to design the objective function *f* ?

Evasion attacks (FGSM)

- Fast gradient sign method (FGSM) [3]:

$$\arg\min_{\delta\in[0,1]^d} \cancel{\|\delta\|} - c \cdot f_{true}(x+\delta) \qquad \arg\min_{\delta\in[0,1]^d} -loss_{true}\,(x+\delta)$$

f = cross entropy loss

$$\arg\min_{\delta\in[0,1]^d} \cancel{\|\delta\|} + c \cdot f_{target}(x+\delta) \longrightarrow \arg\min_{\delta\in[0,1]^d} loss_{target}(x+\delta)$$

- Single step ϵ: fast rather than optimal

$$x' \leftarrow x + \epsilon \cdot sign\left(\frac{\partial loss_{true}}{\partial x}\right)$$

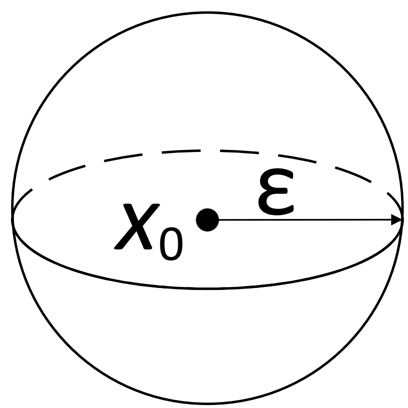
OR $$x' \leftarrow x - \epsilon \cdot sign\left(\frac{\partial loss_{target}}{\partial x}\right)$$

- Not meant to produce the minimal adversarial perturbations

Evasion attacks (Iterative Gradient Sign)

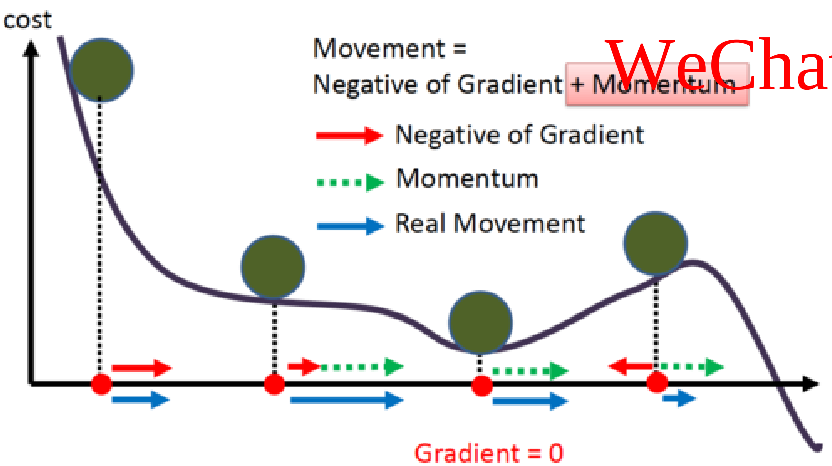
- Iterative gradient sign [4]
  - Single step ϵ → multiple smaller steps α
  - $x_i \leftarrow \text{clip}_\epsilon\left(x_{i-1} + \alpha \cdot \cancel{\text{sign}}\left(\frac{\partial f_{true}}{\partial x_{i-1}}\right)\right)$ OR
  - $x_i \leftarrow \text{clip}_\epsilon\left(x_{i-1} - \alpha \cdot \cancel{\text{sign}}\left(\frac{\partial f_{target}}{\partial x_{i-1}}\right)\right)$
  - clip_ϵ: make sure that x_{ij} is within the range of $\left[x_{0j} - \epsilon, x_{0j} + \epsilon\right]$

    → projection

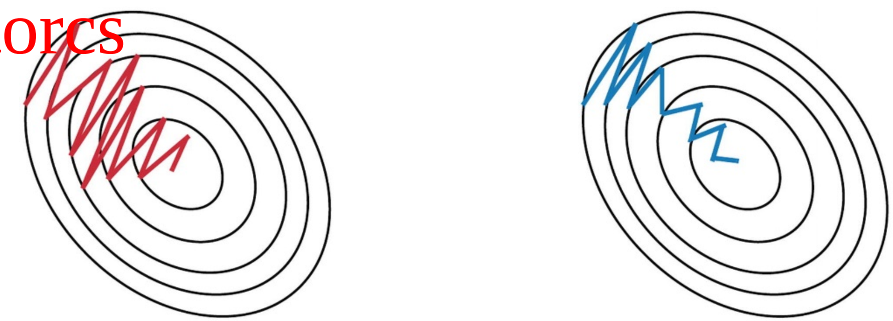


Evasion attacks (Momentum Iterative FGSM)

- Momentum iterative fast gradient sign method
  - $g_i = \mu \cdot g_{i-1} + \frac{\nabla_x J(x_{i-1})}{\|\nabla_x J(x_{i-1})\|_1}, \; x_i \leftarrow x_{i-1} - \alpha \cdot \text{sign}(g_i)$
  - Momentum overcome two problems of vanilla gradient descent
    - Get stuck in local minima
    - Oscillation



https://medium.com/analytics-vidhya/momentum-rmsprop-and-adam-optimizer-5769721b4b19



Stochastic Gradient Descent **withhout** Momentum

Stochastic Gradient Descent **with** Momentum

https://eloquentarduino.github.io/2020/04/stochastic-gradient-descent-on-your-microcontroller/

Evasion attacks (C&W)

C & W attack [2]

$$\arg\min_{\delta \in [0,1]^d} \|\delta\| + c \cdot f(x+\delta)$$

$$C(x+\delta) = l_{target} \text{ if and only if } f(x+\delta) \le 0$$

$$C(x+\delta) \neq l_{target} \Leftrightarrow f(x+\delta) > 0$$

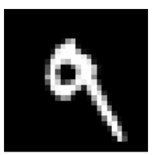
Consistent with the definition of function *f*: how close the prediction and the target are

Evasion attacks (C&W)

$C(x+\delta) = l_{target}$ if and only if $f(x+\delta) = f(x') \leq 0$

- Option 1: $f(x') = max\left(\max_{i \neq t} F(x')_i - F(x')_t,\ 0\right)$
- Option 2: $f(x') = \log(1 + \exp(\max_{i \neq t} F(x')_i - F(x')_t)) - \log(2)$
- Option 3: $f(x') = max(0.5 - F(x')_t,\ 0)$

F(x): output vector for x, i.e., probabilities of the input x belonging to each class. For example:



$\longrightarrow$ Classifier $\longrightarrow$

$$\begin{array}{c} \mathrm{F(x)_0} \\ \mathrm{F(x)_1} \\ \mathrm{F(x)_2} \\ \\ . \\ . \\ . \\ \\ \\ \mathrm{F(x)_9} \end{array} \begin{pmatrix} 0.01 \\ 0.02 \\ 0.01 \\ 0.01 \\ 0.02 \\ 0.01 \\ 0.01 \\ 0.03 \\ 0.03 \\ 0.85 \end{pmatrix} \mathrm{F}(x)$$

CleverHans

- CleverHans
- Do not use the latest version
- Download from: https://github.com/tensorflow/cleverhans/releases/tag/v.3.0.1
- Prerequisite:
  - Python3 (https://www.python.org/downloads/)
  - Tensorflow (https://www.tensorflow.org/install/)
  - Python 3.5/3.6/3.7 and TensorFlow {1.8, 1.12, 1.14}
- Installation:
  - cd cleverhans
  - pip install -e .

Overview

- Week 9: Adversarial Machine Learning – Vulnerabilities
  - Definition + examples
  - Classification
  - Evasion attacks
    - Gradient-descent based approaches
    - Automatic differentiation
    - Real-world example
  - Poisoning attacks
  - Transferability

Evasion attacks (automatic differentiation)

$$\arg\min_{\delta\in[0,1]^d} \|\delta\| - c\cdot f_{true}(x+\delta)$$ Indiscriminate

$$\arg\min_{\delta\in[0,1]^d} \underbrace{\|\delta\| + c\cdot f_{target}(x+\delta)}_{J}$$ Targeted

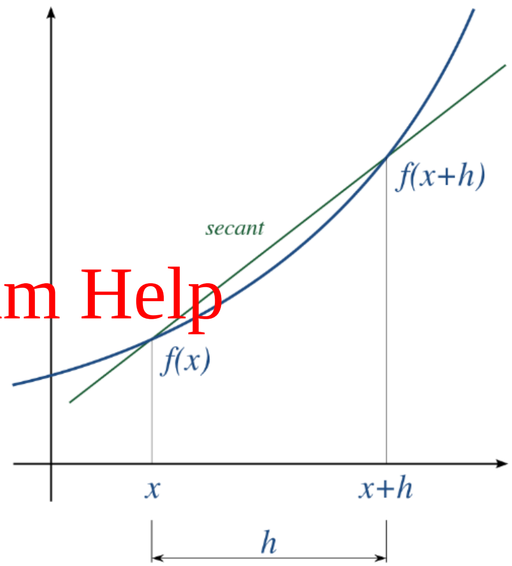
- Start with the initial input x_0
- Repeat $x_i \leftarrow x_{i-1} - \alpha \boxed{\frac{\partial J}{\partial x_{i-1}}}$, $i > 0$
- Until $C(x_i) \neq C(x_0)$ $(or\ C(x_i) = l_{target})$

How to calculate the partial derivatives?

Evasion attacks (automatic differentiation)

Derivative

- Definition: $f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$
- Numerical differentiation
  - $\frac{f(x+h) - f(x)}{h}, \frac{f(x+h) - f(x-h)}{2h}$
  - Significant round-off errors



- Symbolic differentiation: apply chain rules to symbolic expressions
  - Exponentially-long results

Evasion attacks (automatic differentiation)

Automatic differentiation

- A set of techniques to numerically evaluate the derivative of a function specified by a computer program – Wikipedia
- Any complicated function f can be rewritten as the composition of a sequence of primitive functions:

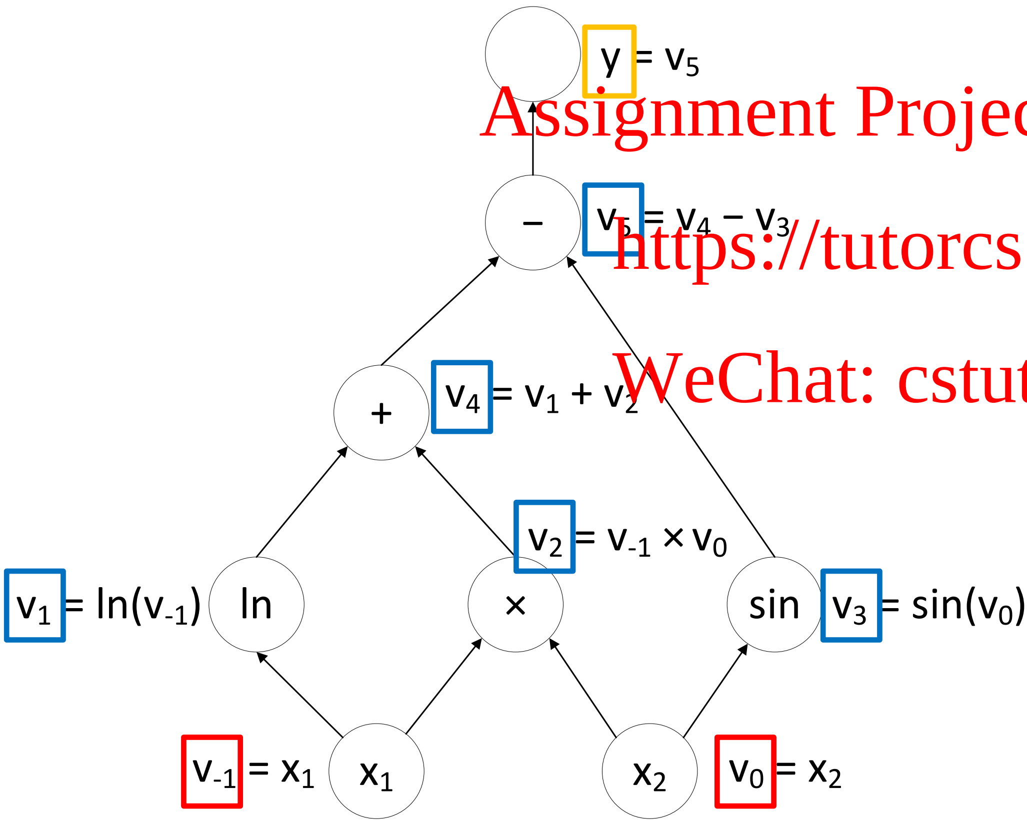
$$f = f_0 \circ f_1 \circ f_2 \circ \cdots \circ f_n$$

- Apply the chain rule

- Forward mode: $\frac{\partial f}{\partial x} = \frac{\partial f_0}{\partial f_1}\left(\frac{\partial f_1}{\partial f_2}\left(\ldots\left(\frac{\partial f_{n-1}}{\partial f_n}\frac{\partial f_n}{\partial x}\right)\right)\right)$
- Reverse mode: $\frac{\partial f}{\partial x} = \left(\left(\left(\frac{\partial f_0}{\partial f_1}\frac{\partial f_1}{\partial f_2}\right)\ldots\right)\frac{\partial f_{n-1}}{\partial f_n}\right)\frac{\partial f_n}{\partial x}$

Evasion attacks (automatic differentiation)

- Given $y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$, calculate $\frac{\partial y}{\partial x_1}$ at (2,5)
- Forward mode [5]



Computational graph

Output variables:

$$y_{m-i} = v_{l-i}, i = m-1, \dots, 0$$

Working variables:

$$v_i, i = 1, \dots, l$$

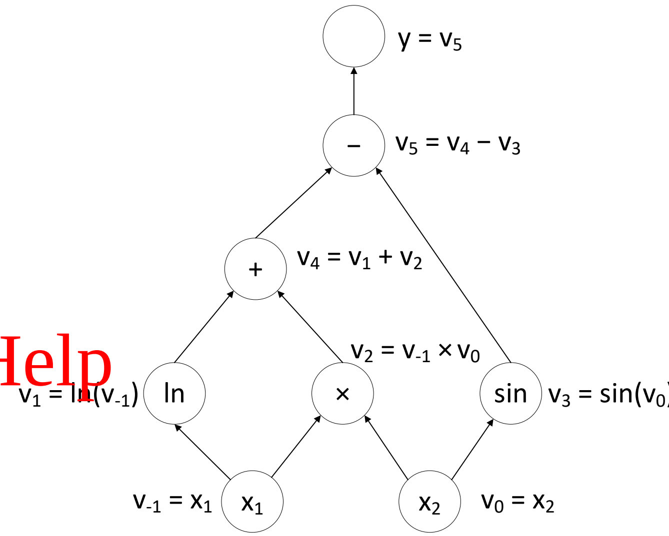
Input variables:

$$v_{i-n} = x_i, i = 1, \dots, n$$

Evasion attacks (automatic differentiation)

Forward evaluation trace

$$
\begin{aligned}
v_{-1} &= x_1 &&= 2\\
v_0 &= x_2 &&= 5\\
\hline
v_1 &= \ln v_{-1} &&= \ln 2\\
v_2 &= v_{-1} \times v_0 &&= 2 \times 5\\
v_3 &= \sin v_0 &&= \sin 5\\
v_4 &= v_1 + v_2 &&= 0.6931 + 10\\
v_5 &= v_4 - v_3 &&= 10.6931 + 0.9589\\
\hline
y &= v_5 &&= 11.6521
\end{aligned}
$$



Forward derivative trace

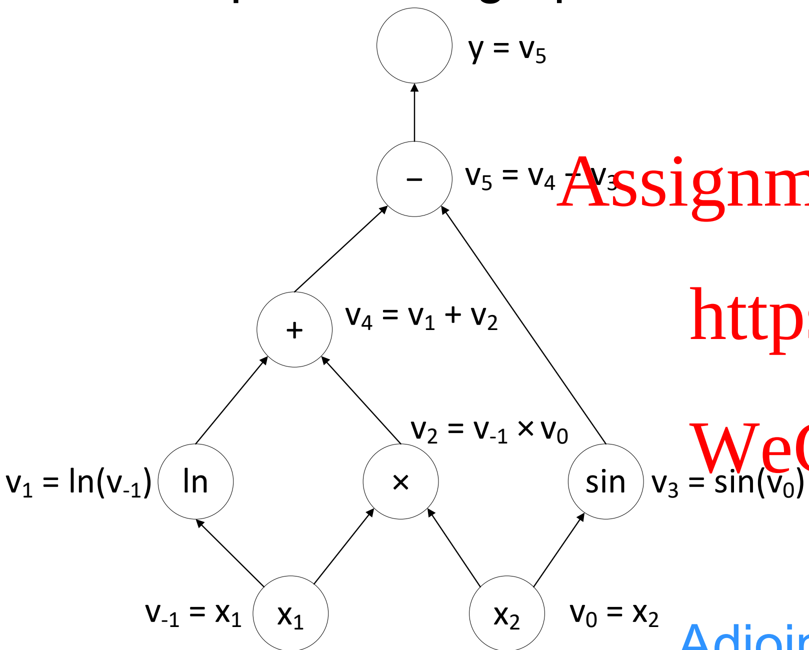
$$\dot{v}_i = \frac{\partial v_i}{\partial x_1}$$

$\dot{v}_{-1}$	$= \dot{x}_1$	$= 1$	0
$\dot{v}_0$	$= \dot{x}_2$	$= 0$	1
$\dot{v}_1$	$= \dot{v}_{-1}/v_{-1}$	$= 1/2$	0
$\dot{v}_2$	$= \dot{v}_{-1} \times v_0 + v_{-1} \times \dot{v}_0$	$= 1 \times 5 + 2 \times 0$	0×5 + 2×1
$\dot{v}_3$	$= \cos v_0 \times \dot{v}_0$	$= \cos 5 \times 0$	cos5×1
$\dot{v}_4$	$= \dot{v}_1 + \dot{v}_2$	$= 0.5 + 5$	0 + 2
$\dot{v}_5$	$= \dot{v}_4 - \dot{v}_3$	$= 5.5 - 0$	2-cos5
$\dot{y}$	$= \dot{v}_5$	$= \mathbf{5.5}$	2-cos5

Evasion attacks (automatic differentiation)

- Reverse mode [5]

Computational graph



Adjoint

$$\bar{v}_i = \frac{\partial y_j}{\partial v_i}$$

Forward evaluation trace

v_{-1}	$= x_1$	$= 2$
v_0	$= x_2$	$= 5$
v_1	$= \ln v_{-1}$	$= \ln 2$
v_2	$= v_{-1} \times v_0$	$= 2 \times 5$
v_3	$= \sin v_0$	$= \sin 5$
v_4	$= v_1 + v_2$	$= 0.6931 + 10$
v_5	$= v_4 - v_3$	$= 10.6931 + 0.9589$
y	$= v_5$	$= 11.6521$

Reverse adjoint trace

$\bar{x}_1$	$= \bar{v}_{-1}$		$= 5.5$	
$\bar{x}_2$	$= \bar{v}_0$		$= 1.7163$	
$\bar{v}_{-1}$	$= \boxed{\bar{v}_{-1} + \bar{v}_1 (\partial v_1 / \partial v_{-1})}$	$= \bar{v}_{-1} + \bar{v}_1 / v_{-1}$		$= 5.5$
$\bar{v}_0$	$= \boxed{\bar{v}_0 + \bar{v}_2 (\partial v_2 / \partial v_0)}$	$= \bar{v}_0 + \bar{v}_2 \times v_{-1}$		$= 1.7163$
$\bar{v}_{-1}$	$= \bar{v}_2 (\partial v_2 / \partial v_{-1})$	$= \bar{v}_2 \times v_0$		$= 5$
$\bar{v}_0$	$= \bar{v}_3 (\partial v_3 / \partial v_0)$	$= \bar{v}_3 \times \cos v_0$		$= -0.2837$
$\bar{v}_2$	$= \bar{v}_4 (\partial v_4 / \partial v_2)$	$= \bar{v}_4 \times 1$		$= 1$
$\bar{v}_1$	$= \bar{v}_4 (\partial v_4 / \partial v_1)$	$= \bar{v}_4 \times 1$		$= 1$
$\bar{v}_3$	$= \bar{v}_5 (\partial v_5 / \partial v_3)$	$= \bar{v}_5 \times (-1)$		$= -1$
$\bar{v}_4$	$= \bar{v}_5 (\partial v_5 / \partial v_4)$	$= \bar{v}_5 \times 1$		$= 1$
$\bar{v}_5$	$= \bar{y}$		$= 1$	

Evasion attacks (automatic differentiation)

- Example 1: $y = \ln(x_1) + x_1 x_2 - \sin(x_2)$
- Calculate $\left(\frac{\partial y}{\partial x_1}, \frac{\partial y}{\partial x_2}\right)$
  - Forward mode: __ time(s)
  - Reverse mode: __ time(s)

- Example 2: $y_1 = \ln(x) + x, y_2 = x - \sin(x)$
- Calculate $\left(\frac{\partial y_1}{\partial x}, \frac{\partial y_2}{\partial x}\right)$
  - Forward mode: __ time(s)
  - Reverse mode: __ time(s)

Evasion attacks (automatic differentiation)

Function $f: R^n \rightarrow R^m$

- n independent x_i as inputs, m dependent y_j as outputs
- Forward mode: $m \gg n$ (one forward run can calculate $\frac{\partial y}{\partial x_i}$)
- Reverse mode: $n \gg m$ (one reverse run can calculate $\frac{\partial y_j}{\partial x}$)

Tensorflow example

```
#http://laid.delanover.com/gradients-in-tensorflow/
import tensorflow as tf

x = tf.Variable(1.)
y = tf.Variable(2.)
z = tf.subtract(2*x, y)
grad = tf.gradients(z, [x, y])

sess = tf.Session()
sess.run(tf.global_variables_initializer())
print(sess.run(grad)) # [2.0, -1.0]
```

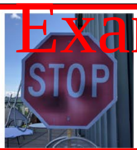
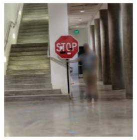
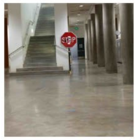
Overview

- Week 9: Adversarial Machine Learning – Vulnerabilities
  - Definition + examples
  - Classification
  - Evasion attacks
    - Gradient-descent based approaches
    - Automatic differentiation
    - Real-world example
  - Poisoning attacks
  - Transferability

Evasion attacks (real-world example)

- Robust Physical-World Attacks on Deep Learning Visual Classification [6]
  - Stop sign, Right Turn sign → Speed Limit 45
  - Drive-By (Field) Tests
  - Start from 250 ft away
  - Classify every 10th frame

Table 1: Sample of physical adversarial examples against LISA-CNN and GTSRB-CNN.

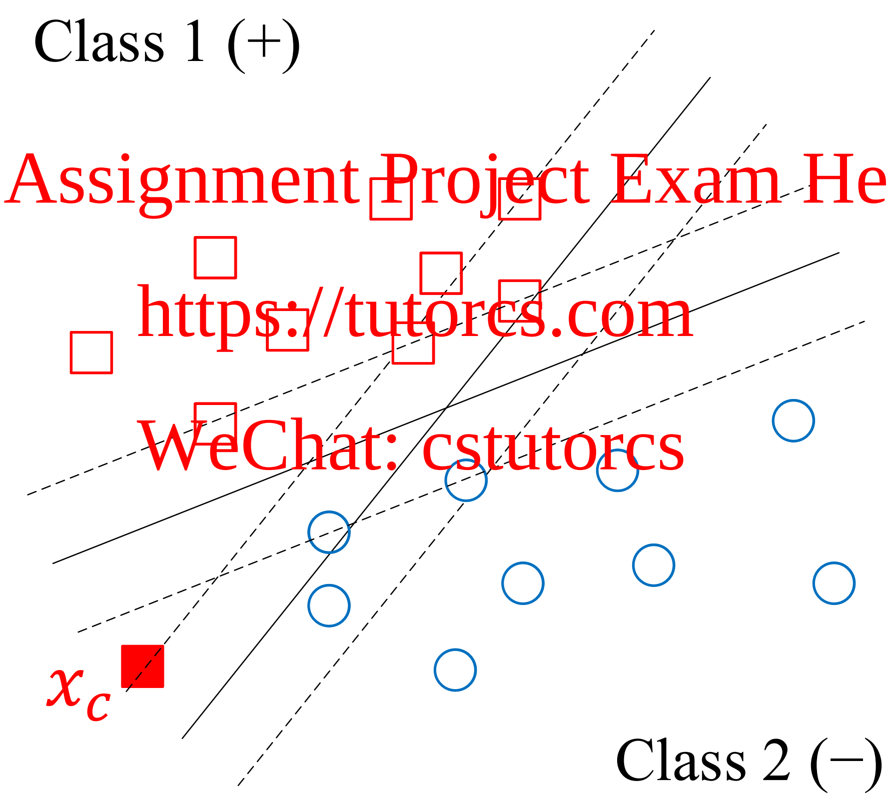
Distance/Angle	Subtle Poster	Subtle Poster Right Turn	Camouflage Graffiti	Camouflage Art (LISA-CNN)	Camouflage Art (GTSRB-CNN)
5′ 0°					
5′ 15°					
10′ 0°					
10′ 30°					
40′ 0°					
Targeted-Attack Success	100%	73.33%	66.67%	100%	80%

Overview

- Week 9: Adversarial Machine Learning – Vulnerabilities
  - Definition + examples
  - Classification
  - Evasion attacks
    - Gradient-descent based approaches
    - Automatic differentiation
    - Real-world example
  - Poisoning attacks
  - Transferability

Poisoning attacks

- Insert extra points to maximally decrease the accuracy [8]



Poisoning attacks

- Attacker's aim: maximise the hinge loss over the validation data $D_{val} = \{x_i, y_i\}_{i=1}^{m}$
- Optimisation problem:

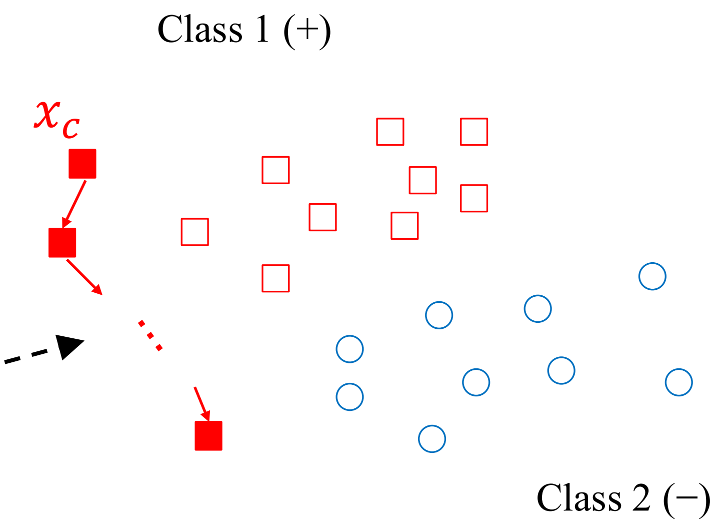
$$\arg\max_{x_c} L(x_c) = \sum_{i=1}^{m} \left(1 - \boxed{y_i}\,\boxed{f_{x_c}(x_i)}\right)$$

To find the optimal poisoning data x_c:

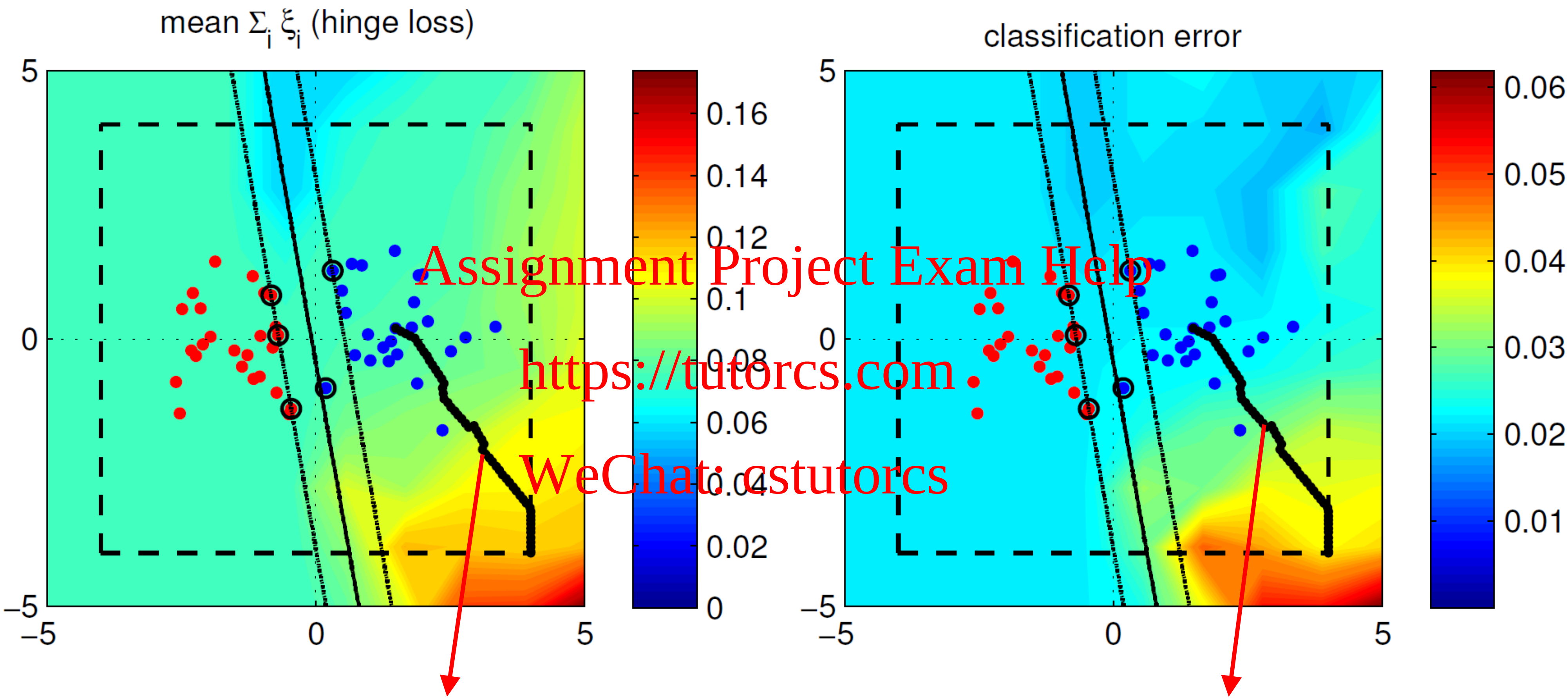
- Random initial attack point x_c
- Update: re-compute the SVM;

$$x_c^p \leftarrow x_c^{p-1} + \alpha \boxed{\frac{\partial L}{\partial x_c^{p-1}}}, p > 0$$

- Until $L(x_c^p) - L(x_c^{p-1}) < \varepsilon$



Poisoning attacks



As the attack point x_c moves towards a local maximum, both the hinge loss and the classification error increase.

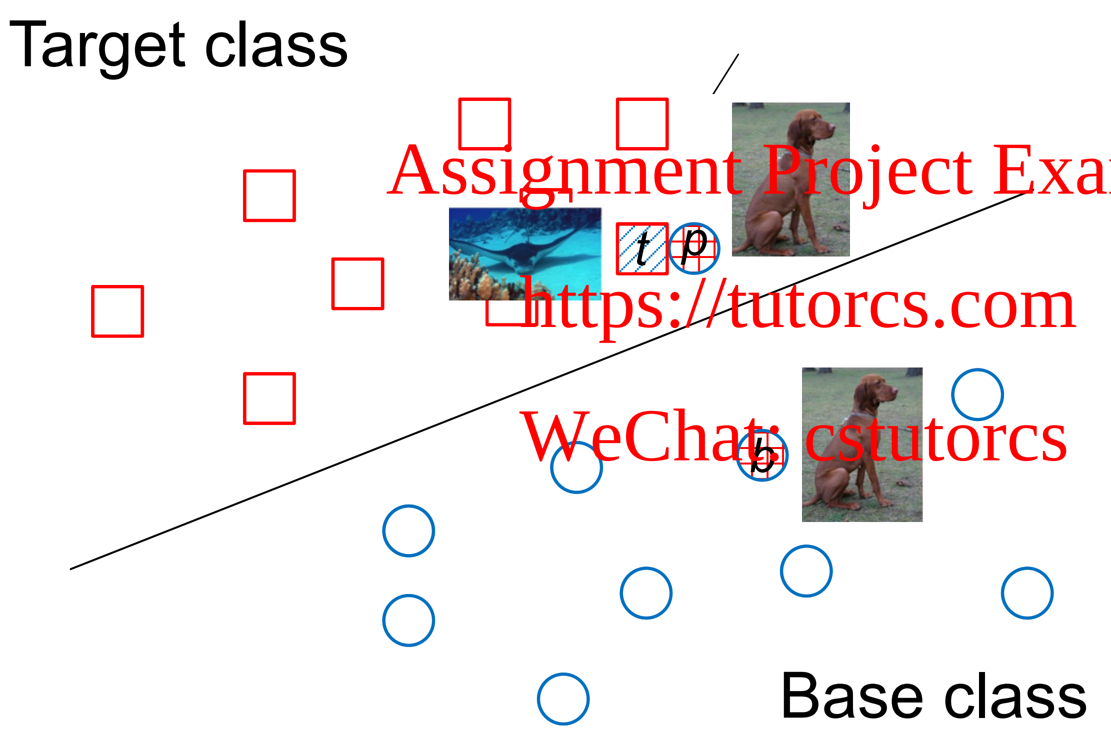
Poisoning attacks

- Poison frog attacks [10]
  - E.g., add a seemingly innocuous image (that is properly labeled) to a training set, and control the identity of a chosen image at test time

Target class

Base class

Poisoning attacks



Step 1: choose an instance from the target class – *t* (target instance)

Step 2: sample an instance from the base class – *b* (base instance)

Step 3: perturb b to create a poison instance – *p*

Step 4: inject *p* into the training dataset

The model is then re-trained. The attack succeeds if the poisoned model labels *t* as the base class

Poisoning attacks

- Generate poison data p
  - Optimisation problem: $p = \underset{x}{\operatorname{argmin}} \|f(x) - f(t)\|_2^2 + \beta\|x - b\|_2^2$
    - $f(x)$: output of the second last layer of the neural network
    - $\|f(x) - f(t)\|_2^2$: makes p move toward the target instance in feature space and get embedded in the target class distribution
    - $\beta\|x - b\|_2^2$: makes p appear like a base class instance to a human labeller

Poisoning attacks

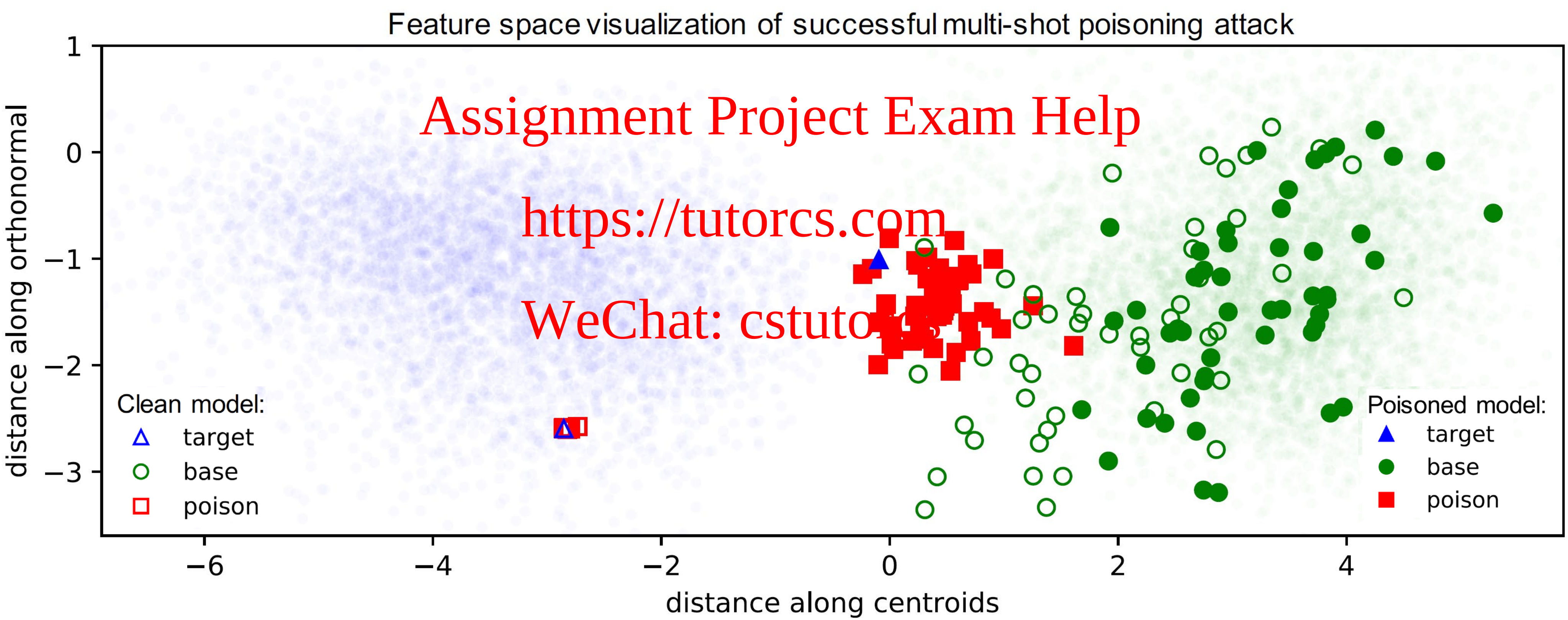
- Forward-backward-splitting iterative procedure [11]
  - Forward step: gradient descent update to minimise the L2 distance to the target instance in feature space
  - Backward step: proximal update that minimises the Euclidean distance from the base instance in input space

Algorithm 1 Poisoning Example Generation

Input: target instance t, base instance b, learning rate λ
Initialize x: $x_0 \leftarrow b$
Define: $\boxed{L_p(x) = \|f(\mathbf{x}) - f(\mathbf{t})\|^2}$
for $i = 1$ **to** $maxIters$ **do**
  Forward step: $\widehat{x_i} = x_{i-1} - \lambda \boxed{\nabla_x L_p(x_{i-1})}$
  Backward step: $\boxed{x_i = (\widehat{x_i} + \lambda\beta b)/(1 + \beta\lambda)}$
end for

Poisoning attacks

- Results



Poisoning attacks

Using Machine Teaching to Identify Optimal Training-Set Attacks on Machine Learners [9]

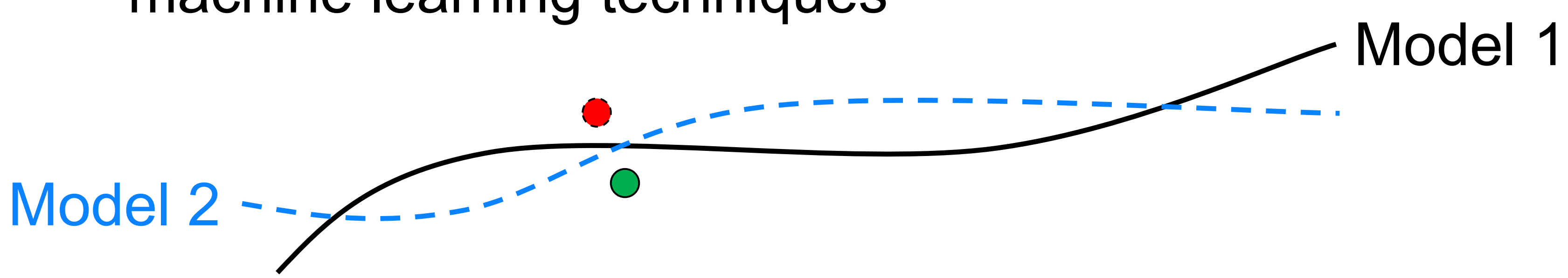
- Attacker's objective : $O_A(D, \hat{\theta}_D) = \|\hat{\theta}_D - \theta^*\| + \|D - D_0\|_2$
  - $\hat{\theta}_D$: parameters of the poisoned model after the attack
  - θ^*: parameters of the attacker's target model, i.e., model that the attacker aims to obtain
  - D: poisoned training data
  - D_0: original training data

Overview

- Week 9: Adversarial Machine Learning – Vulnerabilities
  - Definition + examples
  - Classification
  - Evasion attacks
    - Gradient-descent based approaches
    - Automatic differentiation
    - Real-world example
  - Poisoning attacks
  - Transferability

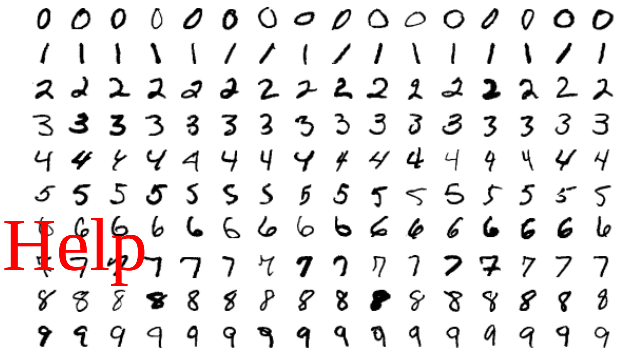
Transferability & Black-box attacks

- Implicit assumption: full knowledge of the target model
- What if the target model is unknown to the attacker?
- Transferability: for two models that perform the same task, trained on different datasets, adversarial samples generated against one model can often fool the other model as well [12][13]
  - Intra-technique: both the target and surrogate model use the same machine learning technique
  - Inter-technique: the target and surrogate model use different machine learning techniques



Transferability & Black-box attacks

- Verification on the MNIST dataset of handwritten digits
  - Grey-scale, 0-255
  - Size: 28px * 28px



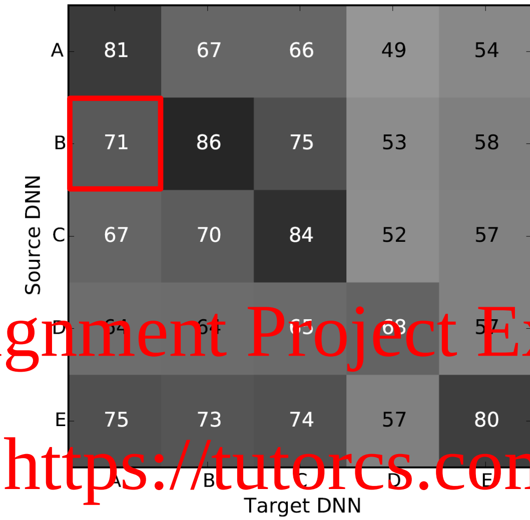
https://upload.wikimedia.org/wikipedia/commons/2/27/MnistExamples.png

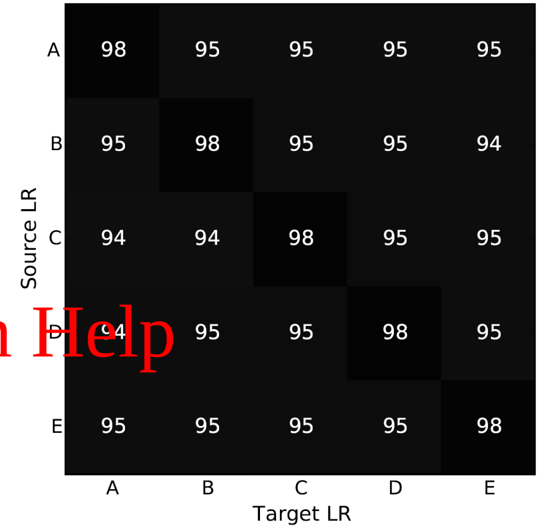
- DNN, SVM, LR, DT, kNN
- Black-box attack
  - Step 1: adversary trains their own model – surrogate/source
  - Step 2: generate adversarial samples against the surrogate
  - Step 3: apply the adversarial samples against the target model

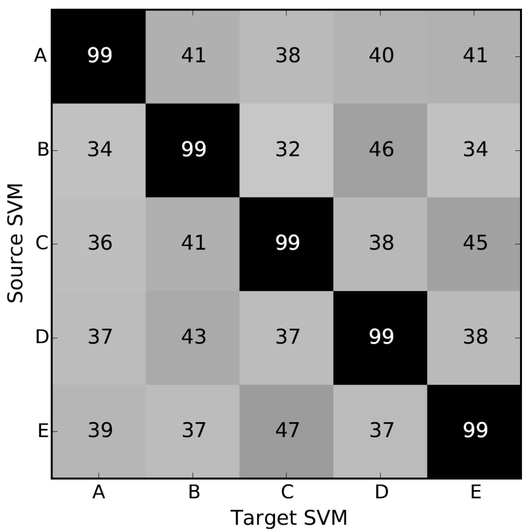
Transferability & Black-box attacks

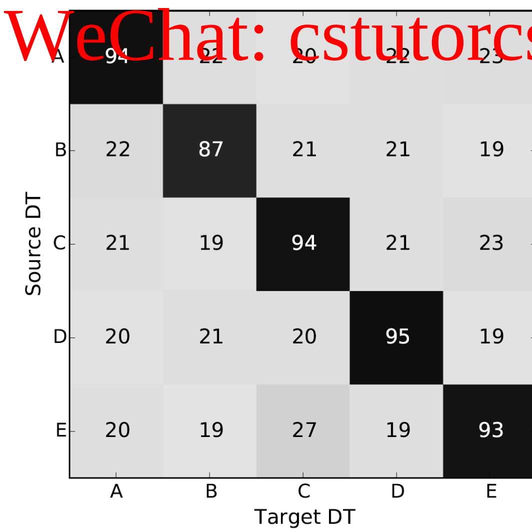
- Intra-technique

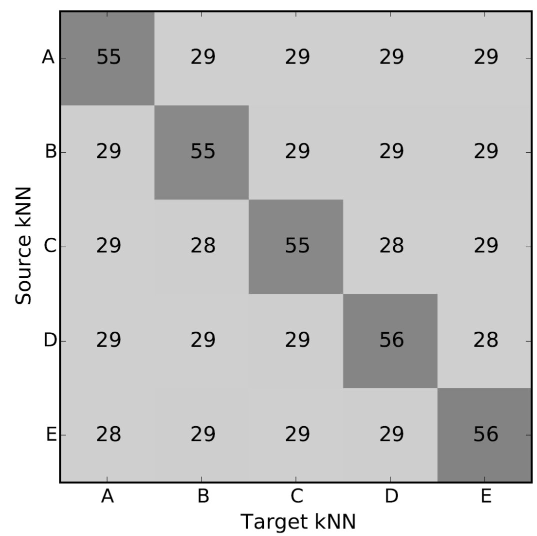
71% adv. samples against the source are also effective against the target





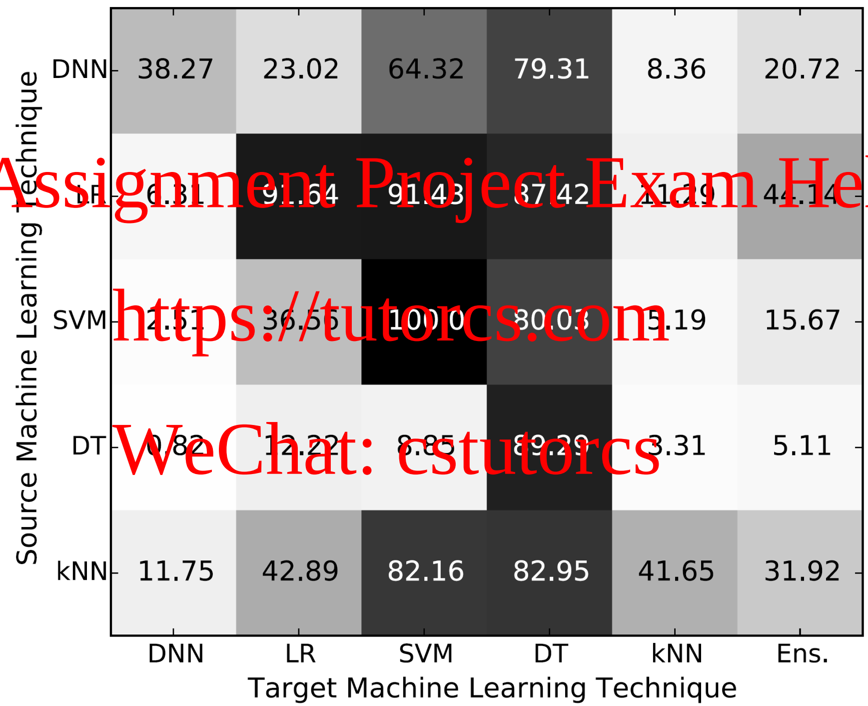






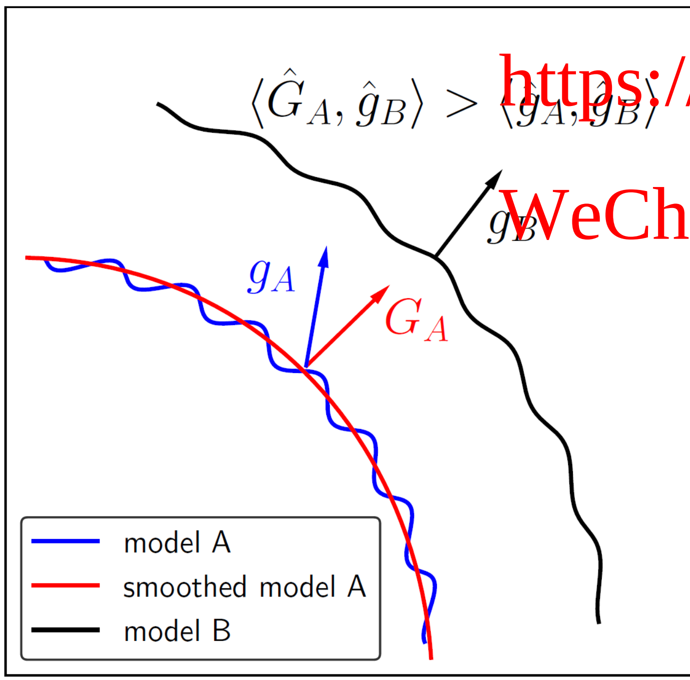
Transferability & Black-box attacks

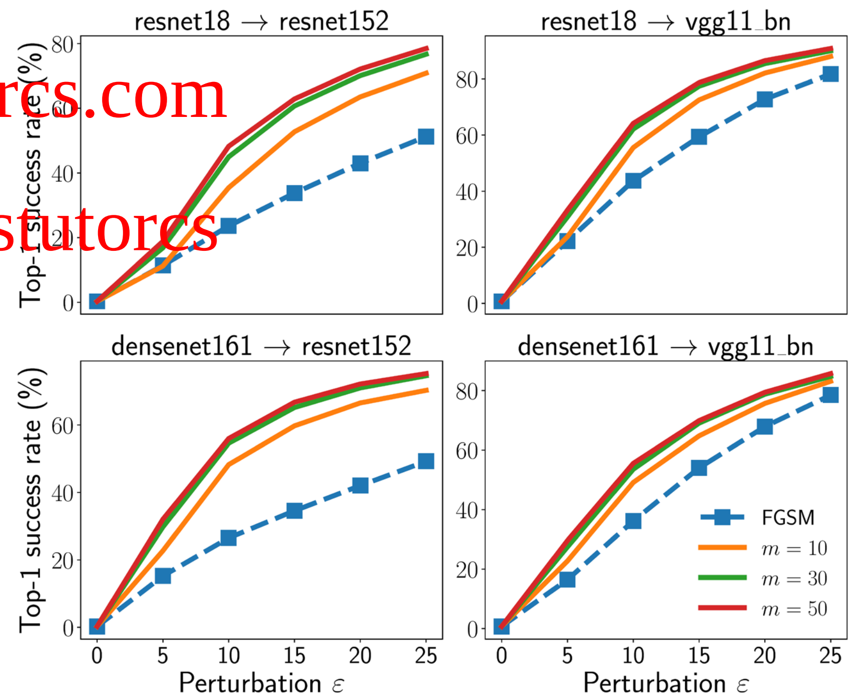
- Inter-technique



Transferability & Black-box attacks

- Non-smoothness can hurt the transferability [7]
  - A is the surrogate model; B is the target model
  - Smoothed loss surface contributes to transferability
  - $x_i \leftarrow x_{i-1} - \alpha \frac{\partial J(x_{i-1})}{\partial x} \Rightarrow x_i \leftarrow x_{i-1} - \alpha \frac{1}{m} \sum_{j=1}^{m} \frac{\partial J(x_{i-1}+\xi_j)}{\partial x}, \xi_j \sim \mathcal{N}(0, \sigma^2)$





Transferability & Black-box attacks

- Input diversity improves transferability [15]
  - Adversarial samples may overfit to the surrogate model
  - Data augmentation
    - Random resizing: resize an input image to a random size
    - Random padding: pad zeros around an image in a random manner

  - Diverse Inputs Iterative Fast Gradient Sign Method (DI^2-FGSM)

$$x_i \leftarrow x_{i-1} - \alpha \cdot \operatorname{sign}\left(\frac{\partial J(x_{i-1})}{\partial x}\right) \Longrightarrow$$

$$x_i \leftarrow x_{i-1} - \alpha \cdot \operatorname{sign}\left(\frac{\partial J(T(x_{i-1};p))}{\partial x}\right), \quad T(x_{i-1};p) = \begin{cases} T(x_{i-1}) & \text{with prob. } p \\ x_{i-1} & \text{with prob. } 1-p \end{cases}$$

  - Momentum Diverse Inputs Iterative Fast Gradient Sign Method (M-DI^2-FGSM)

$$g_i = \mu \cdot g_{i-1} + \frac{\nabla_x J(x_{i-1})}{\|\nabla_x J(x_{i-1})\|_1} \Longrightarrow g_i = \mu \cdot g_{i-1} + \frac{\nabla_x J(T(x_{i-1};p))}{\|\nabla_x J(T(x_{i-1};p))\|_1}$$

Transferability & Black-box attacks

- Backpropagation smoothness [16], backpropagation linearity [17]
  - Non-linear activation functions, e.g., ReLU, sigmoid
  - Non-continuous derivative at zero during backpropagation
  - Continuous derivative property can improve transferability
  - Keep the ReLU function in the forward pass, but during backpropagation approximate the ReLU derivative with a continuous derivative, e.g. using softplus function ($\log(1 + e^x)$)

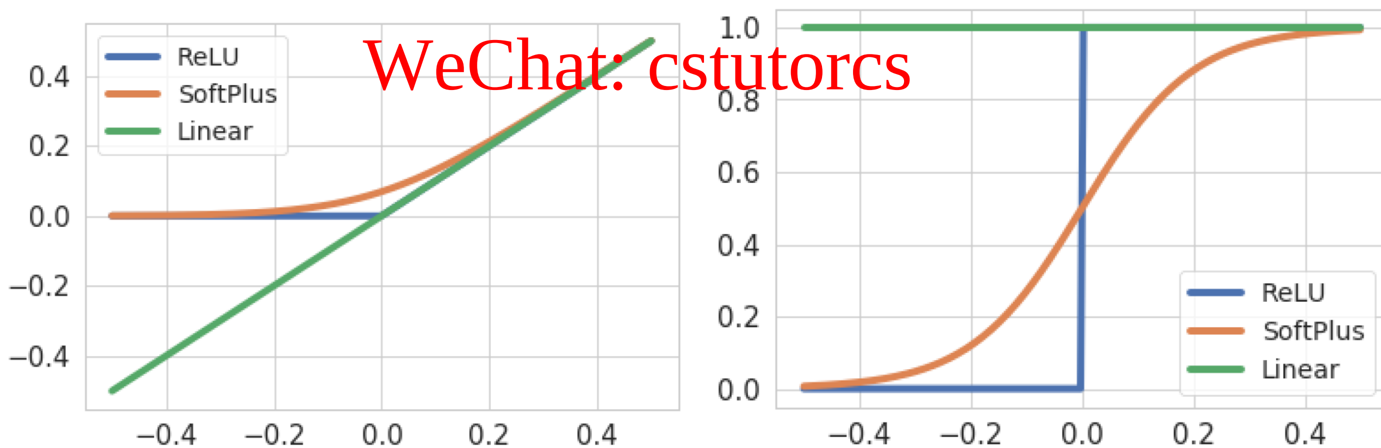


Figure 1. Activation functions (left) and their derivatives (right).

Summary

- Evasion attacks
  - Indiscriminate: $\arg\min_{\delta\in[0,1]^d} \|\delta\| - c \cdot f_{true}(x+\delta)$
  - Targeted: $\arg\min_{\delta\in[0,1]^d} \|\delta\| + c \cdot f_{target}(x+\delta)$
- Poisoning attacks
  - Attacker's objective: $O_A(D, \hat{\theta}_D) = \|\hat{\theta}_D - \theta^*\| + \|D - D_0\|_2$
    - $\hat{\theta}_D$: poisoned model after the attack
    - θ^*: attacker's target, i.e., model that the attacker aims to obtain
    - D: poisoned training data
    - D_0: original training data

- Transferability
  - Intra, inter-technique
  - Black-box attacks

References

- [1] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, “The Security of Machine Learning,” *Machine Learning*, vol. 81, no. 2, pp. 121–148, Nov. 2010.
- **[2] N. Carlini and D. Wagner, “Towards Evaluating the Robustness of Neural Networks,” *eprint arXiv:1608.04644*, 2016.**
- **[3] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and Harnessing Adversarial Examples,” *eprint arXiv:1412.6572*, 2014.**
- [4] A. Kurakin, I. Goodfellow, and S. Bengio, “Adversarial examples in the physical world,” *arXiv preprint arXiv:1607.02533*, 2016.
- [5] A. G. Baydin and B. A. Pearlmutter, “Automatic Differentiation of Algorithms for Machine Learning,” *arXiv:1404.7456 [cs, stat]*, Apr. 2014.
- [6] I. Evtimov *et al.*, “Robust Physical-World Attacks on Machine Learning Models,” *arXiv preprint arXiv:1707.08945*, 2017.
- [7] Wu, L. and Zhu, Z., “Towards Understanding and Improving the Transferability of Adversarial Examples in Deep Neural Networks.” Proceedings of The 12th Asian Conference on Machine Learning:837-850. Available from https://proceedings.mlr.press/v129/wu20a.html.

References

- [8] B. Biggio, B. Nelson, and P. Laskov, "Poisoning Attacks against Support Vector Machines," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, Edinburgh, Scotland, 2012, pp. 1467–1474.
- [9] S. Mei and X. Zhu, "Using Machine Teaching to Identify Optimal Training-set Attacks on Machine Learners," in Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, Texas, 2015, pp. 2871–2877.
- **[10] A. Shafahi et al., "Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks," arXiv:1804.00792 [cs, stat], Apr. 2018.**
- [11] T. Goldstein, C. Studer, and R. Baraniuk, "A field guide to forward-backward splitting with a fasta implementation," arXiv preprint arXiv:1411.3406, 2014
- **[12] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples," *eprint arXiv:1605.07277*, 2016.**
- [13] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples," *eprint arXiv:1602.02697*, 2016.
- [14] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li, "Boosting adversarial attacks with momentum," in CVPR, 2018.

References

- [15] Xie, Cihang and Zhang, Zhishuai and Zhou, Yuyin and Bai, Song and Wang, Jianyu and Ren, Zhou and Yuille, Alan, “Improving Transferability of Adversarial Examples with Input Diversity,” in CVPR, 2019.
- [16] Chaoning Zhang*, Philipp Benz*, Gyusang Cho*, Adil Karjauv, Soomin Ham, Chan-Hyun Youn, In So Kweon, “Backpropagating Smoothly Improves Transferability of Adversarial Examples,” in IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshop on Adversarial Machine Learning, 2021 (*: Equal Contribution)
- [17] Y. Guo, Q. Li, and H. Chen. “Backpropagating linearly improves transferability of adversarial examples,” arXiv preprint arXiv:2012.03528, 2020