智能系统安全实践：对抗样本进阶

复旦白泽智能

系统软件与安全实验室

大纲

- 更多对抗样本生成算法
  - 从FGSM到PGD, JSMA, C&W
  - 有目标对抗样本生成算法
- 实现PGD算法和有目标攻击
- 了解黑盒对抗样本攻击算法

对抗样本：FGSM

■ FGSM攻击

给定一个训练好的模型f_θ

对于**给定的样本x以及对应的标签y**，生成对抗样本：

$$\tilde{x} = x + \epsilon \cdot \text{sign}(\nabla_x \ell(f_\theta(x), y))$$

输入模型，验证$f_\theta(\tilde{x})$的预测结果

$\boldsymbol{x}$
"panda"
57.7% confidence

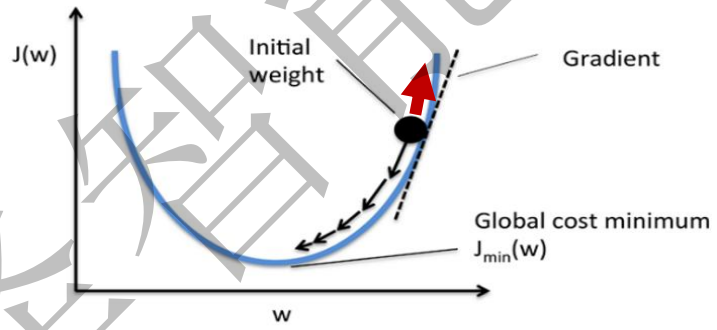
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence

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

对抗样本：FGSM

■ FGSM算法思想

$$\tilde{x} = x + \epsilon \cdot \text{sign}(\nabla_x \ell(f_\theta(x), y))$$



在原样本上对输入**做一步SGD迭代**，等同于在最大化损失函数

存在问题：一步SGD迭代优化效果有限

改进思路：能否**多步SGD迭代**来最大化损失函数？

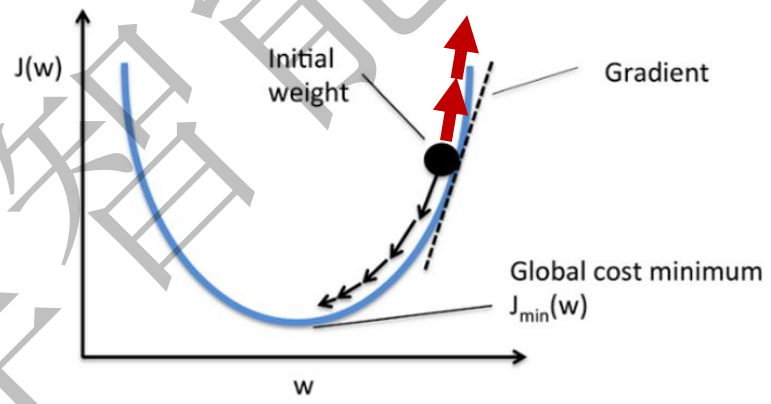
难点：多步迭代后，很可能无法满足对扰动的约束

对抗样本攻击目标：

$$\max_{\delta} \ell(f_\theta(x+\delta), y), \text{ s.t. } \|\delta\|_\infty \leq \epsilon$$

对抗样本：PGD

- Projected Gradient Descent
  每次SGD之后都对扰动**做裁剪**



- 攻击方法
  初始化扰动$\delta_0 = 0$

  for $t = 0, 1, \ldots, T-1$:
  $$\delta_{t+1} = \text{clamp}(\delta_t + \alpha \cdot \text{sign}(\nabla_{x+\delta_t} \ell(f_\theta(x+\delta_t), y)), -\epsilon, \epsilon)$$

α是学习速率，clamp表示把扰动裁剪到$[-\epsilon, \epsilon]$中

FGSM vs PGD

- 对于对抗样本的优化目标

$$\max_{\delta} \ell(f_\theta(x+\delta), y), \quad \text{s.t. } \|\delta\|_\infty \leq \epsilon$$

- FGSM

$$\delta = \epsilon \cdot \text{sign}(\nabla_x \ell(f_\theta(x), y))$$

- PGD

for $t = 0, 1, \ldots, T-1$:

$$\delta_{t+1} = \text{clamp}(\delta_t + \alpha \cdot \text{sign}(\nabla_{x+\delta_t} \ell(f_\theta(x+\delta_t), y)), \ -\epsilon, \epsilon)$$

有目标攻击

- 目前的对抗样本都是无目标攻击

$$\max_{\delta} \ell(f_\theta(x+\delta), y), \quad \text{s.t. } \|\delta\|_\infty \le \epsilon$$

- 能否实现**有目标攻击**？

让模型将添加扰动后的输入$\tilde{x}$分类到$\tilde{y}$

$$\min_{\delta} \ell(f_\theta(x+\delta), \tilde{y}), \quad \text{s.t. } \|\delta\|_\infty \le \epsilon$$

stop sign
Confidence: 0.9153

Adversarial perturbation

flowerpot
Confidence: 0.8374

有目标攻击

■ **攻击方法:** 改变优化方向

有目标攻击 | 无目标攻击

FGSM

$$\tilde{x} = x - \epsilon \cdot \text{sign}(\nabla_x \ell(f_\theta(x), \tilde{y}))$$

$$\delta = -\epsilon \cdot \text{sign}(\nabla_x \ell(f_\theta(x), \tilde{y}))$$

$$\tilde{x} = x + \epsilon \cdot \text{sign}(\nabla_x \ell(f_\theta(x), y))$$

$$\delta = \epsilon \cdot \text{sign}(\nabla_x \ell(f_\theta(x), y))$$

PGD

$$\delta_{t+1} = \text{clamp}(\delta_t - \alpha \cdot \text{sign}(\nabla_{x+\delta_t} \ell(f_\theta(x + \delta_t), \tilde{y})),\ -\epsilon, \epsilon)$$

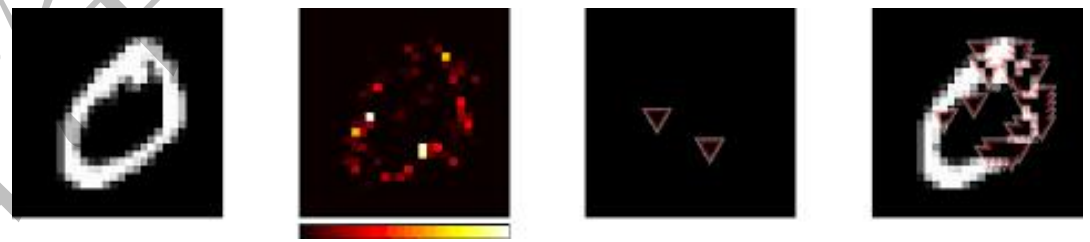
$$\delta_{t+1} = \text{clamp}(\delta_t + \alpha \cdot \text{sign}(\nabla_{x+\delta_t} \ell(f_\theta(x + \delta_t), y)), -\epsilon, \epsilon)$$

Q&A

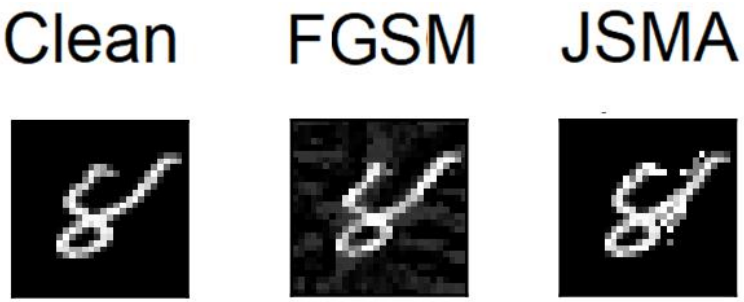
对抗样本：JSMA

- 进一步提升攻击的**隐蔽性**
  Jacobian Saliency Map Attack

  通过**Saliency Map**找到模型分类时关注的区域
  只对这些重要区域进行扰动
  扰动算法可使用FGSM、PGD等



  相比于FGSM，JSMA对于隐蔽性做了进一步的提升



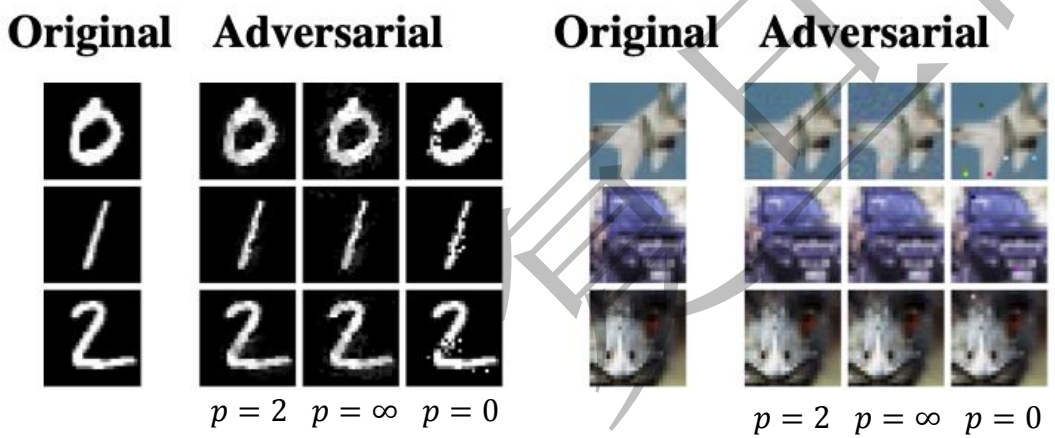
对抗样本：C&W

- 基于优化的扰动方法

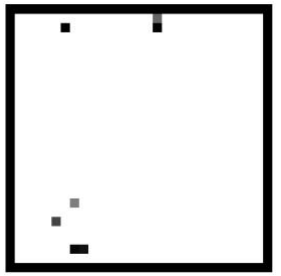
$$\min D(x, x+\delta) \quad \Rightarrow \quad \max_{\delta} \ell(f_\theta(x+\delta), y) + c \cdot \|\delta\|_p$$

$$\text{s.t.} \max_{\delta} \ell(f_\theta(x+\delta), y)$$

对扰动的限制可以有多种形式

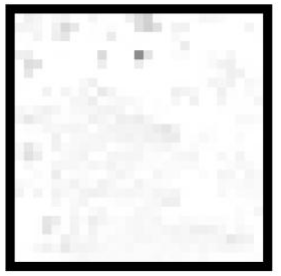


$p = 0$



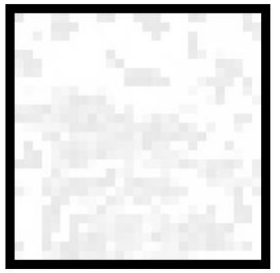
限制扰动位置的个数

$p = 2$



限制扰动大小的总和

$p = \infty$



限制每个扰动的大小

黑盒对抗样本算法

- **在现实情况中**

  攻击者很难拿到模型的代码和参数 ⇨ **黑盒场景**

  e.g., 语音识别、自动驾驶...

- 黑盒场景对抗样本

  攻击者**只能通过查询模型**的方式来实现对抗样本攻击

  攻击者向模型输入x，观察模型的输出值o

  并根据输出值的变化来构造对抗样本

- 黑盒场景**攻击难点**

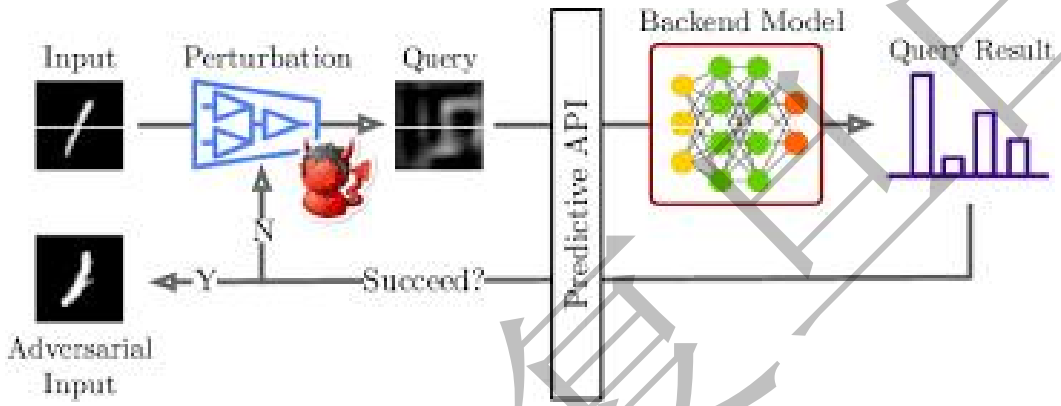
  无法通过loss.backward()来计算梯度信息

黑盒对抗样本算法

- 解决方案1

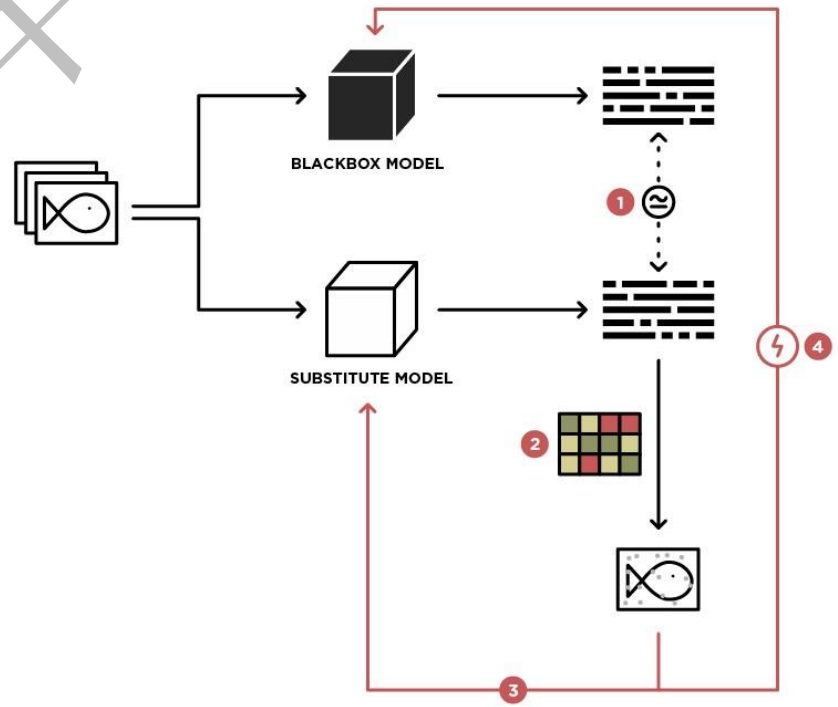
  攻击者利用输入X和输出o
  计算分类损失函数
  并通过**数值微分**来估算梯度值

  $\nabla_X \ell(o, y) \approx ?$



- 解决方案2

  攻击者通过模型蒸馏等方式
  在本地“复制”一个模型



对抗样本攻击：小结

- 核心思想
  - 在输入x上添加微小扰动，导致模型的错误预测
  - 如何生成扰动？ ⇨ FGSM/PGD

- 进阶内容
  - 有目标/无目标对抗样本
  - 更多对扰动的限制（C&W算法）
  - 黑盒/白盒对抗样本

Q&A

实验1：PGD算法

- 在MNIST上实现无目标的PGD攻击算法
  - 在给定测试集上验证模型准确度
  - 与FGSM攻击算法的准确度对比
  - 与FGSM算法可视化的对比

```python
##### Adversarial Attacks #####
def pgd(imgs, epsilon, alpha, iter, model, criterion, labels):
    model.eval()

    adv_xs = imgs.float()
    xs = adv_xs.clone()

    for i in range(iter):
        adv_xs.requires_grad = True

        # TODO: Forward and compute loss, then backward

        # TODO: Retrieve grad and generate adversarial example, note to detach

        # TODO: Clip perturbation

    model.train()

    return adv_xs.detach()
```

$$\delta_{t+1} = \text{clamp}\left(\delta_t + \alpha \cdot \text{sign}\left(\nabla_{x+\delta_t} \ell(f_\theta(x+\delta_t), y)\right), -\epsilon, \epsilon\right)$$

实验2：有目标攻击

- 在MNIST上实现有目标的FGSM和PGD算法
  - 在给定测试集（输入和标签）上验证攻击成功率
  - 与无目标算法可视化对比
  - FGSM与PGD两种算法的效果对比

```python
def fgsm_target(imgs, epsilon, model, criterion, labels):
    model.eval()

    adv_xs = imgs.float()
    adv_xs.requires_grad = True

    # TODO: Forward and compute loss, then backward

    # TODO: Retrieve grad and generate adversarial example, note to detach
    # Note to compute TARGETED loss

    # TODO: Clip perturbation
    # Note the sign of the perturbation

    model.train()

    return adv_xs.detach()
```

$$\delta = -\epsilon \cdot \mathrm{sign}\left(\nabla_x \ell(f_\theta(x), \tilde{y})\right)$$

$$\delta_{t+1} = \mathrm{clamp}\left(\delta_t - \alpha \cdot \mathrm{sign}\left(\nabla_{x+\delta_t} \ell(f_\theta(x+\delta_t), \tilde{y})\right), -\epsilon, \epsilon\right)$$

实验：Bonus

- 实现黑盒对抗样本攻击NES算法
  - Nature Evolutionary Strategies
  - 计算近似后的梯度值
  - 使用FGSM算法生成扰动
  - 与原始FGSM对比效果差异

```python
##### Bonus #####
def nes(imgs, epsilon, model, labels, sigma, n):
    """
    labels: ground truth labels
    sigma: search variance
    n: number of samples used for estimation for each img
    """
    model.eval()

    adv_xs = imgs.reshape(-1, 28 * 28).float()

    grad = torch.zeros_like(adv_xs)
    # TODO: Estimate gradient for each sample adv_x in adv_xs

    adv_xs = adv_xs.detach() - epsilon * grad.sign()
    adv_xs = torch.clamp(adv_xs, min=0., max=1.)

    model.train()

    return adv_xs.detach()
```

Q&A