从局部和全局角度学习通用对抗扰动

1. 项目简介

- 神经网络易受对抗样本的影响，通过向样本中添加人类难以察觉的扰动可以使得模型以高置信度输出错误的类别。先前的图像对抗样本生成方法大多是与图像相关的，有原始图像和模型，每次为输入图像计算扰动，所添加的扰动随输入图像的不同而不同，由于要为每个样本计算扰动，因此在速度上就会比较慢。（补充普通对抗样本和目前通用对抗样本的局限性）
- 局限性：
    - 普通对抗样本
        - 针对的是单张图像生成特定扰动，计算消耗比较大，攻击效果不好。
    - 通用对抗样本
        - 需要大量数据样本，迁移性差，隐蔽性差，攻击效果不佳。
- **在本文中，我们考虑了对抗性攻击的成功率和扰动的不可察觉性，介绍了两种新颖、简单但有效的通用对抗性扰动生成方法。**我们首先主要关注通用对抗样本的攻击成功率，并开发一种基于`优化`的生成方法（SUAN）来实现可见的局部对抗噪声，同时考虑像素强度和扰动量，该方法在有目标攻击和无目标攻击上攻击成功率均高于`95%`，并且在跨数据和跨模型设置中表现良好。然后，考虑到扰动的不可察觉性，我们在此基础上提出了一种结合`深度隐写术（UAP-DS）`的优化算法，将局部对抗性噪声映射到全局扰动。该方法在无目标攻击中达到了相当甚至超越了先前的方法，并且具有良好的可迁移性。最后，我们验证了两种方法产生的通用对抗性扰动的攻击效果，讨论了它们的局限性，并在具体应用场景中对两者进行了权衡。主体实验在`ImageNet`上进行测试，并且，该方法在同一任务下，可以成功地在数据间和模型间迁移。
    - 数据间迁移指的是：不同数据集生成的对抗样本（噪声）可以迁移到目标数据集上。
    - 模型间迁移指的是：同一任务下不同模型生成的对抗样本（噪声）可以迁移到目标模型上，实现黑盒攻击。

2. 项目意义

- **攻击**的角度：我们提出了两种新的、简单有效的通用对抗扰动生成算法，在保证攻击成功率的同时实现了扰动的不可察觉性，同时在有目标攻击和无目标攻击上都有很好的效果。并且有目标攻击可以实现**跨数据**迁移，无目标攻击可以实现**跨模型**迁移。
- 防御角度：对于通用对抗样本的研究，可以快速评估神经网络的鲁棒性和安全性。（因为生成的扰动可以很好地迁移到其他模型中来，无需逐个模型生成，也不用逐样本计算）反过来，对抗扰动的研究也可以用于一些防御场景，利用使用深度学习技术的攻击技术，可以反向防御，例如对抗验证码，向验证码上添加对抗扰动。（浙大纪守领团队）

3. 函数设计

攻击指标

L0范数

- 表示非0元素的个数，对抗样本中表示扰动的非0元素的个数，**修改像素的数量**。

L2范数

- 表示各元素的平方和再开方，对抗样本中表示扰动的各元素的平方和再开平方根；针对图像数据，L2范数越小表示对抗样本人眼越难识别；

L∞范数

- 表示各元素的绝对值的最大值，对抗样本中表示**扰动的各元素的最大值**；

常见的对抗攻击方法

$Method$	$Black/Whitebox$	$Targeted/Non-targeted$	$Perturbation_norm$	Learning
$L-BFGS$	White box	$Non-targeted$	L_∞	$Oneshot$
$FGSM$	White box	$Non-targeted$	L_∞	$Oneshot$
BIM	White box	$Non-targeted$	L_∞	$Iterative$
$JSMA$	White box	$Targeted$	L_0	$Iterative$
$One-pixel$	Black box	$Non-targeted$	L_0	$Iterative$
$C\&W$	White box	$Targeted$	L_0L_2,L_∞	$Iterative$
$DeepFool$	White box	$Non-targeted$	L_2,L_∞	$Iterative$
PGD	White box	$Non-targeted$	L_∞	$Iterative$
UAP	White box	$Non-targeted$	L_2,L_∞	$Iterative$

1. L- BFGS

- Szegedy等人首次证明了可以通过对图像添加小量的人类察觉不到的扰动误导神经网络做出误分类。他们首先尝试`求解让神经网络做出误分类的最小扰动的方程`。

$$\min_p ||p||_2 \tag{1}$$

$$s.t.\, C(I_c+p)=l; I_c+p\in[0,1]^m \tag{2}$$

其中$I_c\in R^m$表示干净图像，p是扰动，l是目标标签。

但由于问题的复杂度太高，他们转而求解简化后的问题，即`寻找最小的损失函数添加项`，使得神经网络做出误分类，这就将问题转化成了凸优化过程，使用L-BFGS来近似它，

$$\min_p c||p||_2+L(I_c+p,l) \tag{3}$$

$$s.t.\, I_c+p\in[0,1]^m \tag{4}$$

- 在攻击成功的情况下，寻找最小的扰动使用L_2度量，基于L-BFGS优化，每生成一个对抗样本都需要优化一遍。
- 基于L-BFGS优化的对抗样本生成算法，每生成一个对抗样本都需要优化一遍，非常耗时。
- 这个方法虽然可靠且稳定有效，但算法的复杂性很高。

- 参考文献：`Szegedy C, Zaremba W, Sutskever I, et al. Intriguing properties of neural networks[J]. arXiv preprint arXiv:1312.6199, 2013.`

2. FGSM(只计算一次梯度叠加到原始数据上)

- Szegedy 等人发现可以通过`对抗训练提高深度神经网络的鲁棒性`，从而提升防御对抗样本攻击的能力。GoodFellow等人开发了一种能有效计算对抗扰动的方法。而求解对抗扰动的方法在原文中就被称为 FGSM，通过梯度来生成对抗噪声，是基于**一步梯度**生成对抗样本。FGSM的全称是Fast Gradient Sign Method(快速梯度下降法），在白盒环境下，通过求出模型对输入的导数，然后用符号函数得到其具体的梯度方向，接着乘以一个步长，得到的“扰动”加在原来的输入上就得到了在FGSM攻击下的样本。
- FGSM是一种一次攻击，即针对一张图加梯度也仅仅增一次梯度。
- 这里采用梯度方向而不是采用梯度值是为了控制扰动的L_∞距离。
- 无目标攻击

$$x' = x + \varepsilon \cdot sign(\bigtriangledown J(I_c, l, \theta)) \tag{5}$$

这里的加号是为了最大化预测标签和原始标签之前的距离，梯度大于0，符号函数取值为1，是为了在梯度方向上添加扰动，影响模型　预测。

- 有目标攻击

$$x' = x - \varepsilon \cdot sign(\bigtriangledown J(I_c, l', \theta)) \tag{6}$$

l'是特定标签，最小化原始标签和特定标签之间的距离。

- 符号函数

$$sign(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases} \tag{7}$$

- 为什么这样做有攻击效果?
  - 攻击成功是模型分类错误，就模型而言，就是加了扰动的样本使得模型的loss增大。而**所有基于梯度的攻击方法都是基于让loss增大这一点来做的。**可以仔细回忆一下，在神经网络的反向传播当中，我们在训练过程时就是沿着梯度方向来更新更新w，b的值。这样做可以使得网络往loss减小的方向收敛。那么现在我们既然是要使得loss增大，而模型的网络系数又固定不变，唯一可以改变的就是输入，因此我们就利用loss对输入求导从而“更新”这个输入。(当然，肯定有人问，神经网络在训练的时候是多次更新参数，这个为什么仅仅更新一次呢？主要因为我们希望产生对抗样本的速度更快，毕竟名字里就有“fast”，当然了，多次迭代的攻击也有，后来的PGD（又叫I-FGSM)以及MIM都是更新很多次，虽然攻击的效果很好，但是速度就慢很多了）。
- 为什么不直接使用导数，而要用符号函数求得其方向?
  - 这个问题我也一直半知半解，我觉得应该是如下两个原因：
    1. FGSM是典型的无穷范数攻击，那么我们在限制扰动程度的时候，只需要使得最大的扰动的绝对值不超过某个阀值即可。而我们对输入的梯度，对于大于阀值的部分我们直接clip到阀值，对于小于阀值的部分，既然对于每个像素扰动方向只有一两个方向，而现在方向已经定了，那么为什么不让其扰动的程度尽量大呢？因此对于小于阀值的部分我们就直接给其提升到阀值，这样一来，相当于我们给梯度加了一个符号函数了。
    2. 由于FGSM这个求导更新只进行一次，如果直接按值更新的话，可能生成的扰动改变就很小，无法达到攻击的目的，因此我们只需要知道这个扰动大概的方向，至于扰动多少我们就可以自己来设定了。

- 参考文献：`Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy,Explaining and Harnessing Adversarial Examples,ICLR 2015`

3. BIM（迭代FGSM（I-FGSM））

- one-step 方法通过一大步运算增大分类器的损失函数而进行图像扰动，因而可以直接将其扩展为通过多个小步增大损失函数的变体，从而我们得到 Basic Iterative Methods（BIM）。
- FGSM 算法把每个像素点都变化了ε这么大，因为每个像素都动了。如果优化目标函数在局部区间是非线性的，那么沿$\bigtriangledown loss_{F,t}(x)$这个方向进行大步长的优化大概率会出现错误。一种方法是把优化区间变小，即假设优化的目标函数在很小的区间内是线性的，那么在这个区间内采用FGSM的优化算法是可行的，因此提出了BIM算法。
- BIM算法提出迭代的方式来找各个像素点的扰动，而不是一次性所有像素都改那么多，即迭代的FGSM，像素范围在$(0, \varepsilon \cdot sign(\bigtriangledown J(I_c, l', \theta)))$之间。

$$X'_{N+1} = clip[X_N + \varepsilon \cdot sign(\bigtriangledown J(X_N, l))] \tag{8}$$

ε是一个很小的值，一般和迭代次数相关，迭代次数越大，区间就越小，ε就越小。通过这种迭代生成的对抗样本攻击性能更好，但在 攻击的迁移性上会变差。

迭代的含义：每次在上一步的对抗样本基础上，各个像素增长ε（减少），然后再进行裁剪，保证新样本的各个像素都在x的ε邻域 内。这种迭代的方法是有可能在各个像素变化小于ε的情况下找到对抗样本，如果找不到，最差的效果和FGSM一样。

- 裁剪：在迭代更新的过程中，随着迭代次数的增加，样本的部分像素值可能会溢出，这时需将这些值用0或1代替，最后才能生成有效的图像。该过程确保了新样本的各个像素在原样本各像素的某一邻域内，不至于图像失真。
- 参考文献：`Kurakin A, Goodfellow I, Bengio S. Adversarial examples in the physical world[J]. 2016.`

4. JSMA

- JSMA提出了限制L_0范数的方法，即**仅改变几个像素的值，而不是扰动整张图，像素的数据有一个阈值。**
- JSMA算法的灵感来自于计算机视觉领域的显著图。简单来说，就是不同输入特征对分类器产生不同输出的影响程度不同。如果我们发现某些特征对应着分类器中某个特定的输出，我们可以通过在输入样本中增强这些特征来使得分类器产生指定类型的输出。JSMA算法主要包括三个过程：`计算前向导数，计算对抗性显著图，添加扰动`，以下给出具体解释。
- 所谓前向导数，其实是计算神经网络最后一层的每一个输出对输入的每个特征的偏导。以MNIST分类任务为例，输入的图片的特征数（即像素点）为784，神经网络的最后一层一般为10个输出（分别对应0-9分类权重），那对于每一个输出我们都要分别计算对784个输入特征的偏导，所以计算结束得到的前向导数的矩阵为（10，784）。前向导数标识了每个输入特征对于每个输出分类的影响程度，其计算过程也是采用链式法则。这里需要说明一下，前面讨论过的FGSM和DeepFool不同在计算梯度时，是通过对损失函数求导得到的，而JSMA中前向导数是通过对神经网络最后一层输出求导得到的。前向导数∇**F(X)**∇F(X)具体计算过程如下所示，`j`表示对应的输出分类，`i`表示对应的输入特征。

$$\nabla \mathbf{F}(\mathbf{X}) = \frac{\partial \mathbf{F}(\mathbf{X})}{\partial \mathbf{X}} = \left[\frac{\partial \mathbf{F_j}(\mathbf{X})}{\partial x_i}\right]_{i \in 1 \ldots M, j \in 1..N}$$

$$\frac{\partial \mathbf{F_j}(\mathbf{X})}{\partial x_i} = \left(\mathbf{W}_{n+1,j} \cdot \frac{\partial \mathbf{H_n}}{\partial x_i}\right) \times \frac{\partial f_{n+1,j}}{\partial x_i}\left(\mathbf{W}_{n+1,j} \cdot \mathbf{H_n} + b_{n+1,j}\right)$$

- 通过得到的前向导数，我们可以计算其对抗性显著图，即对分类器特定输出影响程度最大的输入。首先，根据扰动方式的不同（正向扰动和反向扰动），作者提出了两种计算对抗性显著图的方式，即：

$$S(\mathbf{X}, t)[i] = \begin{cases} 0 & \text{if } \frac{\partial \mathbf{F_t}(\mathbf{X})}{\partial \mathbf{X}_i} < 0 \text{ or } \sum_{j \neq t} \frac{\partial \mathbf{F}_j(\mathbf{X})}{\partial \mathbf{X}_i} > 0 \\ \left(\frac{\partial \mathbf{F_t}(\mathbf{X})}{\partial \mathbf{X_i}}\right) \left| \sum_{j \neq t} \frac{\partial \mathbf{F_j}(\mathbf{X})}{\partial \mathbf{X_i}} \right| & \text{otherwise} \end{cases}$$

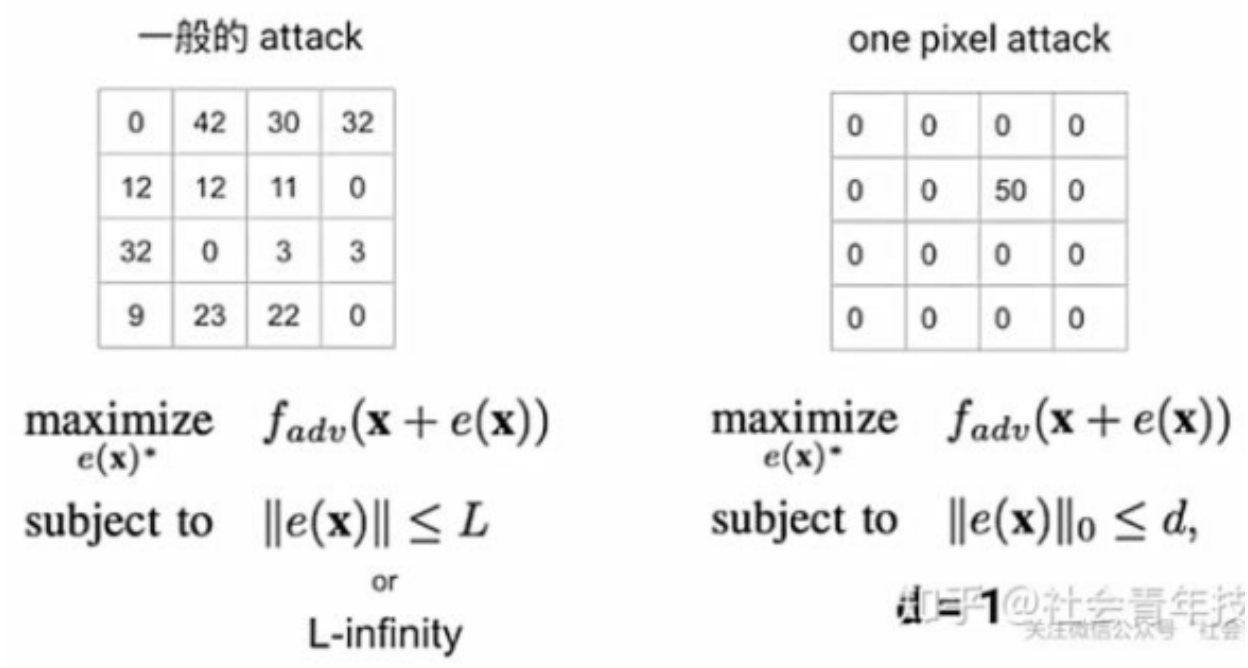
$$S(\mathbf{X}, t)[i] = \begin{cases} 0 & \text{if } \frac{\partial \mathbf{F}_t(\mathbf{X})}{\partial \mathbf{X}_i} > 0 \text{ or } \sum_{j \neq t} \frac{\partial \mathbf{F}_j(\mathbf{X})}{\partial \mathbf{X}_i} < 0 \\ \left(\frac{\partial \mathbf{F}_t(\mathbf{X})}{\partial \mathbf{X}_i}\right) \left| \sum_{j \neq t} \frac{\partial \mathbf{F}_j(\mathbf{X})}{\partial \mathbf{X}_i} \right| & \text{otherwise} \end{cases}$$

找到单个满足要求的特征很困难，所以作者提出了另一种解决方案，通过对抗性显著图寻找对分类器特定输出影响程度最大的输入特　征对，即每次计算得到两个特征。

- 根据对抗性显著图所得到的特征，可以对其添加扰动。扰动方式包括正向扰动和反向扰动（+θ或-θ）。如果添加的扰动不足以使分类结果发生转变，我们利用扰动后的样本可以重复上述过程（计算前向导数->计算对抗性显著图->添加扰动)。这个过程需要注意两点
  - 扰动过程的重复次数需要被约束，即修改的特征数有限；
  - 一旦添加扰动后，该特征达到临界值，那么该特征不再参与扰动过程；
- 缺点：只能进行有目标攻击，不能实现无目标攻击，而且攻击需要指定方向（增加/减少像素）。
- 参考文献：`Papernot N, McDaniel P, Jha S, et al. The limitations of deep learning in adversarial settings[C]//2016 IEEE European symposium on security and privacy (EuroS&P). IEEE, 2016: 372-387.`

5. One-pixel attack（黑盒，L_0）

- 这是一种极端的对抗攻击方法，**仅改变图像中的一个像素值**就可以实现对抗攻击。Su 等人使用了**差分进化算法**，对每个像素进行迭代地修改生成子图像，并与母图像对比，根据选择标准保留攻击效果最好的子图像，实现对抗攻击。这种对抗攻击**不需要知道网络参数或梯度的任何信息**。



- 令f表示分类器，输入为$x = (x_1, \ldots, x_n)$，原始标签为t，在类别t处的概率为$f_t(x)$，扰动为$e(x) = (e_1, \ldots, e_n)$，目标类别为$adv$，其最大限制值为$L$，则生成对抗样本的优化问题描述为：

$$max_{e(x)} f_{adv}(x + e(x)) \tag{9}$$

$$s.t. ||e(x)|| <= L \tag{10}$$

  该问题主要是寻找两个值：1. 需要扰动哪个维度的值；2. 每个维度需要扰动多大的值。

  可以修改为：

$$max_{e(x)} f_{adv}(x + e(x)) \tag{11}$$

$$s.t. ||e(x)||_0 <= d \tag{12}$$

  在one-pixel attack中$d = 1$。可以看作是在一个数据点的n维空间中的一维的方向上进行移动。

- 差分进化算法
  - 差分进化算法是一种解决复杂多模态优化问题的优化算法。
  - 在每次的迭代过程中，根据当前总体（父项）生成另一组候选解决方案（子项）。然后将这些孩子与他们相应的父母进行比较，如果他们比他们的父母更适合（拥有更高的适应值），他们就可以存活下来。这样，只有将父母和孩子进行比较，才能同时达到保持多样性和提高适应值的目的。
  - 使用差分进化算法生成对抗样本的优点：
    - **Higher probability of Finding Global Optima 找到全局最优解的概率较高**。DE是一种元启发式算法，与梯度下降或贪婪搜索算法相比，它相对较少受到局部极小的影响（这部分是由于多样性保持机制和一组候选解的使用）。此外，本文考虑的问题有一个严格的约束（只能修改一个像素），这使得它相对困难。
    - **Require Less Information from Target System 需要较少的目标系统的信息**。DE不要求优化问题如梯度下降法、拟牛顿法等经典优化方法所要求的那样是可微的。这在生成敌对图像的情况下是至关重要的，因为，1）有些网络是不可微的。2） 计算梯度需要更多关于目标系统的信息，这在很多情况下是不现实的。
    - **Simplicity 简单**。这里提出的方法与使用的分类器无关。要使攻击发生，只需知道概率标签就足够了。

- 方法
  - 我们将扰动编码成一个矩阵（候选解），矩阵通过差分进化进行优化（进化）。一个候选解包含固定数量的扰动，每个扰动是一个包含五个元素的元组：x-y坐标和扰动的RGB值。一个扰动修改一个像素。候选解（总体）的初始数目为400，在每次迭代中，将使用通常的DE公式生成另外400个候选解（子解）：

$$x_i(g+1) = x_{r1}(g) + F(x_{r2}(g) - x_{r3}(g)) \tag{13}$$

$$r_1 \neq r_2 \neq r_3 \tag{14}$$

    其中x_i是候选解的某个元素，r_1, r_2, r_3表示随机数，F是一个范围参数，设为0.5，g表示目前迭代的index。
  - 在每一次迭代后，我们会将每一个候选解，与其对应的父解进行对比，胜者进入下一轮迭代过程。
- 使用差分进化算法的优点：
  - 有更高的几率可以找到全局最优点。
  - 需要目标模型很少的信息。
- 参考文献：`Su J, Vargas D V, Sakurai K. One pixel attack for fooling deep neural networks[J]. IEEE Transactions on Evolutionary Computation, 2019, 23(5): 828-841.`

6. C&W

- Carlini 和 Wagner提出了三种对抗攻击方法，通过限制 l_∞、l_2 和l_0 范数使得扰动无法被察觉。实验证明 defensive distillation 完全无法防御这三种攻击。该算法生成的对抗扰动可以从 unsecured 的网络迁移到 secured 的网络上，从而实现黑箱攻击。
- CW攻击的原理
  - CW是一种基于优化的攻击。攻击算法的公式表达：

$$r_n = \frac{1}{2}(\tanh(\omega_n) + 1) - X_n$$

$$\min_{\omega_n} \|r_n\| + c \cdot f(\frac{1}{2}(\tanh(\omega_n) + 1))$$

$$\text{Where } f(x') = \max(\max\{Z(x')_i : i \neq t\} - Z(x')_t, -k)$$

  - 该算法将对抗样本当成一个变量，那么现在如果要使得攻击成功就要满足两个条件：（1）对抗样本和对应的干净样本应该差距越小越好；（2）对抗样本应该使得模型分类错，且错的那一类的概率越高越好。
  - 其实上述公式的两部分loss也就是基于这两点而得到的，首先说第一部分，r_n对应着干净样本和对抗样本的差，但作者在这里有个小trick，他把对抗样本映射到了$tanh$空间里面，$-1 \leq tanh(w_i) \leq 1$，所以$0 \leq x_i + \delta_i \leq 1$是成立的，这样的转化允许我们使用其他不支持盒约束的算法进行优化。这样做有什么好处呢？这里将 ω 变量用来代替原来的样本，主要就是将样本映射到tanh空间，可以在$(-\infty, +\infty)$ 中进行变换，如果不做变换，那么x只能在$(0，1)$这个范围内变换，做了这个变换，x可以在$-inf$到 $+inf$ 做变换，有利于优化。但论文中并没有说明ω是怎么来的，源代码中，作者将原样本做$arctan(2x-1)$ 变换生成ω，那为什么要这么做呢，我们看这个等式，加入噪声

$$\begin{aligned}\delta &= \frac{1}{2}(tanh(w)+1) \\ &= \frac{1}{2}(tanh(\arctan(2x-1))+1) \\ &= x\end{aligned} \tag{15}$$

所以我们只需要将w中加入一点点噪声就生成了对抗样本。

- 再来说说第二部分，公式中的Z(x)表示的是样本x通过模型未经过softmax的输出向量，对于干净的样本来说，这个向量的最大值对应的就是正确的类别（如果分类正确的话），现在我们将类别t（也就是我们最后想要攻击成的类别）所对应的逻辑值记为$Z(x')_t$，将最大值（对应类别不同于t）记为$max(Z(x')_i : i \neq t)$，如果通过优化使得$max(Z(x')_i : i \neq t) - Z(x')_t$变小，攻击不就离成功更近了嘛。那么式子中的k是什么呢？k其实就是置信度，可以理解为k越大，那么模型分错，且错成的那一类的概率越大。但与此同时，这样的对抗样本就更难找了。最后就是常数c，这是一个超参数，用来权衡两个loss之间的关系，在原论文中，作者使用二分查找来确定c值。

- CW是一个基于优化的攻击，主要调节的参数是c和k，看你自己的需要了。它的**优点**在于，可以调节置信度，生成的扰动小，可以破解很多的防御方法，**缺点**是很慢。最后在说一下，就是在某些防御论文中，它实现CW攻击，是直接用$f(\frac{1}{2}(tanh(w_n)+1))$替换PGD中的loss，其余步骤和PGD一模一样。

7. DeepFool

- Moosavi-Dezfooli 等人通过迭代计算的方法**生成最小规范对抗扰动**，将位于分类边界内的图像逐步推到边界外，直到出现错误分类。作者证明他们生成的扰动比 FGSM 更小，同时有相似的欺骗率。
- 鲁棒性
  - 给定一个分类器，样本鲁棒性是使得模型出现误分类的最小扰动，具体形式如下：

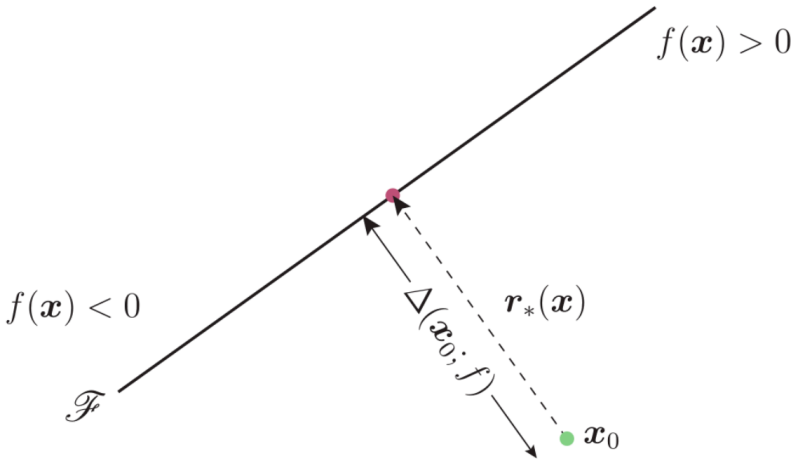
$$\Delta(\boldsymbol{x};\hat{k}) := \min_{\boldsymbol{r}} \|\boldsymbol{r}\|_2 \text{ subject to } \hat{k}(\boldsymbol{x}+\boldsymbol{r}) \neq \hat{k}(\boldsymbol{x})$$

其中，x为干净的样本，$\hat{k}(x)$为模型预测的标签。 $\triangle(x;\hat{k})$为样本 x在模型分类器$\hat{k}$的鲁棒性。进而作者又定义出了模型在整个数据集上的鲁棒性，具体形式为：

$$\rho_{\text{adv}}(\hat{k}) = \mathbb{E}_{\boldsymbol{x}} \frac{\Delta(\boldsymbol{x};\hat{k})}{\|\boldsymbol{x}\|_2}$$

作者的这种定义是在分母中都除以一个样本的 2 范数。

- 攻击二分类器



上图为对抗样本攻击线性分类器的图示。其中$f(x) = w^T x + b$为一个二分类器。$\triangle(x_0; f)$为干净样本点x_0的最短距离，即为样本点x_0在分类器f中的鲁棒性。

目标函数：

$$\begin{aligned}\boldsymbol{r}_*(\boldsymbol{x}_0) &:= \arg\min \|\boldsymbol{r}\|_2 \\ &\text{s.t. } \operatorname{sign}(f(\boldsymbol{x}_0 + \boldsymbol{r})) \neq \operatorname{sign}(f(\boldsymbol{x}_0)) \\ &= -\frac{f(\boldsymbol{x}_0)}{\|\boldsymbol{w}\|_2^2}\boldsymbol{w}\end{aligned}$$

可转换为如下的优化形式：

$$\arg\min_{\boldsymbol{r}_i} \|\boldsymbol{r}_i\|_2 s.t.\ f(\boldsymbol{x}_i) + \nabla f(\boldsymbol{x}_i)^T \boldsymbol{r}_i = 0$$

Algorithm 1 DeepFool for binary classifiers

input: Image $\boldsymbol{x}$, classifier f.
output: Perturbation $\hat{\boldsymbol{r}}$.
Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
while $\operatorname{sign}(f(\boldsymbol{x}_i)) = \operatorname{sign}(f(\boldsymbol{x}_0))$ **do**
    $\boldsymbol{r}_i \leftarrow -\frac{f(\boldsymbol{x}_i)}{\|\nabla f(\boldsymbol{x}_i)\|_2^2}\nabla f(\boldsymbol{x}_i)$,
    $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$,
    $i \leftarrow i + 1$.
end while
return $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$.

- 攻击多分类器
  - 分类器预测标签：

$$\hat{k}(x) = argmax f_k(x) \tag{16}$$

其中，$f_k(x)$是预测概率向量$f(x)$的第k类的概率分量。多分类器模型误分类的优化函数：

$$\arg\min_{r} \|r\|_2$$

$$\text{s.t. } \exists k : w_k^\top (x_0 + r) + b_k \geq w_{k(x_0)}^\top (x_0 + r) + b_{k(x_0)}$$

多分类的对抗扰动为：

$$r_*(x_0) = \frac{\left| f_{\hat{l}(x_0)}(x_0) - f_{\hat{k}(x_0)}(x_0) \right|}{\left\| w_{\hat{l}(x_0)} - w_{\hat{k}(x_0)} \right\|_2^2} \left(w_{\hat{l}(x_0)} - w_{\hat{k}(x_0)} \right)$$

也就是样本分别到各个边界的最短距离。

Algorithm 2 DeepFool: multi-class case

input: Image x, classifier f.
output: Perturbation $\hat{r}$.

Initialize $x_0 \leftarrow x$, $i \leftarrow 0$.
while $\hat{k}(x_i) = \hat{k}(x_0)$ **do**
  for $k \neq \hat{k}(x_0)$ **do**
    $w'_k \leftarrow \nabla f_k(x_i) - \nabla f_{\hat{k}(x_0)}(x_i)$
    $f'_k \leftarrow f_k(x_i) - f_{\hat{k}(x_0)}(x_i)$
  end for
  $\hat{l} \leftarrow \arg\min_{k \neq \hat{k}(x_0)} \frac{|f'_k|}{\|w'_k\|_2}$
  $r_i \leftarrow \frac{|f'_{\hat{l}}|}{\|w'_{\hat{l}}\|_2^2} w'_{\hat{l}}$
  $x_{i+1} \leftarrow x_i + r_i$
  $i \leftarrow i + 1$
end while
return $\hat{r} = \sum_i r_i$

- 样本离分类边界越远，样本的L_2范数越小，评测数值越鲁棒。

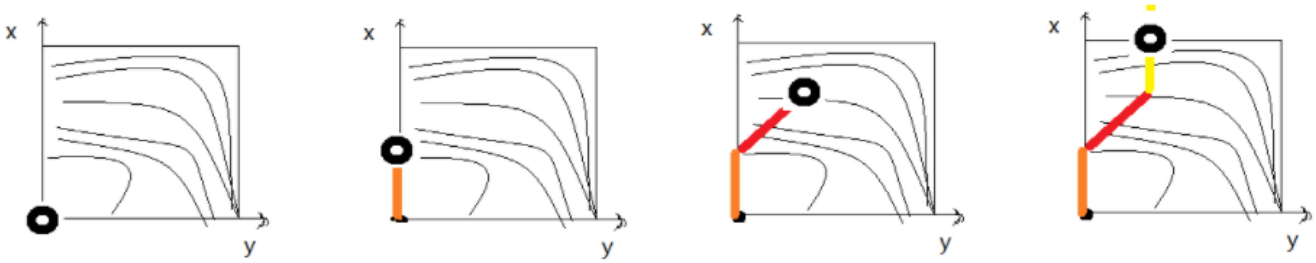
- FGSM算法能够快速简单的生成对抗性样例，但是它没有对原始样本扰动的范围进行界定（扰动程度ϵ是人为指定的），我们希望通过最小程度的扰动来获得良好性能的对抗性样例。2016年，Seyed等人提出的DeepFool算法很好的解决了这一问题。文章的核心思想是希望找到一种对抗性扰动的方法来作为对不同分类器对对抗性扰动鲁棒性评估的标准。简单来说就是，现在我需要两个相同任务的分类器A、B针对同一个样本生成各自的对抗性样例。对于分类器A而言，其生成对抗性样例所需要添加的最小扰动为a；对于分类器B而言，其生成对抗性样例所需要添加的最小扰动为b；通过对a、b的大小进行比较，我们就可以对这两个分类器对对抗性样例的鲁棒性进行评估。由于FGSM产生扰动是人为界定的，所以它不能作为评估的依据。DeepFool可以生成十分接近最小扰动的对抗性样例，因此它可以作为衡量分类器鲁棒性的标准。

8. PGD

- PGD（Project Gradient Descent)攻击是一种**迭代攻击**，可以看作是FGSM的翻版—K-FGSM (K表示迭代的次数），大概的思路就是，FGSM是仅仅做一次迭代，走一大步，而PGD是做多次迭代，每次走一小步，每次迭代都会将扰动clip到规定范围内。

$$x_{t+1} = \Pi_{x+S}(x_t + \alpha \cdot sign(\bigtriangledown_x J(x_t, y))) \tag{17}$$

一般来说，PGD的攻击效果比FGSM要好，首先，如果目标模型是一个线性模型，那么用FGSM就可以了，因为此时loss对输入的导数是固定的，换言之，使得loss下降的方向是明确的，即使你多次迭代，扰动的方向也不会改变。而对于一个非线性模型，仅仅做一次迭代，方向是不一定完全正确的，这也是为什么FGSM的效果一般的原因了。



用画图软件画了一个很丑的图，但大致能够表达我的看法，黑圈是输入样本，假设样本只有两维，那么样本可以改变的就有八个方向，坐标系中显示了loss等高线,以及可以扰动的最大范围（因为是无穷范数，所以限制范围是一个方形，负半轴的范围没有画出来），黑圈每一次改变，都是以最优的方向改变，最后一次由于扰动超出了限制，所以直接截断，如果此时迭代次数没有用完，那么就在截断处继续迭代，直到迭代次数用完。

- PGD攻击是最强的一阶攻击，以均匀的随机噪声作为初始化，如果防御方法对这个攻击能够有很好的防御效果，那么其他攻击也不在话下了。
- PGD和BIM的区别就是PGD增加迭代轮数，并且增加了一层随机化处理。

9. Universal Adversarial Perturbations

- 诸如 FGSM、DeepFool等方法只能生成单张图像的对抗扰动，而 Universal Adversarial Perturbations能生成对任何图像实现攻击的扰动，这些扰动同样对人类是几乎不可见的。该论文中使用的方法和 DeepFool 相似，都是用对抗扰动将图像推出分类边界，不过同一个扰动针对的是所有的图像。虽然文中只针对单个网络 ResNet 进行攻击，但已证明这种扰动可以泛化到其它网络上。universal是指同一个扰动加入到不同的图片中，能够使图片被分类模型误分类，而不管图片到底是什么。
- 特点：
  - 扰动与输入图片无关，仅与模型本身有关。
  - 具有小的范数，从而不改变图片本身的结构。即不用像之前的方法如FGSM一样，要针对每一个样本进行梯度计算求得扰动。
- 形式化定义：
  - 对于d维度数据分布，里面的每一个样本$x \in R_d$，存在一个分类器$k(x) \rightarrow [1, \ldots, k]$，$v$是一个扰动，满足：

$$\hat{k}(x+v) \neq \hat{k}(x) \tag{18}$$

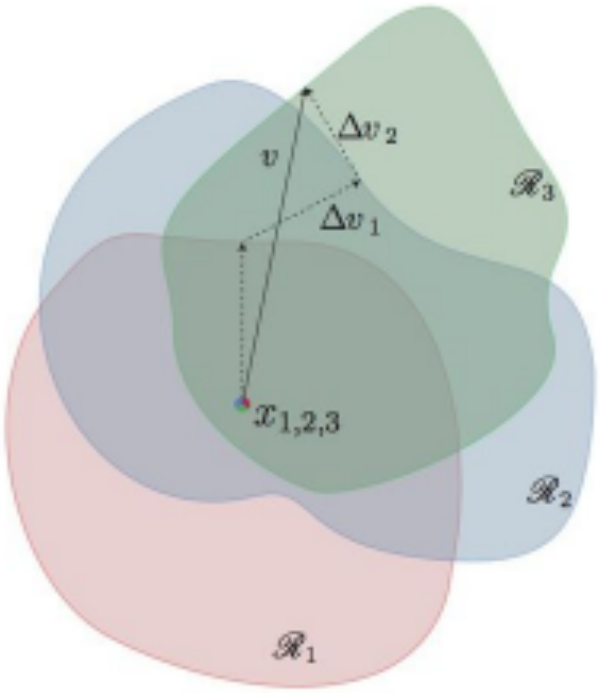
分类错误。

  - 同时还有一个范数约束：

$$||v||_p \leq \varepsilon, \tag{19}$$

$$P(\hat{k}(x+v) \neq \hat{k}(x)) \geq 1-\sigma \tag{20}$$

换句说，需要找到一个对抗扰动v，这个扰动可以加到所有的样本点上，而且会以$1-\sigma$的概率让对抗样本被分类错误。



Algorithm 1 Computation of universal perturbations.

input: Data points X, classifier $\hat{k}$, desired ℓ_p norm of the perturbation ξ, desired accuracy on perturbed samples δ.
output: Universal perturbation vector v.
Initialize $v \leftarrow 0$.
while $\text{Err}(X_v) \leq 1 - \delta$ **do**
    for each datapoint $x_i \in X$ **do**
        if $\hat{k}(x_i + v) = \hat{k}(x_i)$ **then**
            Compute the *minimal* perturbation that sends $x_i + v$ to the decision boundary:

$$\Delta v_i \leftarrow \arg\min_r \|r\|_2 \text{ s.t. } \hat{k}(x_i + v + r) \neq \hat{k}(x_i).$$

            Update the perturbation:

$$v \leftarrow \mathcal{P}_{p,\xi}(v + \Delta v_i).$$

        end if
    end for
end while

其中：

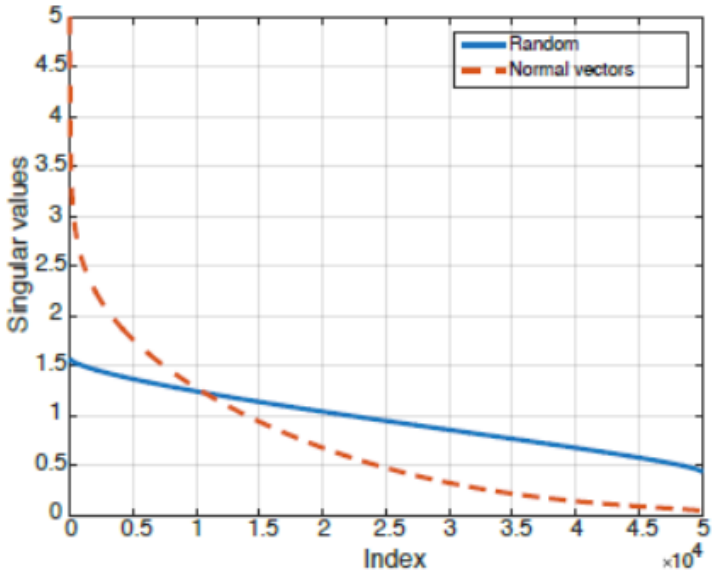
$$\mathcal{P}_{p,\xi}(v) = \arg\min_{v'} \|v - v'\|_2 \text{ subject to } \|v'\|_p \leq \xi.$$

表示把寻到的扰动v限制在L_p范数下以ε为半径的球上。

- 这个算法的思想是：
  - 从分布μ里面采样出一个样本集X，里面有m个图片，然后迭代地寻找能够让m个样本以$1-\delta$概率被分类错误的对抗扰动。
  - 一开始$v=0$，没有什么扰动，然后对于每个样本x_i，看它加上扰动v后，会不会分类错误，如果分类错误，则下一个样本；否则寻找一个微小的扰动Δv_i，使得$x_i+v+\Delta v_i x_i$被分类错误。持续这个过程，直到在这m个样本中错误样本满足错误率。
- UAP存在性解释：
  - 对于验证集里面的每个样本x，我们寻找它的对抗扰动$r(x) = argmin_r ||r||_2 s.t. \hat{k}(x+r) \neq \hat{k}(x)$，这种$r(x)$其实可以近似看作是分类模型在x处的决策界的法向量，因为它很小，只改x的一点点就让分类器得到其他的标签。
  - 作者提取n个样本处的这种法向量，并对它们进行单位化，形成正规矩阵N：

$$N = \left[\frac{r(x_1)}{\|r(x_1)\|_2} \cdots \frac{r(x_n)}{\|r(x_n)\|_2} \right]$$

通过对N进行SVG分解，作者发现N的奇异值有一些特别大，而另外一些特别小：



- 这种现象意味着，这些法向量其实可以存在冗余的，换句话说这些法向量所在决策界存在着冗余性和相关性。基于SVG分解的前100个向量生成的对抗扰动，也能取得38%的对抗准确性。这就说明了，神经网络学习得到的决策界，在高维空间是存在相似的相关性的。
- 通过样本子集X 可以获得m个样本的决策界相关性，这种相关性在其它不同的样本周围的决策界上依然存在。Universal Pertubation则是以最大化成功率的使用这些法向量构建扰动，因而它也会学习到决策界的相关性。

MI- FGSM（Momentum iterative attack）动量迭代攻击

- 引入动量迭代快速梯度符号方法来以非目标攻击方式生成满足L_∞范数限制的对抗样本。动量法是一种通过在迭代过程中沿损失函数的梯度方向累积速度矢量来加速梯度下降算法的技术。
- FGSM通过在数据点周围决策边界的线性假设下仅一次将梯度的符号应用于真实示例来生成一个对抗样本。但是，实际上，当失真较大时，线性假设可能不成立，这使得FGSM生成的对抗样本不足于模型，从而限制了其攻击能力。相反，I-FGSM在每次迭代中将对抗样本沿梯度符号的方向贪婪地移动。因此，对抗样本很容易掉入**不良的局部最大值**并"过度拟合"模型，这不太可能在模型之间转移。
- 在（6）式中使用衰减因子μ收集前t次迭代的梯度，可以保留梯度的大致方向，防止陷入不好的局部最优值。每次迭代中使用L_1距离做归一化。
- 文献提出的方案是以一组集成模型为目标，在黑盒/灰盒设置下攻击一个不可见的模型。其基本思想是考虑多个模型相对于输入的梯度，并综合确定一个梯度方向，这种攻击方法生成的对抗样本更可能转移攻击其他黑盒/灰盒模型。
- MIM攻击全称是 Momentum Iterative Method，其实这也是一种类似于PGD的基于梯度的迭代攻击算法。它的本质就是，在进行迭代的时候，每一轮的扰动不仅与当前的梯度方向有关，还与之前算出来的梯度方向相关。其中的衰减因子就是用来调节相关度的，decay_factor在（0，1）之间，decay_factor越小，那么迭代轮数靠前算出来的梯度对当前的梯度方向影响越小。其实仔细想想，这样做也很有道理，由于之前的梯度对后面的迭代也有影响，那么这使得，迭代的方向不会跑偏，使得总体的大方向是对的。到目前为止都是笔者对MIM比较感性的认识，下面贴出论文中比较学术的观点。
- 其实为了加速梯度下降，通过累积损失函数的梯度方向上的矢量，从而（1）稳定更新（2）有助于通过narrow valleys, small humps and poor local minima or maxima.(专业名词不知道怎么翻译，可以脑补函数图像，大致意思就是，可以有效避免局部最优）

Algorithm 1 MI-FGSM

Input: A classifier f with loss function J; a real example $\boldsymbol{x}$ and ground-truth label y;

Input: The size of perturbation ϵ; iterations T and decay factor μ.

Output: An adversarial example $\boldsymbol{x}^*$ with $\|\boldsymbol{x}^* - \boldsymbol{x}\|_\infty \leq \epsilon$.

1: $\alpha = \epsilon/T$;

2: $\boldsymbol{g}_0 = 0$; $\boldsymbol{x}_0^* = \boldsymbol{x}$;

3: **for** $t = 0$ to $T - 1$ **do**

4: Input $\boldsymbol{x}_t^*$ to f and obtain the gradient $\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_t^*, y)$;

5: Update $\boldsymbol{g}_{t+1}$ by accumulating the velocity vector in the gradient direction as

$$\boldsymbol{g}_{t+1} = \mu \cdot \boldsymbol{g}_t + \frac{\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_t^*, y)}{\|\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_t^*, y)\|_1}; \tag{6}$$

6: Update $\boldsymbol{x}_{t+1}^*$ by applying the sign gradient as

$$\boldsymbol{x}_{t+1}^* = \boldsymbol{x}_t^* + \alpha \cdot \text{sign}(\boldsymbol{g}_{t+1}); \tag{7}$$

7: **end for**

8: **return** $\boldsymbol{x}^* = \boldsymbol{x}_T^*$.