

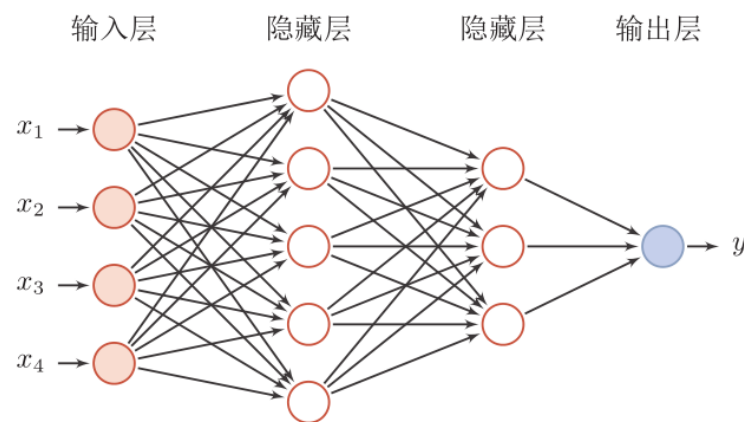


## 前馈神经网络

# 网络结构

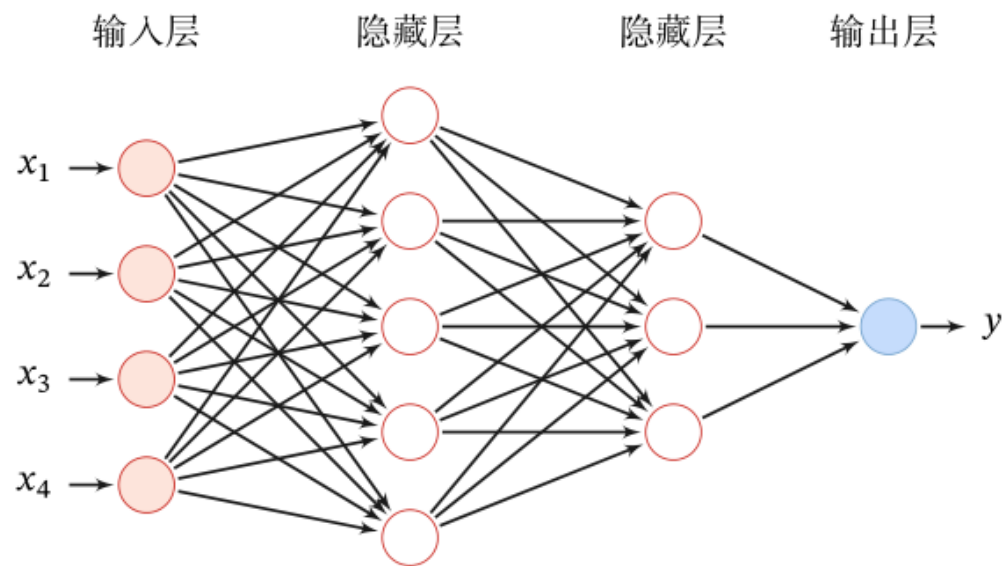
## ▶ 前馈神经网络（全连接神经网络、多层感知器）

- ▶ 各神经元分别属于不同的层，层内无连接。
- ▶ 相邻两层之间的神经元全部两两连接。
- ▶ 整个网络中无反馈，信号从输入层向输出层单向传播，可用一个有向无环图表示。



# 前馈网络

给定一个前馈神经网络，用下面的记号来描述这样网络：



记号	含义
$L$	神经网络的层数
$M_l$	第 $l$ 层神经元的个数
$f_l(\cdot)$	第 $l$ 层神经元的激活函数
$\mathbf{W}^{(l)} \in \mathbb{R}^{M_l \times M_{l-1}}$	第 $l-1$ 层到第 $l$ 层的权重矩阵
$\mathbf{b}^{(l)} \in \mathbb{R}^{M_l}$	第 $l-1$ 层到第 $l$ 层的偏置
$\mathbf{z}^{(l)} \in \mathbb{R}^{M_l}$	第 $l$ 层神经元的净输入（净活性值）
$\mathbf{a}^{(l)} \in \mathbb{R}^{M_l}$	第 $l$ 层神经元的输出（活性值）

# 信息传递过程

---

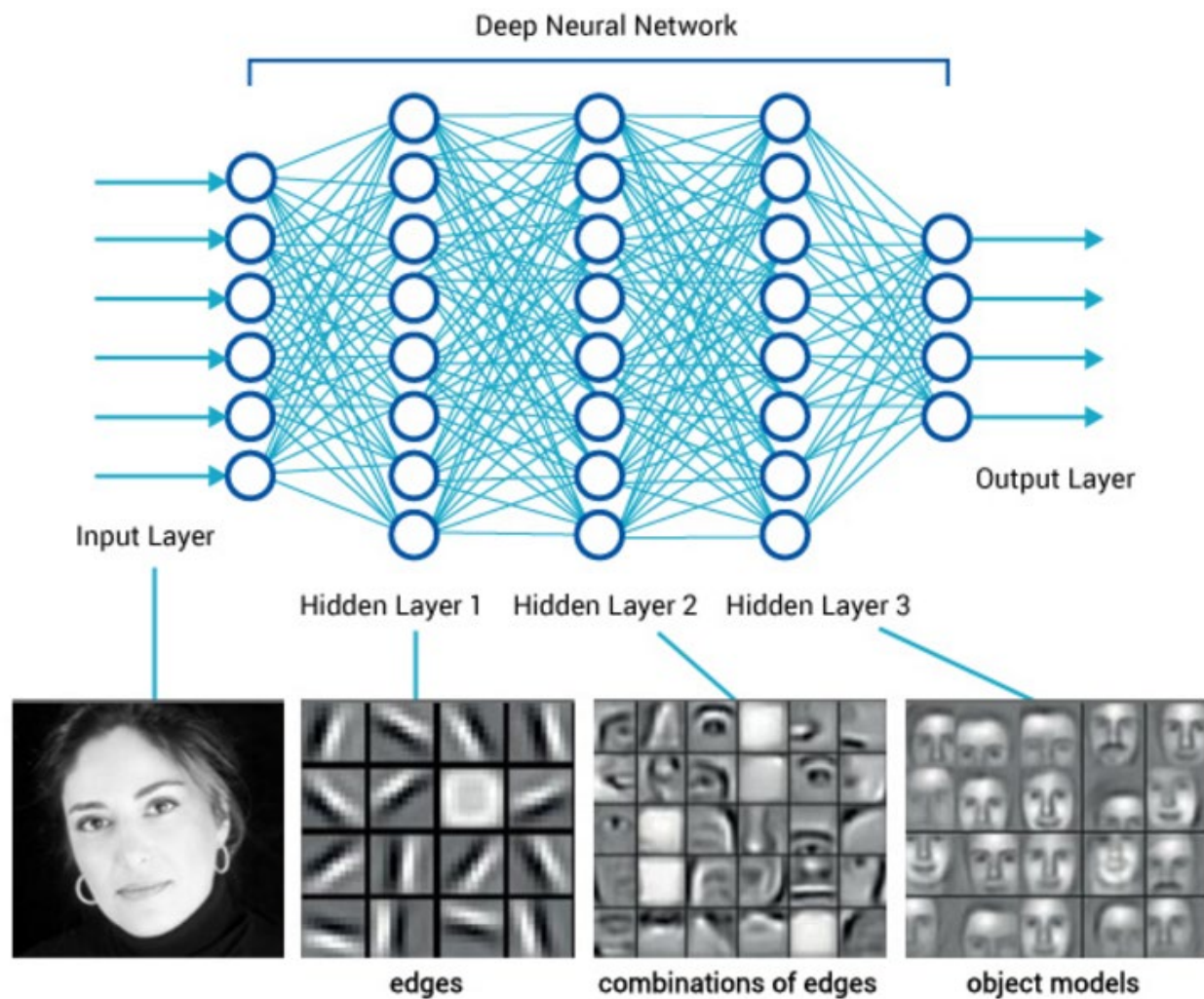
► 前馈神经网络通过下面公式进行信息传播。

$$\begin{aligned}\mathbf{z}^{(l)} &= \mathbf{W}^{(l)} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}, \\ \mathbf{a}^{(l)} &= f_l(\mathbf{z}^{(l)}).\end{aligned}$$

► 前馈计算：

$$\mathbf{x} = \mathbf{a}^{(0)} \rightarrow \mathbf{z}^{(1)} \rightarrow \mathbf{a}^{(1)} \rightarrow \mathbf{z}^{(2)} \rightarrow \dots \rightarrow \mathbf{a}^{(L-1)} \rightarrow \mathbf{z}^{(L)} \rightarrow \mathbf{a}^{(L)} = \phi(\mathbf{x}; \mathbf{W}, \mathbf{b}))$$

# 深层前馈神经网络



# 通用近似定理

**定理 4.1 – 通用近似定理 (Universal Approximation Theorem)**

**[Cybenko, 1989, Hornik et al., 1989]:** 令  $\varphi(\cdot)$  是一个非常数、有界、单调递增的连续函数,  $\mathcal{I}_d$  是一个  $d$  维的单位超立方体  $[0, 1]^d$ ,  $C(\mathcal{I}_d)$  是定义在  $\mathcal{I}_d$  上的连续函数集合。对于任何一个函数  $f \in C(\mathcal{I}_d)$ , 存在一个整数  $m$ , 和一组实数  $v_i, b_i \in \mathbb{R}$  以及实数向量  $\mathbf{w}_i \in \mathbb{R}^d$ ,  $i = 1, \dots, m$ , 以至于我们可以定义函数

$$F(\mathbf{x}) = \sum_{i=1}^m v_i \varphi(\mathbf{w}_i^T \mathbf{x} + b_i), \quad (4.33)$$

作为函数  $f$  的近似实现, 即

$$|F(\mathbf{x}) - f(\mathbf{x})| < \epsilon, \forall \mathbf{x} \in \mathcal{I}_d. \quad (4.34)$$

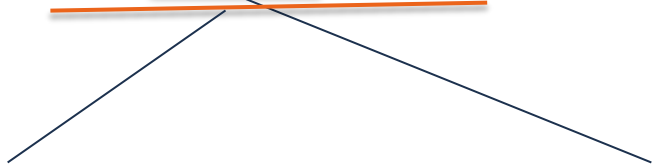
其中  $\epsilon > 0$  是一个很小的正数。

根据通用近似定理, 对于具有线性输出层和至少一个使用“挤压”性质的激活函数的隐藏层组成的前馈神经网络, 只要其隐藏层神经元的数量足够, 它可以以任意的精度来近似任何从一个定义在实数空间中的有界闭集函数。

# 应用到机器学习

---

- ▶ 神经网络可以作为一个“万能”函数来使用，可以用来进行复杂的特征转换，或逼近一个复杂的条件分布。

$$\hat{y} = g(\varphi(\mathbf{x}), \theta)$$


分类器                      神经网络

- ▶ 如果  $g(\cdot)$  为 Logistic 回归，那么 Logistic 回归分类器可以看成神经网络的最后一层。

# 应用到机器学习

---

[3D Visualization of a Fully-Connected Neural Network \(adamharley.com\)](http://adamharley.com)





## 机器学习的四要素

# 机器学习的四个要素

---

## ▶ 数据

## ▶ 模型

▶ 线性方法:  $f(\mathbf{x}, \theta) = \mathbf{w}^T \mathbf{x} + b$

▶ 广义线性方法:  $f(\mathbf{x}, \theta) = \mathbf{w}^T \phi(\mathbf{x}) + b$

▶ 如果 $\phi(\mathbf{x})$ 为可学习的非线性基函数,  $f(\mathbf{x}, \theta)$ 就等价于神经网络。

## ▶ 学习准则

▶ 期望风险

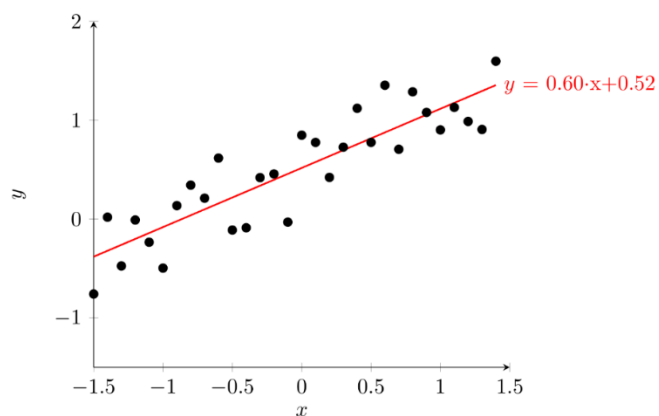
$$\mathcal{R}(f) = \mathbb{E}_{(\mathbf{x}, y) \sim p(\mathbf{x}, y)} [\mathcal{L}(f(\mathbf{x}), y)],$$

## ▶ 优化

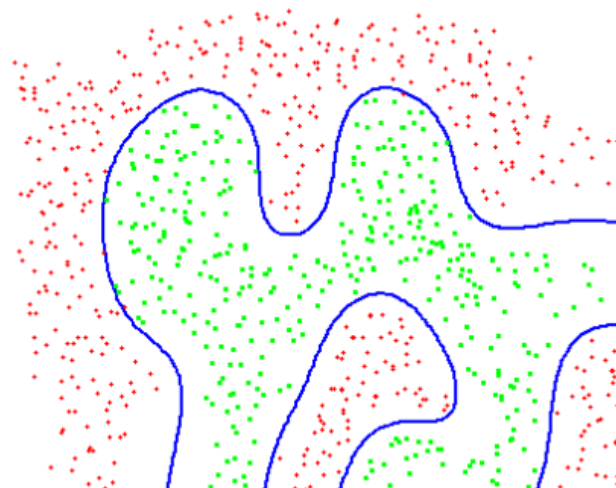
▶ 梯度下降

# 常见的机器学习问题

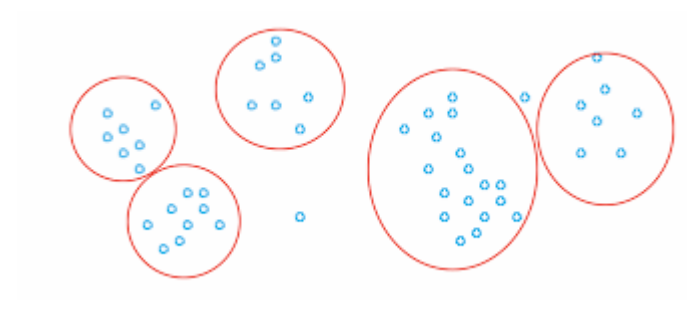
---



回归



分类



聚类

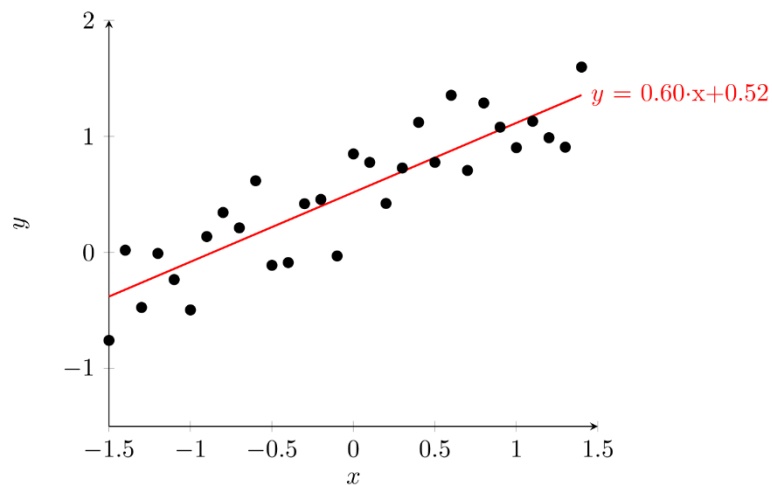
# 模型

---

▶ 以线性回归（Linear Regression）为例

▶ 模型：

$$f(\mathbf{x}; \mathbf{w}, b) = \mathbf{w}^T \mathbf{x} + b$$



# 学习准则

---

## ► 损失函数

### ► 0-1 损失函数

$$\mathcal{L}(y, f(x, \theta)) = \begin{cases} 0 & \text{if } y = f(x, \theta) \\ 1 & \text{if } y \neq f(x, \theta) \end{cases}$$

### ► 平方损失函数

$$\mathcal{L}(y, \hat{y}) = (y - f(x, \theta))^2$$

# 学习准则

---

► 期望风险未知，通过经验风险近似

► 训练数据：  $\mathcal{D} = \{x^{(n)}, y^{(n)}\}, i \in [1, N]$

$$\mathcal{R}_{\mathcal{D}}^{emp}(\theta) = \frac{1}{N} \sum_{n=1}^N \mathcal{L}(y^{(n)}, f(x^{(n)}, \theta))$$

► 经验风险最小化

► 在选择合适的风险函数后，我们寻找一个参数 $\theta^*$ ，使得经验风险函数最小化。

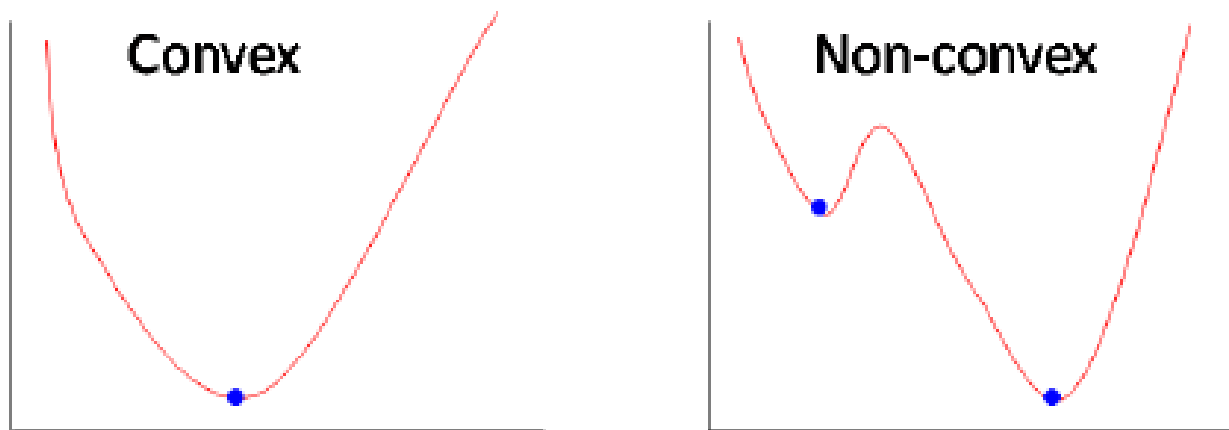
$$\theta^* = \arg \min_{\theta} \mathcal{R}_{\mathcal{D}}^{emp}(\theta)$$

► 机器学习问题转化为一个最优化问题

# 最优化问题

---

► 机器学习问题转化为一个最优化问题



$$\min_{\mathbf{x}} f(\mathbf{x})$$

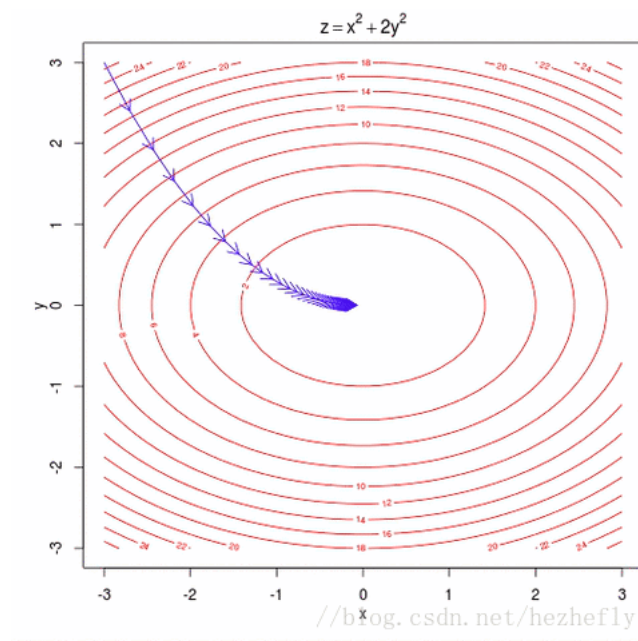
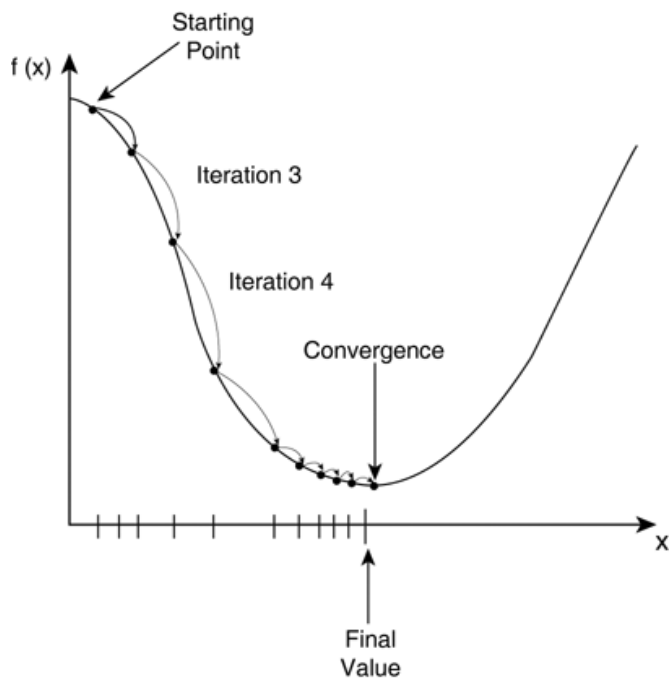
# 梯度下降法 ( Gradient Descent )

---

- ▶ 回顾：对于可微函数 $f(x)$ ，在点 $x$ 处，负梯度方向是函数的最速下降方向。



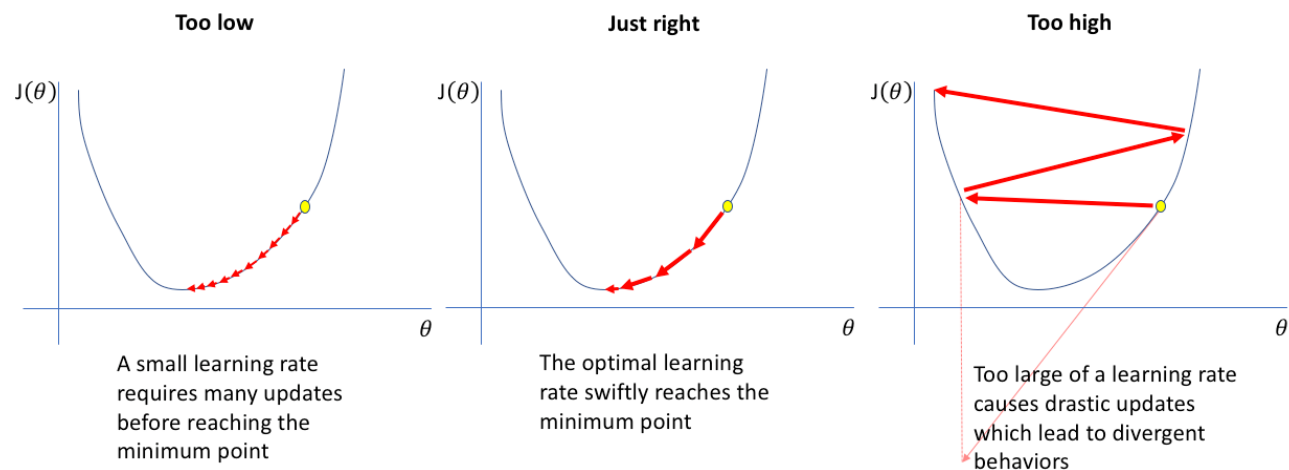
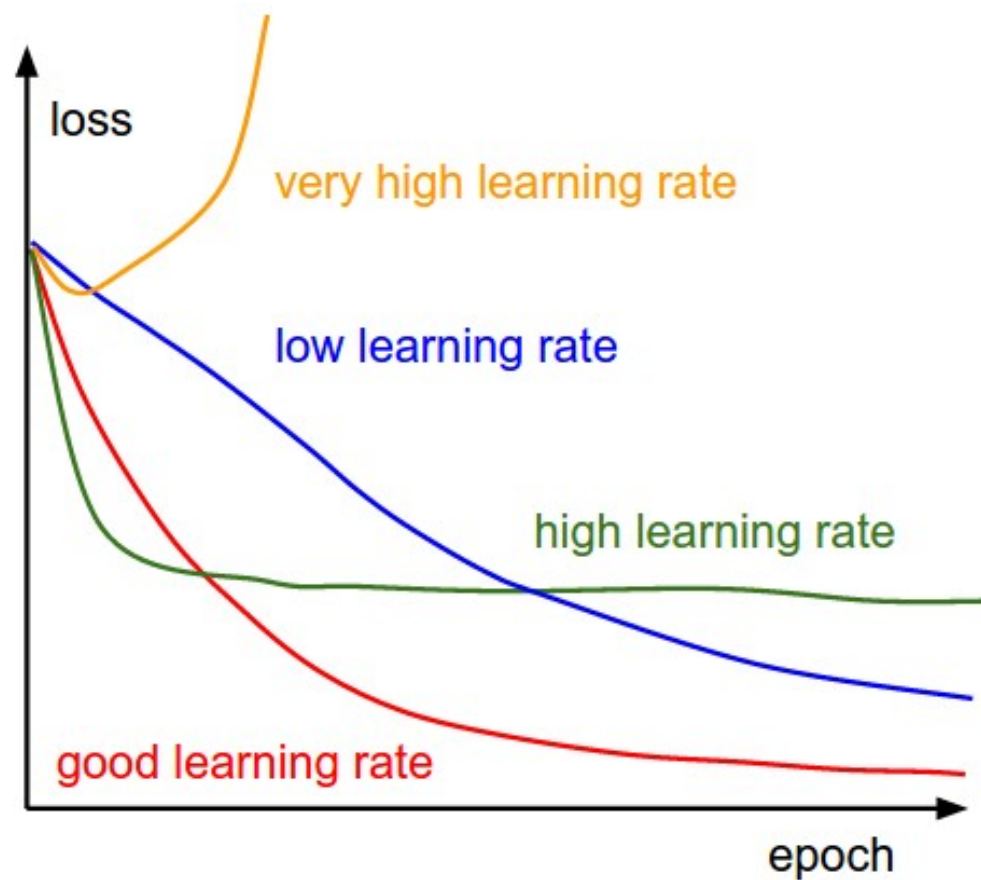
# 梯度下降法 ( Gradient Descent )



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

搜索步长 $\alpha$ 中也叫作学习率 (Learning Rate)

# 学习率是十分重要的超参数！



# 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  $\alpha$

1 随机初始化  $\theta$ ;

2 **repeat**

3     对训练集  $\mathcal{D}$  中的样本随机重排序;

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

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

        // 更新参数

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

7     **end**

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

输出:  $\theta$

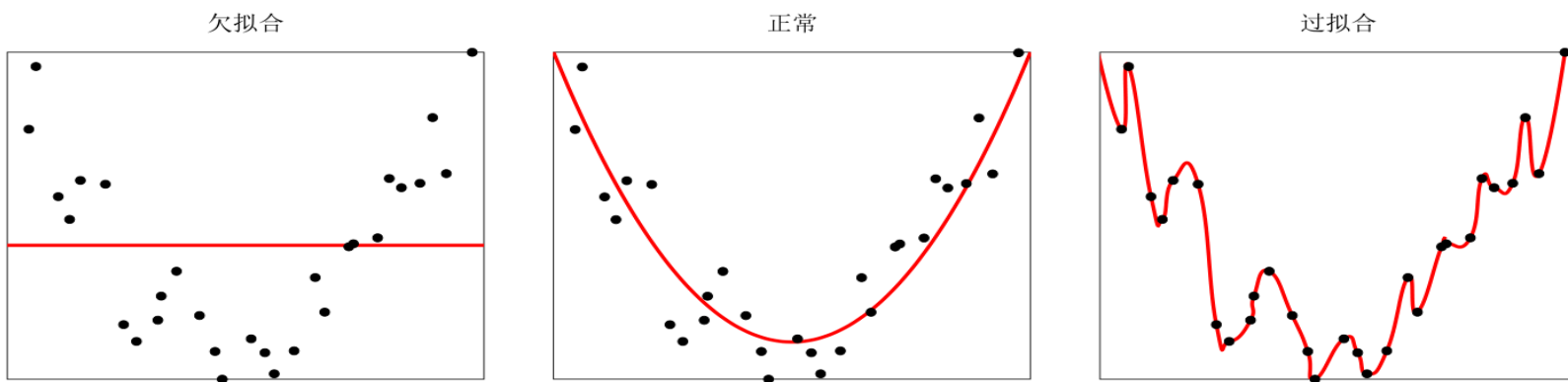
---



Why?

# 机器学习 = 优化?

机器学习 = 优化? NO!



过拟合：**经验风险最小化原则**很容易导致模型在训练集上错误率很低，但是在未知数据上错误率很高。

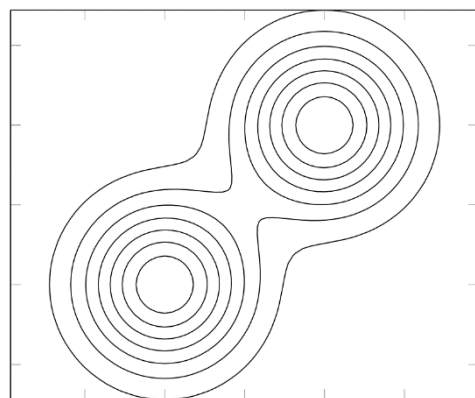
过拟合问题往往是由于训练数据少和噪声等原因造成的。

# 泛化错误

期望风险

$$\mathcal{R}(f) = \mathbb{E}_{(\mathbf{x}, y) \sim p(\mathbf{x}, y)} [\mathcal{L}(f(\mathbf{x}), y)],$$

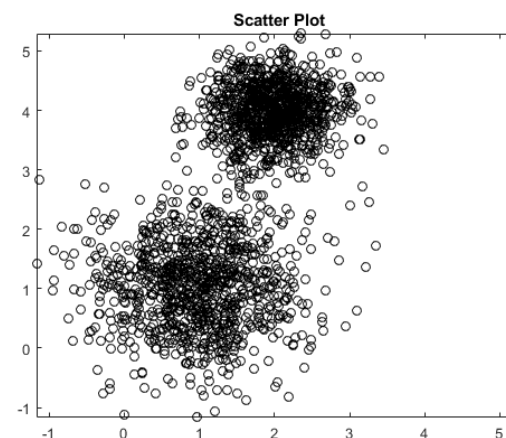
真实分布  $p_r$



$\neq$

经验风险

$$\mathcal{R}_D^{emp}(\theta) = \frac{1}{N} \sum_{n=1}^N \mathcal{L}(y^{(n)}, f(x^{(n)}, \theta))$$



$$\mathcal{G}_D(f) = \mathcal{R}(f) - \mathcal{R}_D^{emp}(f)$$

泛化错误

# 如何减少泛化错误?

---

优化

经验风险最小

正则化

降低模型复杂度





# 正则化 (regularization)

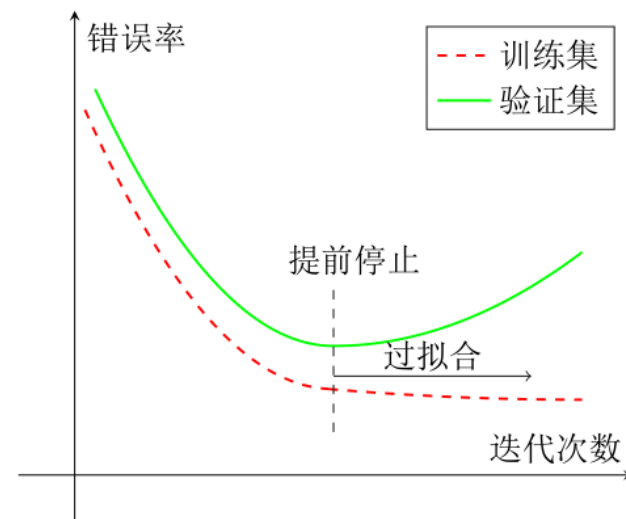
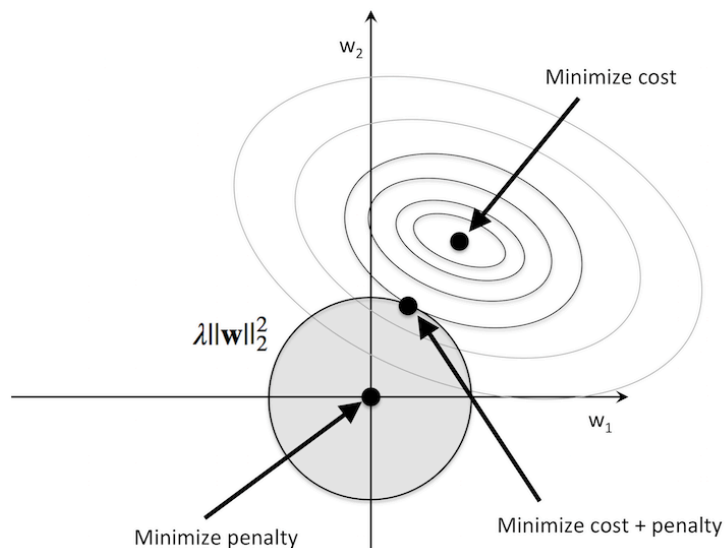
所有损害优化的方法都是正则化。

增加优化约束

L1/L2约束、数据增强

干扰优化过程

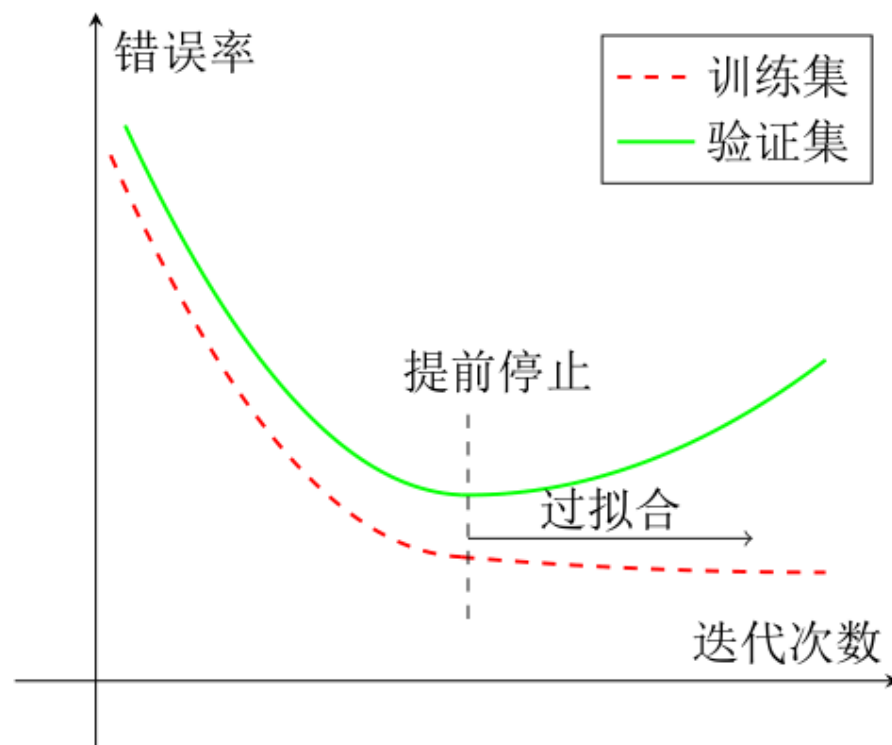
权重衰减、随机梯度下降、提前停止





# 提前停止

- ▶ 我们使用一个验证集（Validation Dataset）来测试每一次迭代的参数在验证集上是否最优。如果在验证集上的错误率不再下降，就停止迭代。





## 损失函数

# 平方损失

---

□ **定义**: 真实值与预测值的平方误差, 也称L2 Loss;

□ **形式**:  $L(y, f(x)) = \frac{1}{2}(y - f(x))^2$

□ **应用**: 用于回归问题。

# 0-1损失

---

□ **定义**: 预测值和真实值不相等时为1, 否则为0;

□ **形式**: 
$$L(Y, f(X)) = \begin{cases} 1, Y \neq f(x) \\ 0, Y = f(x) \end{cases}$$

□ **应用**: 用于二分类问题。

# 二元交叉熵损失函数

---

□ **定义**：对数损失函数，也称交叉熵损失，在二分类模型中，预测结果只有两种情况，对每个类别得到的概率分别 $p$ 和 $1-p$ ；

□ **形式**：
$$L_i = L(p_i, y_i) = \begin{cases} -\log(p_i), & y_i = 1 \\ -\log(1 - p_i), & y_i = 0 \end{cases} = -y_i \log(p_i) - (1 - y_i) \log(1 - p_i)$$

□ **应用**：用于二分类问题。

# 多元交叉熵损失函数

---

□ **定义**: 是对二分类损失函数的扩展;

□ **形式**:  $L_i = -\sum_{c=1}^K y_{ic} \log(p_{ic})$  当属于某一类时, 该类的y为1

□ **应用**: 用于多分类问题。 **思考**: 为何交叉熵损失是这种形式?

$$\text{Softmax: } P(y_i) = \frac{\exp(y_i)}{\sum_{j=1}^n \exp(y_j)}$$

# 合页损失函数 (Hinge Loss)

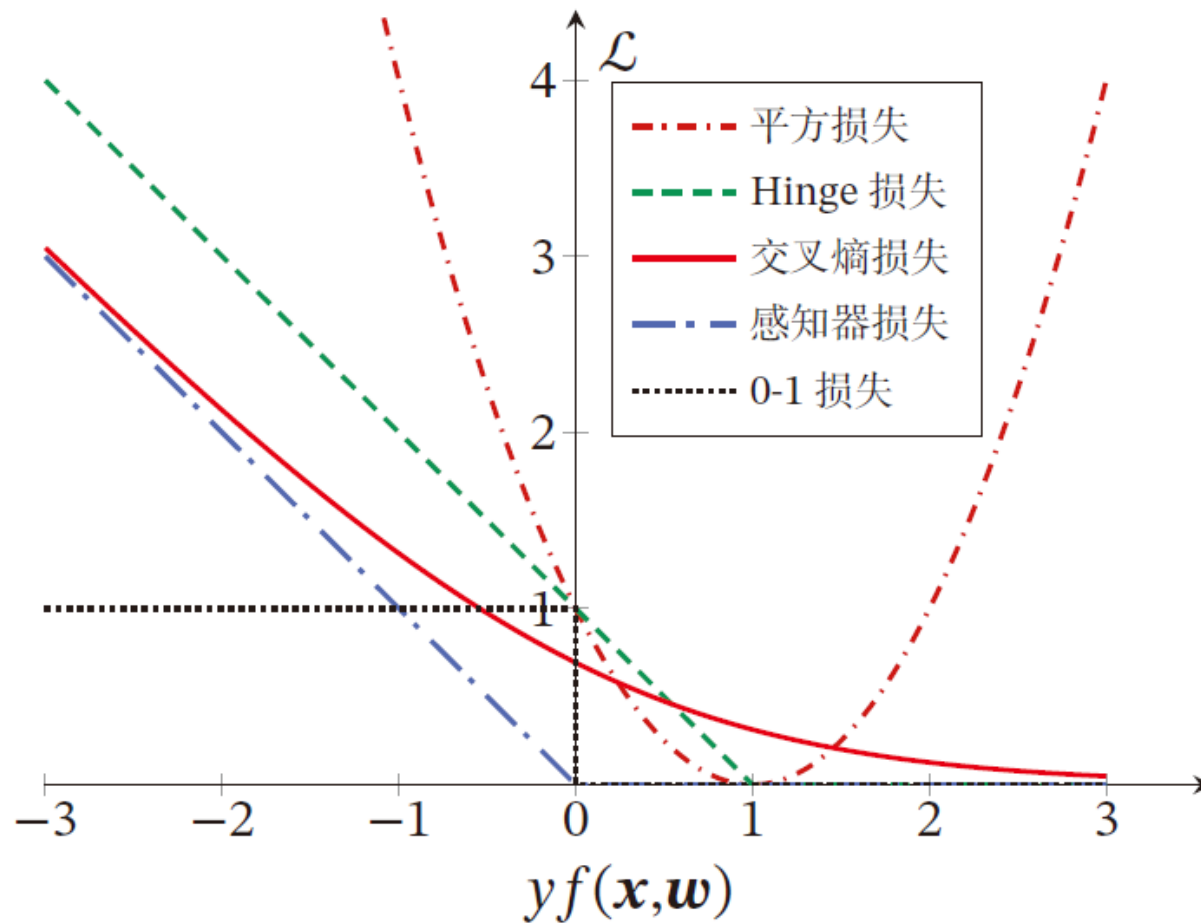
---

□ **定义**: 样本被正确分类, 则损失为0, 否则为 $1 - f \cdot y(x)$ ;

□ **形式**:  $L(y, f(x)) = \max(0, 1 - f \cdot y(x))$

□ **应用**: 用于二分类问题, 例如, SVM采用的就是合页损失函数。

# 损失函数对比







## 参数学习

# 应用到机器学习

---

## ► 对于多分类问题

- 如果使用Softmax回归分类器，相当于网络最后一层设置C个神经元，其输出经过Softmax函数进行归一化后可以作为每个类的条件概率。

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{z}^{(L)})$$

- 采用交叉熵损失函数，对于样本(x,y)，其损失函数为

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = -\mathbf{y}^T \log \hat{\mathbf{y}}$$

# 参数学习

▶ 给定训练集为  $D = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^N$ ，将每个样本  $\mathbf{x}^{(n)}$  输入给前馈神经网络，得到网络输出为  $\hat{\mathbf{y}}^{(n)}$ ，其在数据集  $D$  上的结构化风险函数为：

$$\mathcal{R}(W, \mathbf{b}) = \frac{1}{N} \sum_{n=1}^N \mathcal{L}(\mathbf{y}^{(n)}, \hat{\mathbf{y}}^{(n)}) + \frac{1}{2} \lambda \|W\|_F^2$$

辨析：

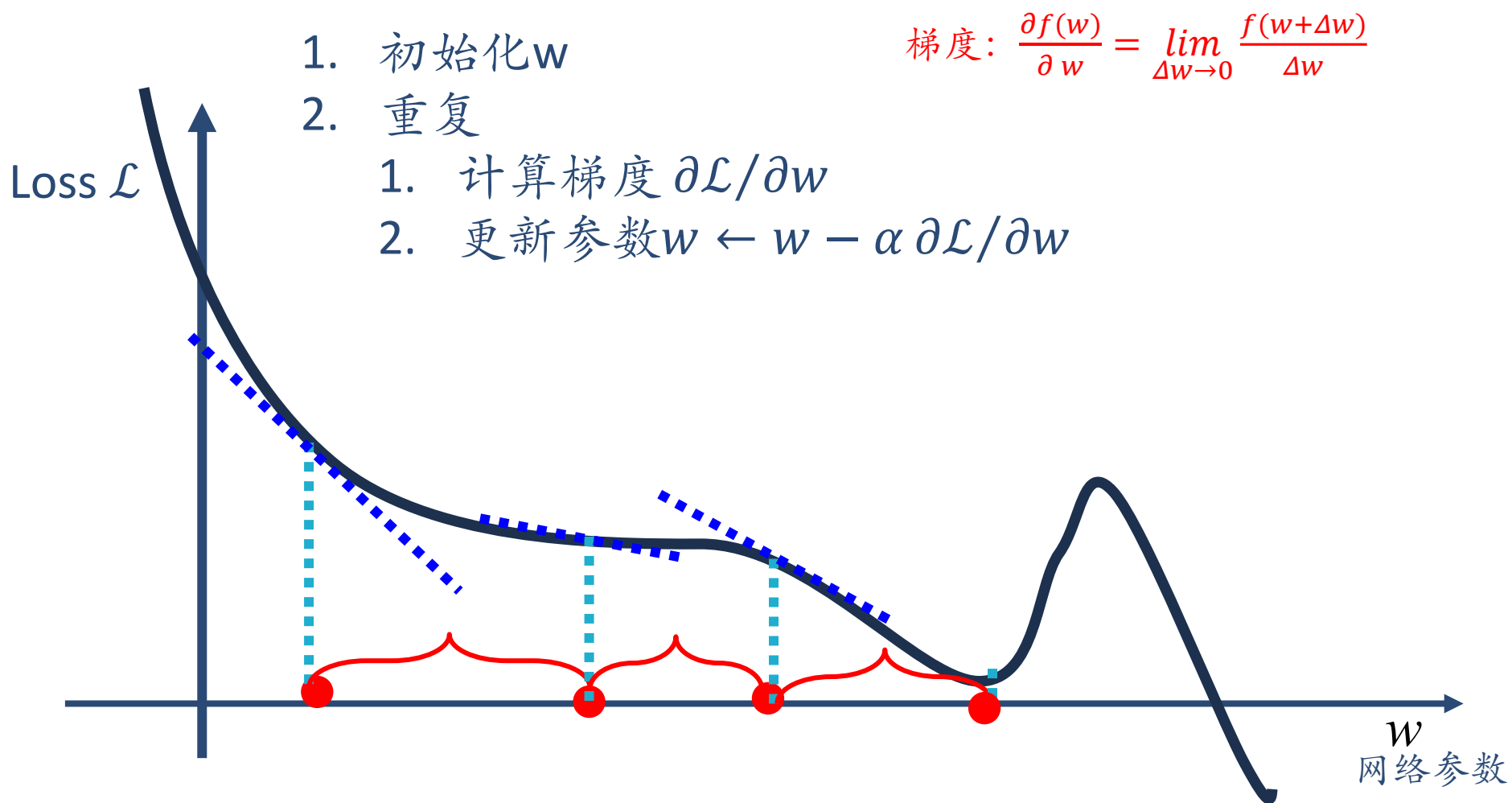
- 损失函数
- 代价函数
- 结构化风险函数

▶ 梯度下降

$$W^{(l)} \leftarrow W^{(l)} - \alpha \frac{\partial \mathcal{R}(W, \mathbf{b})}{\partial W^{(l)}}$$

$$\mathbf{b}^{(l)} \leftarrow \mathbf{b}^{(l)} - \alpha \frac{\partial \mathcal{R}(W, \mathbf{b})}{\partial \mathbf{b}^{(l)}}$$

# 梯度下降



# 如何计算梯度？

---

## ▶ 神经网络为一个复杂的复合函数

### ▶ 链式法则

$$y = f^5(f^4(f^3(f^2(f^1(x))))) \rightarrow \frac{\partial y}{\partial x} = \frac{\partial f^1}{\partial x} \frac{\partial f^2}{\partial f^1} \frac{\partial f^3}{\partial f^2} \frac{\partial f^4}{\partial f^3} \frac{\partial f^5}{\partial f^4}$$

## ▶ 反向传播算法

### ▶ 根据前馈网络的特点而设计的高效方法

## ▶ 一个更加通用的计算方法

### ▶ 自动微分 (Automatic Differentiation, AD)

# 矩阵微积分

---

- ▶ 矩阵微积分 (Matrix Calculus) 是多元微积分的一种表达方式, 即使用矩阵和向量来表示因变量每个成分关于自变量每个成分的偏导数。
- ▶ 分母布局
  - ▶ 标量关于向量的偏导数

$$\frac{\partial y}{\partial \mathbf{x}} = \left[ \frac{\partial y}{\partial x_1}, \dots, \frac{\partial y}{\partial x_p} \right]^T$$

- ▶ 向量关于向量的偏导数

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \dots & \frac{\partial y_q}{\partial x_1} \\ \vdots & \vdots & \vdots \\ \frac{\partial y_1}{\partial x_p} & \dots & \frac{\partial y_q}{\partial x_p} \end{bmatrix} \in \mathbb{R}^{p \times q}$$

# 链式法则

---

► 链式法则 (Chain Rule) 是在微积分中求复合函数导数的一种常用方法。

(1) 若  $x \in \mathbb{R}$ ,  $\mathbf{u} = u(x) \in \mathbb{R}^s$ ,  $\mathbf{g} = g(\mathbf{u}) \in \mathbb{R}^t$ , 则

$$\frac{\partial \mathbf{g}}{\partial x} = \frac{\partial \mathbf{u}}{\partial x} \frac{\partial \mathbf{g}}{\partial \mathbf{u}} \in \mathbb{R}^{1 \times t}.$$

(2) 若  $\mathbf{x} \in \mathbb{R}^p$ ,  $\mathbf{y} = g(\mathbf{x}) \in \mathbb{R}^s$ ,  $\mathbf{z} = f(\mathbf{y}) \in \mathbb{R}^t$ , 则

$$\frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \frac{\partial \mathbf{z}}{\partial \mathbf{y}} \in \mathbb{R}^{p \times t}.$$

(3) 若  $X \in \mathbb{R}^{p \times q}$  为矩阵,  $\mathbf{y} = g(X) \in \mathbb{R}^s$ ,  $z = f(\mathbf{y}) \in \mathbb{R}$ , 则

$$\frac{\partial z}{\partial X_{ij}} = \frac{\partial \mathbf{y}}{\partial X_{ij}} \frac{\partial z}{\partial \mathbf{y}} \in \mathbb{R}.$$

# 反向传播算法

---

已知：

□ 训练集：  $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N$

□ 前馈神经网络：  $L$  ——神经网络总层数

$M_l$  ——第 $l$ 层的神经元个数

$\mathbf{W}^{(l)} \in \mathbb{R}^{M_l \times M_{l-1}}$  ——第 $l-1$ 层到第 $l$ 层的权重矩阵

$\mathbf{b}^{(l)} \in \mathbb{R}^{M_l}$  ——第 $l-1$ 层到第 $l$ 层的偏置

$\mathbf{z}^{(l)} \in \mathbb{R}^{M_l}$  ——第 $l$ 层神经元的净活性值

$\mathbf{a}^{(l)} \in \mathbb{R}^{M_l}$  ——第 $l$ 层神经元的活性值



# 反向传播算法

---

已知：

□ 损失函数：  $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$

惩罚项，暂时忽略

□ 结构化风险函数：  $\mathcal{R}(\mathbf{W}, \mathbf{b}) = \frac{1}{N} \sum_{n=1}^N \mathcal{L}(\mathbf{y}^{(n)}, \hat{\mathbf{y}}^{(n)}) + \frac{1}{2} \lambda \|\mathbf{W}\|_F^2$

问题：

□  $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$  是如何指导权重  $\mathbf{W}$ 、偏置  $\mathbf{b}$  的更新的？

□ 也就是说，计算出了  $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$ ， $\mathbf{W}$ 、 $\mathbf{b}$  应如何更新？

# 反向传播算法

“媒介”

$$\mathbf{z}^{(l)} = W^{(l)} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$$

$$\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial w_{ij}^{(l)}} = \frac{\partial \mathbf{z}^{(l)}}{\partial w_{ij}^{(l)}} \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}}$$

$$\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{b}^{(l)}} = \frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{b}^{(l)}} \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}}$$

$$\begin{aligned} \frac{\partial \mathbf{z}^{(l)}}{\partial w_{ij}^{(l)}} &= \left[ \frac{\partial z_1^{(l)}}{\partial w_{ij}^{(l)}}, \dots, \frac{\partial z_i^{(l)}}{\partial w_{ij}^{(l)}}, \dots, \frac{\partial z_{m^{(l)}}^{(l)}}{\partial w_{ij}^{(l)}} \right] \\ &= \left[ 0, \dots, \frac{\partial (\mathbf{w}_{i:}^{(l)} \mathbf{a}^{(l-1)} + b_i^{(l)})}{\partial w_{ij}^{(l)}}, \dots, 0 \right] \\ &= \left[ 0, \dots, a_j^{(l-1)}, \dots, 0 \right] \\ &\triangleq \mathbb{I}_i(a_j^{(l-1)}) \in \mathbb{R}^{m^{(l)}}, \end{aligned}$$

$$\delta^{(l)} = \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}} \in \mathbb{R}^{m^{(l)}} \quad \text{误差项}$$

$$\frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{b}^{(l)}} = \mathbf{I}_{m^{(l)}} \in \mathbb{R}^{m^{(l)} \times m^{(l)}}$$

计算  $\delta^{(l)} = \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}} \in \mathbb{R}^{m^{(l)}}$

$$\mathbf{a}^{(l)} = f_l(\mathbf{z}^{(l)})$$

“媒介”

$$\mathbf{z}^{(l+1)} = W^{(l+1)} \mathbf{a}^{(l)} + \mathbf{b}^{(l+1)}$$

$$\begin{aligned} \frac{\partial \mathbf{z}^{(l+1)}}{\partial \mathbf{a}^{(l)}} &= (W^{(l+1)})^T & \delta^{(l)} &\triangleq \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}} & \frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}} &= \frac{\partial f_l(\mathbf{z}^{(l)})}{\partial \mathbf{z}^{(l)}} \\ & & & & &= \text{diag}(f'_l(\mathbf{z}^{(l)})) \\ & & & & &= \text{diag}(f'_l(\mathbf{z}^{(l)})) \cdot \frac{\partial \mathbf{z}^{(l+1)}}{\partial \mathbf{a}^{(l)}} \cdot \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l+1)}} \\ & & & & &= \text{diag}(f'_l(\mathbf{z}^{(l)})) \cdot (W^{(l+1)})^T \cdot \delta^{(l+1)} \\ & & & & &= f'_l(\mathbf{z}^{(l)}) \odot ((W^{(l+1)})^T \delta^{(l+1)}), \quad \text{递推关系式} \end{aligned}$$

# 反向传播算法

---

在计算出上面三个偏导数之后, 公式 (4.49) 可以写为

$$\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial w_{ij}^{(l)}} = \mathbb{I}_i(a_j^{(l-1)})\delta^{(l)} = \delta_i^{(l)} a_j^{(l-1)}.$$

进一步,  $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$  关于第  $l$  层权重  $W^{(l)}$  的梯度为

$$\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial W^{(l)}} = \delta^{(l)} (\mathbf{a}^{(l-1)})^\top.$$

同理,  $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$  关于第  $l$  层偏置  $\mathbf{b}^{(l)}$  的梯度为

$$\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{b}^{(l)}} = \delta^{(l)}.$$

[illegible]

— **EXERCISE 10** — **THE FUTURE OF THE FUTURE** —

\_\_\_\_\_

Figure 1. The proposed research model.

© 2013 Pearson Education, Inc. or its affiliate(s). All rights reserved.

\_\_\_\_\_

\_\_\_\_\_

// 计算每一层参数的导数

[illegible]
$$L_1 = \{ \langle M, x \rangle \mid M \text{ is a Turing machine and } x \text{ is a string such that } M \text{ accepts } x \}$$

$$0 \leq \nu_i \leq 1, \quad \sum_{i=1}^n \nu_i = 1, \quad \text{and} \quad \sum_{i=1}^n \nu_i \log \nu_i = -1. \quad (1.10)$$

$$\partial_{\theta} \log \pi(\theta) = \frac{1}{\pi(\theta)} \frac{\partial \pi(\theta)}{\partial \theta}$$
$$\frac{\partial \mathcal{L}}{\partial \mathbf{y}} = \delta(l). \quad // \text{ 公式 (4.69)}$$
$$W^{(l)} \leftarrow W^{(l)} - \alpha(\rho^{(l)}(g^{(l-1)}))^+ \pm \lambda W^{(l)}.$$
$$\mathbf{l}(l) \quad \mathbf{l}(l) \quad \dots \mathbf{s}(l)$$

---



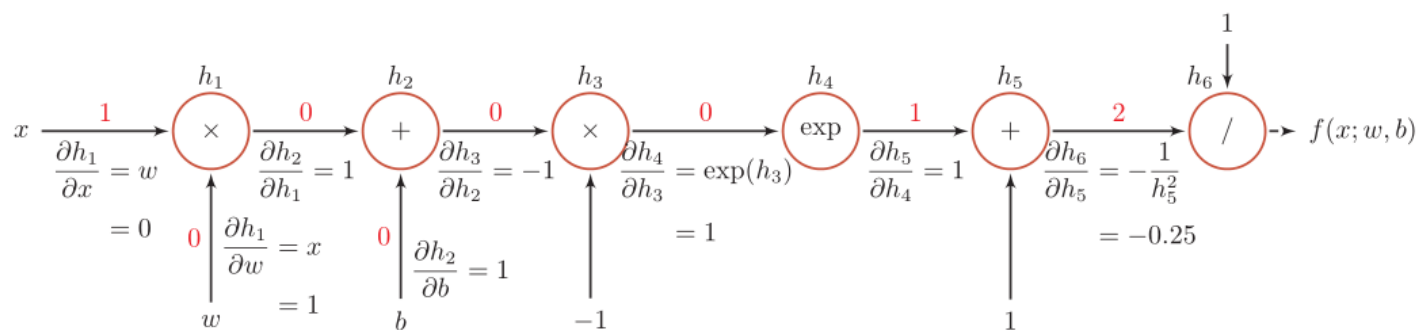
## 计算图与自动微分

# 计算图与自动微分

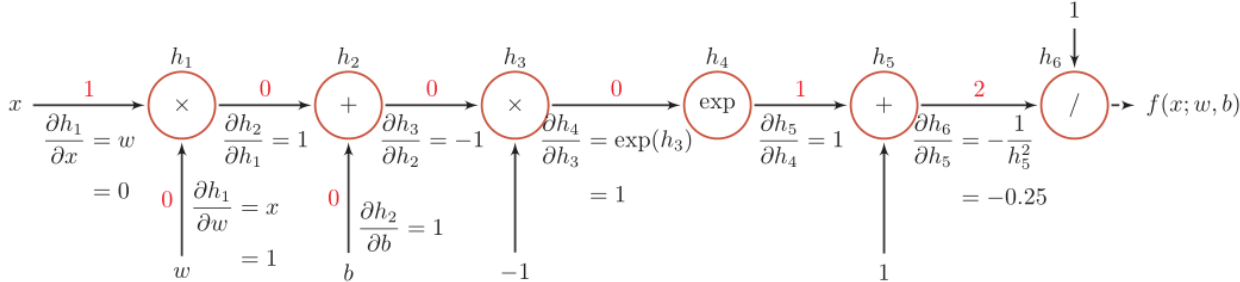
► 自动微分是利用链式法则来自动计算一个复合函数的梯度。

$$f(x; w, b) = \frac{1}{\exp(-(wx + b)) + 1}.$$

► 计算图



# 计算图



函数	导数	
$h_1 = x \times w$	$\frac{\partial h_1}{\partial w} = x$	$\frac{\partial h_1}{\partial x} = w$
$h_2 = h_1 + b$	$\frac{\partial h_2}{\partial h_1} = 1$	$\frac{\partial h_2}{\partial b} = 1$
$h_3 = h_2 \times -1$	$\frac{\partial h_3}{\partial h_2} = -1$	
$h_4 = \exp(h_3)$	$\frac{\partial h_4}{\partial h_3} = \exp(h_3)$	
$h_5 = h_4 + 1$	$\frac{\partial h_5}{\partial h_4} = 1$	
$h_6 = 1/h_5$	$\frac{\partial h_6}{\partial h_5} = -\frac{1}{h_5^2}$	

当 $x = 1, w = 0, b = 0$ 时，可以得到

$$\begin{aligned} \frac{\partial f(x; w, b)}{\partial w} \Big|_{x=1, w=0, b=0} &= \frac{\partial f(x; w, b)}{\partial h_6} \frac{\partial h_6}{\partial h_5} \frac{\partial h_5}{\partial h_4} \frac{\partial h_4}{\partial h_3} \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial w} \\ &= 1 \times -0.25 \times 1 \times 1 \times -1 \times 1 \times 1 \\ &= 0.25. \end{aligned}$$



# 自动微分

---

## ▶ 前向模式和反向模式

- ▶ 反向模式和反向传播的计算梯度的方式相同
- ▶ 如果函数和参数之间有多条路径，可以将这多条路径上的导数再进行相加，得到最终的梯度。

# 反向传播算法 (自动微分的反向模式)

---

- ▶ 前馈神经网络的训练过程可以分为以下三步
  - ▶ 前向计算每一层的状态和激活值，直到最后一层
  - ▶ 反向计算每一层的参数的偏导数
  - ▶ 更新参数

# 静态计算图和动态计算图

---

- ▶ 静态计算图是在编译时构建计算图，计算图构建好之后在程序运行时不能改变。
  - ▶ Theano和Tensorflow
- ▶ 动态计算图是在程序运行时动态构建。两种构建方式各有优缺点。
  - ▶ DyNet, Chainer和PyTorch
- ▶ 静态计算图在构建时可以进行优化，并行能力强，但灵活性比较低。动态计算图则不容易优化，当不同输入的网络结构不一致时，难以并行计算，但是灵活性比较高。

# 如何实现?



# Getting started: 30 seconds to Keras

---

```
from keras.models import Sequential
from keras.layers import Dense, Activation
from keras.optimizers import SGD

model = Sequential()
model.add(Dense(output_dim=64, input_dim=100))
model.add(Activation("relu"))
model.add(Dense(output_dim=10))
model.add(Activation("softmax"))

model.compile(loss='categorical_crossentropy',
              optimizer='sgd', metrics=['accuracy'])

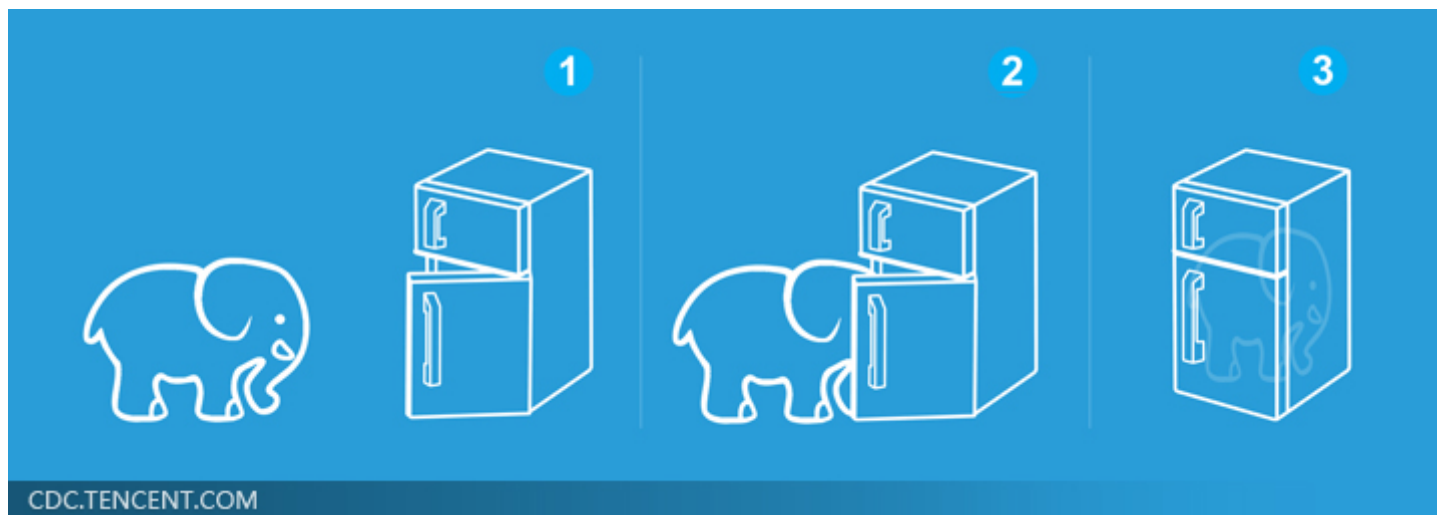
model.fit(X_train, Y_train, nb_epoch=5, batch_size=32)

loss = model.evaluate(X_test, Y_test, batch_size=32)
```

# 深度学习的三个步骤



Deep Learning is so simple .....

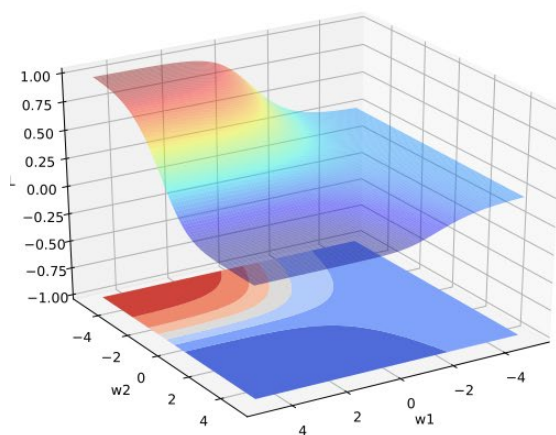




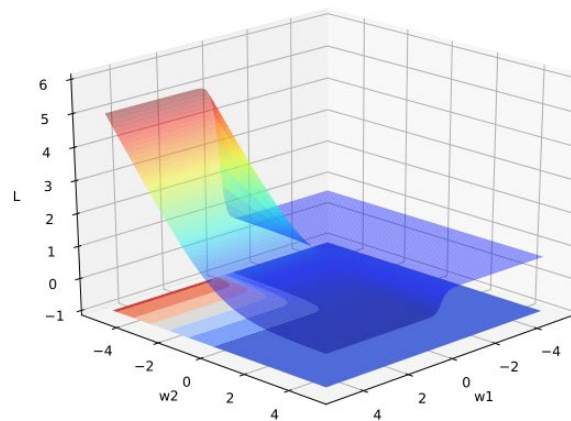
## 优化问题

# 优化问题

## ► 非凸优化问题



(a) 平方误差损失



(b) 交叉熵损失

图 4.9 神经网络  $y = \sigma(w_2\sigma(w_1x))$  的损失函数

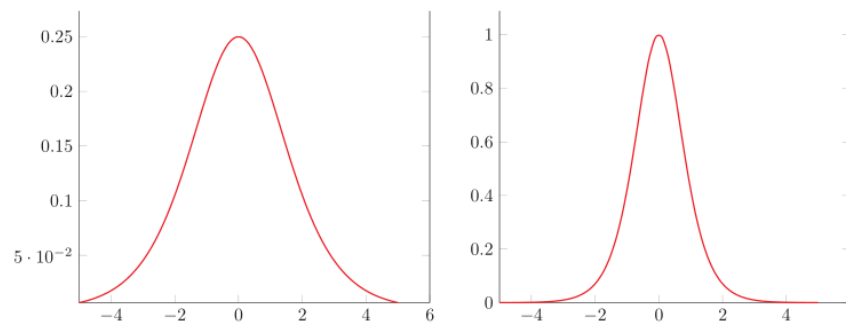


# 优化问题

## ► 梯度消失问题 (Vanishing Gradient Problem)

$$y = f^5(f^4(f^3(f^2(f^1(x)))))$$

$$\frac{\partial y}{\partial x} = \frac{\partial f^1}{\partial x} \frac{\partial f^2}{\partial f^1} \frac{\partial f^3}{\partial f^2} \frac{\partial f^4}{\partial f^3} \frac{\partial f^5}{\partial f^4}$$



(a) logistic 函数的导数

(b) tanh 函数的导数

# 优化问题

---

## ▶ 难点

- ▶ 参数过多，影响训练
- ▶ 非凸优化问题：即存在局部最优而非全局最优解，影响迭代
- ▶ 梯度消失问题，下层参数比较难调
- ▶ 参数解释起来比较困难

## ▶ 需求

- ▶ 计算资源要大
- ▶ 数据要多
- ▶ 算法效率要好：即收敛快

# 实验准备

---

- ▶ 复习Python语法；
- ▶ 了解PyTorch基本语法。

谢 谢

<https://nndl.github.io/>