# 人工智能安全课程

## 第2讲：人工智能鲁棒性之对抗样本(一)

主讲人：王志波 杨子祺

浙江大学计算机科学与技术学院/网络空间安全学院

# 课程大纲

一 深度学习基础

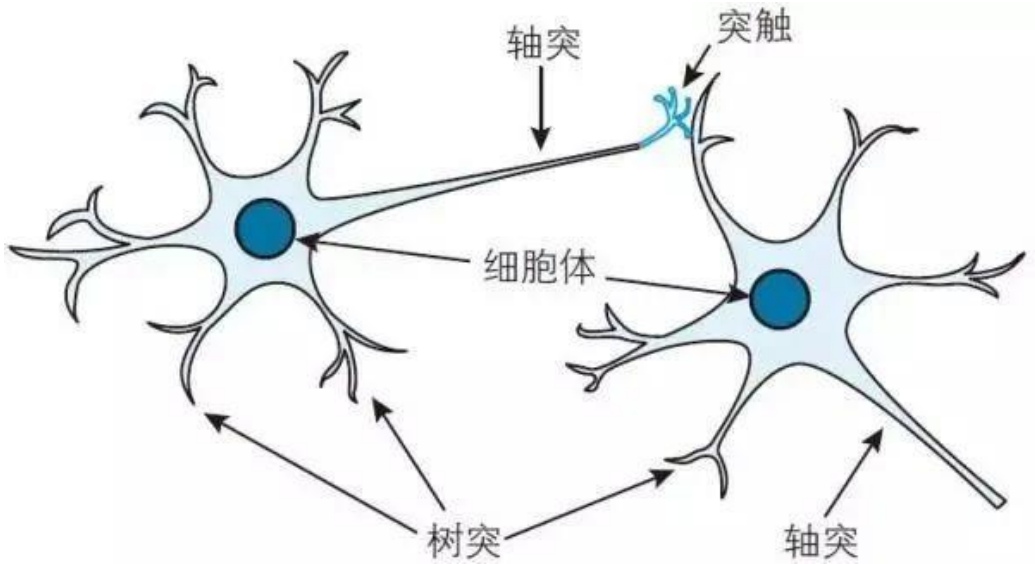
二 人工智能模型特性

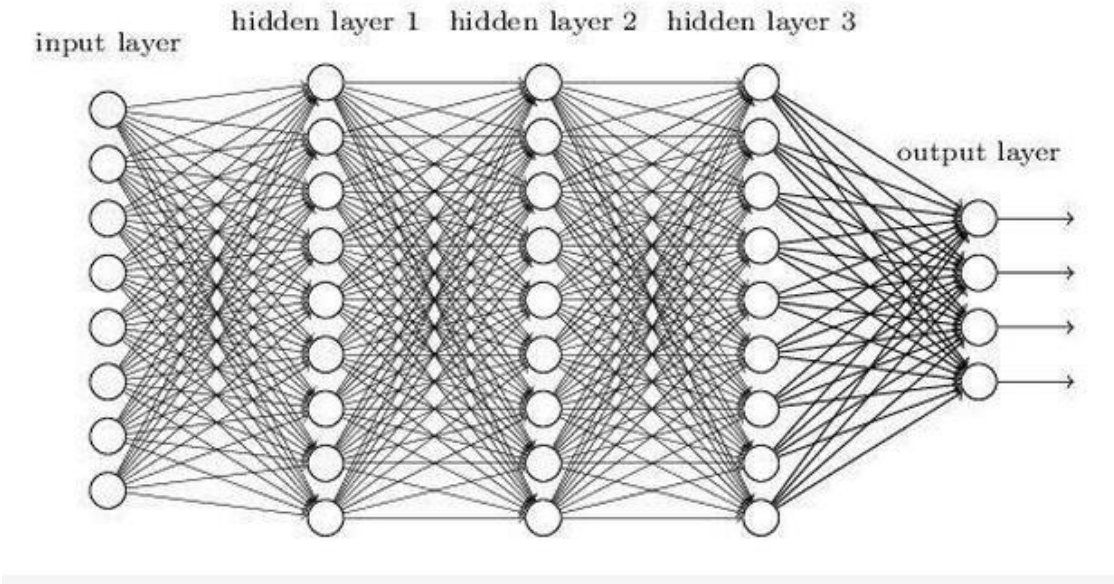
三 对抗样本攻击原理

四 经典对抗样本算法

# 神经网络

## □ 人工神经元网络（ANN或者NN）




PPT 4-39 参考文献：跟我学AI之神经网络是如何实现的—马少平（清华大学）

# 神经网络

## 从数字识别谈起

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

数字8的图像

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -1 | -1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -1 | -1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 |
| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -1 | -1 |

数字3的模式

# 神经网络

## 口 模式匹配

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |

•

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | -1 |
| -1 | -1 | -1 | -1 | 1 |
| 1 | 1 | 1 | 1 | -1 |
| -1 | -1 | -1 | -1 | 1 |
| 1 | 1 | 1 | 1 | -1 |

=

$0\times1+1\times1+1\times1+1\times1+0\times(-1)+$
$1\times(-1)+0\times(-1)+0\times(-1)+0\times(-1)+1\times1+$
$0\times1+1\times1+1\times1+1\times1+0\times(-1)+$
$1\times(-1)+0\times(-1)+0\times(-1)+0\times(-1)+1\times1+$
$0\times1+1\times1+1\times1+1\times1+0\times(-1)$
$=9$

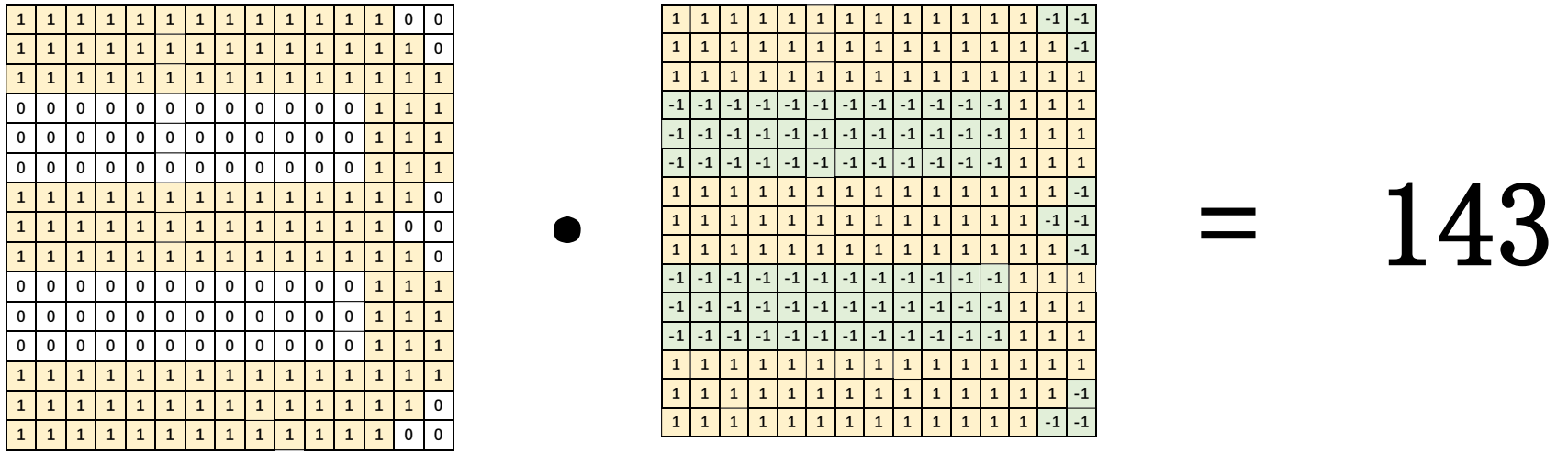
$$net = w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_n \cdot x_n$$

其中：$w_i(i = 1,2,\ldots,n)$：表示数字模式

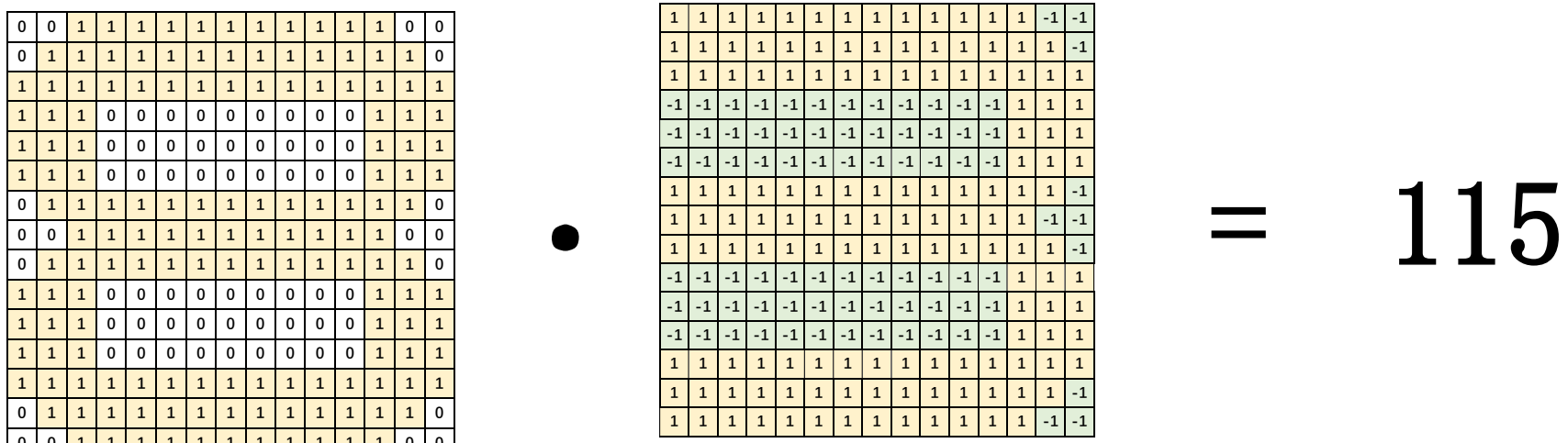
$x_i(i = 1,2,\ldots,n)$：表示数字图像

# 神经网络

## □ 数字3、数字8与模式3的匹配



$$\bullet \quad = \quad 143$$



$$\bullet \quad = \quad 115$$

# 神经网络

## 存在的问题

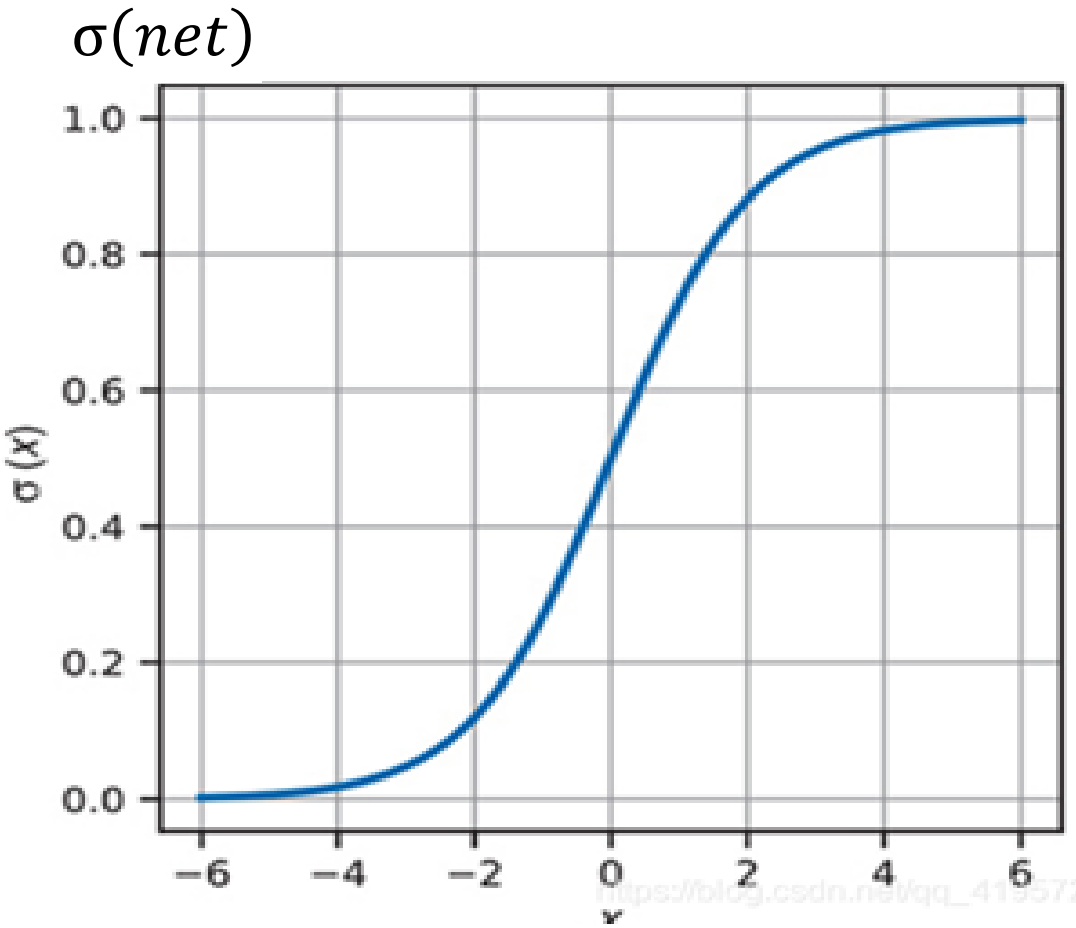
- 笔画多少带来的问题
  - 1的笔画少
  - 8的笔画多
- 如何评判匹配的程度

# 神经网络

## 使用Sigmoid函数

- 使用sigmoid函数将匹配结果变化到0~1之间

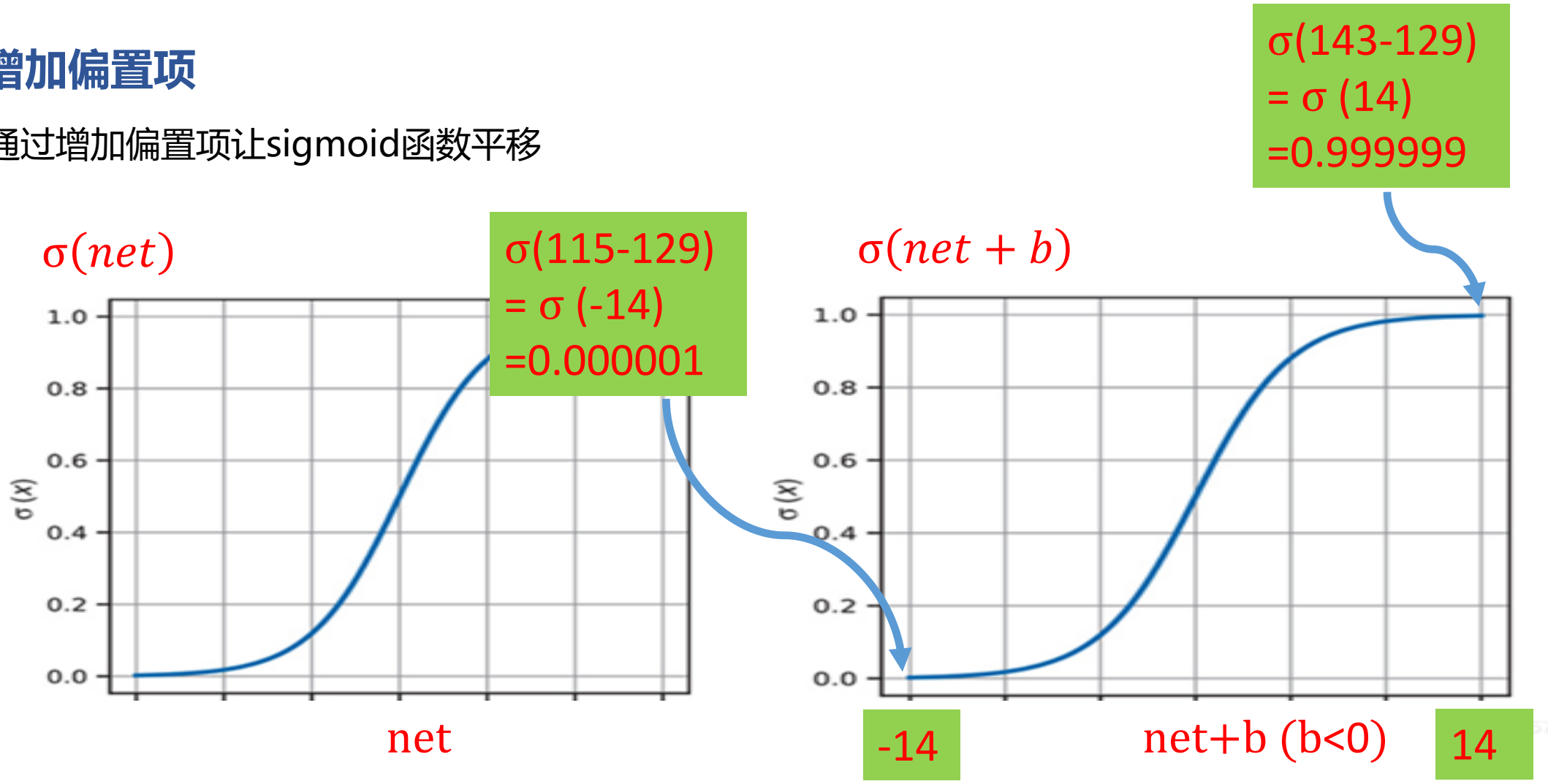
$$\sigma(net) = \frac{1}{1+e^{-net}}$$



# 神经网络

## 增加偏置项

- 通过增加偏置项让sigmoid函数平移



# 神经网络

## 口 增加偏置项

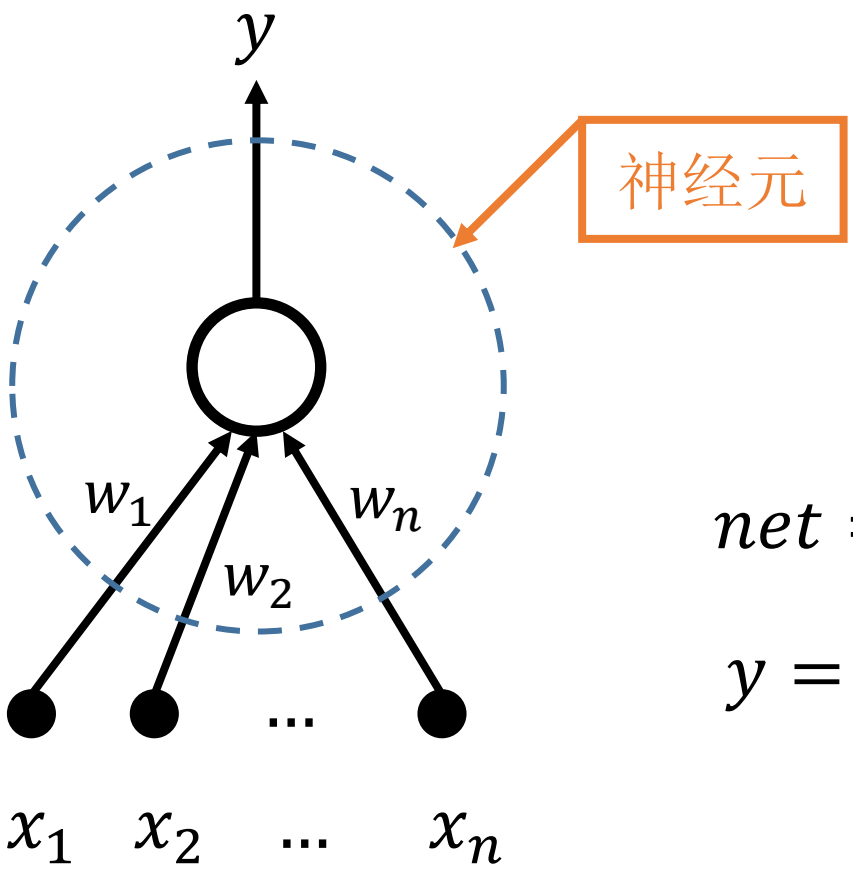
■ 模式匹配

$$net = w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_n \cdot x_n + b$$

$$y = sigmoid(net)$$

# 神经网络

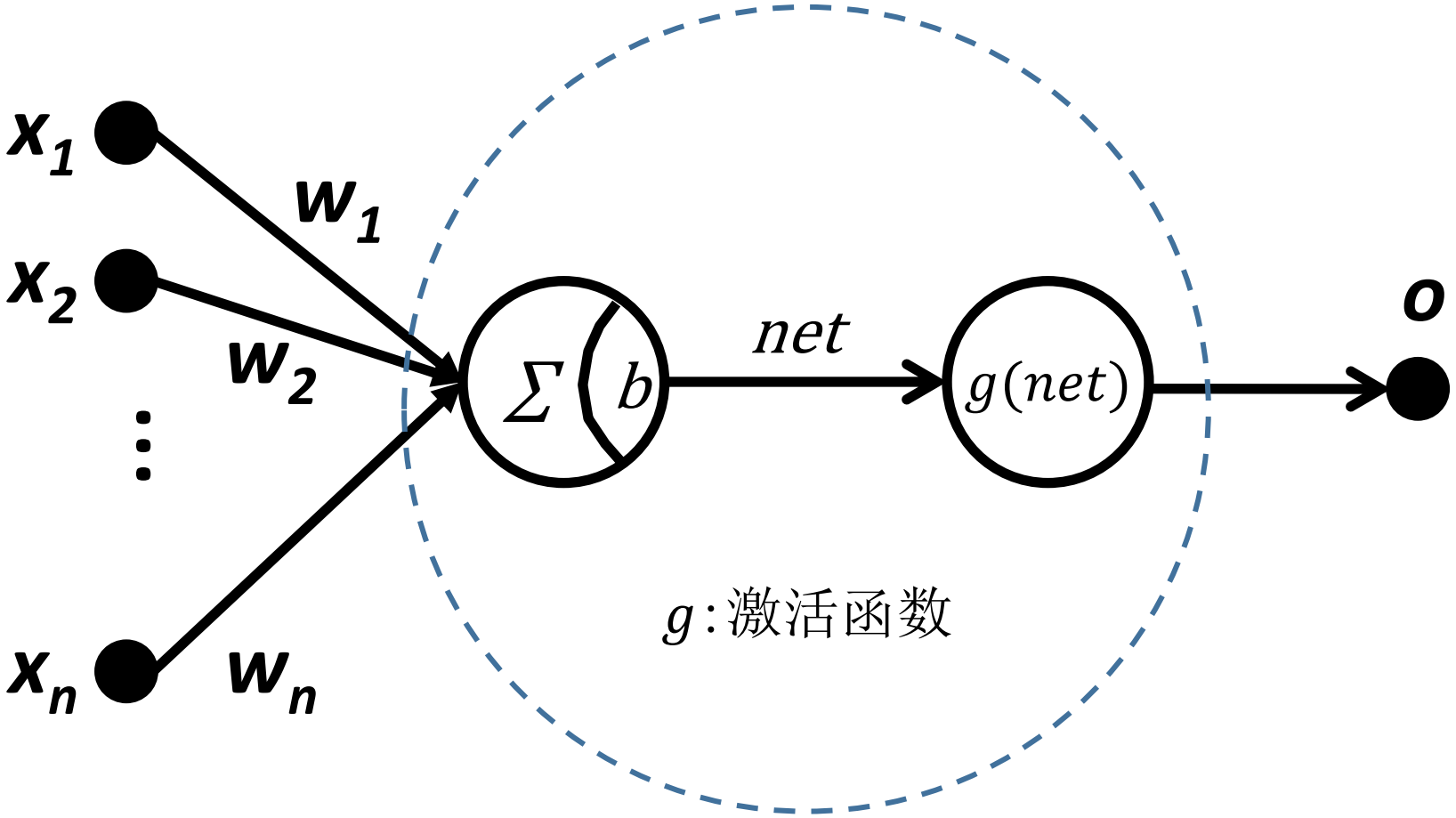
## 图示表示




$$net = w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_n \cdot x_n + b$$

$$y = sigmoid(net)$$

# 神经元与神经网络



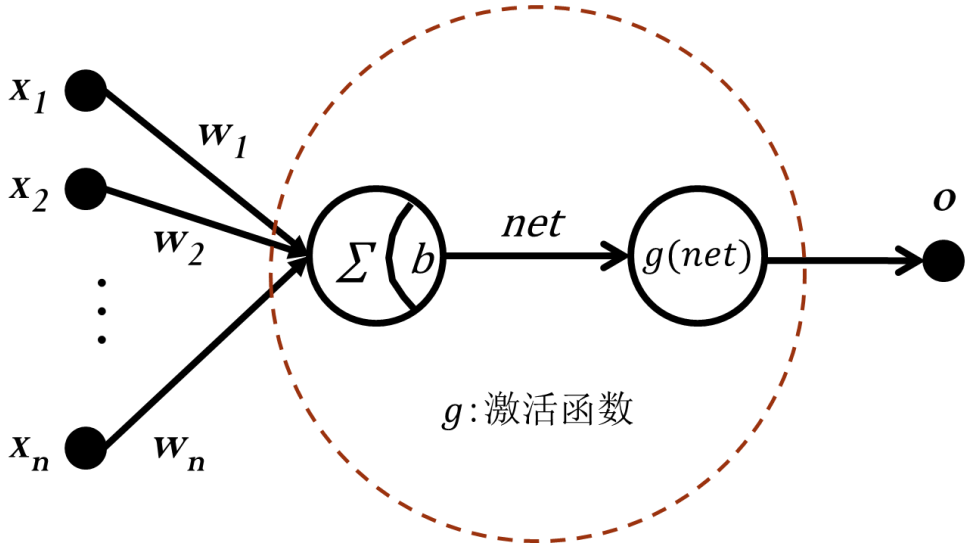
# 神经元与神经网络

## 口 神经元

$$net = w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_n \cdot x_n + b$$

$$= \sum_{i=1}^{n} w_i \cdot x_i + b$$

$$o = g(net)$$




令：$x_0 = 1, \quad w_0 = b$

则：$net = w_0 \cdot x_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_n \cdot x_n$

$$= \sum_{i=0} w_i \cdot x_i$$

# 神经元与神经网络

## 神经元的向量表示

- 输入：$x = [x_0, x_1, \ldots, x_n]$
- 权重：$w = [w_0, w_1, \ldots, w_n]$

$$net = w \cdot x$$

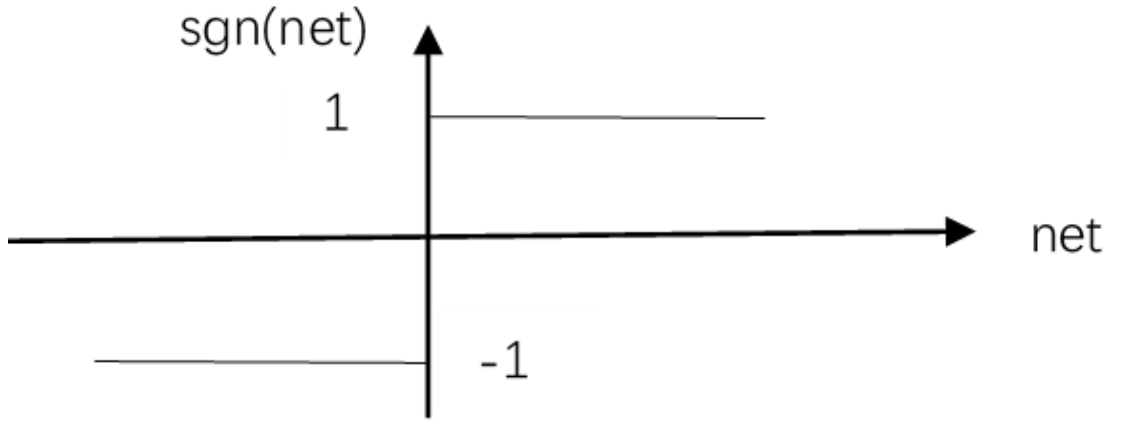
$$o = g(net)$$

# 神经元与神经网络

## 激活函数

- 符号函数

$$g(net) = \mathrm{sgn}(net) = \begin{cases} 1, & \text{当} net \geq 0 \\ -1 & \text{当} net < 0 \end{cases}$$

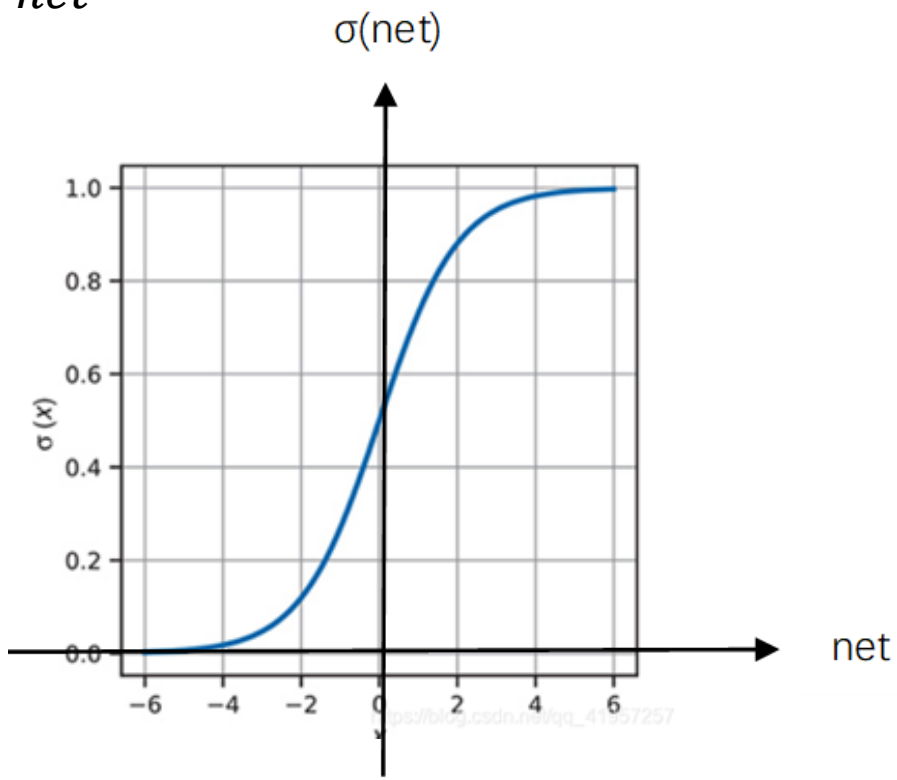


# 神经元与神经网络

## 激活函数

- Sigmoid函数

$$g(net) = \sigma(net) = \frac{1}{1 + e^{-net}}$$

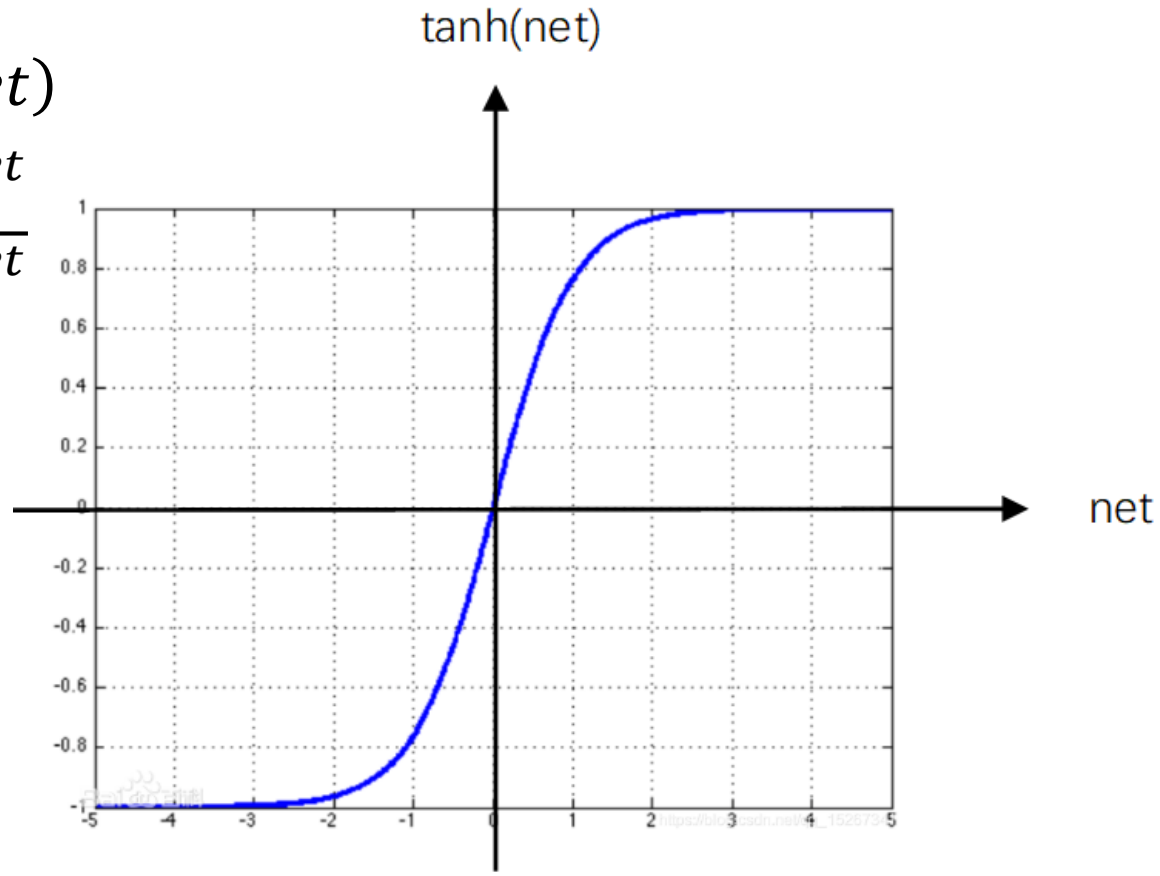


# 神经元与神经网络

## 激活函数

- 双曲正切函数

$$g(net) = \tanh(net) = \frac{e^{net} - e^{-net}}{e^{net} + e^{-net}}$$

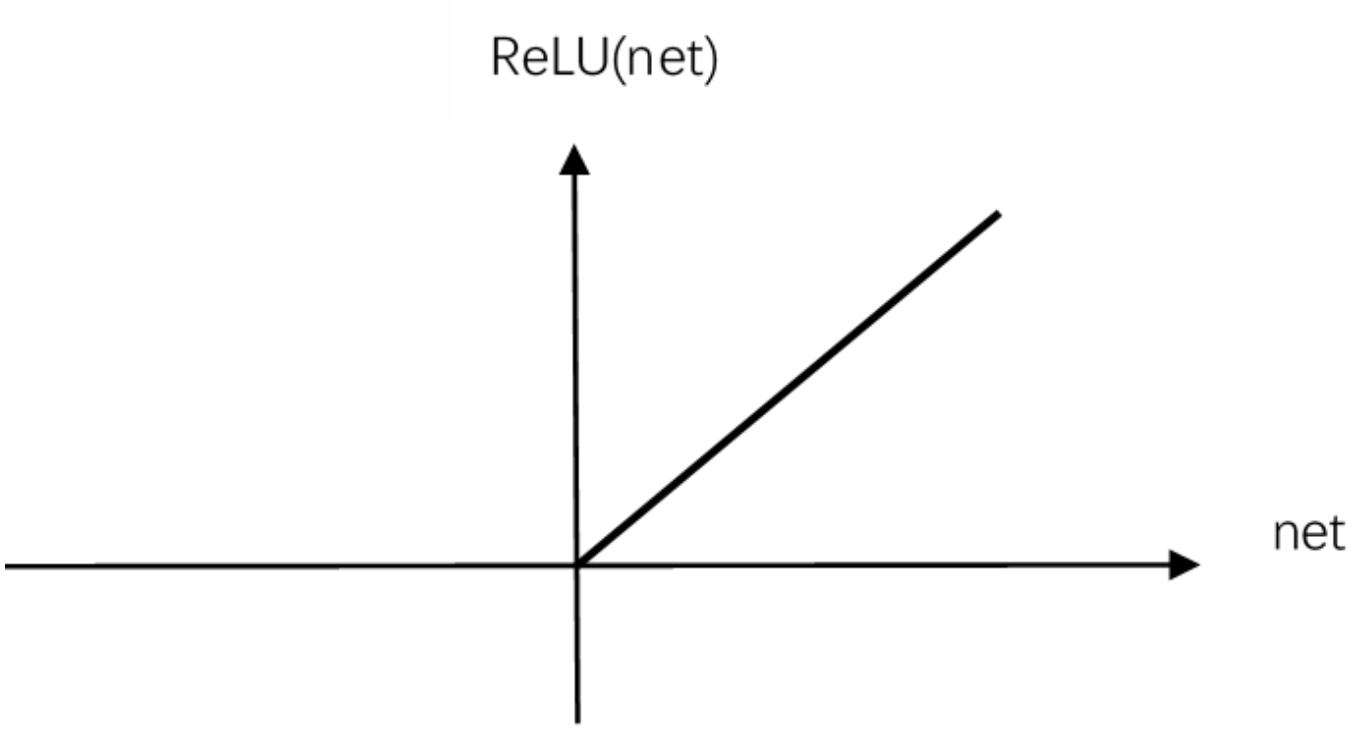


# 神经元与神经网络

## 激活函数

- 线性整流函数

$$g(net) = \mathrm{ReLU}(net) = \max(0, net)$$



# 神经元与神经网络

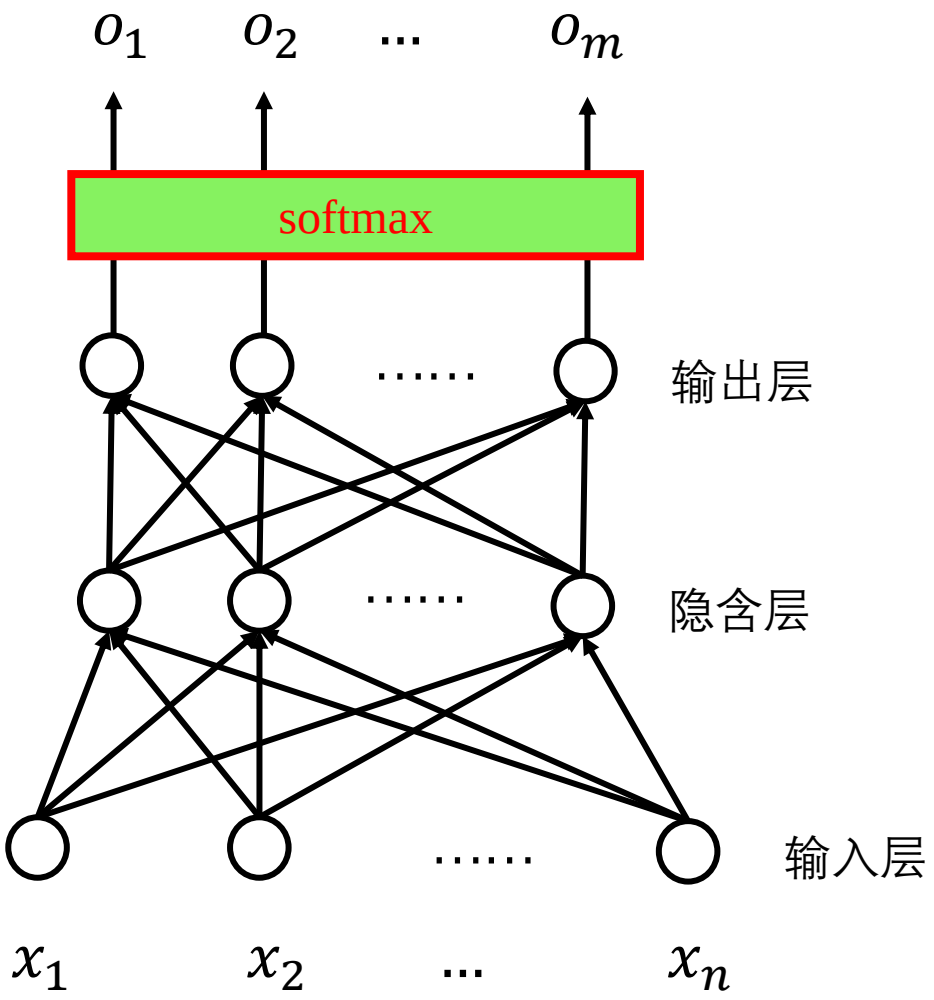
## 激活函数

- Softmax函数

$$o_k = \frac{e^{net_k}}{\sum_{i=1}^{m} e^{net_i}}$$

$$0 \le o_k \le 1$$

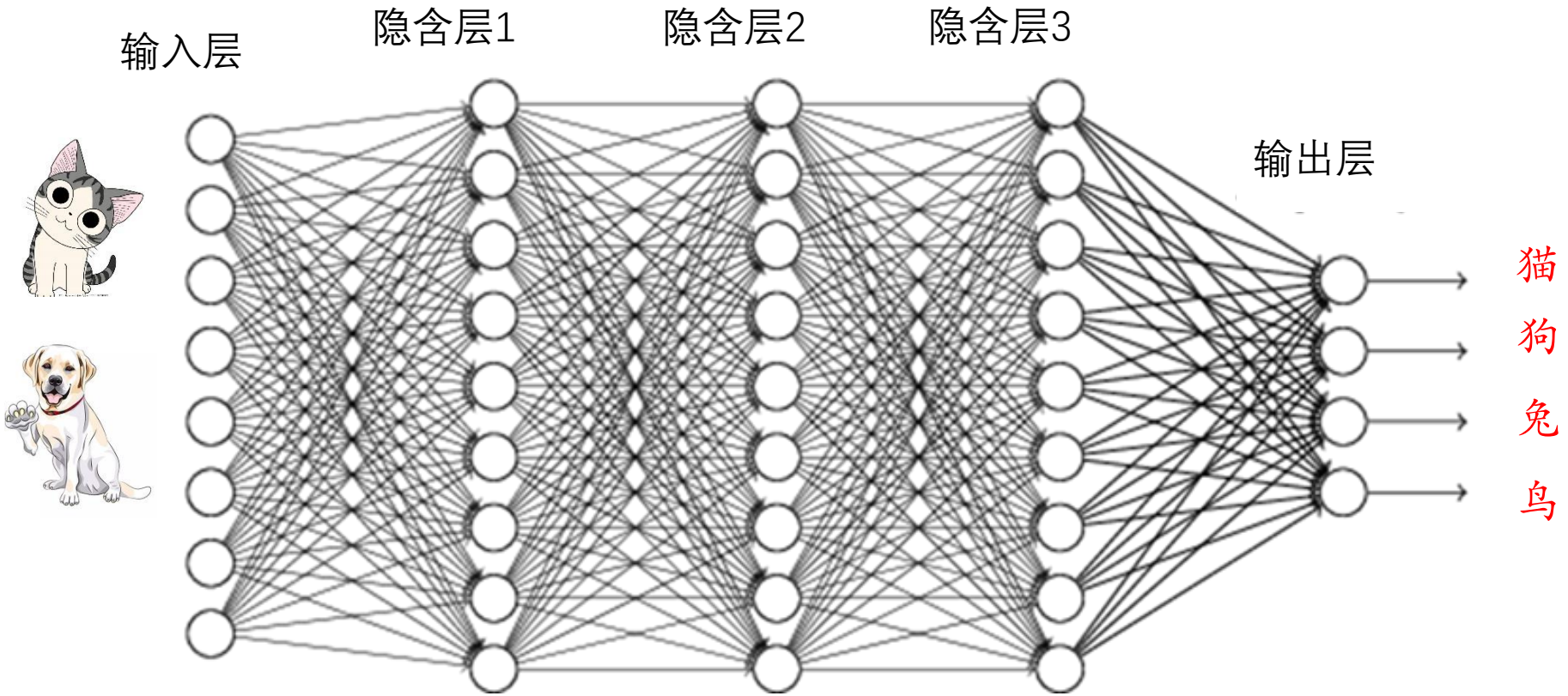
$$\sum_{k=1}^{m} o_k = 1$$



# 神经元与神经网络

## 口 全连接网络

■ 术语：全连接网络，前馈网络，多层感知机，全连接层，稠密层



# 神经网络如何训练

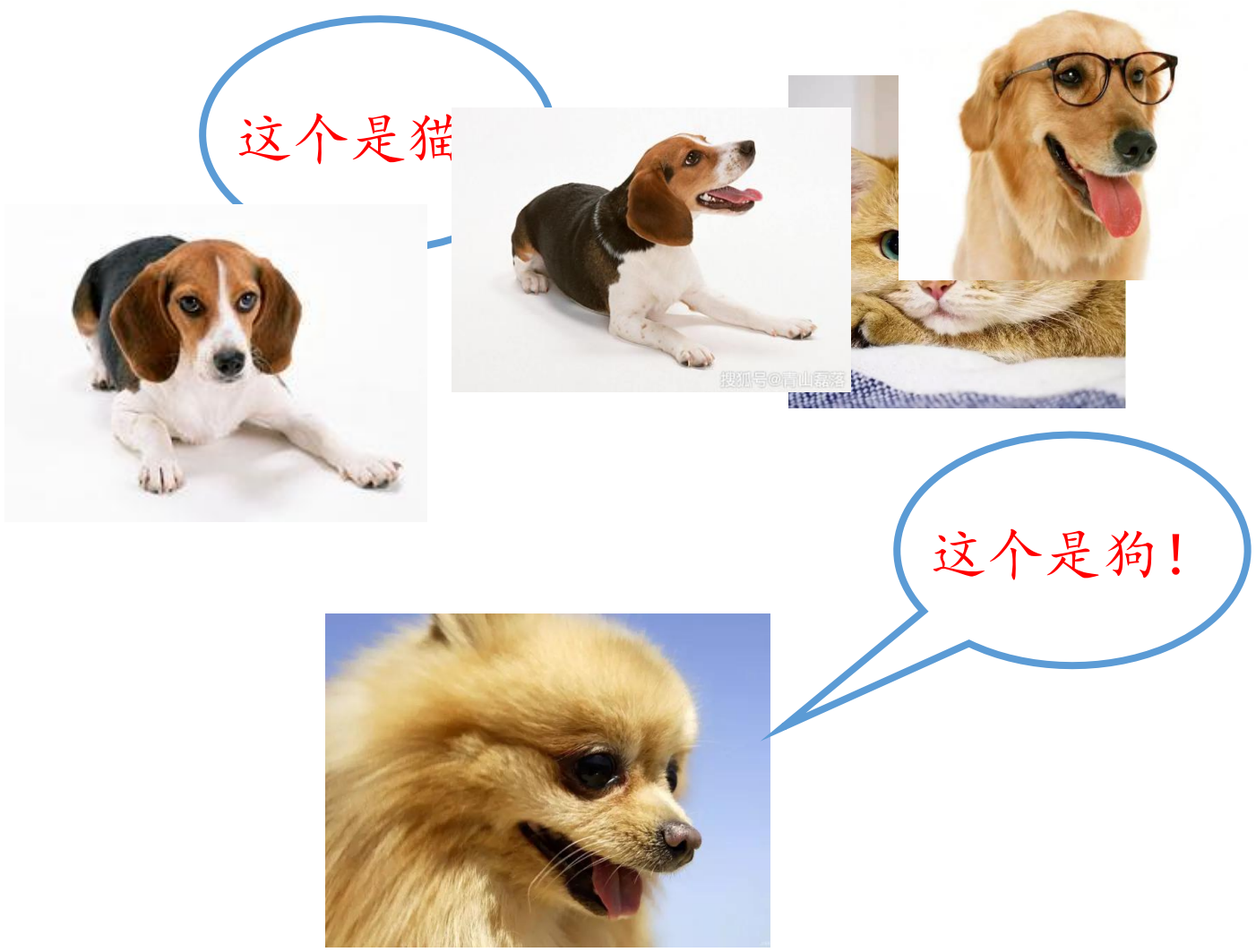
浙江大学 ZHEJIANG UNIVERSITY

## □ 小朋友如何认识小动物

# 神经网络如何训练

## 建立数据集

- 收集各种动物的照片
  - 不同姿势
  - 不同角度
  - 不同大小
- 数据标注
  - 每张照片标注上动物的名字
- 训练集与测试集
  - 样本



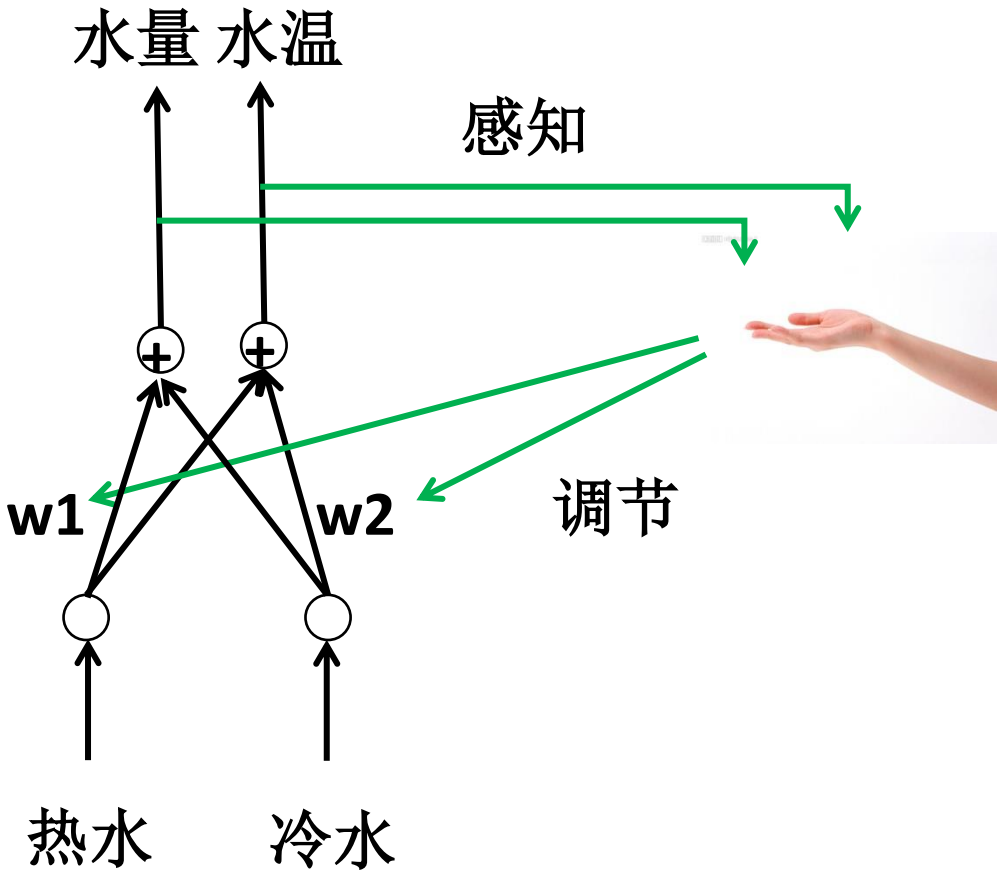
# 神经网络如何训练

浙江大学 ZHEJIANG UNIVERSITY

## 如何训练

# 神经网络如何训练

## 淋雨器示意图



# 神经网络如何训练

## 口 如何评价调节效果

■ 损失函数

$$E(\text{阀门}) = \left(t_{\text{水温}} - o_{\text{水温}}\right)^2 + \left(t_{\text{水量}} - o_{\text{水量}}\right)^2$$

$t_{\text{水温}}$：希望水温

$t_{\text{水量}}$：希望水量

$o_{\text{水温}}$：实际水温

$o_{\text{水量}}$：实际水量

# 神经网络如何训练

## 口 损失函数——误差平方和

■ 对样本d的误差

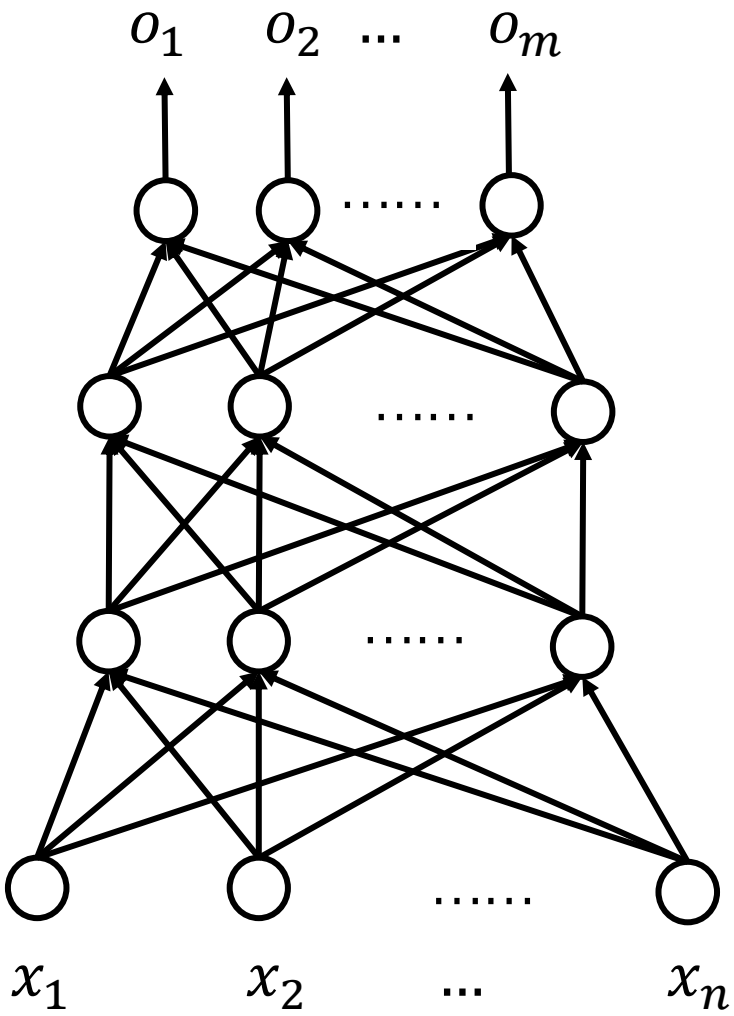
$$\mathrm{E}_d(w) = \frac{1}{2}\sum_{k=1}^{m}(t_{kd} - o_{kd})^2$$

■ 对所有样本的误差

$$E(w) = \frac{1}{2}\sum_{d=1}^{n}\mathrm{E}_d(\mathrm{w})$$
$$= \frac{1}{2}\sum_{d=1}^{n}\sum_{k=1}^{m}(t_{kd} - o_{kd})^2$$

■ $t_{kd}$：对应样本d的第k个希望输出

■ $o_{kd}$：对应样本d的第k个实际输出



# 神经网络如何训练

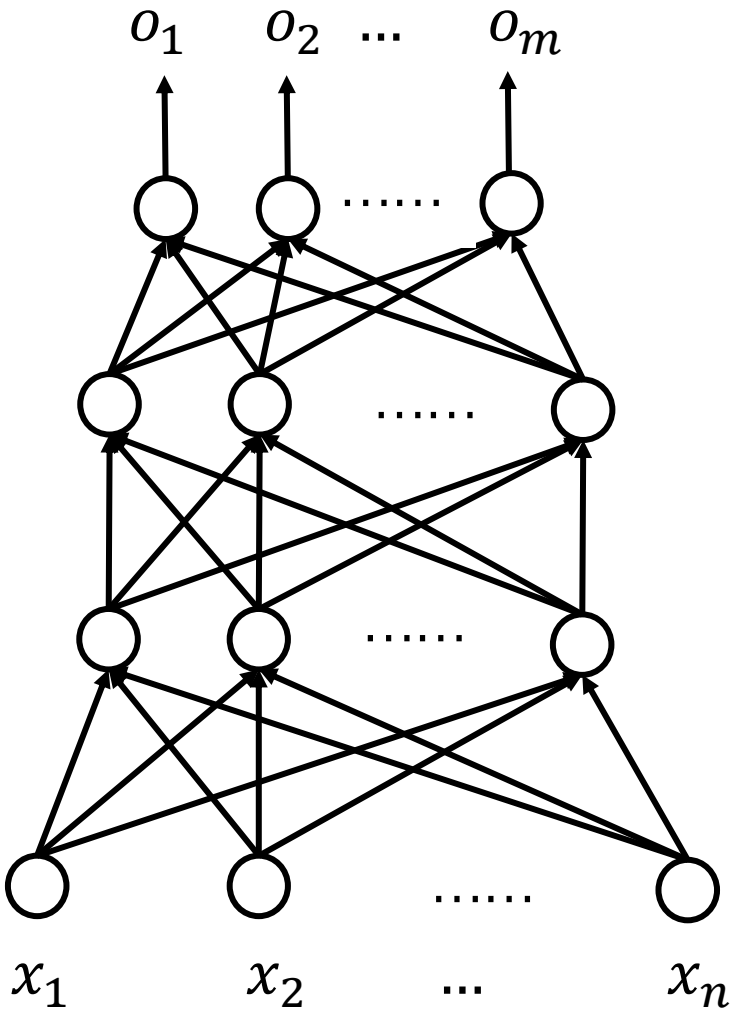
## 如何训练

- 损失函数最小化问题

$$\mathrm{E}_d(w) = \frac{1}{2}\sum_{k=1}^{m}(t_{kd} - o_{kd})^2$$

$$E(w) = \frac{1}{2}\sum_{d=1}^{n}\mathrm{E}_d(\mathrm{w})$$

$$= \frac{1}{2}\sum_{d=1}^{n}\sum_{k=1}^{m}(t_{kd} - o_{kd})^2$$



# 神经网络如何训练

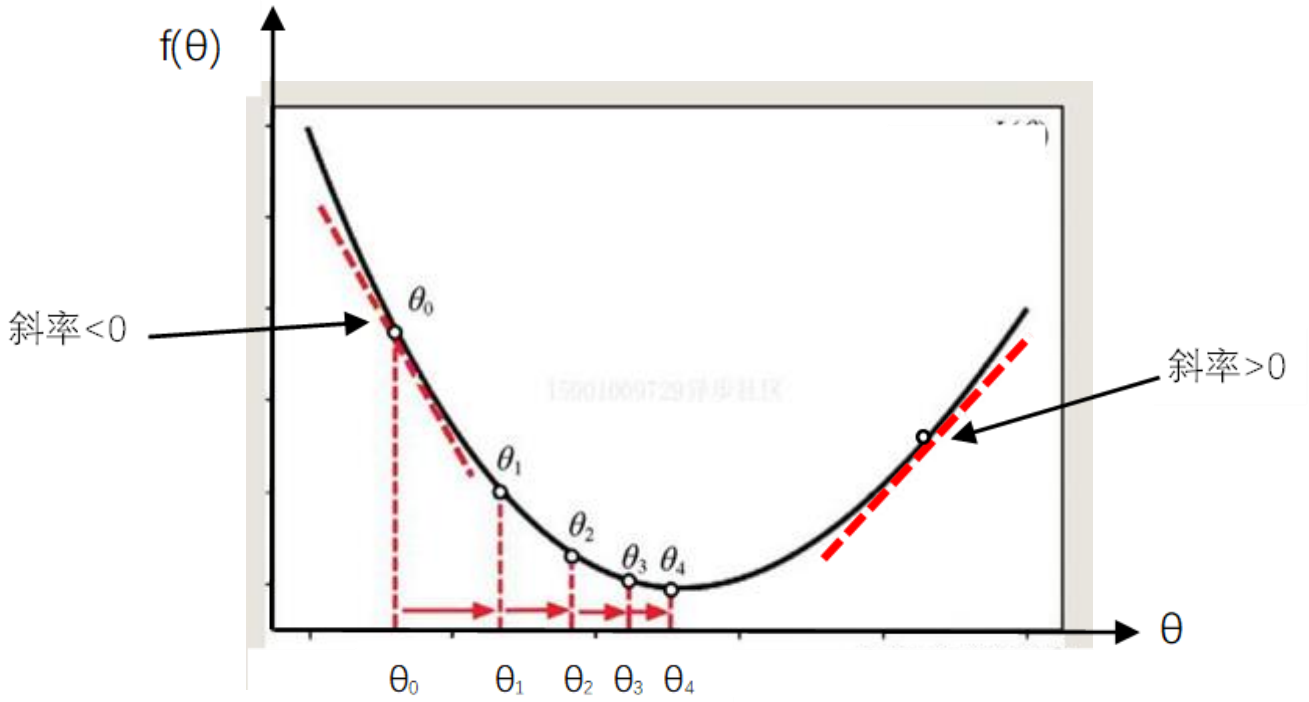
## 口 梯度下降法

- 求$f(\theta)$的最小值
- 迭代逼近

$$\theta_{i+1} = \theta_i + \triangle\,\theta_i$$

- 两个问题：
  - 修改量的大小
  - 修改的方向

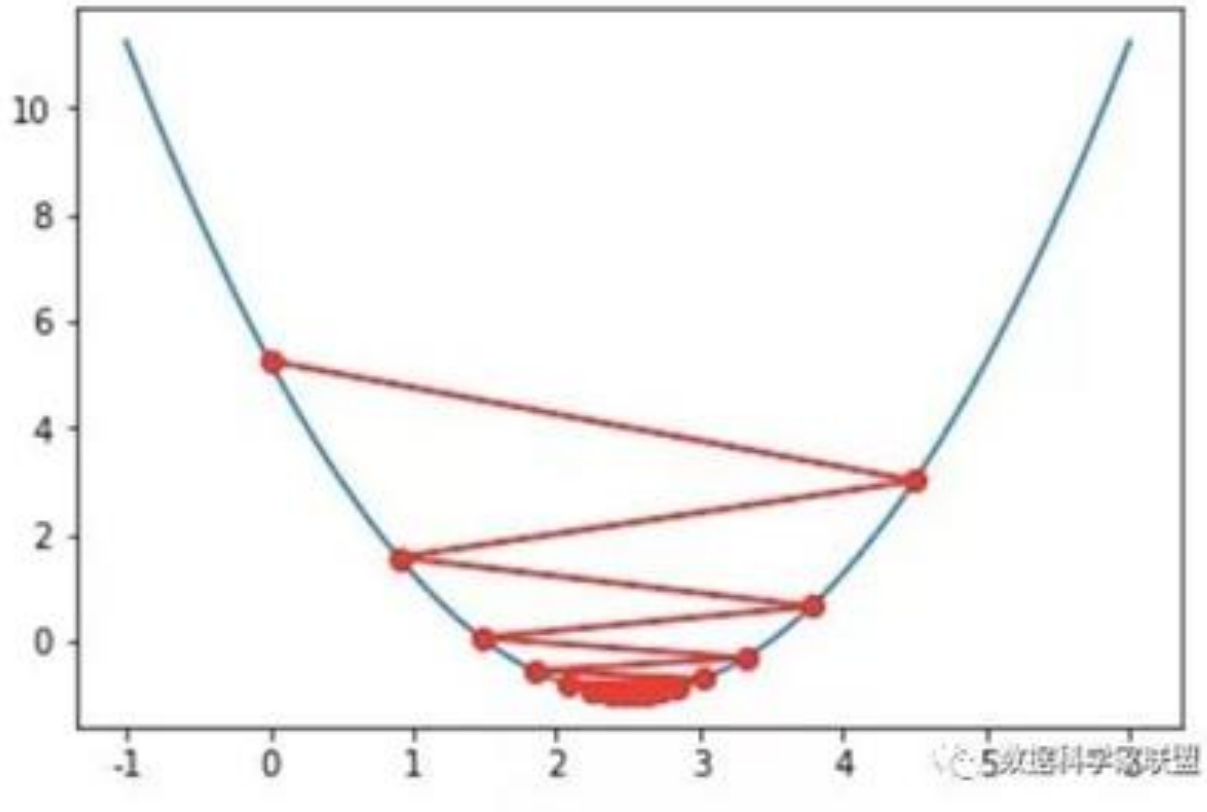
$$\triangle\,\theta_i = -\frac{df}{d\theta}$$



# 神经网络如何训练

## 梯度下降法

- 震荡问题
- 步长$\eta$
- $\triangle \theta_i = -\eta \frac{df}{d\theta}$



# 神经网络如何训练

## 梯度下降法

- 迭代公式

$$w_i^{new} = w_i^{old} + \triangle w_i$$

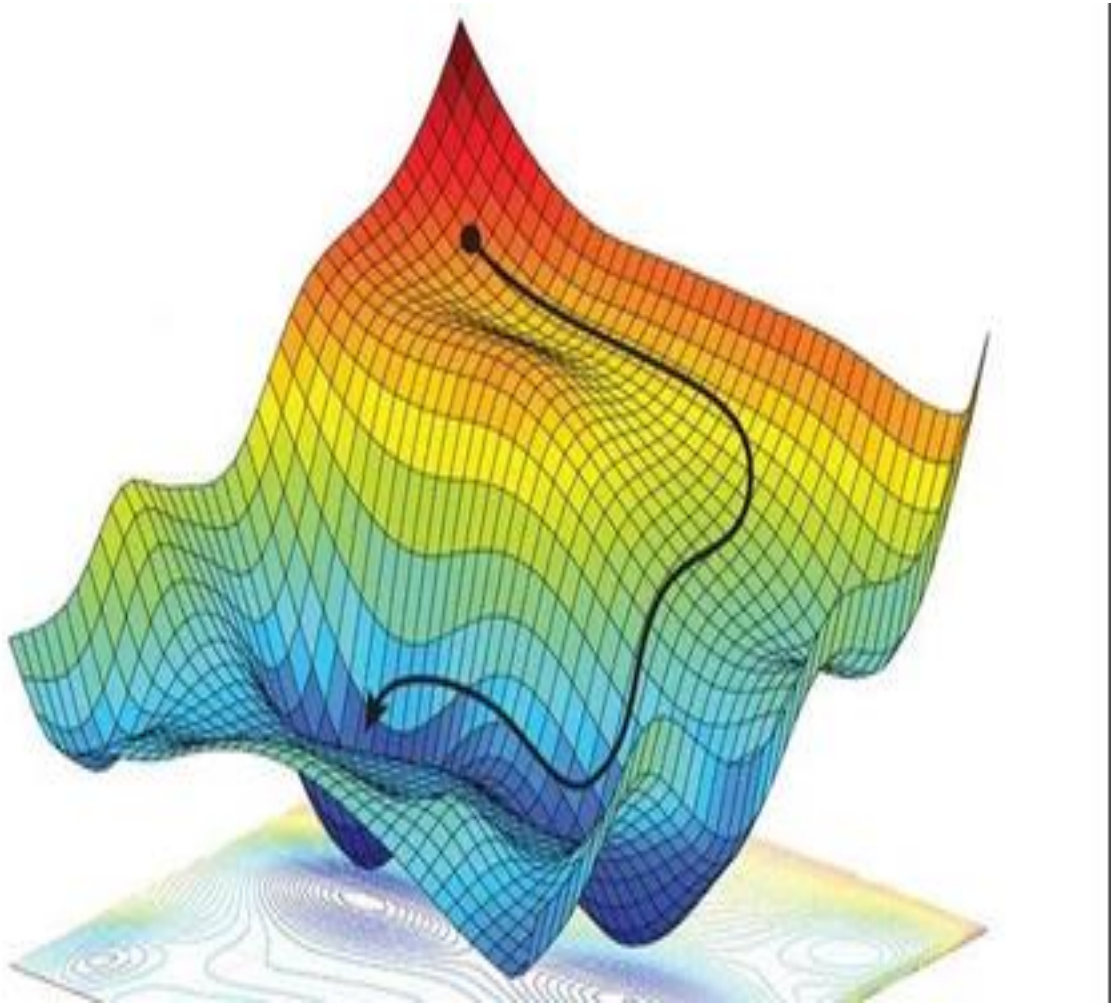
$$\triangle w_i = -\eta \frac{\partial E(w)}{\partial w_i}$$

- 梯度

$$\nabla_w E(w) = \left[\frac{\partial E(w)}{\partial w_1}, \frac{\partial E(w)}{\partial w_2}, \ldots, \frac{\partial E(w)}{\partial w_n}\right]$$

$$w^{new} = w^{old} + \triangle w$$

$$\triangle w = -\eta \nabla_w E(w)$$



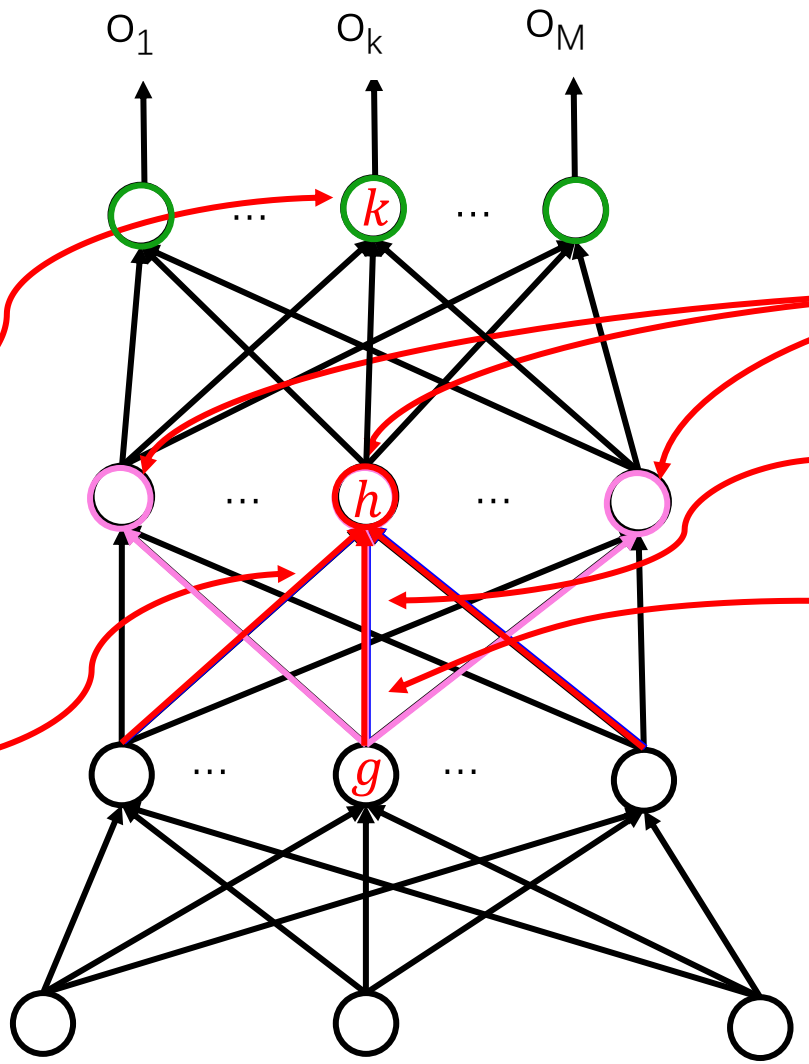
# 神经网络如何训练

## 梯度下降法

- 批量梯度下降算法 $E(w)=\frac{1}{2}\sum_{d=1}^{n}\sum_{k=1}^{m}(t_{kd}-o_{kd})^2$
  - 每次处理全部样本
- 随机梯度下降算法 $\mathrm{E}_d(w)=\frac{1}{2}\sum_{k=1}^{m}(t_{kd}-o_{kd})^2$
  - 每次处理一个样本
- 小批量梯度下降算法 $E(w)=\frac{1}{2}\sum_{d=k}^{k+B}\sum_{k=1}^{m}(t_{kd}-o_{kd})^2$
  - 每次处理小批量的样本

# 神经网络如何训练

## 符号说明




$t_k$：k的希望输出

$o_k$：k的实际输出

$x_{hi}$：h的第i个输入

$w_{hi}$：h的第i个权重

g的后续

$$w_{hi}^{new} = w_{hi}^{old} + \triangle w_{hi}$$

$$\triangle w_{hi} = -\eta \frac{\partial E(w)}{\partial w_{hi}}$$

$$\frac{\partial E(w)}{\partial w_{hi}} = -\delta_h x_{hi}$$

$$\triangle w_{hi} = \eta \delta_h x_{hi}$$

# 神经网络如何训练

## 口 反向传播算法（BP:Back Propagation ）

- 又称作误差反向传播算法
- 给出了一种计算偏导数的方法

# 神经网络如何训练

## 口 反向传播算法（BP）

3，计算输出层神经元的 $\delta_k$

$$\delta_k = (t_k - o_k)o_k(1 - o_k)$$

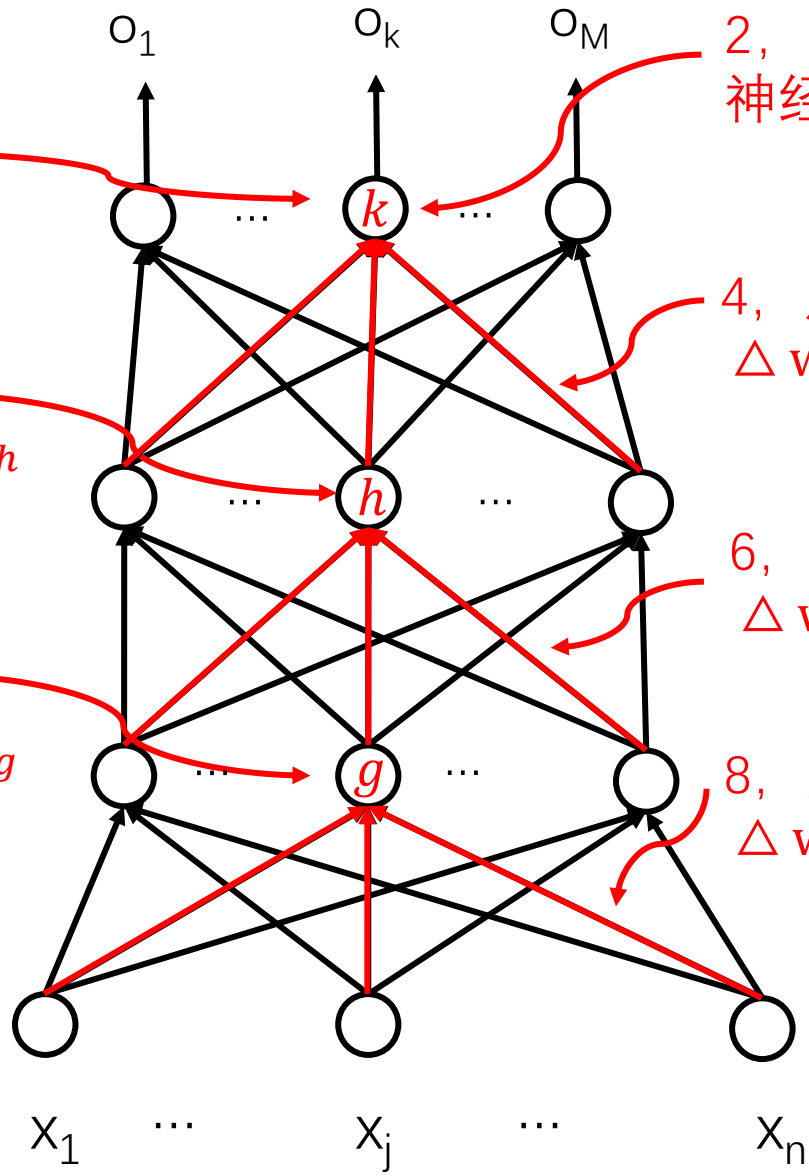
5，用 $\delta_k$ 计算隐含层2的 $\delta_h$

$$\delta_h = o_h(1 - o_h) \sum_{k\in后续(h)} \delta_k w_{kh}$$

7，用 $\delta_h$ 计算隐含层2的 $\delta_g$

$$\delta_g = o_g(1 - o_g) \sum_{k\in后续(g)} \delta_k w_{kg}$$

1. 输入一个样本计算所有神经元的输出




2，得到输出层神经元k的输出

4，用 $\delta_k$ 值更新权重

$$\triangle w_{ki} = \eta\delta_k x_{ki},\ w_{ki} = w_{ki} + \triangle w_{ki}$$

6，用 $\delta_h$ 值更新权重

$$\triangle w_{hi} = \eta\delta_h x_{hi},\ w_{hi} = w_{hi} + \triangle w_{hi}$$

8，用 $\delta_g$ 值更新权重

$$\triangle w_{gi} = \eta\delta_g x_{gi},\ w_{gi} = w_{gi} + \triangle w_{gi}$$

# 神经网络如何训练

## 该BP算法的条件

- 具体条件不同，算法会有差异，总体思路一样
- 全连接网络
  - 其他形式的网络
- 随机梯度下降算法
  - 批量、小批量梯度下降算法
- 激活函数：sigmoid函数
  - 其他的激活函数
- 损失函数：误差平方和
  - 其他的损失函数

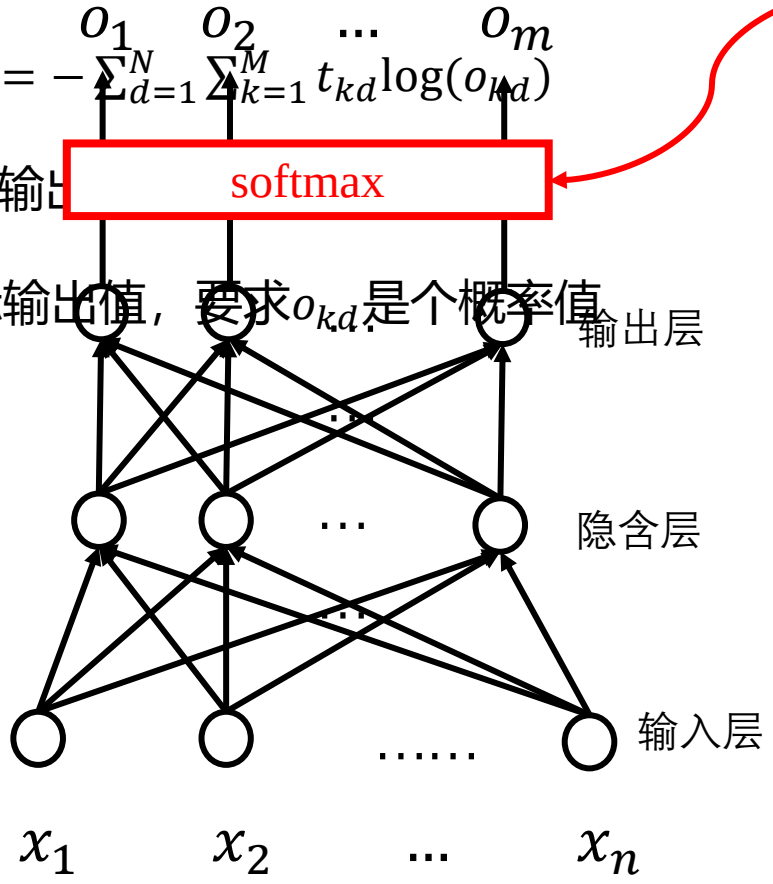
# 神经网络如何训练

## 交叉熵损失函数

- $\mathrm{H}_d(w) = -\sum_{k=1}^{M} t_{kd}\log(o_{kd})$
- $H(w) = \sum_{d=1}^{N} \mathrm{H}_d(\mathrm{w}) = -\sum_{d=1}^{N}\sum_{k=1}^{M} t_{kd}\log(o_{kd})$

$t_{kd}$：样本d对应的希望输出

$o_{kd}$：样本d对应的实际输出值，要求$o_{kd}$是个概率值




$$o_k = \frac{e^{net_k}}{\sum_{i=1}^{m} e^{net_i}}$$

$$0 \le o_k \le 1$$

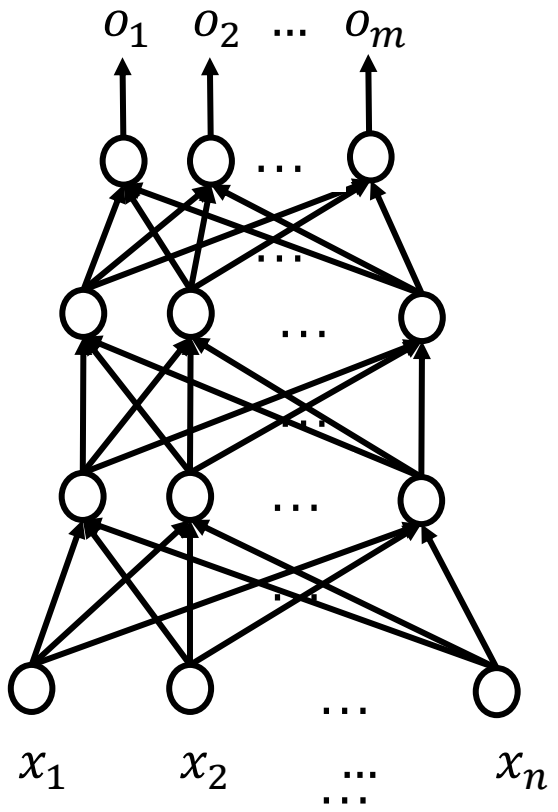
$$\sum_{k=1}^{m} o_k = 1$$

# 神经网络如何训练

## 交叉熵损失函数

- 对于分类问题：
- 对于给定输入样本d，只有d对应的希望输出为1，其他为0

$$\mathrm{H}_d(w) = -\log(o_{kd})$$



# 神经网络如何训练

## 两种损失函数的作用

- 误差平方和损失函数
  - 用于输出是具体数值的问题
- 交叉熵损失函数
  - 用于分类问题

# 优化方法

## 机器学习优化问题

- 几乎所有的机器学习算法最后都归结为求一个目标函数的极值，即优化问题
- 机器学习的核心步骤：

1. 给出一个模型（一般是映射函数）
2. 定义对这个模型好坏的评价函数（目标函数、损失函数）
3. 求解目标函数的极大值或者极小值，以确定模型的参数，从而得到我们想要的模型。

# 优化方法

## Jacobian矩阵

- 有时我们需要计算输入和输出都为向量的函数的所有偏导数。包含所有这样的偏导数的矩阵被称为**Jacobian矩阵**。具体来说，如果我们有一个函数$f: R^m \to R^n$, $f$的Jacobian矩阵$\boldsymbol{J} \in R^{n*m}$定义为

$$\boldsymbol{J}_{i,j} = \frac{\partial}{\partial x_j} f(x)_i$$

## Hessian矩阵

- 当函数具有多维输入时，二阶导数也有很多。我们可以将这些导数合并成一个矩阵，称为**Hessian矩阵**。Hessian矩阵$\boldsymbol{H}(f)(\boldsymbol{x})$定义为

$$\boldsymbol{H}(f)(\boldsymbol{x})_{i,j} = \frac{\partial^2}{\partial x_i \partial x_j} f(\boldsymbol{x})$$

- Hessian等价于梯度的Jacobian矩阵。微分算子在任何二阶偏导连续的点处可交换，也就是说它们的顺序可换

$$\frac{\partial^2}{\partial x_i \partial x_j} f(\boldsymbol{x}) = \frac{\partial^2}{\partial x_j \partial x_i} f(\boldsymbol{x})$$

PPT 49-53 参考文献：《Deep Learning》P56-P61

# 优化方法

## 口 二阶优化算法

■ 仅使用梯度信息的优化算法称为一阶优化算法，如梯度下降，随机梯度下降等。使用Hessian矩阵的优化算法称为二阶优化算法，如牛顿法，拟牛顿法。

- 牛顿法是二阶优化技术，利用了函数的一阶（梯度）和二阶（Hessian矩阵）导数信息，直接寻找梯度为0的点。核心思想是在某点处用二次函数来近似目标函数(二阶泰勒展开)，得到导数为0的方程，求解该方程，得到下一个迭代点。

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}_0) + (\boldsymbol{x} - \boldsymbol{x}_0)^T \nabla_x f(\boldsymbol{x}_0) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_0)^T H(f)(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0)$$

- 两边同求梯度并令为0，计算可得这个函数的驻点：

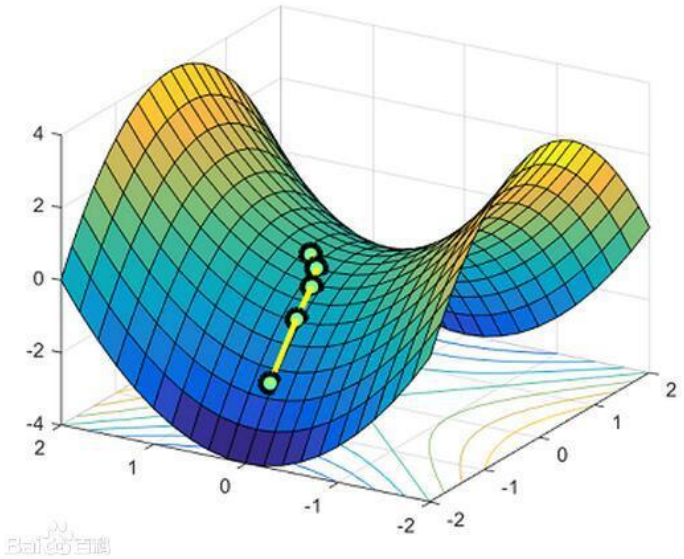
$$\boldsymbol{x}^* = \boldsymbol{x}_0 - H(f)(\boldsymbol{x}_0)^{-1} \nabla_x f(\boldsymbol{x}_0)$$

- 如果$f$是一个正定的二次函数，牛顿法只要应用一次上式就能跳到函数最小点；
- 如果$f$不是一个真正的二次函数，但在局部内近似为正定二次，牛顿法需要多次迭代上式。
- 迭代地更新近似函数和跳到近似函数的最小点可以比梯度下降更快地到达临界点。

# 优化方法

## 二阶优化算法缺陷

- 牛顿法使用的是目标函数的二阶导数，高维情况下这个矩阵非常大，计算和存储代价大是最主要的问题。
- 牛顿法容易受到鞍点或者最大值点的吸引（高维空间中鞍点激增或许解释了为什么神经网络训练中二阶方法无法成功取代梯度下降）
- 二阶方法能够更快地求得更高精度的解，但是在神经网络这类深层模型中，不高的精度对模型还有益处，能够提高模型的泛化能力。



# 优化方法

## 口 约束优化

- 有时候，在$x$的所有可能值下最大化或者最小化一个函数$f(x)$不是我们所希望的。相反，我们可能希望在$x$的某些集合$\boldsymbol{S}$中找$f(x)$ 的最大值或者最小值。**这称为约束优化。**
- 约束条件分为两种，一种是等式约束；另一种是不等式约束。
- 求解方法

- 等式约束的优化问题：拉格朗日乘数法
- 不等式约束的优化问题：KKT方法（通用方法）

# 优化方法

## 拉格朗日乘数法

- 基本思想：通过引入拉格朗日乘子来将含有n个变量和k个约束条件的约束优化问题转化为含有（n+k）个变量的无约束优化问题。
- 对于如下等式约束优化问题：

$$\begin{aligned} &minf(\boldsymbol{x}) \\ &h_i(\boldsymbol{x}) = 0, i = 1, \dots, p \end{aligned}$$

为什么等价？

- 构造拉格朗日函数：

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i=1}^{p} \lambda_i h_i(\boldsymbol{x})$$

- 在最优点处对 $\boldsymbol{x}$ 和乘子变量$\lambda_i$的导数都必须为0：

$$\begin{aligned} &\nabla_x f + \sum_{i=1}^{p} \lambda_i \nabla_x h_i = 0 \\ &h_i(\boldsymbol{x}) = 0, i = 1, \dots, p \end{aligned}$$

- 解这个方程即可得到最优解

## 拉格朗日乘数法

■ 举例：

$$minf(x,y) = x^2 + y^2$$

$$xy = 3$$

由上式子可知是一个典型的约束优化问题，约束条件是等式，可以用拉格朗日乘子。

$$L(x,y,\lambda) = x^2 + y^2 + \lambda(xy - 3)$$

原来的条件约束问题，转化为无约束方程组问题。

$$2x + \lambda y = 0$$

$$2y + \lambda x = 0$$

$$xy - 3 = 0$$

求得$\lambda = 2$或$\lambda = -2$，当$\lambda = 2$时，$x = \sqrt{3}, y = \sqrt{3}$;当$\lambda = -2$时，$x = -\sqrt{3}, y = -\sqrt{3}$

# 优化方法

## KKT方法

■ KKT方法是拉格朗日乘数法的推广，用于求解既带有等式约束，又带有不等式约束的函数极值。对于如下优化问题：

$$\begin{aligned} &minf(\boldsymbol{x}) \\ &g_i(\boldsymbol{x}) \le 0, i = 1, \dots, q \\ &h_i(\boldsymbol{x}) = 0, i = 1, \dots, p \end{aligned}$$

■ 为每个约束引入新的变量$\lambda_i$和$\mu_i$,这些新变量被称为KKT乘子。KKT条件构造如下广义拉格朗日函数:

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{x}) + \sum_{i=1}^{p} \lambda_i h_i(\boldsymbol{x}) + \sum_{j=1}^{q} \mu_j g_j(\boldsymbol{x})$$

■ 在最优解$x^*$处应该满足如下性质(即KKT条件)

① 广

② 所

③ 不等式约束显示的“互补松弛性”：$\boldsymbol{\mu} \odot \boldsymbol{g}(\boldsymbol{x}^*) = 0$

KKT条件只是确定最优点的的必要条件而不是充分条件。

① $\nabla_x L(\boldsymbol{x}^*) = 0$

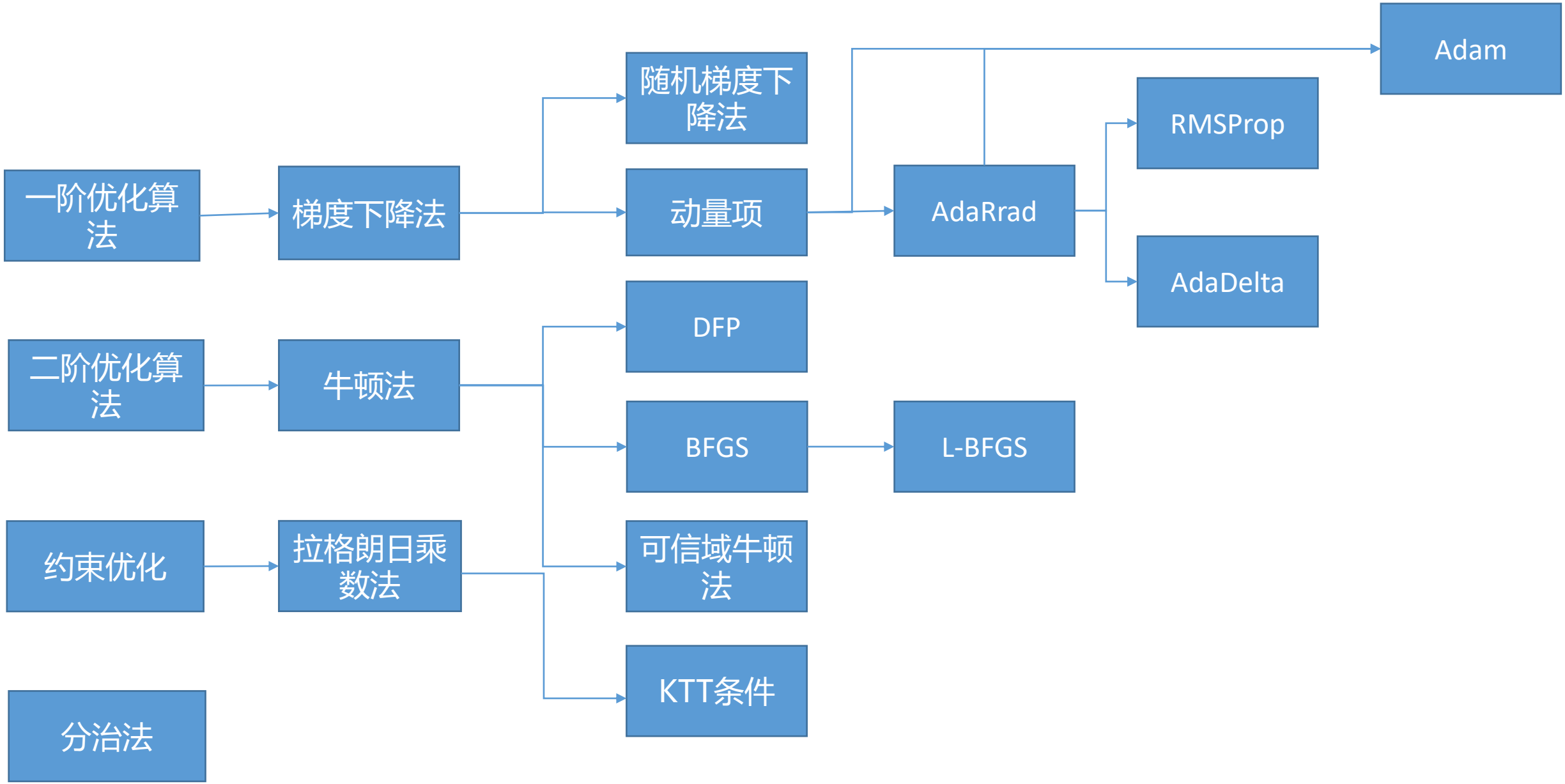
② $\begin{cases} h_i(\boldsymbol{x}^*) = 0 \\ g_i(\boldsymbol{x}^*) \le 0 \end{cases}$

③ $\mu_j g_j(\boldsymbol{x}^*) = 0$

$\boldsymbol{A} \odot \boldsymbol{B}$ 表示$\boldsymbol{A}$与$\boldsymbol{B}$的逐元素乘积

KKT:**Karush-Kuhn-Tucker**(Karush,1939; Kuhn and Tucker,1951)

# 优化方法


一阶优化算法
梯度下降法
随机梯度下降法
动量项
AdaRrad
Adam
RMSProp
AdaDelta
二阶优化算法
牛顿法
DFP
BFGS
L-BFGS
约束优化
拉格朗日乘数法
可信域牛顿法
KTT条件
分治法

# 课程大纲

一 深度学习基础

二 人工智能模型特性

三 对抗样本攻击原理

四 经典对抗样本算法

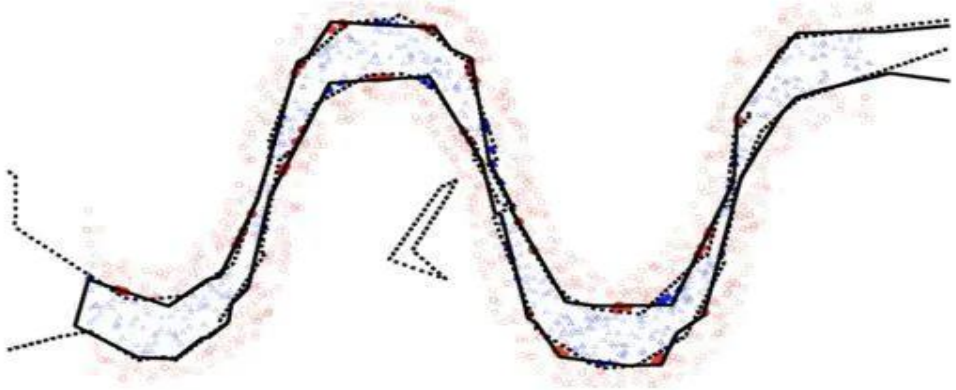
# 模型特性一

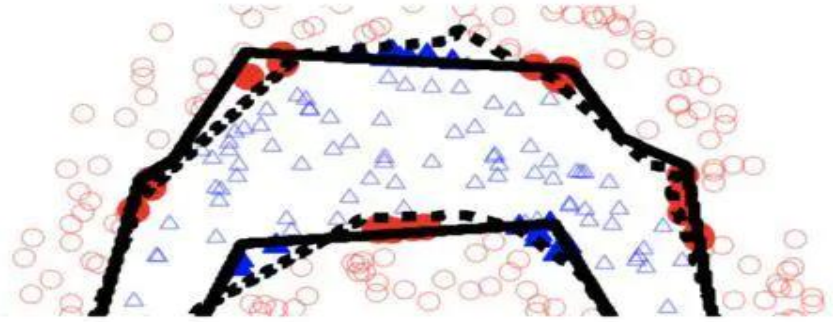
## 网络深，结构复杂

- 近年来随着深度学习的发展，深度学习模型结构越来越复杂，越来越丰富，同样网络深度也随之增加。从8层的AlexNet增加到16层、19层的VGG模型（2014）到152层的ResNet（2015）。

WHY?

- 要解决比较复杂的问题，要么增加深度，要么增加宽度，而增加宽度的代价往往远高于深度。更深的模型，意味着更好的非线性表达能力，可以学习更加复杂的变换，从而可以拟合更加复杂的特征输入

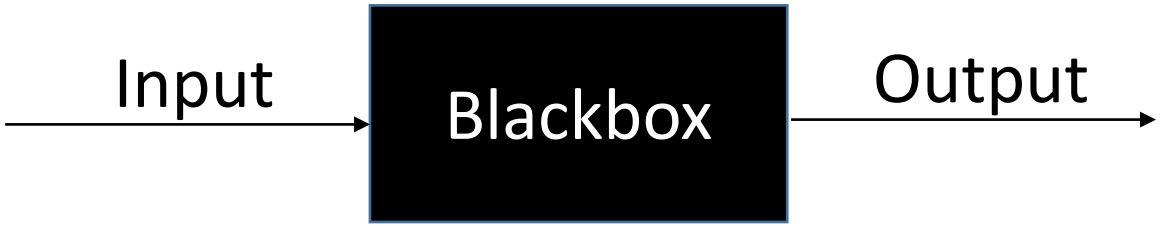




# 模型特性二

## 可解释性不足

- 由于大多数深度学习模型复杂度高、参数多、透明性低。如**黑盒**一般，人们无法理解这种“端到端”模型做出决策的机理，无法判断决策是否可靠。

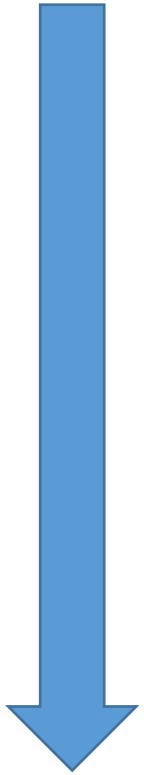



- 深度学习模型在近几年迅速发展，被成功地应用于图像分类、语音识别、自然语言处理等多个领域，取得了与人类相媲美的准确率。然而，可解释性的缺失使深度学习模型在实际应用中产生了各种问题。
- 例如在图像处理领域，高度精确的神经网络面对对抗攻击却很脆弱。只需对图像的像素进行不可察觉的改变，图像处理神经网络就可以将其任何预测改变为任何其他可能的预测

# 模型特性三

## 高维空间

| | | |
|---|---|---|
| 向量：数据存储在一条线上 | = | 一维的张量 |
| 矩阵：数据存储在一个平面上 | = | 二维的张量 |
| 方块：数据存储在立方体上 | = | 三维的张量 |
| RGB-D相机采集图像 | = | 四维的张量 |



点动成线，线动成面，面动成体。 张量的出现是为了实现最大的一般化，张量通常可以具有任意维度，并且包含标量、向量和矩阵。

张量维度越高，表达能力越强

# 模型特性三

## 高维空间

- 维度越高，它对于同一事物的刻画能力将会更强。尽管在某些环境因素下，低维空间就可以满足条件，但是，一般情况下想利用深度学习的方法来解决的问题，抽象性和复杂度不是普通方法能够做到的，所以足见高维空间出现的必要性。但是，并不是维度越高、深度越深越好。理论上维度越高，它的表达能力也就越强，对低维度空间也就越抽象，越复杂。
- 在计算机视觉领域，如人脸识别，对一幅300x300的图像，特征维数就达到10万的量级。因为高维特征才包含尽可能多的、具有可分辨性地有效人脸信息，以此提高识别准确率。
- 但是维度越高会带来 处理数据、特征工程的困难；在进行深度学习算法计算时高维会几何倍的增加计算难度；平面衡量距离用欧氏距离，随着维度增大，损失距离难以定义衡量。

# 模型特性四

## 口 理论研究不足

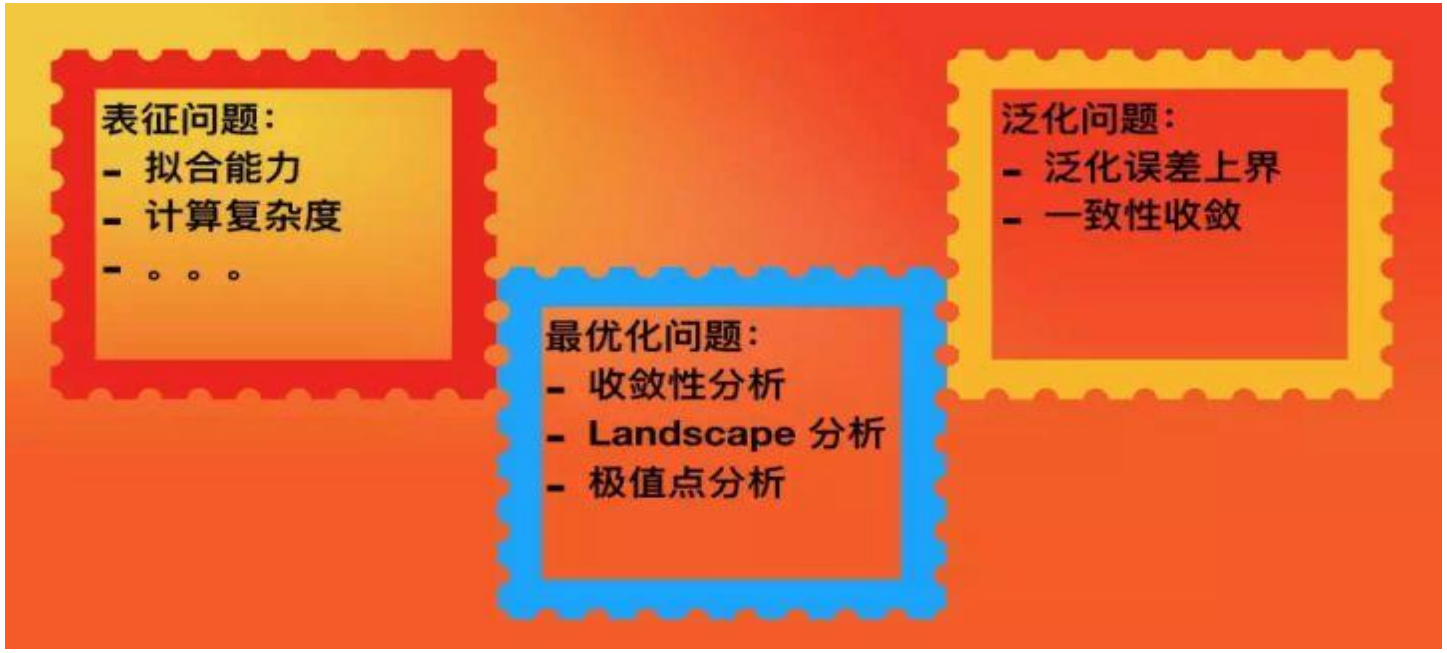
- 许多应用领域对AI算法的安全性有很高的要求，例如自动驾驶系统，一个微小的错误都有可能会导致致命的灾难。许多**现有算法缺乏适当的理论基础**，我们对**这些算法“为什么能成功”**并没有准确的把握；另外，AI应用领域也持续遇到一些挑战，这都使得大众对AI模型的信任不断减少。
- 目前深度学习的基础理论研究还处在初级阶段。**深度学习的成功主要建立在实验之上，缺乏坚实的理论基础。**
- 在深度学习中，有很多模型的数学推导都非常优美。也有很多模型从实验与直观概念出发，描述整个学习过程的数学表达。它们都非常重要，但并不能解决深度学习最基本的疑问：**为什么深度模型能够高效学习？为什么深度模型要比浅层模型的性质好？为什么深度学习泛化性也不错？**

# 模型特性四

## 口 理论研究不足

■ MIT 教授 Tomaso Poggio 曾在他的系列研究中表示深度学习理论研究可以分为三大类：

- 表征问题（Representation）：为什么深层网络比浅层网络的表达能力更好?
- 最优化问题（Optimization）：为什么梯度下降能找到很好的极小值解，好的极小值有什么特点?
- 泛化问题（Generalization）：为什么过参数化仍然能拥有比较好的泛化性，不过拟合?




深度学习理论基础研究**关键词**

# 课程大纲

一 深度学习基础

二 人工智能模型特性

三 对抗样本攻击原理

四 经典对抗样本算法

# 对抗样本

## 口 对抗样本概述

分类为熊猫　　微小的对抗噪音　　分类为长臂猿

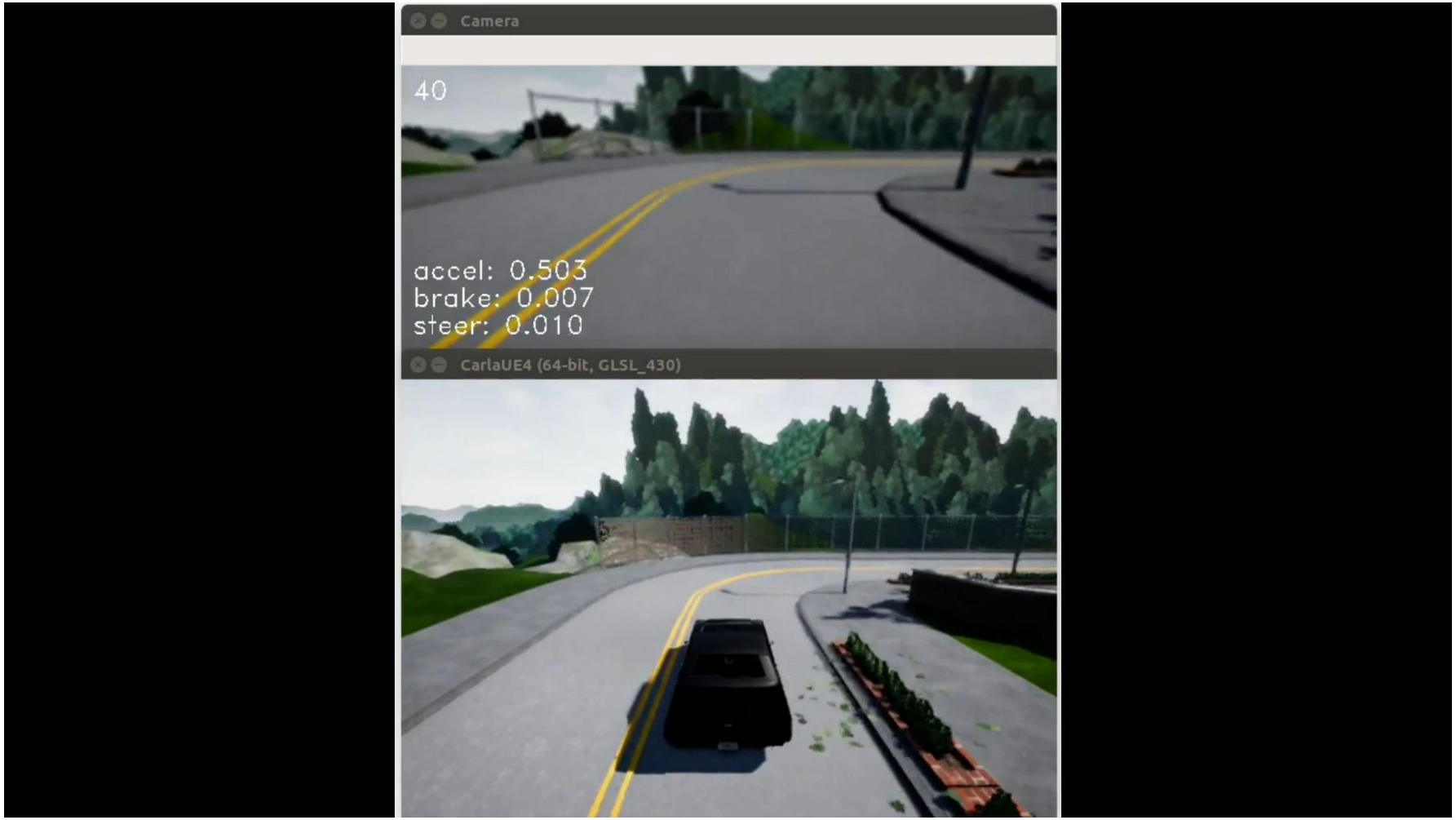
- 通过在源数据上增加人类难以通过感官辨识到的细微改变，但是却可以让机器学习模型做出错误的分类决定

第57-77页 参考文献：[1] Adversarial machine learning (tutorial) by Yevgeniy Vorobeychik and Bo Li
[2] Adversarial Machine Learning by Ian Molloy, Mathieu Sinn, and Irina Nicolae

# 面向自动驾驶的攻击

- 如果对于熊猫图片的攻击没有意义，那么下面的这些场景呢

# 针对现实场景下自动驾驶汽车的攻击

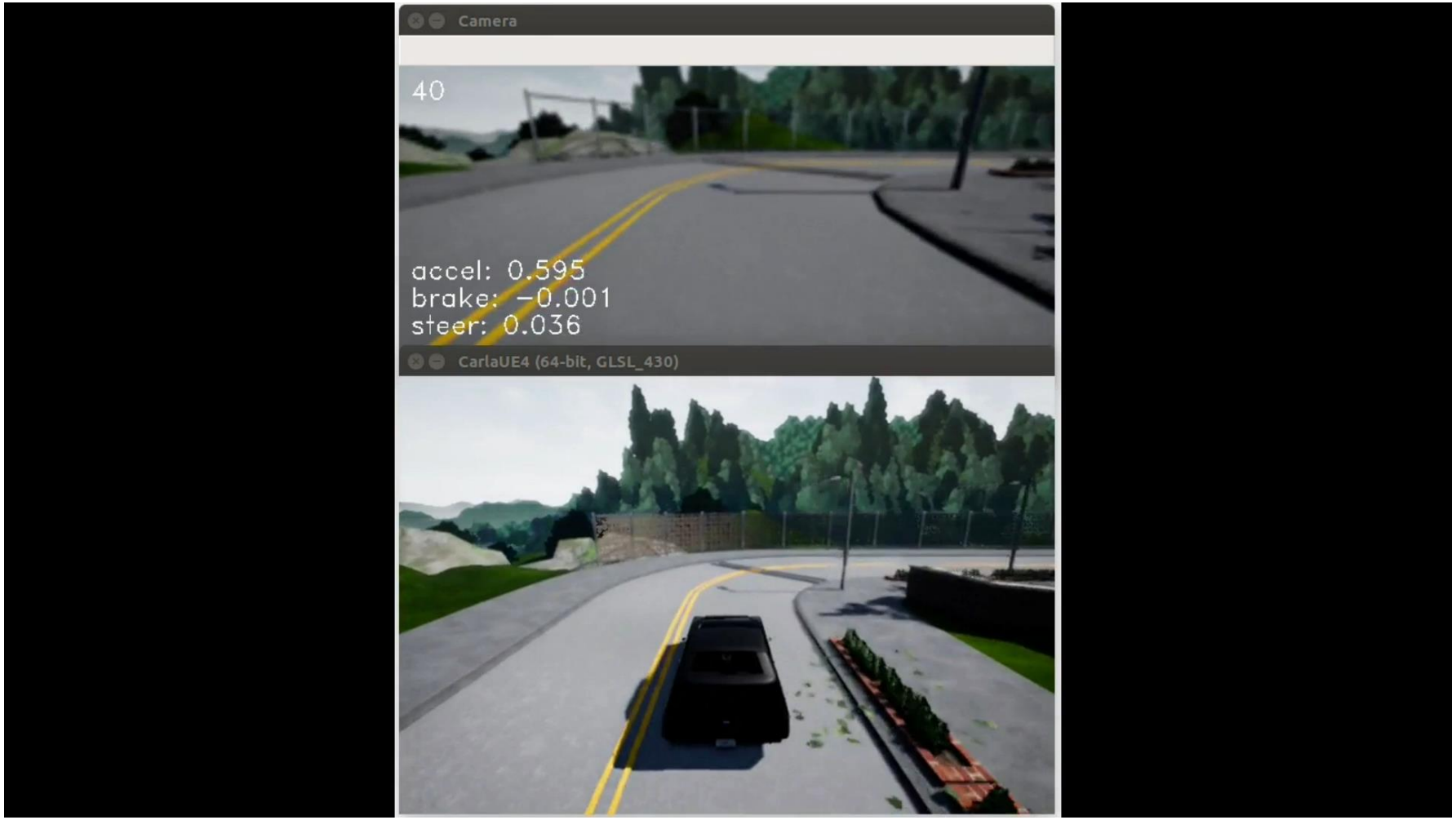


# 针对现实场景下自动驾驶汽车的攻击

## 攻击方法

- 将下面的图添加到马路上（并且旋转这张图）

# 针对现实场景下自动驾驶汽车的攻击



# 攻击者

## 对抗机器学习的关键问题: 攻击者的目标

- 不被检测到
- 产生攻击者希望的结果
- 攻击者会试图改变观察到的情况、模型的行为等，以实现目标

# 对抗攻击分类

浙江大学 ZHEJIANG UNIVERSITY

## 机器学习攻击的两个维度

| 1. 攻击者信息 | 白盒 vs. 黑盒 |
| --- | --- |
| **2. 攻击者目标** | 有目标 vs. 无目标 |

# 按攻击者信息分类

## 白盒 vs. 黑盒 攻击

- 白盒攻击
  - 攻击者知道目标模型 $f$ 的所有信息：
    - 模型结构
    - 模型参数
    - 训练算法(包括超参数)
    - 训练数据集
  - 攻击者可以此来了解目标模型
- 黑盒攻击
  - 攻击者得到的信息有限，不能确切地知道模型结构、模型参数和训练算法

# 按攻击目标分类

## 口 无目标 vs. 有目标 攻击

- 无目标攻击
  - 攻击者只需要让目标模型对样本预测错误即可，但并不指定错误预测的类别
  - 最大化预测的错误率 (普遍应用于监督学习)

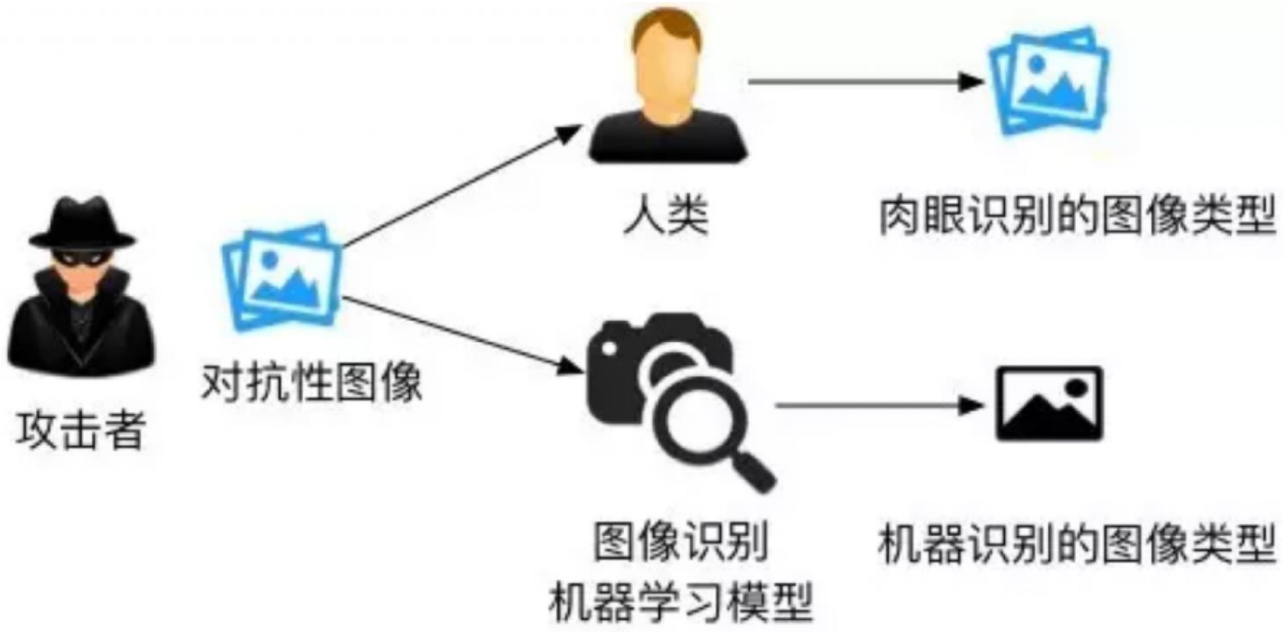
$$\min_{x'} c(x, x') \quad \text{subject to:} \quad f(x') \neq y$$

- 有目标攻击
  - 攻击者指定某一类别，使得目标模型不仅对样本分类错误并且需要错误分类成指定的类别
  - 通用的情况：假设 $S$ 是样本空间的子集；对于$S$中的样本$(x, y) \in S$，攻击者希望其被错误地分类为$(x, y')$，其中 $y' = l(x)$是攻击者期待的错误标签

$$\min_{x'} c(x, x') \quad \text{subject to:} \quad f(x') = y_T$$

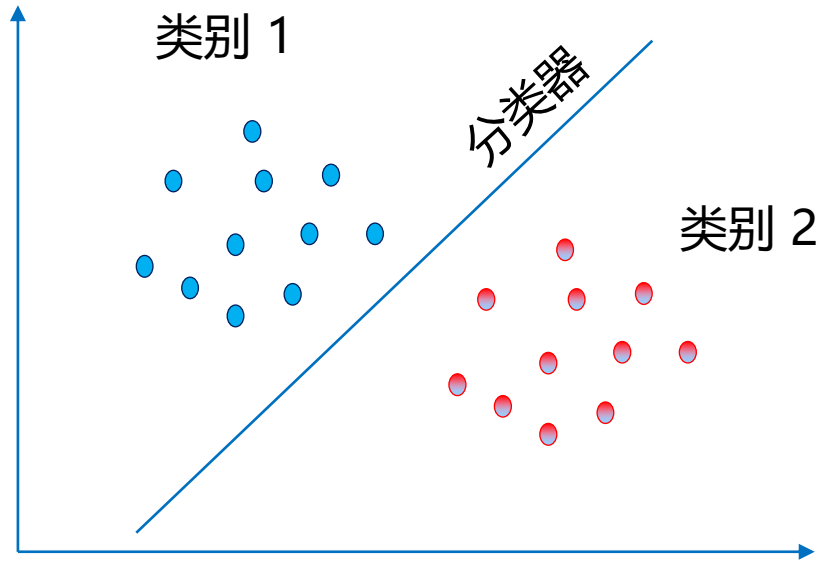
# 重要特例：逃逸攻击

## □ 逃逸攻击概览



# 对于分类问题的逃逸攻击

## 口 分类问题简述

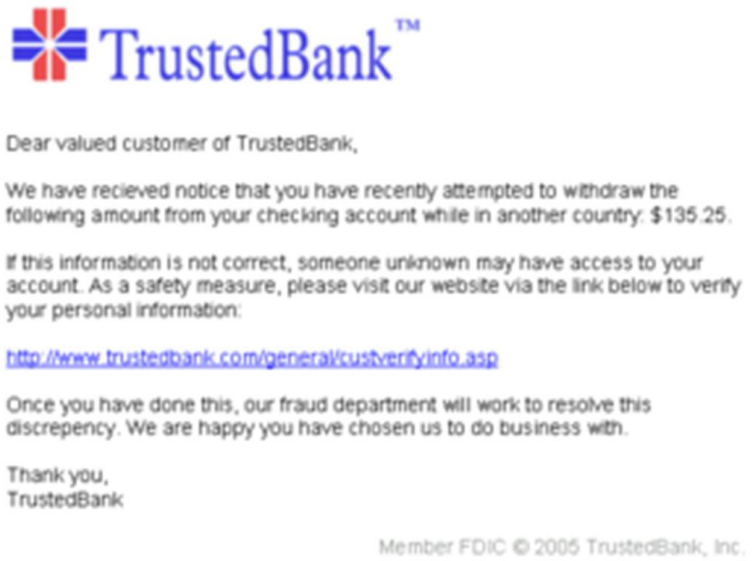



数据集: $\{(x_1, y_1), \ldots, (x_n, y_n)\}, (x, y) \sim D$
$x$: 特征向量
$y$: 关于{-1, +1}的二分类标签，用来表示$x$的类别
训练: 在数据上训练分类器$f(x)$
预测: 使用 $f(x)$ 来预测任意的 $x$ 的类别

# 对于分类问题的逃逸攻击

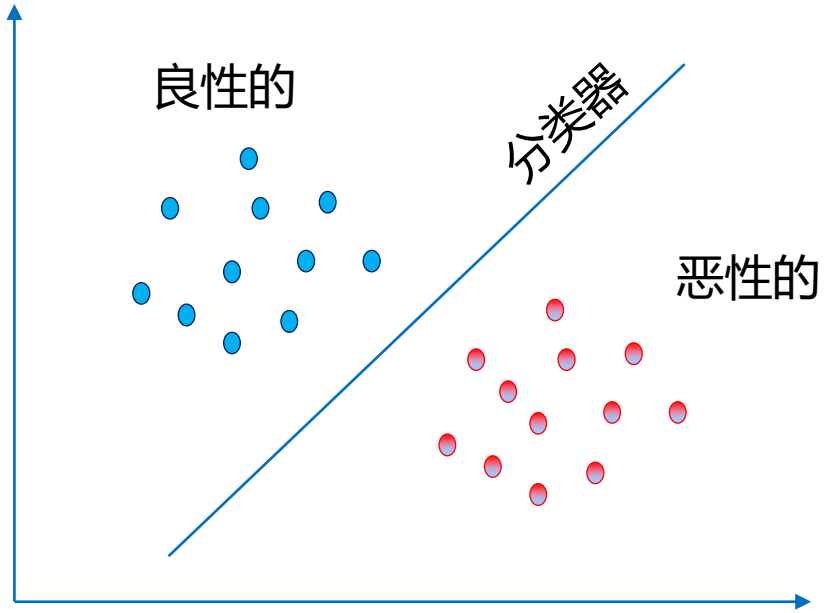
## □ 对于分类问题的逃逸攻击

- 通常，分类器的任务是区分“好” 和“坏”
  - 垃圾邮件 vs. 非垃圾邮件
  - 正常软件 vs. 恶意软件
  - 入侵检测

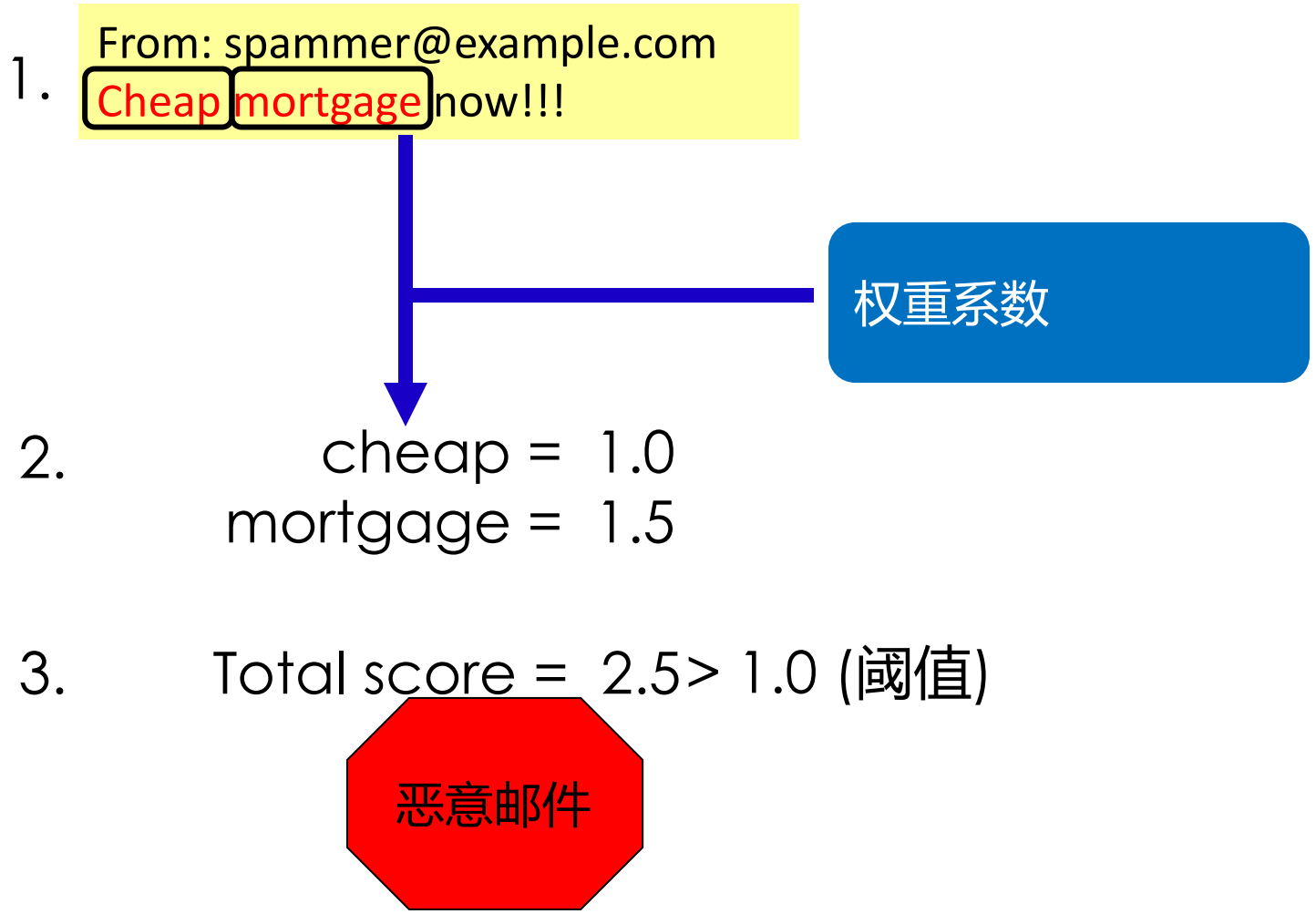


# 逃逸攻击

- 攻击者选择一个被分类为“恶性的”样本$x$，以此构造对抗样本$x'$，使其被认为是“良性的”

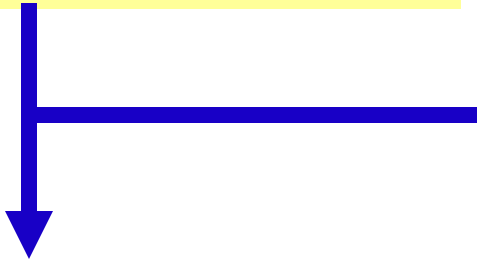


# 逃逸攻击的例子


1.
From: spammer@example.com
Cheap mortgage now!!!
权重系数
2.
cheap = 1.0
mortgage = 1.5
3.
Total score = 2.5> 1.0 (阈值)
恶意邮件

# 逃逸攻击的例子

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



权重系数

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

非恶意邮件

# 逃逸攻击的建模

## 逃逸攻击建模过程

- 攻击者拥有“理想的”特征向量$x$
  - 即训练数据中的原始的恶意的特征向量$x$
  - 将 $x$ 修改为另一个特征向量 $x'$会消耗代价 $c(x, x')$
- 攻击者的目标是使分类器将“恶性的”特征向量分类为“良性的”

# 对抗攻击目标函数基本模型

## 无目标攻击:

$$\min_{x'} c(x, x') \quad \text{subject to:} \quad f(x') \neq y$$

## 有目标攻击:

$$\min_{x'} c(x, x') \quad \text{subject to:} \quad f(x') = y_T$$

## 衡量攻击的代价

- 假设攻击者以恶性特征向量 $x$ 为输入开始攻击。设 $c(x, x')$ 是将 $x$转换为另一个特征向量 $x'$的代价函数，则 $lp$ 范式距离是对该函数建模的标准方法：
  - $c(x, x') = ||x - x'||_p$, for $p \in \{0, 1, 2, \ldots, \infty\}$
  - $\|x\|_p = (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}}$

# 对抗攻击目标函数一

## 函数一

- 最小化代价函数，以确保$x'$看起来足够良性
- $\min_{x'} c(x, x')$ subject to: $g(x') \leq \theta$
  - *对于二元分类，有目标和无目标攻击之间是没有区别的*

## 函数一推广（多元分类条件下）

- 无目标攻击:
  - $\min_{x'} c(x, x')$ subject to: $\exists\, y' \quad g_y(x') \leq g_{y'}(x')$
  - 对于所有的类别，只需要找到最小化扰动使得存在一个非正确的类别得到的预测分数大于等于正确的类别得分即可
- 有目标攻击:
  - $\min_{x'} c(x, x')$ subject to: $\forall y \quad g_{y_T}(x') \geq g_y(x')$
  - 对于所有的类别，需要找到最小的扰动使得我们希望模型非正确分类的类别得到的预测分数大于等于所有其他类别的得分

# 对抗攻击目标函数二

## 函数二

- 让$x'$看起来越良性越好，只要代价函数的变化限制在$\epsilon$之内
- $\min_{x'} g(x')$ subject to: $c(x, x') \leq \epsilon$

## 函数二扩展

- 在给定真实标签 $y$ 的情况下，最大化目标分类器的损失函数，且将代价函数变化限制在 $\epsilon$ 内
- $\max_{x'} l(g(x'), y)$ subject to: $c(x, x') \leq \epsilon$

# 对抗攻击目标函数二的推广

## 函数二推广

- 无目标攻击:
  - $\min_{x'} g_y(x')$ subject to: $c(x, x') \leq \epsilon$
  - 将代价函数变化限制在 $\epsilon$ 内，并使得添加扰动后模型输出正确类别的分数最小
- 有目标攻击:
  - $\max_{x'} g_{y_T}(x')$ subject to: $c(x, x') \leq \epsilon$
  - 将代价函数变化限制在 $\epsilon$ 内，使得添加扰动后模型输出的我们希望模型非正确分类的类别得到的预测分数最大

# 对抗攻击目标函数三

## 函数三

- 同时最小化对抗攻击目标函数和代价函数
- $\min_{x'} g(x') + \lambda c(x, x')$

## 函数三推广

- 无目标攻击
  - $\min_{x'} \max\{g_y(x') - \max_{y' \neq y} g_{y'}(x'), 0\} + \lambda c(x, x')$
  - 同时最小化正确标签得分与错误标签得分中最大值的差与代价函数的加权和
- 有目标攻击
  - $\max_{x'} \min\{g_{y_T}(x') - \max_{y' \neq y_T} g_{y'}(x'), 0\} - \lambda c(x, x')$
  - 同时最小化我们希望模型非正确分类的类别得分与其他类别得分最大值的差与代价函数的加权和

# 课程大纲

一 深度学习基础

二 人工智能模型特性

三 对抗样本攻击原理

四 经典对抗样本算法

# 经典对抗样本算法

浙江大学 ZHEJIANG UNIVERSITY

## 介绍对抗样本的一些经典算法：

- 白盒攻击
  - 快速梯度符号法（FGSM）
  - 投影梯度下降法（PGD）
  - 单像素攻击(One Pixel Attack)
- 黑盒攻击
  - 不同设定下的黑盒攻击
  - 基于访问的黑盒攻击

# 快速梯度符号法（FGSM）

## FGSM

- FGSM全称Fast Gradient-Sign Method，即快速梯度符号法
- FGSM是最著名的无目标攻击方法之一，它能够直接推广到有目标攻击中
- FGSM是梯度上升的单步（Fast）优化方法，优化方向沿着训练模型梯度下降的反方向

## 主要思想

- 作者认为，深度网络容易收到对抗性扰动的主要原因是它们的线性性质，高维空间中的线性行为足以引起对抗样本
- 在输入图像中加上计算得到的梯度方向，这样修改后的图像经过分类网络时的损失值就比修改前的图像经过分类网络时的损失值要大，换句话说，模型预测对的概率变小了

参考文献：[1] Goodfellow et al. Explaining and Harnessing Adversarial Examples, ICLR 2015.
[2] Eykholt et al., Robust Physical-World Attacks on Deep Learning Models, CVPR 2018.

# 快速梯度符号法（FGSM）

## 对抗样本的线性解释

- 对于一个线性分类器：$\boldsymbol{f(x) = w^T x + b}$ 对抗扰动随着权重维度的增加而增加
  - 如果$w$有$n$个维度，权重向量中每个元素的平均大小为$m$，且$||\eta||_\infty \leq \epsilon$，则产生的偏移最大可达到$\epsilon nm$
  - 虽然的$\epsilon$值很小，但是当$w$的维度足够大时，可以使激活值提高$\epsilon nm$，就有可能使分类器分类错误
- DNN一般是高度非线性模型
  - DNN的非线性单元赋予了其强大的表达能力，但也降低了学习效率
  - 为了提高学习效率，需要对非线性单元进行改进，通常做法是通过降低其非线性来实现
  - 非线性单元的线性行为不断增强，导致DNN的线性能力增强，导致对抗样本的存在

参考文献：[1] Goodfellow et al. Explaining and Harnessing Adversarial Examples, ICLR 2015.
[2] Eykholt et al., Robust Physical-World Attacks on Deep Learning Models, CVPR 2018.

# 快速梯度符号法（FGSM）

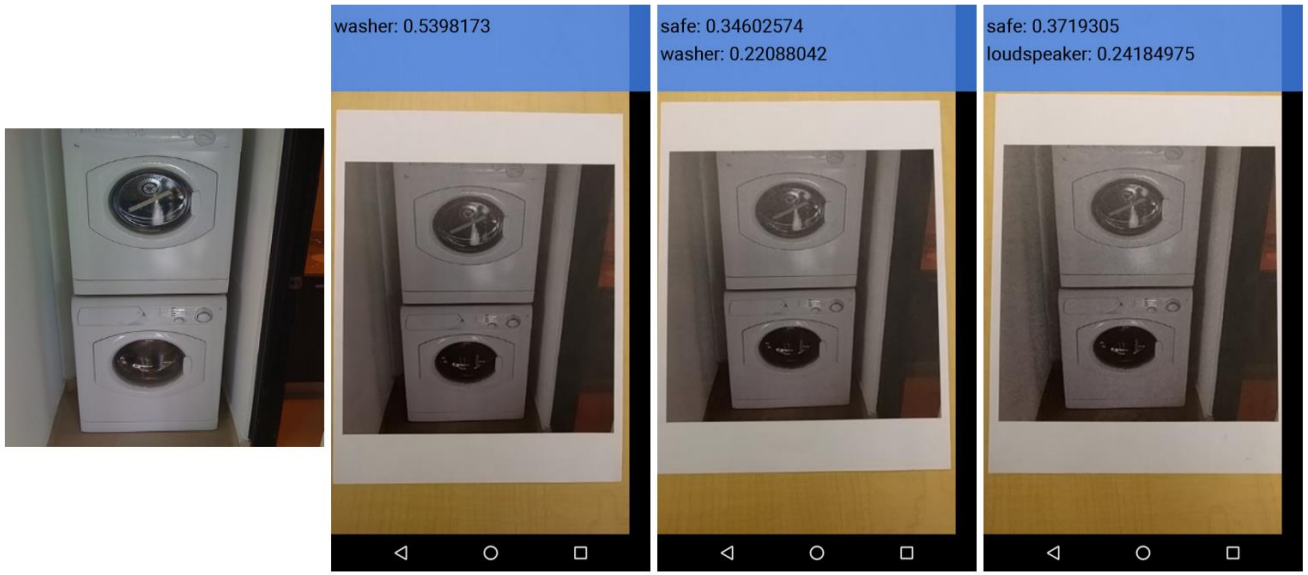
## 对非线性模型施加线性扰动

- 对抗目标：将代价函数变化限制在 $\epsilon$ 内，使得正确标签与模型预测结果的损失最大化
- $\max_{\delta} l(f(x+\delta), y)$ subject to: $||\delta||_{\infty} \leq \epsilon$
- 现在，将损失函数在点$(x, y)$附近进行线性近似:
- $l(f(x+\delta), y) \approx l(f(x), y) + \nabla_x l(f(x), y)\delta$
- 损失函数是关于 $\delta$ 的线性函数，且 $\delta$ 受 $l_{\infty}$ 约束，在$x$的每个分量上独立地添加最大化的扰动（梯度上升），就能得到最优解：$\delta^* = \epsilon \, sign\{\nabla_x l(f(x), y)\}$

参考文献：[1] Goodfellow et al. Explaining and Harnessing Adversarial Examples, ICLR 2015.
[2] Eykholt et al., Robust Physical-World Attacks on Deep Learning Models, CVPR 2018.

# 快速梯度符号法（FGSM）

## □ FGSM攻击举例




(a) Image from dataset (b) Clean image (c) Adv. image, $\epsilon = 4$ (d) Adv. image, $\epsilon = 8$

- 图为对于洗衣机的图片的分类
- 在添加扰动后，模型将洗衣机错误地分类为保险柜

参考文献：[1] Goodfellow et al. Explaining and Harnessing Adversarial Examples, ICLR 2015.
[2] Eykholt et al., Robust Physical-World Attacks on Deep Learning Models, CVPR 2018.

# 投影梯度下降法（PGD）

## □ PGD主要思想

- PGD（Project Gradient Descent)攻击是一种迭代攻击，可以看作是FGSM攻击的翻版——K-FGSM（K表示迭代的次数）。其主要思路是：FGSM是仅仅做一次迭代（单步），而PGD是做多次迭代（多步），每次走一小步，每次迭代都会将扰动调整到规定范围内
- 如果目标是一个线性模型，此时损失对输入的倒数是固定的，扰动的方向是明确的，可以用FGSM
- 对于一个非线性模型，如果仅做一次迭代，方向不一定是正确的，FGSM的效果就不好

参考文献：[1] Madry et al. Towards Deep Learning Models Resistant to Adversarial Attacks, ICLR 2018.
[2] Carlini and Wagner. Towards Evaluating the Robustness of Neural networks, IEEE S&P 2017.

# 投影梯度下降法（PGD）

## □ PGD目标函数

$$\underset{(x,y)\sim\mathcal{D}}{} \left[ \max_{\delta\in\mathcal{S}} L(\theta, x+\delta, y) \right]$$

$(x, y)$：样本空间的样本　　$L$：损失函数

$D$：样本空间的分布　　$\delta$：扰动

$\theta$：模型参数　　$S$：扰动的最大范围

■ PGD的目标是在扰动范围S的球体内寻找到一个点$x^* = x + \delta$，使得$x^*$处的损失函数的值最大

参考文献：[1] Madry et al. Towards Deep Learning Models Resistant to Adversarial Attacks, ICLR 2018.
[2] Carlini and Wagner. Towards Evaluating the Robustness of Neural networks, IEEE S&P 2017.

# 投影梯度下降法（PGD）

## □ PGD算法描述

$$x^{t+1} = \Pi_{x+\mathcal{S}}\left(x^t + \alpha\,\mathrm{sgn}(\nabla_x L(\theta, x, y))\right).$$

$(x, y)$：样本空间的样本　　$L$：损失函数，其中L内的$x$指的是$x^t$

$\prod$：投影操作　　$\alpha$：每次移动的步长

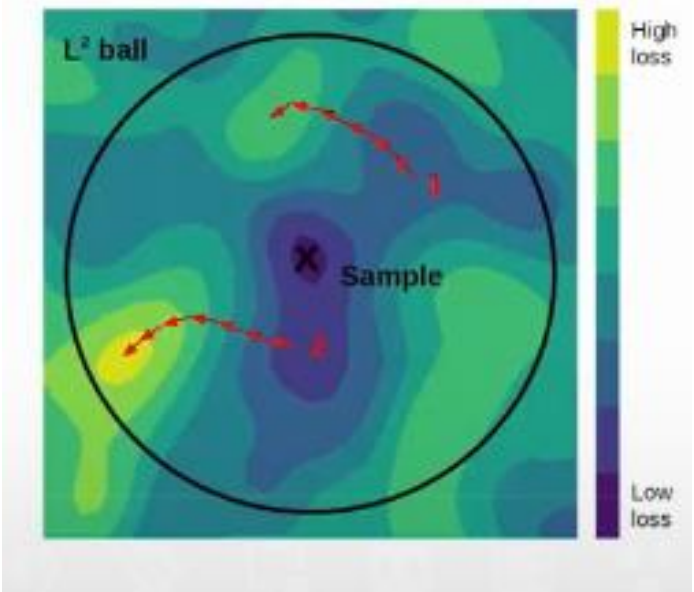
$\theta$：模型参数　　$S$：扰动的最大范围

- PGD算法在扰动范围内逐步向损失函数更大的方向以$\alpha$的速度移动
- 投影的作用：当扰动超过了最大范围$S$时，通过投影的操作将其限制回扰动范围内

参考文献：[1] Madry et al. Towards Deep Learning Models Resistant to Adversarial Attacks, ICLR 2018.
[2] Carlini and Wagner. Towards Evaluating the Robustness of Neural networks, IEEE S&P 2017.

# 投影梯度下降法（PGD）

## □ PGD算法与FGSM算法比较




- 在扰动范围的球体内，最佳对抗样本出现的点不一定在球面上，也可能出现在球体的内部
- 如图中的1，2两条路径，PGD算法可以沿着1，2的路径找到使损失函数最大的对抗样本点，但是FGSM算法只能找到球面上的点，但显然这不是最好的对抗样本点
- 所以一般认为PGD算法比FGSM算法更有效且更难防御

参考文献：[1] Madry et al. Towards Deep Learning Models Resistant to Adversarial Attacks, ICLR 2018.
[2] Carlini and Wagner. Towards Evaluating the Robustness of Neural networks, IEEE S&P 2017.

# 单像素攻击(One Pixel Attack)

## □ One Pixel Attack的主要思想

- 使用差分进化的方法来修改单个像素点并以此来改变分类器的输出


参考文献：[1] Su J，Vargas D V，Kouichi S . One pixel attack for fooling deep neural networks[J]. IEEE Transactions on Evolutionary Computation, 2017.

# 单像素攻击(One Pixel Attack)

## □ One Pixel Attack的目标函数

- **有目标攻击**

$$\begin{aligned} \underset{e(\mathbf{x})^*}{\text{maximize}} \quad & f_{adv}(\mathbf{x}+e(\mathbf{x})) \\ \text{subject to} \quad & \|e(\mathbf{x})\|_0 \leq d, \end{aligned}$$

$x$：输入图像
$e(x)$：添加的扰动

$f_{adv}$：分类器输出目标类别$adv$的置信度
$d$：改变的像素个数，单像素攻击时$d = 1$

- 在只改变单个像素的限制下，最大化分类器输出的目标类别的得分

参考文献：[1] Su J，Vargas D V，Kouichi S . One pixel attack for fooling deep neural networks[J]. IEEE Transactions on Evolutionary Computation, 2017.

# 单像素攻击(One Pixel Attack)

## 口 差分进化（Differential Evolution，下文简称DE)

- 使用原因
  - One Pixel Attack显然无法通过对于所有像素点的搜索与改变的穷举来发动攻击，暴力枚举的办法效率太低
  - 在攻击时，不需要得到的扰动是全局最优解，只需要得到的扰动可以使分类器输出设定的目标类别即可
  - DE是一种高效的全局优化算法
- 主要思想
  - 在每次迭代中，根据当前解(父亲解)生成另一组候选解(孩子解)
  - 然后将孩子解与他们相应的父亲解进行比较，如果孩子解比他们的父亲解更适合，他们就会被保留下来，否则他们被淘汰

参考文献：[1] Su J，Vargas D V，Kouichi S . One pixel attack for fooling deep neural networks[J]. IEEE Transactions on Evolutionary Computation, 2017.

# 单像素攻击(One Pixel Attack)

## □ DE在One Pixel Attack中的使用

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

$x_i$：标号为$i$的候选解　　$g$：迭代的代数

F：设定的超参数　　$r1, r2, r3$：随机选择的父亲解的标号

- 在One Pixel Attack中，候选解$x_i$为单个像素点，即$\boldsymbol{x} + \boldsymbol{e}(\boldsymbol{x})$中改动的像素点，每个候选解是由五个元素组成的向量，分别为x轴坐标，y轴坐标和图片RGB的三个通道值
- 每次随机选择不同的三个父亲解，使用上述的DE公式来生成孩子解，把孩子解对应轮次的图片输入模型$f$得到目标类别的置信度，因为所有类别的置信度之和为1，如果让分类器在目标类别上的置信度不断增大，超过0.5后，那么分类器就会将其预测为目标类别。
- 当出现解满足攻击目标或者达到最大迭代次数（超参数）时，停止搜索

参考文献：[1] Su J，Vargas D V，Kouichi S . One pixel attack for fooling deep neural networks[J]. IEEE Transactions on Evolutionary Computation, 2017.

# 单像素攻击(One Pixel Attack)

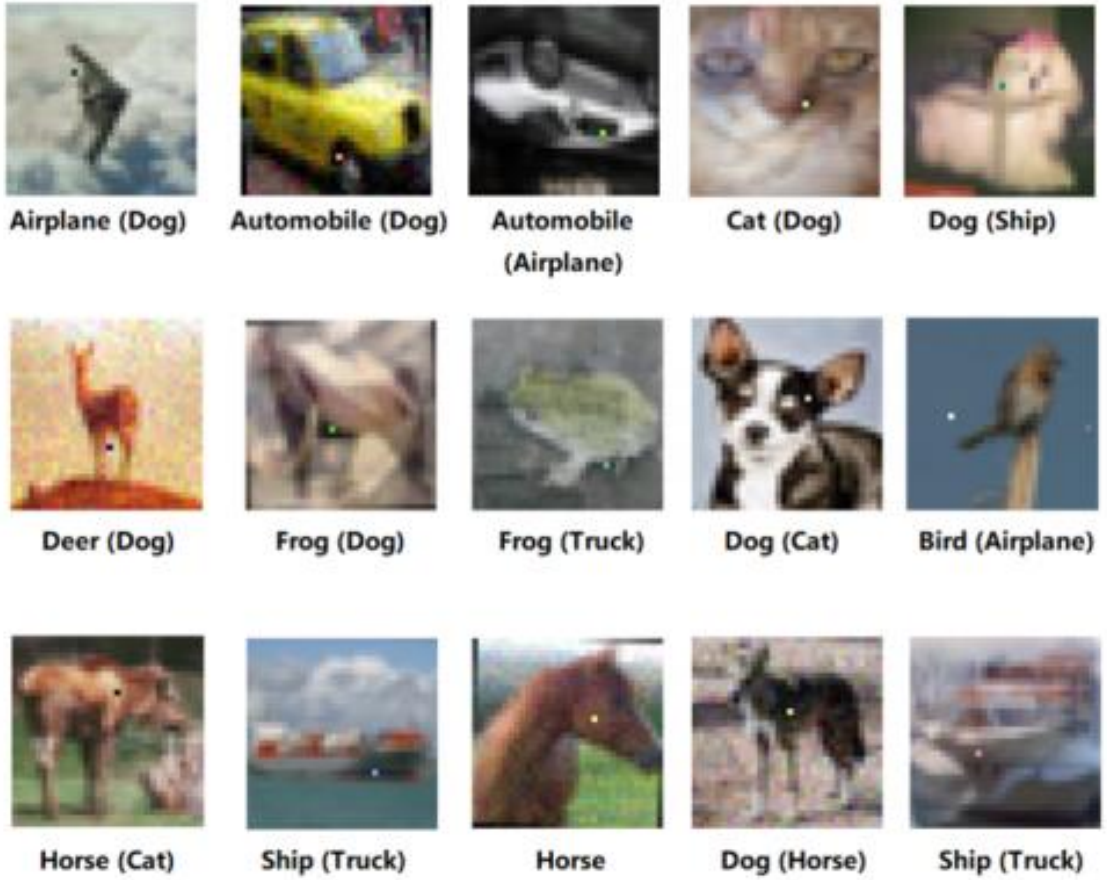



Fig. 3: Illustration of one pixel adversarial attacks [68]: The correct label is mentioned with each image. The corresponding predicted label is given in parentheses.

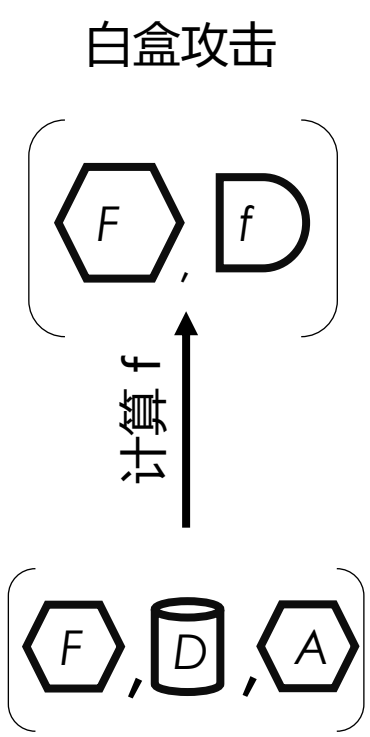


参考文献：[1] Su J，Vargas D V，Kouichi S . One pixel attack for fooling deep neural networks[J]. IEEE Transactions on Evolutionary Computation, 2017.

# 黑盒攻击算法

## 不同设定下的黑盒攻击算法

- 主要思想：如果我们知道特征空间、学习算法，并且有原始的训练数据，我们就可以恢复模型（在训练数据上训练学习算法），从而将其转化为白盒攻击





参考文献：[1] Adversarial machine learning (tutorial) by Yevgeniy Vorobeychik and Bo Li
[2] Adversarial Machine Learning by Ian Molloy, Mathieu Sinn, and Irina Nicolae

# 黑盒攻击算法

## 基于访问的黑盒攻击


- 黑盒攻击中的一个常见设置是只能对模型进行访问，并使用它来展开攻击
- 访问（query）：得到分类器对于特定实例x的分类标签

参考文献：[1] Adversarial machine learning (tutorial) by Yevgeniy Vorobeychik and Bo Li
[2] Adversarial Machine Learning by Ian Molloy, Mathieu Sinn, and Irina Nicolae

# 黑盒攻击算法

## 基于访问的黑盒攻击

- **通过访问构建代理模型：**
- 假设我们对训练数据的分布有所了解
- 询问输入实例（根据原始分布生成）， 这为我们提供了一个训练数据集
- 在这个训练数据集上运行一个代理算法， 会得到一个代理模型

参考文献：[1] Adversarial machine learning (tutorial) by Yevgeniy Vorobeychik and Bo Li
[2] Adversarial Machine Learning by Ian Molloy, Mathieu Sinn, and Irina Nicolae

# 黑盒攻击算法

## 基于访问的黑盒攻击

- **黑盒攻击的迁移性：**
- 针对代理模型设计的攻击，仍能成功针对真实模型的攻击被认为是可转移的
- 可转移性非常普遍

参考文献：[1] Adversarial machine learning (tutorial) by Yevgeniy Vorobeychik and Bo Li
[2] Adversarial Machine Learning by Ian Molloy, Mathieu Sinn, and Irina Nicolae

# 谢 谢

浙江大学网络空间安全学院

https://icsr.zju.edu.cn/