日期: /

神经网络

一. 神经元模型:

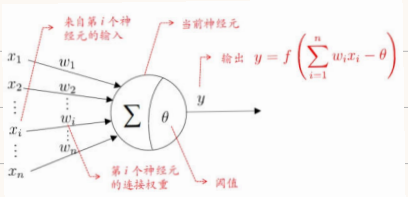
1. 神经网络的定义

神经网络是由具有适应性的简单单元组成的广泛并行互联的网络. 它的组织能够模拟生物神经系统对真实世界物体作出的反应

机器学习中的神经网络指"神经网络学习"

2. 神经元模型: 上述定义的简单单元是神经网络的基本成分

3. M-P 神经元模型.



M-P神经元模型

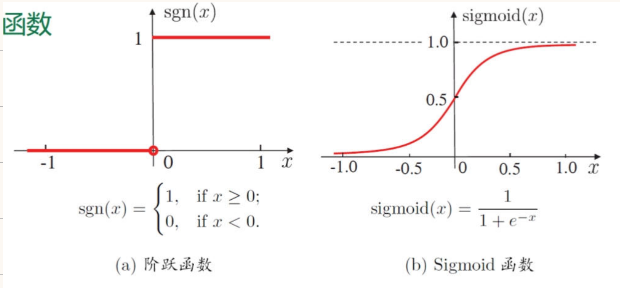
① 输入: 来自其他 n 个神经元传递过来的输入信号 $(x_1, x_2, \cdots x_n)$

② 处理: 输入信号通过带权重 $(w_1, w_2, \cdots, w_n)$ 的连接进行传递, 神经元接收到总输入值 $\sum_{i=1}^{n} w_i x_i$ 将与神经元的阈值 θ 进行比较

③ 输出: 通过激活函数的处理得到神经元输出

4. 激活函数:

日期:　　/



(a) 阶跃函数　　(b) Sigmoid 函数

① Sgn(x) 阶跃函数.

0 表示抑制神经元, 1 表示激活神经元.

缺点: 0处不可导, 具有不连续, 不光滑的性质

② Sigmoid(x)

定义域 $[-\infty, +\infty]$, 值域 $(0,1)$

优点: 均可导.

5. 多层网络:

① 多层前馈神经网络

(1) 定义: 每层神经元与下一层神经元全互联, 神经元之间不存在同层连接也不存在跨层连接.

(2) 前馈: 输入层接受外界输入, 隐含层与输出层神经元对信号进行加工, 最终结果由输出层神经元输出.

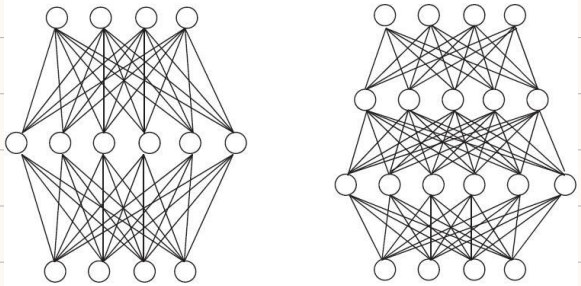
(3) 学习: 根据训练数据来调整神经元之间的连接权以及每个功能神经元的"阈值"

日期: /

② 多层网络：包含隐层的神经网络。

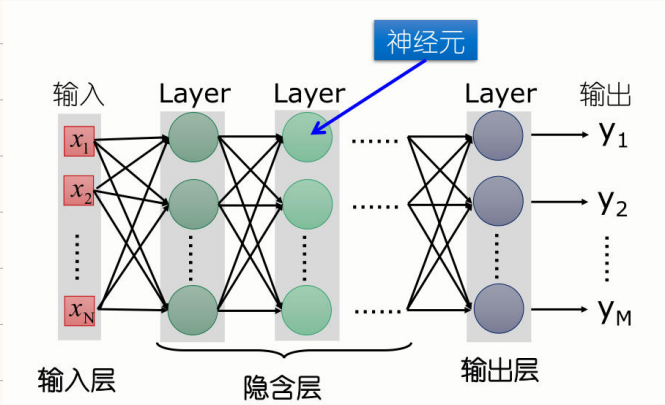
{ 浅层神经网络：仅含有一个隐含层。
  深度神经网络：至少含有二个隐含层。

层数：有权重个数，即二个神经元之间的连线。



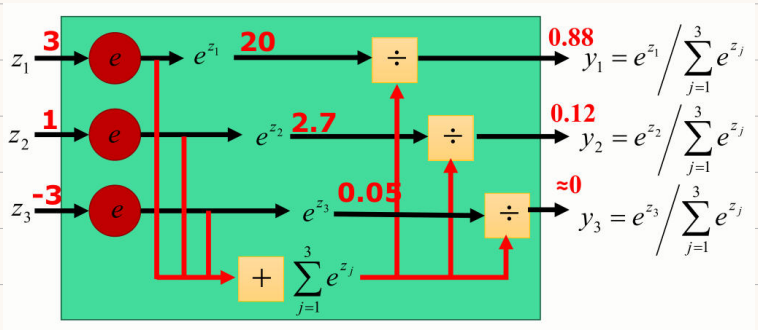
(a) 单隐层前馈网络　　(b) 双隐层前馈网络

③ 多层网络要素。



多分类任务输出层 softmax 层：

日期: /



二. 神经网络模型架构

1. 训练数据：图像以及相应标记 labels

2. 优化函数：网络实际输出和目标值之间的欧几里得距离或交叉熵.

$\hat{y}$：预测为1的概率. y：标签

① 欧几里得距离：$L(\theta) = \| y - \hat{y} \|_2^2$

$= \sqrt{\frac{1}{2} \sum_{i=1} (y_i - \hat{y}_i)^2}$

② 交叉熵：

$$L = -\frac{1}{n} \sum y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log (1 - \hat{y}^{(i)})$$

当 $y=1$ 时 $L = -\log(\hat{y})$

当 $y=0$ 时 $L = -\log(1 - \hat{y})$

日期: /

三. 反向传播算法

1. 目的：通过设置网络参数使得误差最小.

$\theta = \{W^1, b^1, W^2, b^2, \dots W^L, b^L\}$

⇓ 最小化

$C(\theta) = \sum_k \|y_k - f(x_k;\theta)\|^2$ 对所有样例

2. 模型解析：

假设模型函数集 f_1, f_2

训练数据 $(x_1, y_1)\ (x_2, y_2)$ ⇒ 训练：挑选最佳模型函数 f^*

⇓

最佳 f^*

3. 模型求解：最优化问题.

① 目标函数：$\theta^* = \arg\min_\theta C(\theta)$ [C 代价函数，θ 参数]

② 求解：梯度降 (Gradient Descent)

假设 θ 有两个变量 $\{\theta_1, \theta_2\}$

随机选择初始点 $\theta^0 = \begin{bmatrix}\theta_1^0 \\ \theta_2^0\end{bmatrix}$

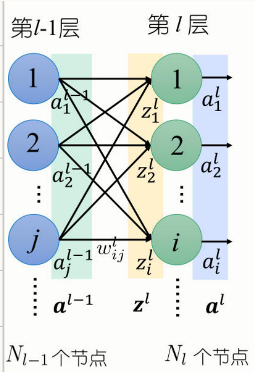
$$\nabla C(\theta) = \begin{bmatrix}\partial C(\theta_1)/\partial\theta_1 \\ \partial C(\theta_2)/\partial\theta_2\end{bmatrix}$$

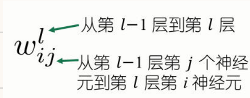
$$\begin{bmatrix}\theta_1^1 \\ \theta_2^1\end{bmatrix} = \begin{bmatrix}\theta_1^0 \\ \theta_2^0\end{bmatrix} - \eta \begin{bmatrix}\partial C(\theta_1^0)/\partial\theta_1 \\ \partial C(\theta_2^0)/\partial\theta_2\end{bmatrix} \Rightarrow \theta^1 = \theta^0 - \eta\nabla C(\theta^0)$$

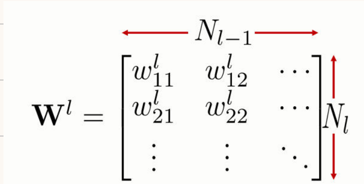
$$\begin{bmatrix}\theta_1^2 \\ \theta_2^2\end{bmatrix} = \begin{bmatrix}\theta_1^1 \\ \theta_2^1\end{bmatrix} - \eta \begin{bmatrix}\partial C(\theta_1^1)/\partial\theta_1 \\ \partial C(\theta_2^1)/\partial\theta_2\end{bmatrix} \Rightarrow \theta^2 = \theta^1 - \eta\nabla C(\theta^1)$$

4. 反向传播算法

① 符号说明:







$$\mathbf{W}^l = \begin{bmatrix} w_{11}^l & w_{12}^l & \cdots \\ w_{21}^l & w_{22}^l & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

两层之间的权构成的矩阵

记号	含义
a_i^l	第 l 层第 i 个神经元的激活值
$\boldsymbol{a}^l$	第 l 层的神经元的激活向量
z_i^l	第 l 层第 i 个神经元的激活函数的带权输入
$\boldsymbol{z}^l$	第 l 层神经元的带权输入
w_{ij}^l	从第 $l-1$ 层第 j 个神经元到第 l 层第 i 个神经元的权重
$\boldsymbol{W}^l$	连接从第 $l-1$ 层到第 l 层的权重矩阵
b_i^l	第 l 层的第 i 个神经元的偏置(bias)
$\boldsymbol{b}^l$	第 l 层的偏置向量(bias vector)

(1) a^{l-1} 与 z^l 关系: 加权

$$z_i^l = \sum_k w_{ik}^l a_k^{l-1} + b_i^l \overset{广}{\Rightarrow} z^l = w^l a^{l-1} + b^l$$

(2) z^l 与 a^l 关系: 激活函数.

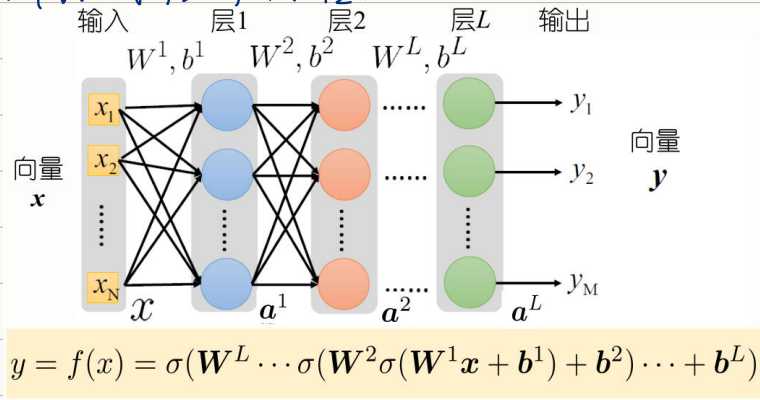
日期: /

$a_i^l = \sigma(z_i^l) \Rightarrow a^l = \sigma(z^l)$

(3) a^l 与 a^{l-1} 关系.

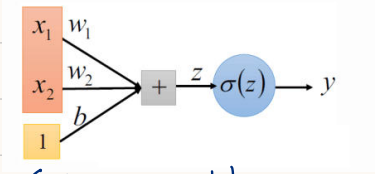
$a^l = \sigma(w^l a^{l-1} + b^l)$

② 神经网络前向传播



$$y = f(x) = \sigma(W^L \cdots \sigma(W^2 \sigma(W^1 x + b^1) + b^2) \cdots + b^L)$$

③ 梯度降 (3个参数为例)



$\theta = \{w, b\} = \{w_1, w_2, b\}$ $y = \sigma(wx+b)$

计算梯度:

$$\nabla_\theta C(\theta) = \left(\frac{\partial C(\theta)}{\partial w_1}; \frac{\partial C(\theta)}{\partial w_2}; \frac{\partial C(\theta)}{\partial b}\right)$$

$$\begin{bmatrix} w_1^{i+1} \\ w_2^{i+1} \\ b^{i+1} \end{bmatrix} = \begin{bmatrix} w_1^{i} \\ w_2^{i} \\ b^{i} \end{bmatrix} - \eta \begin{bmatrix} \frac{\partial C(\theta)}{\partial w_1} \\ \frac{\partial C(\theta)}{\partial w_2} \\ \frac{\partial C(\theta)}{\partial b} \end{bmatrix}$$

日期: /

预测值 ↑ → 目标值

其中 $C(\theta) = \|y - \hat{y}\|^2 = \|G(wx+b) - \hat{y}\|^2$

$G(x) = \frac{1}{1+e^{-x}}$ $G'(x) = G(x)(1-G(x))$

$$w_1^{(t+1)} = w_1^{(t)} - \eta \frac{\partial C(\theta^{(t)})}{\partial w_1} \qquad \frac{\partial C(\theta)}{\partial w_1} = 2(\sigma(Wx+b) - \hat{y})[1-\sigma(Wx+b)]\sigma(Wx+b)x_1$$

$$w_2^{(t+1)} = w_2^{(t)} - \eta \frac{\partial C(\theta^{(t)})}{\partial w_2} \qquad \frac{\partial C(\theta)}{\partial w_2} = 2(\sigma(Wx+b) - \hat{y})[1-\sigma(Wx+b)]\sigma(Wx+b)x_2$$

$$b^{(t+1)} = b^{(t)} - \eta \frac{\partial C(\theta^{(t)})}{\partial b} \qquad \frac{\partial C(\theta)}{\partial b} = 2(\sigma(Wx+b) - \hat{y})[1-\sigma(Wx+b)]\sigma(Wx+b)$$

(4) BP: Error Back propagation

(1) 目的: 为了高效计算梯度

(2) 前向传播与反向传播对比

- 前向传播
  - 从输入 x 到输出 y，信息通过网络前向传播
  - 在训练阶段，前向传播可以继续向前，直到它产生标量代价函数$C(\theta)$
- 反向传播
  - 允许来自代价函数的信息然后通过网络反向流动，以便计算**梯度**
  - 可以被应用到任何函数

(3) 导数:

$$f'(x) = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon) - f(x)}{\varepsilon}$$

$$\Rightarrow \quad f(x+\varepsilon) = f(x) + \varepsilon f'(x) + O(\varepsilon)$$

日期: /

记 $f'(x)=\frac{dy}{dx}$，即 $x\to x+\Delta x$；$y\to y+\frac{dy}{dx}\Delta x$

x 变化为 $x+\Delta x$ 时，y 变化为 $y+\frac{dy}{dx}\Delta x$

4) 链式法则 $y=f(x)$ $z=g(y)$ $x\xrightarrow{f} y\xrightarrow{g} z$.

$$\begin{cases} x\to x+\Delta x \Rightarrow y\to y+\frac{dy}{dx}\Delta x \\ y\to y+\Delta y \Rightarrow z\to z+\frac{dz}{dy}\Delta y. \end{cases}$$

$$x\to x+\Delta x \Rightarrow z\to z+\frac{dz}{dy}\cdot\frac{dy}{dx}\Delta x$$

之同于神经网络：

$w=f(w)$，$y=f(x)$，$z=f(y)$

$\Delta w\to \Delta x\to \Delta y\to \Delta z$.

$$\frac{dz}{dw}=\frac{dz}{dy}\cdot\frac{dy}{dx}\cdot\frac{dx}{dw}=f'(y)\cdot f'(x)\cdot f'(w)$$

$$=f'(f(f(w)))f'(f(w))\cdot f'(w)$$

5) 代价函数的2个假设

$$C=\frac{1}{2n}\sum_x \|y(x)-a^L(x)\|^2$$

- 代价函数写为每个训练样本上函数 C_x 的均值

$$C=\frac{1}{n}\sum_x C_x \qquad C_x=\frac{1}{2}\|y-a^L\|^2$$

对每个样本算 $\frac{\partial C_x}{\partial w}$ 和 $\frac{\partial C_x}{\partial b}$，然后对获得 $\frac{\partial C}{\partial w}$ $\frac{\partial C}{\partial b}$

日期: /

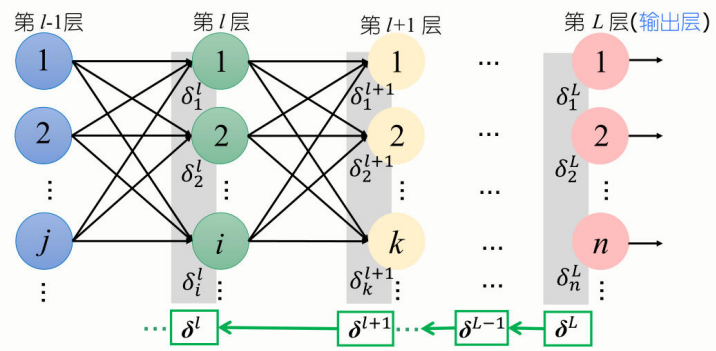
- 代价函数可写成神经网络输出的激活函数

$$C = \frac{1}{2}\|y - a^L\|^2 = \frac{1}{2}\sum_j (y_j - a_j^L)^2$$

$$C = C(a^L)$$

(1) 误差 δ_i^l：定义第 l 层第 i 个神经元上误差为

$$\delta_i^l = \frac{\partial C}{\partial z_i^l} \quad \left[z_i^l = \sum_k w_{ik}^L a_k^{l-1} + b_i^l\right]$$



计算所有层误差：从第 L 层到第 1 层.

{ 首先计算 δ^L
  由 δ^{l+1} 计算 δ^l

$a_i^l = G(z_i^l)$

δ^l 计算：

$$\delta_i^l = \frac{\partial C}{\partial z_i^l} = \frac{\partial C}{\partial a_i^l} \cdot \frac{\partial a_i^l}{\partial z_i^l} = \frac{\partial C}{\partial a_i^l} \cdot G'(z_i^l)$$

$\partial z_i^l \to \partial a_i^l \to \partial C$

日期: /

$$\nabla C(a) = \begin{bmatrix} \frac{\partial C}{\partial a_1} \\ \frac{\partial C}{\partial a_2} \\ \vdots \\ \frac{\partial C}{\partial a_n} \end{bmatrix} \qquad \sigma'(z^l) = \begin{bmatrix} \sigma'(z_1^l) \\ \sigma'(z_2^l) \\ \vdots \\ \sigma'(z_n^l) \end{bmatrix}$$

$$\delta^l = \nabla C(a) \odot \sigma'(z^l)$$

$\delta^{l+1} \Rightarrow \delta^l$ 计算

$$\Delta z_i^l \to \Delta a_i^l \to \begin{matrix} \Delta z_1^{l+1} \\ \Delta z_2^{l+1} \\ \Delta z_3^{l+1} \\ \vdots \end{matrix} \to \Delta C$$

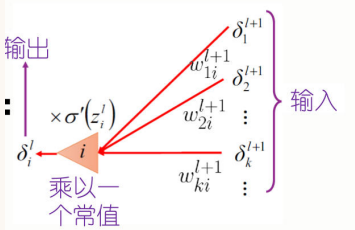
$$\delta_i^l = \frac{\partial C}{\partial z_i^l} = \sum_k \left(\frac{\partial C}{\partial z_k^{l+1}} \cdot \frac{\partial z_k^{l+1}}{\partial a_i^l} \cdot \frac{\partial a_i^l}{\partial z_i^l} \right)$$

$$= \frac{\partial a_i^l}{\partial z_i^l} \sum_k \underbrace{\frac{\partial C}{\partial z_k^{l+1}}}_{\delta_k^{l+1}} \cdot \frac{\partial z_k^{l+1}}{\partial a_i^l}$$

$$\sum w_{ki}^{l+1} \cdot a_i^l + b_k^{l+1}$$

$$= \sigma'(z_i^l) \sum \delta_k^{l+1} \cdot \frac{\partial z_k^{l+1}}{\partial a_i^l}$$

$$= \sigma'(z_i^l) \sum \delta_k^{l+1} \cdot w_{ki}^{l+1}$$



$$\delta^l = \sigma'(z^l) \odot (W^{l+1})^T \delta^{l+1}$$

日期: /

① 首先：计算 δ^L $\quad \delta^L = \nabla C(a) \odot \sigma'(z^L)$

② 其次：基于 δ^{l+1} 计算 δ^l $\delta^l = \sigma'(z^l) \odot \left(W^{l+1}\right)^{\mathrm{T}} \delta^{l+1}$

1. 求梯度：$\frac{\partial L}{\partial w_{ij}^l}$ 通过误差计算

$$\frac{\partial L}{\partial w_{ij}^l} = \frac{\partial L}{\partial z_i^l} \cdot \frac{\partial z_i^l}{\partial w_{ij}^l} = \begin{cases} a_j^{l-1}, & z_i^l = \sum w_{ij}^l a_j^{l-1} + b \\ x_j, & l = 1 \text{ [输入层]} \end{cases}$$

$$\frac{\partial L}{\partial b_i^l} = \frac{\partial L}{\partial z_i^l} \cdot \frac{\partial z_i^l}{\partial b_i^l} = 1$$

$$= \delta_i^l$$

向和反的项

$$\frac{\partial C}{\partial w_{ij}^l} = \frac{\partial z_i^l}{\partial w_{ij}^l} \frac{\partial C}{\partial z_i^l}$$

误差信号

$$\begin{cases} a_j^{l-1} & l > 1 \\ x_j & l = 1 \end{cases}$$

δ_i^l

正向传递

$$z^1 = W^1 x + b^1$$
$$a^1 = \sigma(z^1)$$
……
$$z^{l-1} = W^{l-1} a^{l-2} + b^{l-1}$$
$$a^{l-1} = \sigma(z^{l-1})$$

反向传递

$$\delta^L = \sigma'(z^L) \odot \nabla_a C$$
$$\delta^{L-1} = \sigma'(z^{L-1}) \odot (W^L)^{\mathrm{T}} \delta^L$$
……
$$\delta^l = \sigma'(z^l) \odot \left(W^{l+1}\right)^{\mathrm{T}} \delta^{l+1}$$
……

日期: /

反向传播算法

1. **输入x：** 为输入层设置对应的激活值a^1 。
2. **前向传播：** 对每个$l=2,3,...,L$ 计算相应的$z^l=w^la^{l-1}+b^l$ 和$a^l=\sigma(z^l)$
3. **输出层误差δ^L：** 计算向量$\delta^L=\nabla_aC\odot\sigma'(z^L)$
4. **反向误差传播：** 对每个$l=L-1,L-2,...,2$，计算$\delta^l=((w^{l+1})^T\delta^{l+1})\odot\sigma'(z^l)$
5. **输出：** 代价函数的梯度由$\frac{\partial C}{\partial w^l_{jk}}=a^{l-1}_k\delta^l_j$ 和$\frac{\partial C}{\partial b^l_j}=\delta^l_j$ 得出

⑤ 批量梯度降优化.

1. **输入训练样本的集合** $D=\{(x_i,y_i)\}$, $i=1,2,\ldots,N$, N 为样本总数.
2. **对每个训练样本** x**:** 设置对应的输入激活 $a^{x,1}$，并执行下面的步骤:
   - **前向传播:** 对每个 $l=2,3,...,L$ 计算 $z^{x,l}=w^la^{x,l-1}+b^l$ 和 $a^{x,l}=\sigma(z^{x,l})$。
   - **输出误差** $\delta^{x,L}$**:** 计算向量$\delta^{x,L}=\nabla_aC_x\odot\sigma'(z^{x,L})$。
   - **误差:** 对每个 $l=L-1,L-2,...,2$ 计算 $\delta^{x,l}=((w^{l+1})^T\delta^{x,l+1})\odot\sigma'(z^{x,l})$。
3. **梯度下降:** 对每个$l=L-1,L-2,...,2$根据 $w^l\leftarrow w^l-\eta\frac{1}{N}\sum_x\delta^{x,l}(a^{x,l-1})^T$ 和 $b^l\leftarrow b^l-\eta\frac{1}{N}\sum_x\delta^{x,l}$更新权重和偏置。

i) 缺点：(每一次迭代都需要所有样本梯度参与计算，求和平均后迭代更新权重和偏置)；当样本数目为n，训练过程会很慢.

ii) 优点：全局最优解；易于并行实现

⑥ SGD 随机梯度降优化.

日期: /

1. 输入训练样本的集合 $D=\{(x_i,y_i)\}$
2. 对每个训练样本 x：设置对应的输入激活 $a^{x,1}$，并执行下面的步骤：
   - 前向传播：对每个 $l=2,3,...,L$ 计算 $z^{x,l}=w^l a^{x,l-1}+b^l$ 和 $a^{x,l}=\sigma(z^{x,l})$。
   - 输出误差 $\delta^{x,L}$：计算向量$\delta^{x,L}=\nabla_a C_x \odot \sigma'(z^{x,L})$。
   - 误差：对每个 $l=L-1,L-2,...,2$ 计算 $\delta^{x,l}=((w^{l+1})^T\delta^{x,l+1})\odot\sigma'(z^{x,l})$。
3. 梯度下降：对每个 $l=L-1,L-2,...,2$ 根据 $w^l\leftarrow w^l-\eta\delta^{x,l}(a^{x,l-1})^T$ 和 $b^l\leftarrow b^l-\eta\delta^{x,l}$ 更新权重和偏置。

每一次迭代，使用一个样本的梯度参与计算，再迭代更新权重和偏置。

优点：训练速度快

缺点：准确率下降，不是全局最优；不易于并行实现。：噪声

※ SGD 对每一个样本都调整权重，训练过程有随机性

(2) 小批量（mini-batch）梯度下降优化

1. 输入训练样本的集合 $D=\{(x_i,y_i)\}$
2. 对每个训练样本 x：设置对应的输入激活 $a^{x,1}$，并执行下面的步骤：
   - 前向传播：对每个 $l=2,3,...,L$ 计算 $z^{x,l}=w^l a^{x,l-1}+b^l$ 和 $a^{x,l}=\sigma(z^{x,l})$。
   - 输出误差 $\delta^{x,L}$：计算向量$\delta^{x,L}=\nabla_a C_x \odot \sigma'(z^{x,L})$。
   - 误差：对每个 $l=L-1,L-2,...,2$ 计算 $\delta^{x,l}=((w^{l+1})^T\delta^{x,l+1})\odot\sigma'(z^{x,l})$。
3. 梯度下降：对每个 $l=L-1,L-2,...,2$ 根据 $w^l\leftarrow w^l-\eta\frac{1}{m}\sum_x\delta^{x,l}(a^{x,l-1})^T$ 和 $b^l\leftarrow b^l-\eta\frac{1}{m}\sum_x\delta^{x,l}$ 更新权重和偏置。

日期: /

每一次迭代选出 m 个样本的梯度等与计算，求和平均后迭代更新权重和偏置

Batch size (批量大小)：1次迭代使用的样本量

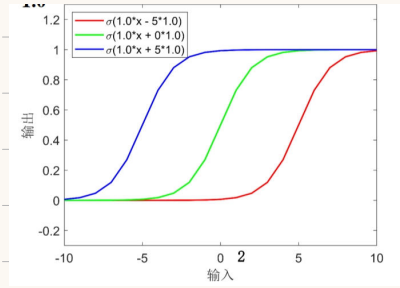
Iteration (迭代)：1次 iteration，使用1个 batch size 样本训练网络参数1次

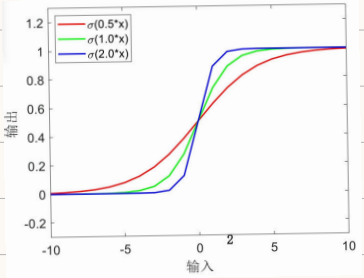
总 iteration = 训练集大小 / batch size

Epoch (期)：1个 epoch 表示遍历了1遍训练集中的所有样本

- 假设训练集有 2560000个样本。现在选择 Batch size = 256 对模型进行训练。
- 总的迭代（iteration）次数： 2560000/256 = 10000
- 每个 Epoch 要训练的样本数量： 2560000
- 需要10000次iteration完成一个epoch

5、bias和w在神经网络作用





w：控制这个S形的陡度

日期: /

bias: 平移曲线.

6. 非线性激活函数

① 举例.

$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$$

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

$$\text{relu}(x) = \max(x, 0)$$

② 使用目的: