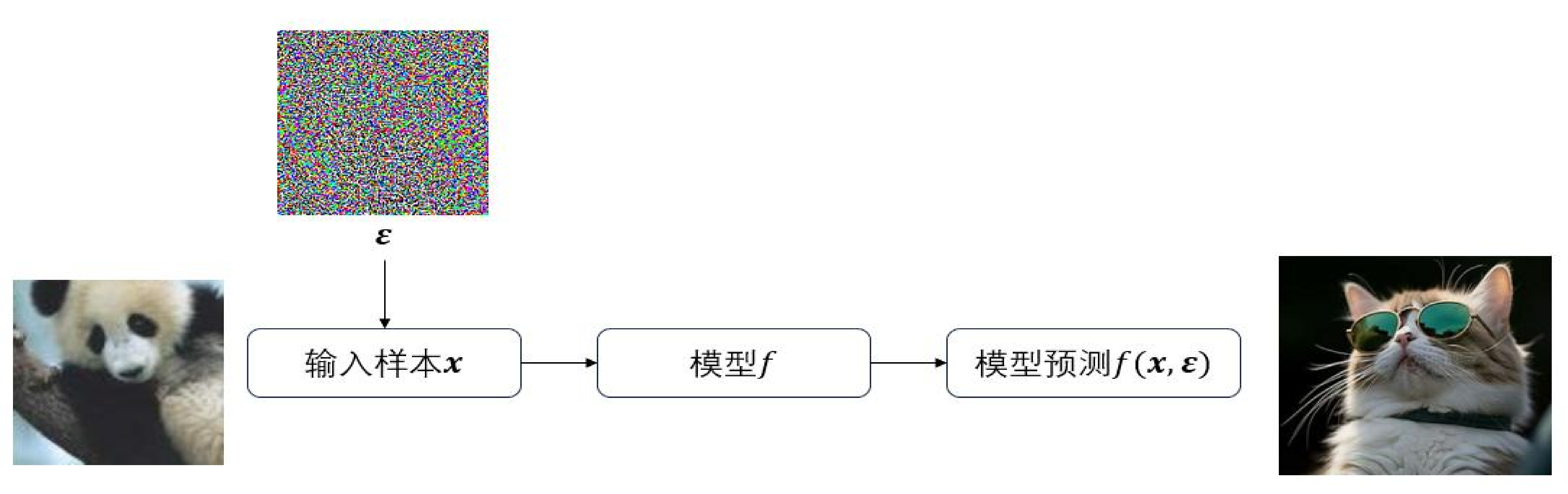
《对抗攻击》

对抗攻击概述

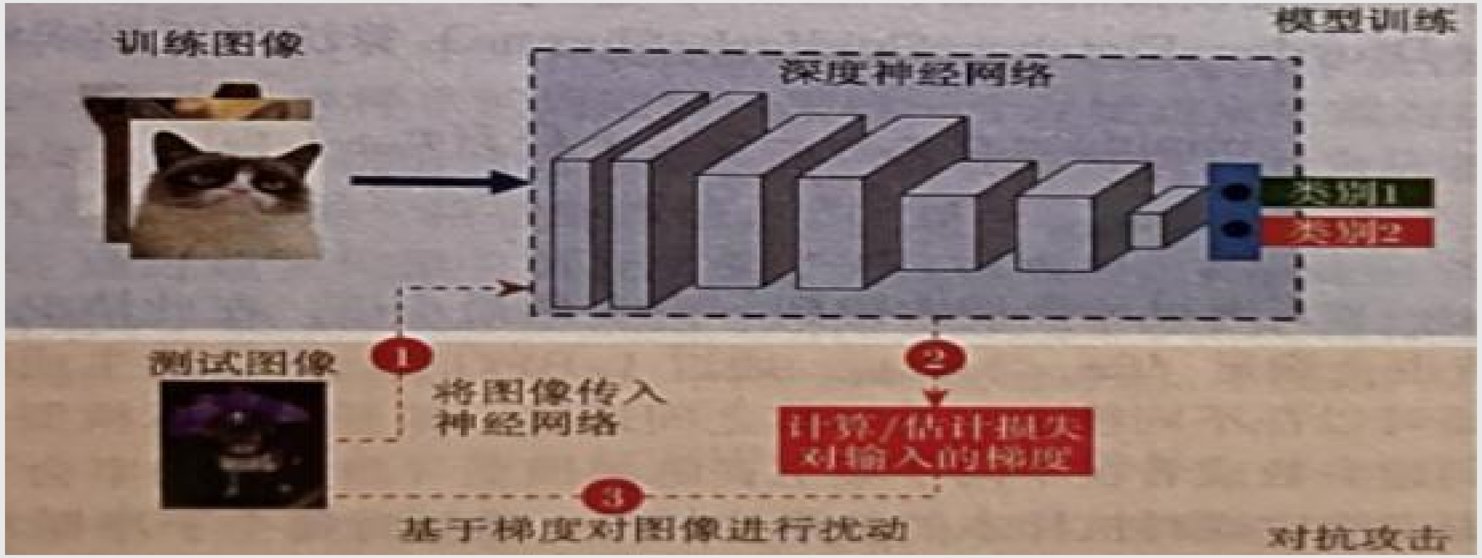
对抗攻击

- 对抗攻击（adversarial attack）一般通过向干净测试样本中添加细微的、人眼无法察觉的（对图像数据来说）噪声来构造对抗样本(adversarial example)。



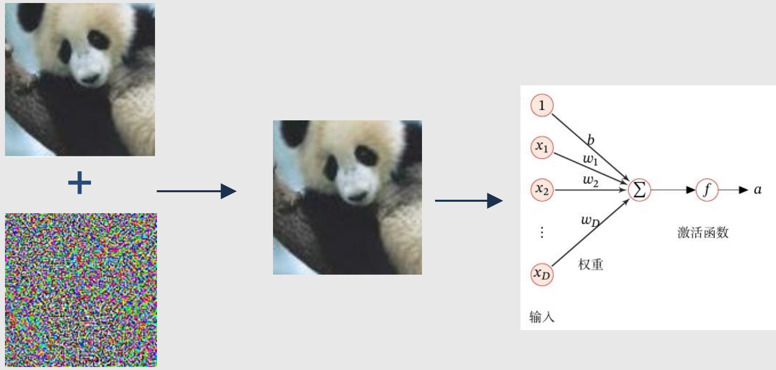
对抗攻击与模型训练的区别

- 对抗攻击的目标是一个已经训练完成的模型，是一种测试阶段攻击。
- 对抗攻击改变的是输入样本而模型训练改变的是模型参数。
- 对抗攻击通过梯度上升最大化模型的错误而模型训练通过梯度下降最小化模型的错误。



攻击分类

- 无目标攻击
  - 生成的对抗样本x_{adv}会被目标模型错误预测为除真实类别以外的任意类别，即$f(x_{adv}) \neq y$
- 目标攻击
  - 生成的对抗样本x_{adv}会被目标模型f错误预测为攻击者预先指定的目标类别y_t，即$f(x_{adv}) = y_t$ 且$y_t \neq y$



y

y_1

y_2

y_3

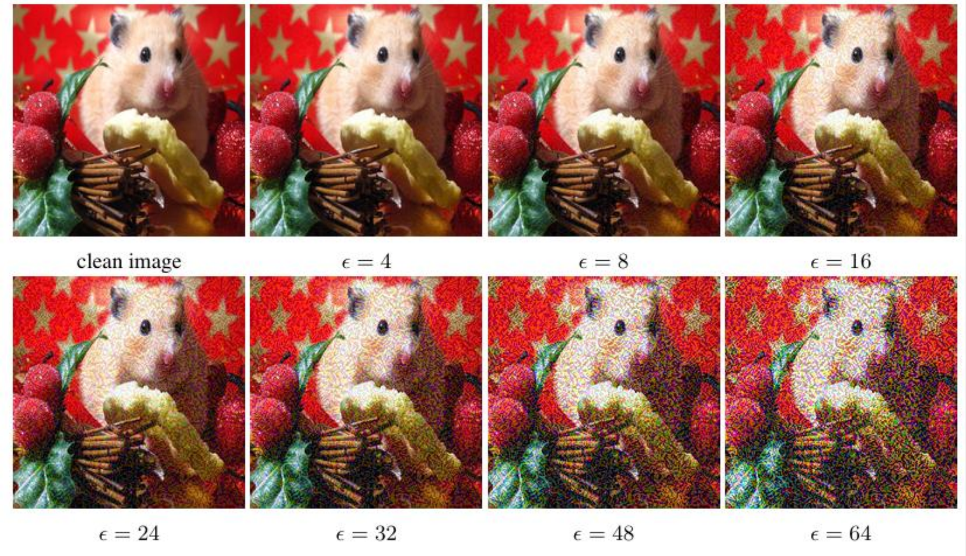
攻击分类

- 白盒攻击(White-Box Attack)
  - 假设攻击者可以获得目标模型的全部信息，包括训练数据、超参数、激活函数、模型架构与参数等。
- 黑盒攻击(Black-Box Attack)
  - 假设攻击者无法获得目标模型的相关信息，只能获得目标模型的输出信息（逻辑值或概率）。

与白盒攻击相比，黑盒攻击更贴合实际应用场景，也更加具有挑战性。

常见衡量指标

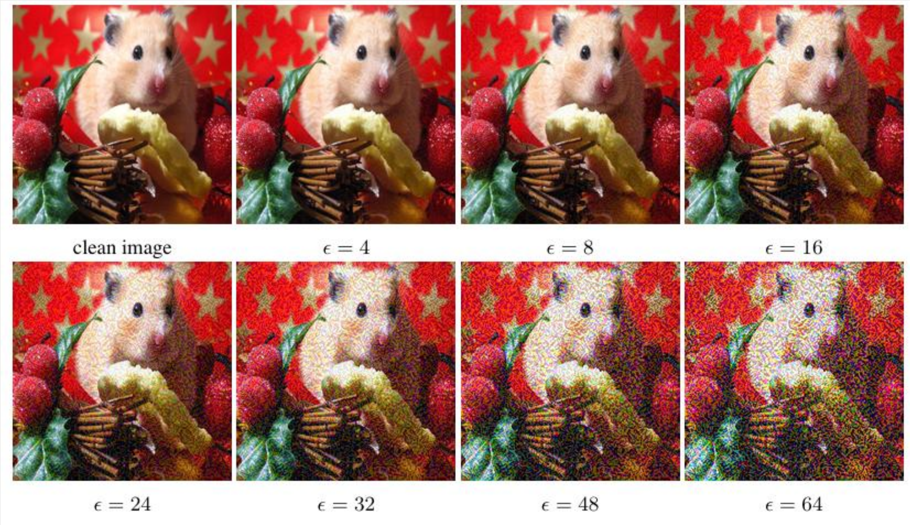
- 对抗样本在攻击成功的基础上，要保证数据的主体信息不变，即要保证添加的扰动肉眼不可见。



  - 扰动量用下面的式子表示：
  - $L_p = ||\varepsilon||_p = \left(\sum_{i=1}^{n} |\varepsilon_i|^p\right)^{\frac{1}{p}}$
  - 当$p = 0$时，为0范数攻击，又称为单像素攻击，物理含义为修改的数据总点数。这种放松会限制可以改变的数据点个数，不关心每个点具体改变了多少。
  - 当$p = 1$时，为1范数攻击，物理含义为前后两个数据的总改变量。这种方式从全局上考虑修改幅度较小。

常见衡量指标

- 对抗样本在攻击成功的基础上，要保证数据的主体信息不变，即要保证添加的扰动肉眼不可见。
  - 扰动量用下面的式子表示：



  - $L_p = ||\varepsilon||_p = \left(\sum_{i=1}^{n} |\varepsilon_i|^p\right)^{\frac{1}{p}}$
  - 当$p = 2$时，为2范数攻击，物理含义为前后两个数据的欧氏距离。与1范数攻击相似，这种方式从全局上考虑修改幅度较小。
  - 当$p = \infty$时，为∞范数攻击，物理含义为修改前后单个数据点的最大扰动量。这种方式最终会修改更多的数据点，但是修改点的值仅会轻微改变。

L-BFGS

- 通过解决以下边界约束优化问题构造对抗样本：
  - $\min ||x_{adv} - x||_2$
  - $s.t. f(x_{adv}) = y_t , x_{adv} \in [0,1]^d$
  - 其中，y_t 为攻击者预先指定的目标类别，x_{adv}是x的对抗样本。
- 由于上述优化问题难以求解，所以使用边界约束的L-BFGS算法近似求解：
  - $\min c||x_{adv} - x||_2 + L(f(x_{adv}), y_t)$
  - $s.t.\ x_{adv} \in [0,1]^d$

局部线性假说

- 对抗样本是深度神经网络局部线性化的必然产物。
- 核心思想：
  - 尽管深度神经网络整体呈非线性，但其内部存在大量局部线性操作，基于局部线性性质进行攻击足以产生对抗样本。

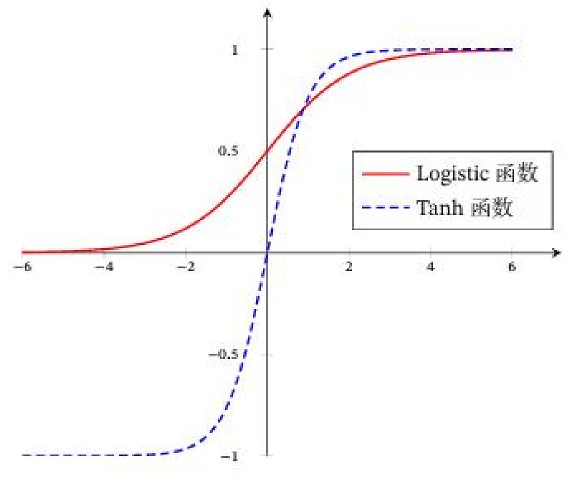


图 4.2 Logistic 函数和 Tanh 函数

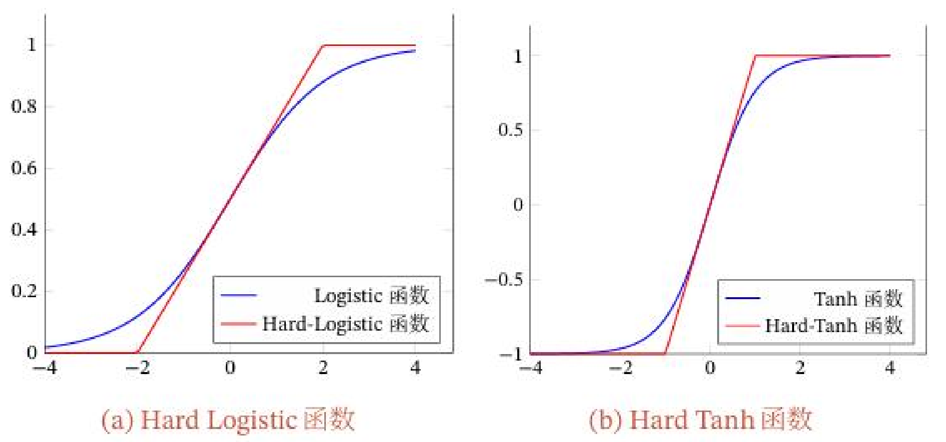


图 4.3 Hard Sigmoid 型激活函数

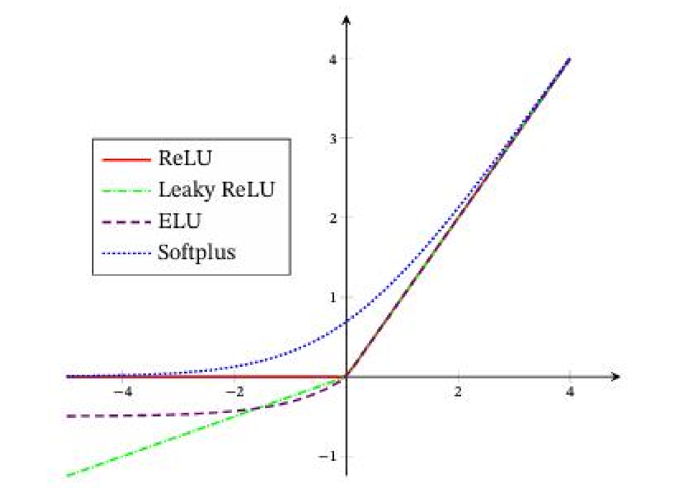
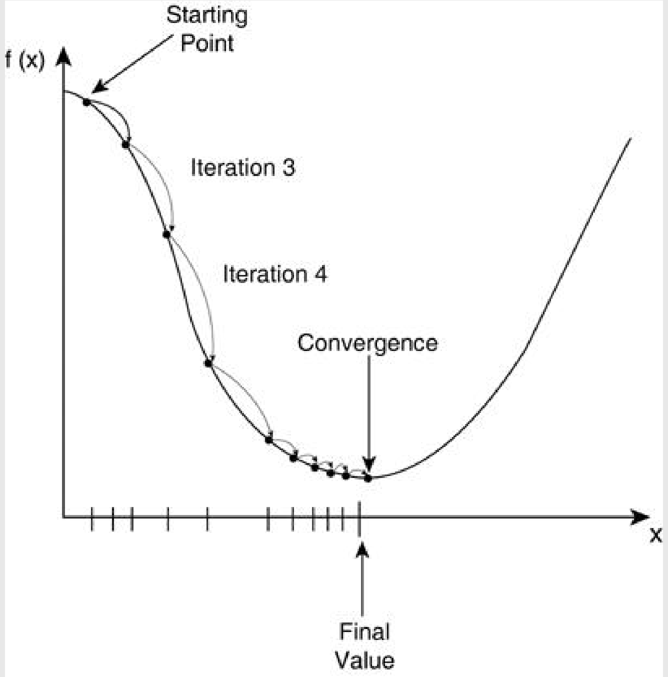
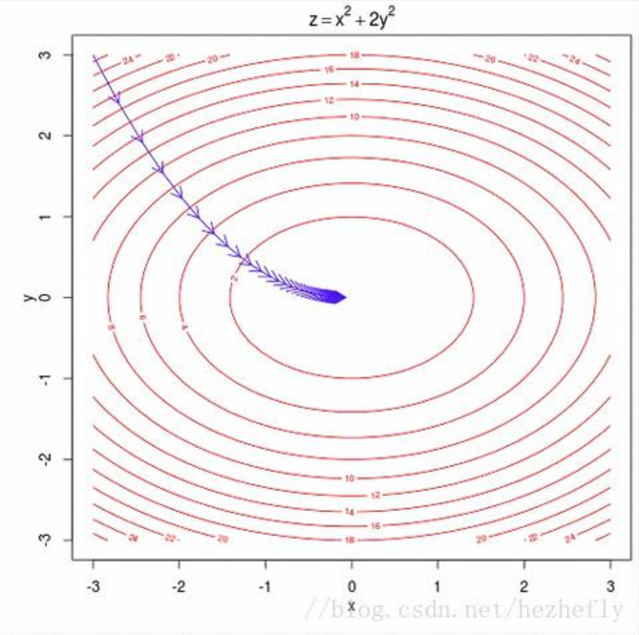


图 4.4 ReLU、Leaky ReLU、ELU 以及 Softplus 函数

梯度下降法（Gradient Descent）





搜索步长α中也叫作学习率（Learning Rate）

$$
\begin{aligned}
\theta_{t+1} &= \theta_t - \alpha \frac{\partial \mathcal{R}(\theta)}{\partial \theta_t} \\
&= \theta_t - \alpha \frac{1}{N} \sum_{i=1}^{N} \frac{\partial \mathcal{L}(\theta_t; x^{(i)}, y^{(i)})}{\partial \theta}.
\end{aligned}
$$

随机梯度下降法

- 随机梯度下降法（Stochastic Gradient Descent，SGD）也叫增量梯度下降，每个样本都进行更新

$$\theta_{t+1} = \theta_t - \alpha \frac{\partial \mathcal{L}(\theta_t; x^{(t)}, y^{(t)})}{\partial \theta},$$

- 小批量（Mini-Batch）随机梯度下降法

随机梯度下降法

算法 2.1: 随机梯度下降法

输入: 训练集 $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$，验证集 $\mathcal{V}$，学习率 α

随机初始化 θ;

repeat

  对训练集 $\mathcal{D}$ 中的样本随机重排序;

  for $n = 1 \cdots N$ **do**

    从训练集 $\mathcal{D}$ 中选取样本 $(\mathbf{x}^{(n)}, y^{(n)})$;

    // 更新参数

$$\theta \leftarrow \theta - \alpha \frac{\partial \mathcal{L}(\theta; x^{(n)}, y^{(n)})}{\partial \theta};$$

  end

until 模型 $f(\mathbf{x}; \theta)$ 在验证集 $\boxed{\mathcal{V}}$ 上的错误率不再下降;

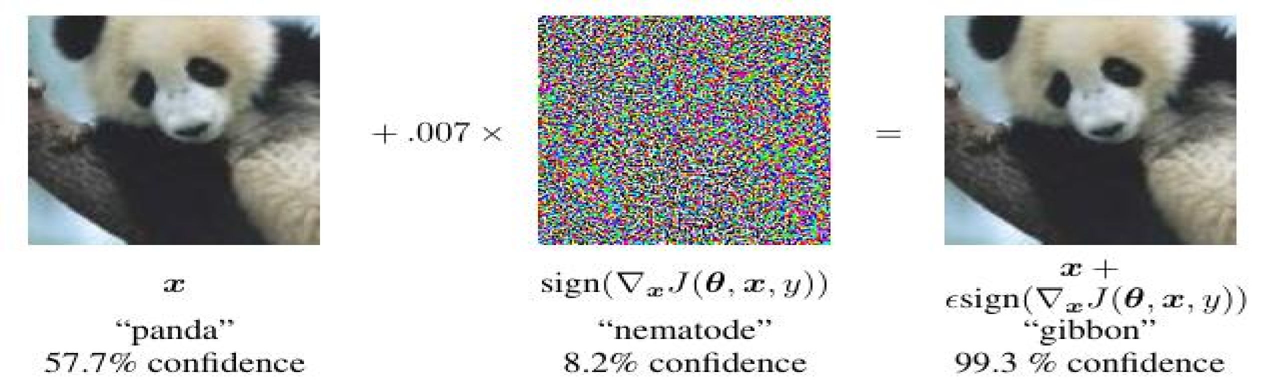
输出: θ

Why?

FGSM(Fast Gradient Sign Method)

- 假设：
  - 损失函数L在样本x周围是线性的，即可以被x处的一阶泰勒展开高度近视。
- 方法：
  - 利用输入梯度（分类损失相对输入的梯度）的符号信息进行一步固定步长的梯度上升来完成攻击。

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\eta} \qquad \boldsymbol{\eta} = \epsilon \operatorname{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)\right)$$



Meet the L_∞norm bound$||x^* - x||_\infty \le \varepsilon$

FGM(Fast Gradient Method)

- 方法：
  - 利用输入梯度（分类损失相对输入的梯度）的单位向量进行一步固定步长的梯度上升来完成攻击。
  - FGM is a generalization of FGSM to meet the L_2 norm bound $||x^* - x||_2 \leq \varepsilon$

$$\boldsymbol{x}^* = \boldsymbol{x} + \epsilon \cdot \frac{\nabla_{\boldsymbol{x}} J(\boldsymbol{x}, y)}{||\nabla_{\boldsymbol{x}} J(\boldsymbol{x}, y)||_2}$$

BIM(Basic Iterative Method)

- 方法：
  - 以更小的步长多次应用FGSM，并在每次迭代后对生成的对抗样本的像素值进行裁剪，以保证每个像素的变化都足够小。

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X}, \quad \boldsymbol{X}_{N+1}^{adv} = Clip_{X,\epsilon}\Big\{\boldsymbol{X}_N^{adv} + \alpha\,\mathrm{sign}\big(\nabla_X J(\boldsymbol{X}_N^{adv}, y_{true})\big)\Big\}$$

$$Clip_{X,\epsilon}\{\boldsymbol{X}'\}(x,y,z) = \min\Big\{255, \boldsymbol{X}(x,y,z)+\epsilon, \max\big\{0, \boldsymbol{X}(x,y,z)-\epsilon, \boldsymbol{X}'(x,y,z)\big\}\Big\}$$

  - 总迭代次数T设置为$\min(\varepsilon + 4, 1.25\varepsilon)$（对应像素值范围[0,255]），步长$\alpha = \varepsilon/T$($T$步迭代后正好达到$\varepsilon$大小)

- BIM本质上是对负损失函数的投影梯度下降。

ILCM(Iterative Least-Likely Class Method)

- 之前的方法只是试图增加正确类的成本，而没有指定模型应该选择哪些不正确的类。
- 方法：
  - 引入ILCM方法。这种迭代方法试图生成一个对抗性的图像，该图像将被归类为特定的期望目标类。对于期望的类别，根据训练好的网络对图像X的预测来选择最不可能的类别。

$$y_{LL} = \arg\min_{y} \{ p(y|\boldsymbol{X}) \}$$

To make an adversarial image which is classified as y_{LL} we maximize $\log p(y_{LL}|\boldsymbol{X})$ by making iterative steps in the direction of $\text{sign}\{\nabla_X \log p(y_{LL}|\boldsymbol{X})\}$. This last expression equals $\text{sign}\{-\nabla_X J(\boldsymbol{X}, y_{LL})\}$ for neural networks with cross-entropy loss. Thus we have the following procedure:

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X}, \quad \boldsymbol{X}_{N+1}^{adv} = Clip_{X,\epsilon}\left\{\boldsymbol{X}_N^{adv} - \alpha \,\text{sign}\left(\nabla_X J(\boldsymbol{X}_N^{adv}, y_{LL})\right)\right\}$$

MI-FGSM

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

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

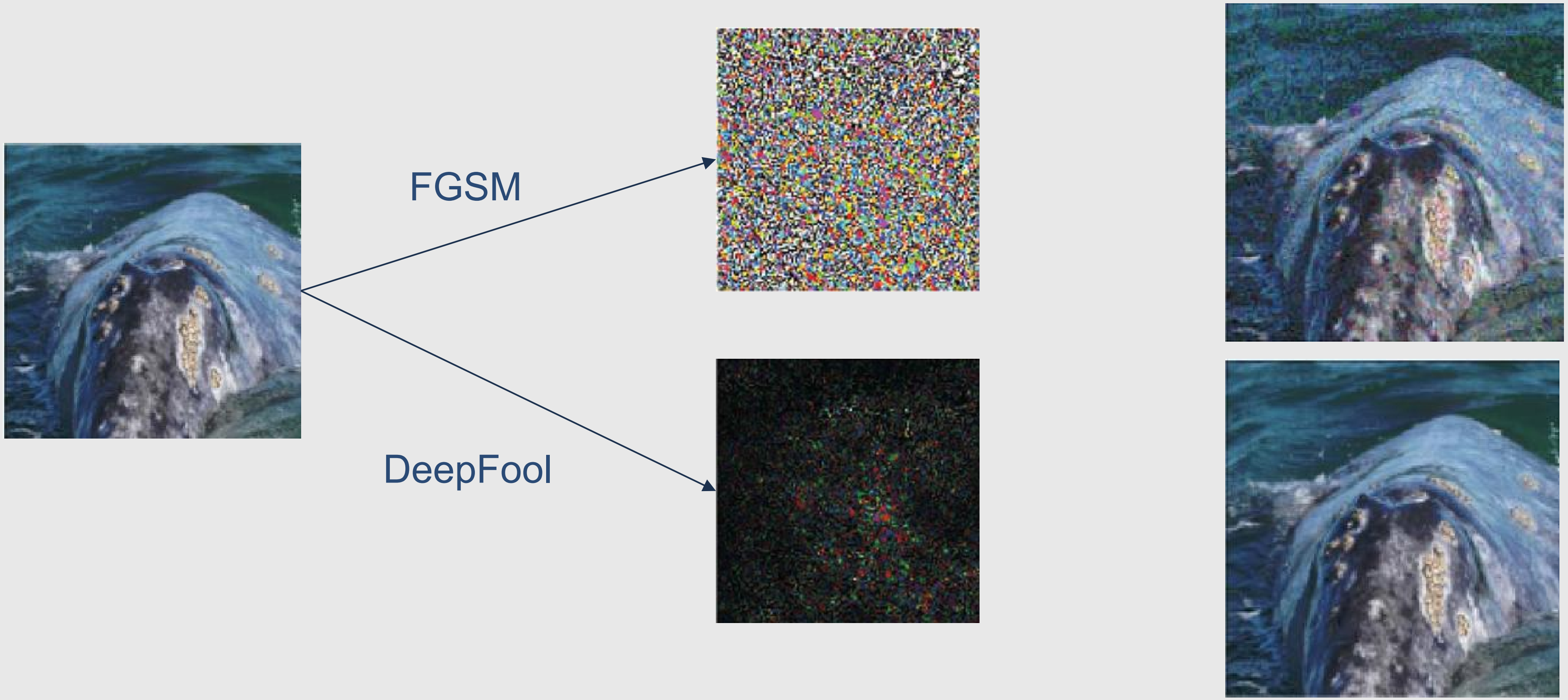
6: Update $\boldsymbol{x}_{t+1}^*$ by applying the sign gradient as

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

7: **end for**

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

FGSM的不足之处

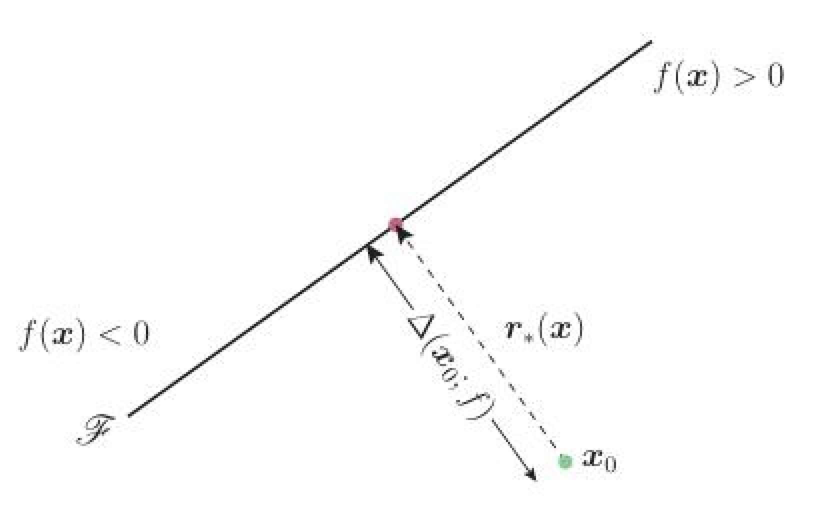


- FGSM的扰动太大，容易被人发现，且扰动大小需要人为设置。

DeepFool for binary classifiers

- As a multiclass classifier can be be viewed as aggregation of binary classifiers, first propose the algorithm for binary classifiers.
  - The minimal perturbation to change the classifier's decision corresponds to the orthogonal projection of x_0 onto F. It is given by the closed-form formula:

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



DeepFool for binary classifiers

- Assuming now that f is a general binary differentiable classifier, we adopt an iterative procedure to estimate the robustness $\Delta(x_0; f)$. Specifically, at each iteration, f is linearized around the current point x_i and the minimal perturbation of the linearized classifier is computed as:

$$\arg\min_{\boldsymbol{r}_i} \|\boldsymbol{r}_i\|_2 \text{ subject to } f(\boldsymbol{x}_i) + \nabla f(\boldsymbol{x}_i)^T \boldsymbol{r}_i = 0$$

- In practice, the above algorithm can often converge to a point on the zero level set F. In order to reach the other side of the classification boundary, the final perturbation vector $\hat{r}$ is multiplied by a constant $1 + \eta$, with $\eta \ll 1$.In our experiments, we have used η =0.02.

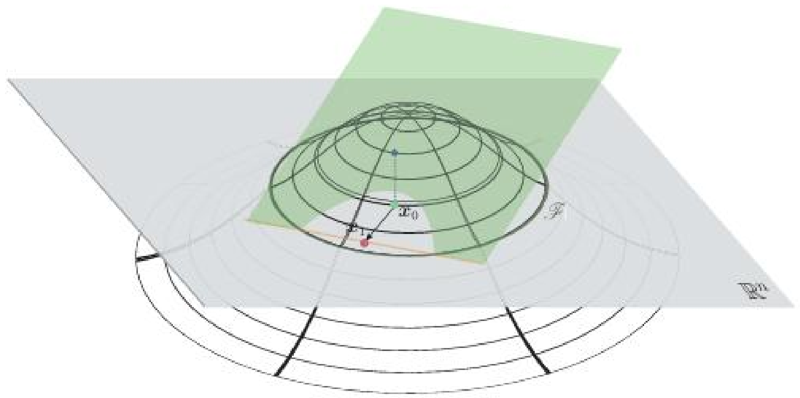
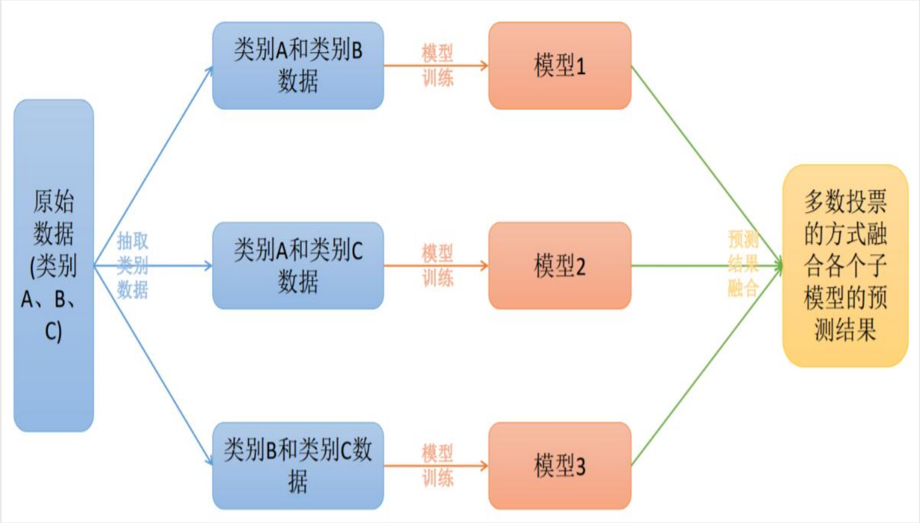


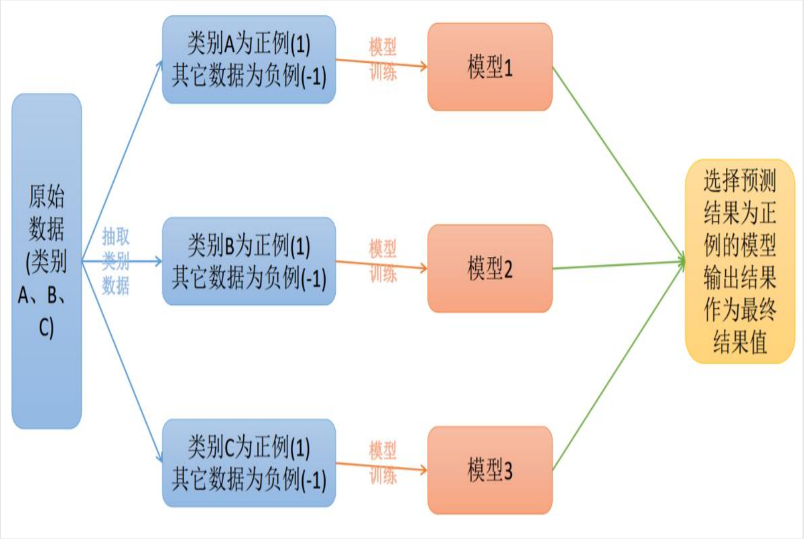
Figure 3: Illustration of Algorithm 1 for $n = 2$. Assume $\boldsymbol{x}_0 \in \mathbb{R}^n$. The green plane is the graph of $\boldsymbol{x} \mapsto f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)^T(\boldsymbol{x} - \boldsymbol{x}_0)$, which is tangent to the classifier function (wire-framed graph) $\boldsymbol{x} \mapsto f(\boldsymbol{x})$. The orange line indicates where $f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)^T(\boldsymbol{x} - \boldsymbol{x}_0) = 0$. $\boldsymbol{x}_1$ is obtained from $\boldsymbol{x}_0$ by projecting $\boldsymbol{x}_0$ on the orange hyperplane of $\mathbb{R}^n$.

多分类学习

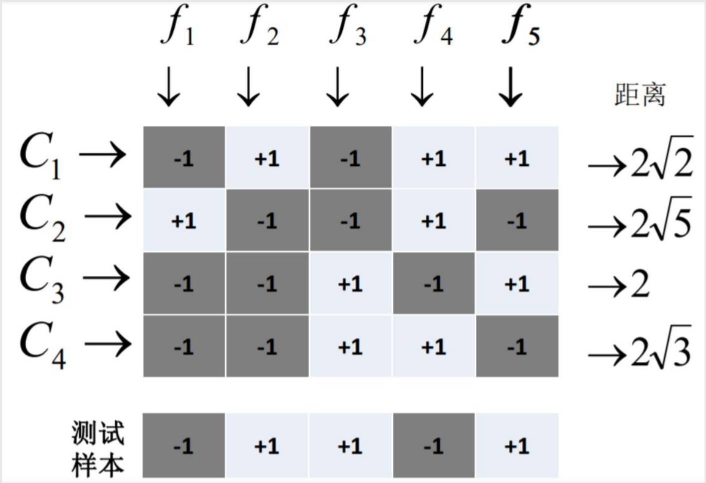
- 多分类学习的基本思路是“拆解法”，即将多分类任务拆解为若干个二分类任务求解。
- 最经典的拆分策略有三种：One vs. One (OvO)、One vs. Rest (OvR)、Many vs. Many (MvM)



OvO



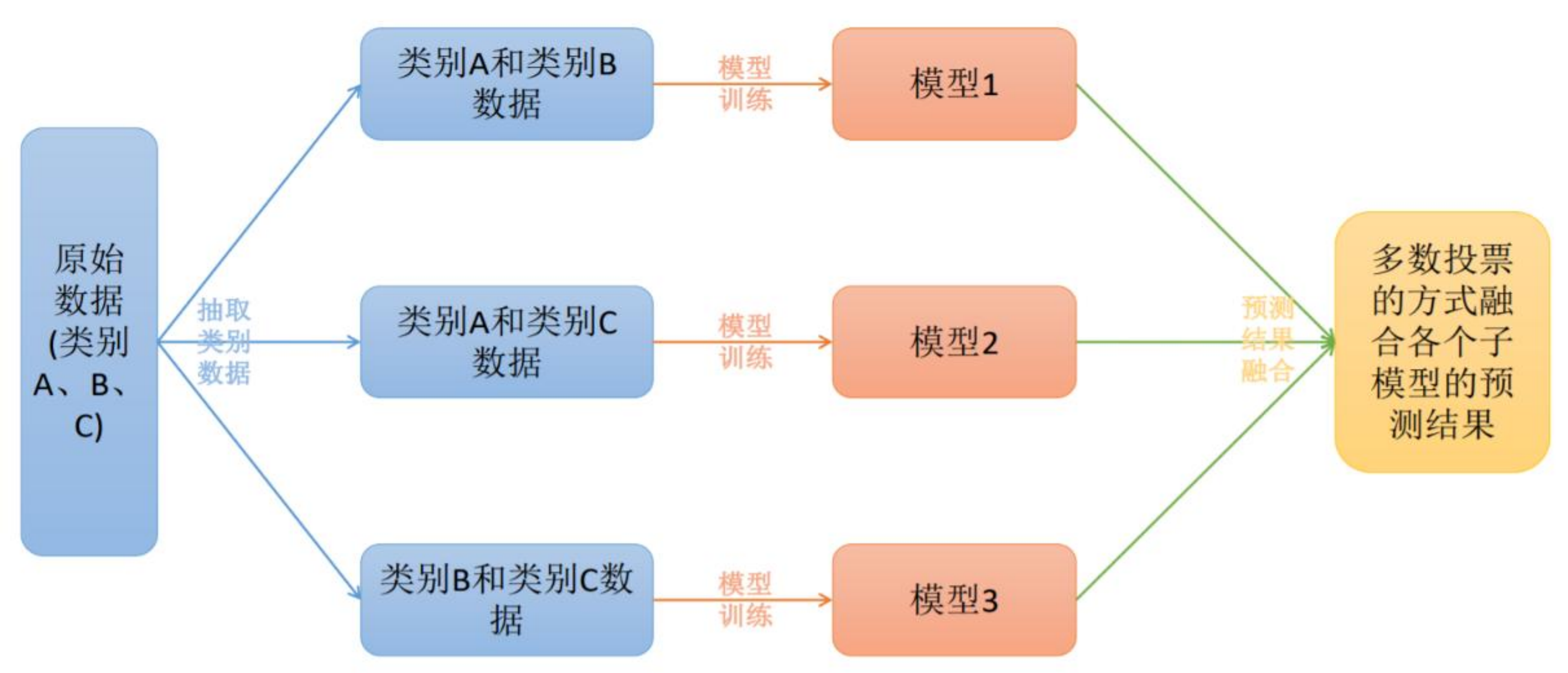
OvR



MvM

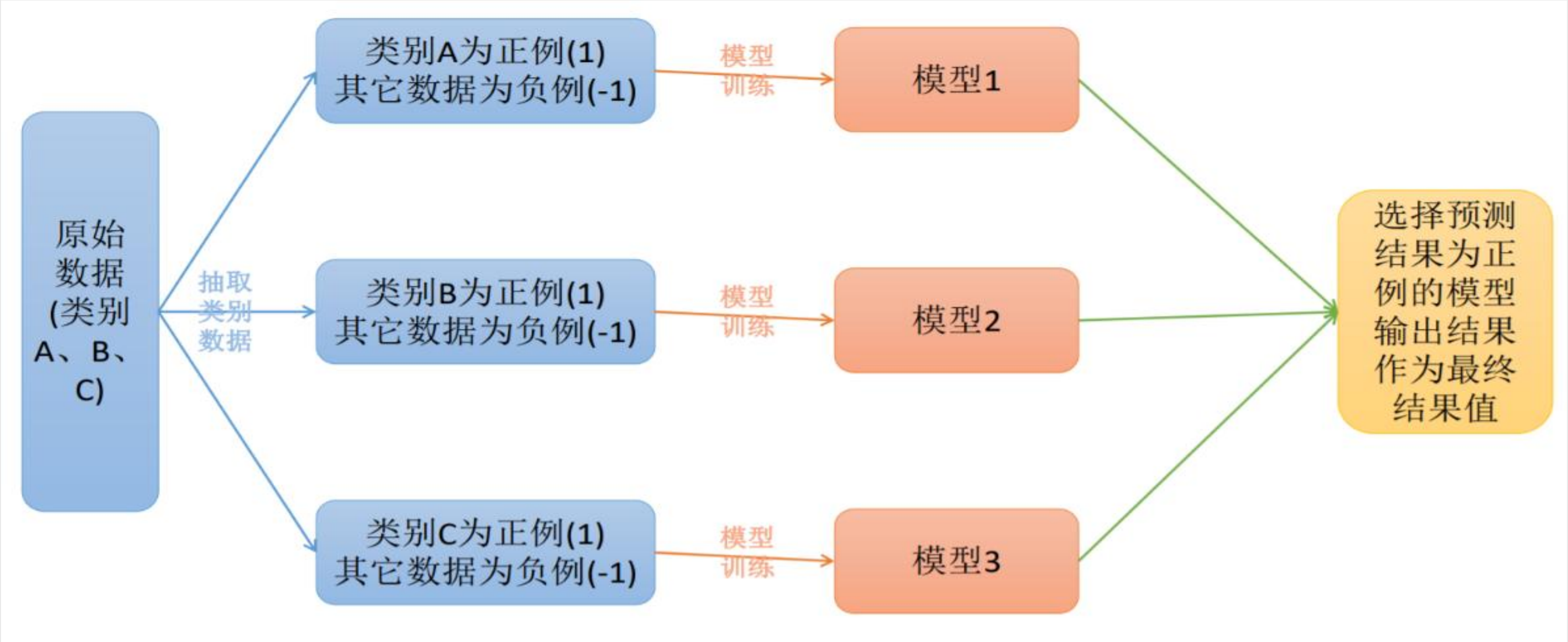
One vs. One（OvO）

- 给定数据集$D=\{(x_1,y_1),(x_2,y_2),\dots,(x_m,y_m)\}, y_i\in\{C_1,C_2,\dots,C_N\}$.OvO将这$N$个类别两两配对，从而产生$N(N-1)/2$个二分类任务，例如OvO将为区分类别$C_i$和$C_j$训练一个分类器，该分类器把$D$中的$C_i$类样例作为正例，$C_j$类样例作为反例。在测试阶段，新样本将同时提交给所以分类器，于是我们将得到$N(N-1)/2$个分类结果，最终结果可通过投票产生：即把预测得最多的类别作为最终分类结果。

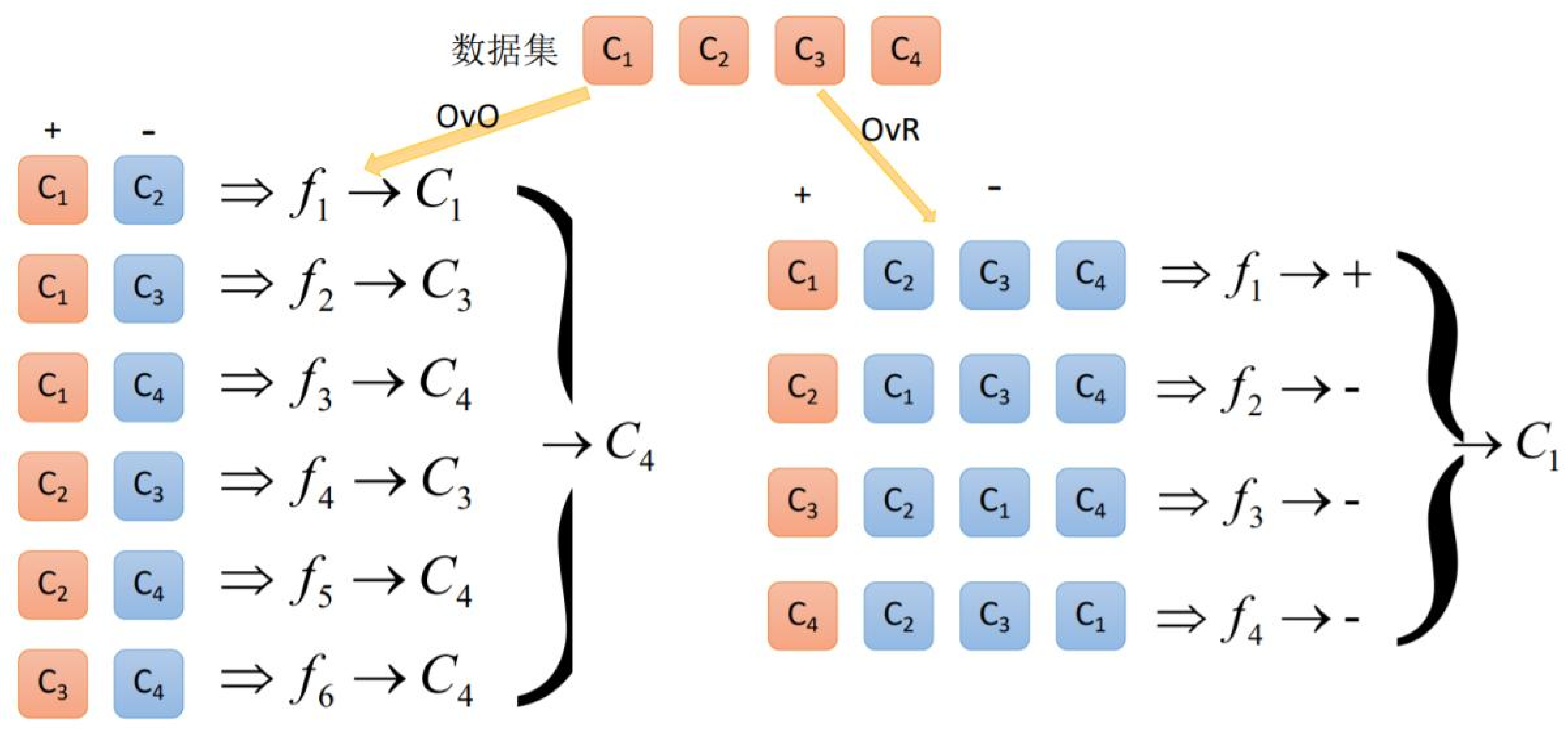


One vs. Rest（OvR）

- OvR亦称OvA（One vs. All），但OvA这个说法不严格，因为不可能把“所有类”作为反类。
- OvR则是每次将一个类的样例作为一个正例、所有其他类的样例作为反例来训练N个分类器。在测试时若仅有一个分类器预测为正类，则对应的类别标记为最终分类结果。



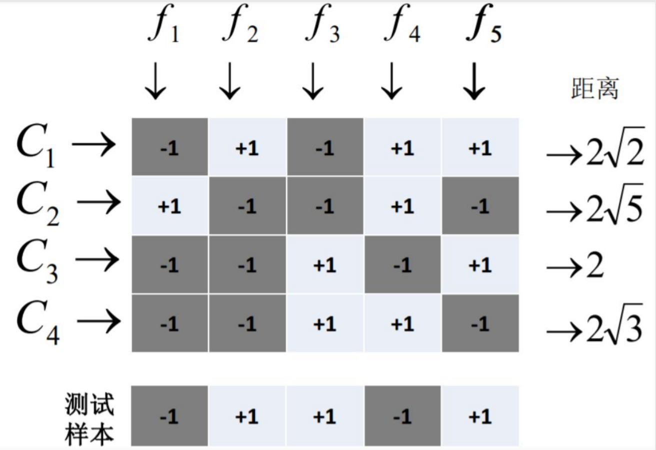
OvO与OvR的区别


数据集
C1
C2
C3
C4
OvO
OvR
+
-
$\Rightarrow f_1 \rightarrow C_1$
$\Rightarrow f_2 \rightarrow C_3$
$\Rightarrow f_3 \rightarrow C_4$
$\Rightarrow f_4 \rightarrow C_3$
$\Rightarrow f_5 \rightarrow C_4$
$\Rightarrow f_6 \rightarrow C_4$
$\rightarrow C_4$
$\Rightarrow f_1 \rightarrow +$
$\Rightarrow f_2 \rightarrow -$
$\Rightarrow f_3 \rightarrow -$
$\Rightarrow f_4 \rightarrow -$
$\rightarrow C_1$

Many vs. Many（MvM）

MvM原理

- 将模型构建应用分为两个阶段：编码阶段和解码阶段。
- 编码阶段：对k个类别中进行m次划分，每次划分将一部分数据分为正类，一部分数据分为反类，每次划分都构建出来一个模型，模型的结果是在空间中对于每个类别都定义了一个点。
- 解码阶段：使用训练出来的模型对测试样例进行预测，将预测样本对应的点和类别之间的点求距离，选择距离最近的类别作为最终的预测类别。



DeepFool for multiclass classifiers

- Extend the DeepFool method to the multiclass case.
- Method based on one-vs-all.
- The classifier has c outputs where c is the number of classes. Therefore, a classifier can be defined as $f: R^n \rightarrow R^c$and the classification is done by the following mapping:

$$\hat{k}(\boldsymbol{x}) = \arg\max_k f_k(\boldsymbol{x})$$

- where $f_k(x)$ is the output of $f(x)$ that corresponds to the k^{th}class.

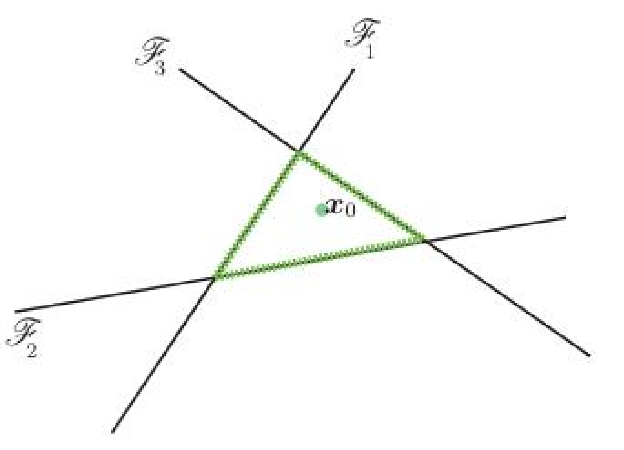
DeepFool for multiclass classifiers

- Classifier : $f(x) = W^T x + b$, given W and b
- Since the mapping $\hat{k}$ is the outcome of a one-vs-all classification scheme, the minimal perturbation to fool the classifier can be rewritten as follows

$$\arg\min_{\boldsymbol{r}} \|\boldsymbol{r}\|_2$$
$$\text{s.t. } \exists k : \boldsymbol{w}_k^\top(\boldsymbol{x}_0 + \boldsymbol{r}) + b_k \geq \boldsymbol{w}_{\hat{k}(\boldsymbol{x}_0)}^\top(\boldsymbol{x}_0 + \boldsymbol{r}) + b_{\hat{k}(\boldsymbol{x}_0)}$$

- Convex polyhedron P(defines the region of the space where f outputs the label $\hat{k}(x_0)$):

$$P = \bigcap_{k=1}^{c} \{\boldsymbol{x} : f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}) \geq f_k(\boldsymbol{x})\}$$



- Distance between x_0 and the complement of the convex polyhedron P:$dist(x_0, P^c)$

DeepFool for multiclass classifiers

- Define $\hat{l}(x_0)$ to be the closest hyperplane of the boundary of P

$$\hat{l}(\boldsymbol{x}_0) = \underset{k \neq \hat{k}(\boldsymbol{x}_0)}{\arg\min} \frac{\left|f_k(\boldsymbol{x}_0) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_0)\right|}{\|\boldsymbol{w}_k - \boldsymbol{w}_{\hat{k}(\boldsymbol{x}_0)}\|_2}$$

- e.g. $\hat{l}(x_0) = 3$ in Figure 4
- Minimum perturbation $r_*(x_0)$:

$$\boldsymbol{r}_*(\boldsymbol{x}_0) = \frac{\left|f_{\hat{l}(\boldsymbol{x}_0)}(\boldsymbol{x}_0) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_0)\right|}{\|\boldsymbol{w}_{\hat{l}(\boldsymbol{x}_0)} - \boldsymbol{w}_{\hat{k}(\boldsymbol{x}_0)}\|_2^2}(\boldsymbol{w}_{\hat{l}(\boldsymbol{x}_0)} - \boldsymbol{w}_{\hat{k}(\boldsymbol{x}_0)}). \tag{9}$$

Vector that projects x_0 on the hyperplane indexed by $\hat{l}(x_0)$

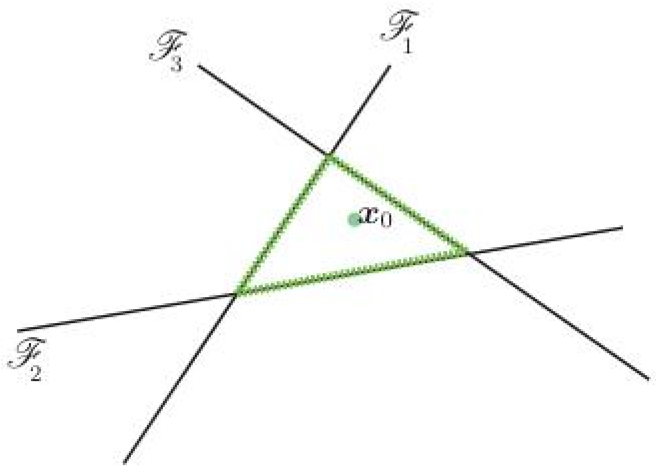


Figure 4: For $\boldsymbol{x}_0$ belonging to class 4, let $\mathscr{F}_k = \{\boldsymbol{x} : f_k(\boldsymbol{x}) - f_4(\boldsymbol{x}) = 0\}$. These hyperplanes are depicted in solid lines and the boundary of P is shown in green dotted line.

DeepFool for multiclass classifiers

- Extend the DeepFool to the general case of multiclass differentiable classifiers
- Approximate the set P at iteration i by a Polyhedron $\widetilde{P}_i$

$$\tilde{P}_i = \bigcap_{k=1}^{c} \Big\{ \boldsymbol{x} : f_k(\boldsymbol{x}_i) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i) + \nabla f_k(\boldsymbol{x}_i)^\top \boldsymbol{x} - \nabla f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)^\top \boldsymbol{x} \le 0 \Big\}$$

- At iteration i ,the distance between x_i and the complement of P,$dist(x_i, P^c)$,by $dist(x_i, \widetilde{P}_i^c)$

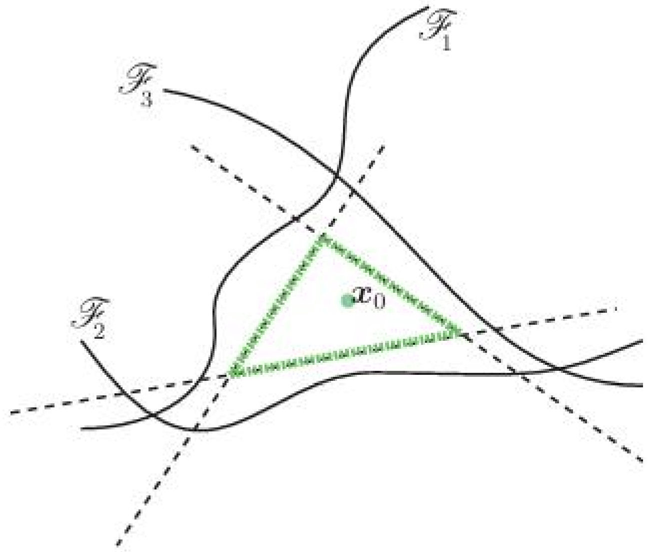


Figure 5: For $\boldsymbol{x}_0$ belonging to class 4, let $\mathscr{F}_k = \{\boldsymbol{x} : f_k(\boldsymbol{x}) - f_4(\boldsymbol{x}) = 0\}$. The linearized zero level sets are shown in dashed lines and the boundary of the polyhedron $\tilde{P}_0$ in green.

DeepFool for multiclass classifiers

- Extend the DeepFool to the general case of multiclass differentiable classifiers
- Approximate the set P at iteration i by a Polyhedron $\widetilde{P}_i$

$$\tilde{P}_i = \bigcap_{k=1}^{c} \Big\{ \boldsymbol{x} : f_k(\boldsymbol{x}_i) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i) + \nabla f_k(\boldsymbol{x}_i)^\top \boldsymbol{x} - \nabla f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)^\top \boldsymbol{x} \le 0 \Big\}$$

- At iteration i ,the distance between x_i and the complement of P,$dist(x_i, P^c)$,by $dist(x_i, \widetilde{P}_i^c)$

Algorithm 2 DeepFool: multi-class case

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

Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
while $\hat{k}(\boldsymbol{x}_i) = \hat{k}(\boldsymbol{x}_0)$ **do**
  for $k \neq \hat{k}(\boldsymbol{x}_0)$ **do**
    $\boldsymbol{w}'_k \leftarrow \nabla f_k(\boldsymbol{x}_i) - \nabla f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
    $f'_k \leftarrow f_k(\boldsymbol{x}_i) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
  end for
  $\hat{l} \leftarrow \arg\min_{k \neq \hat{k}(\boldsymbol{x}_0)} \frac{|f'_k|}{\|\boldsymbol{w}'_k\|_2}$
  $\boldsymbol{r}_i \leftarrow \frac{|f'_{\hat{l}}|}{\|\boldsymbol{w}'_{\hat{l}}\|_2^2} \boldsymbol{w}'_{\hat{l}}$
  $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$
  $i \leftarrow i + 1$
end while
return $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$