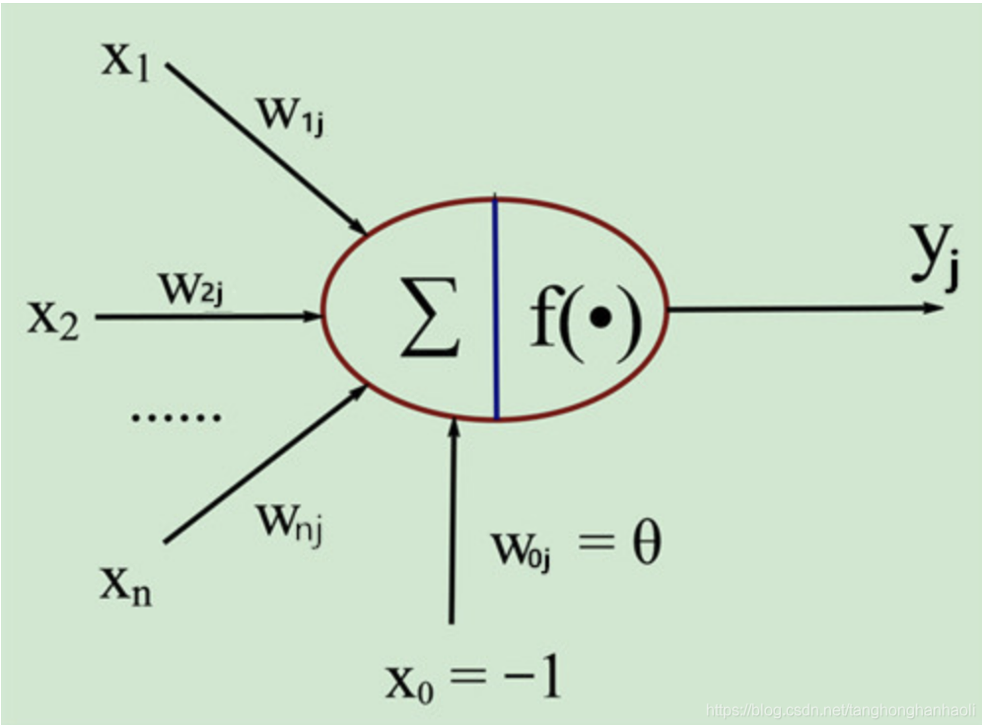
第五章 神经网络

> 深度学习
> 大量data的特殊性
> 算法

5.1神经元模型

neuron = linear(wx+b) + activation激活
model = architecture架构 + parameters 参数



$$
\begin{aligned}
z &= \sum_{d=1}^{D} w_d x_d + b \\
&= w^T x + b \\
&= w^T x \\
&(\ if\ x_0 = -1, w_0 = b)
\end{aligned}
$$

输入值z 经过一个非线性函数$f(\cdot)$ (激活函数) 得到活性值(Activation)

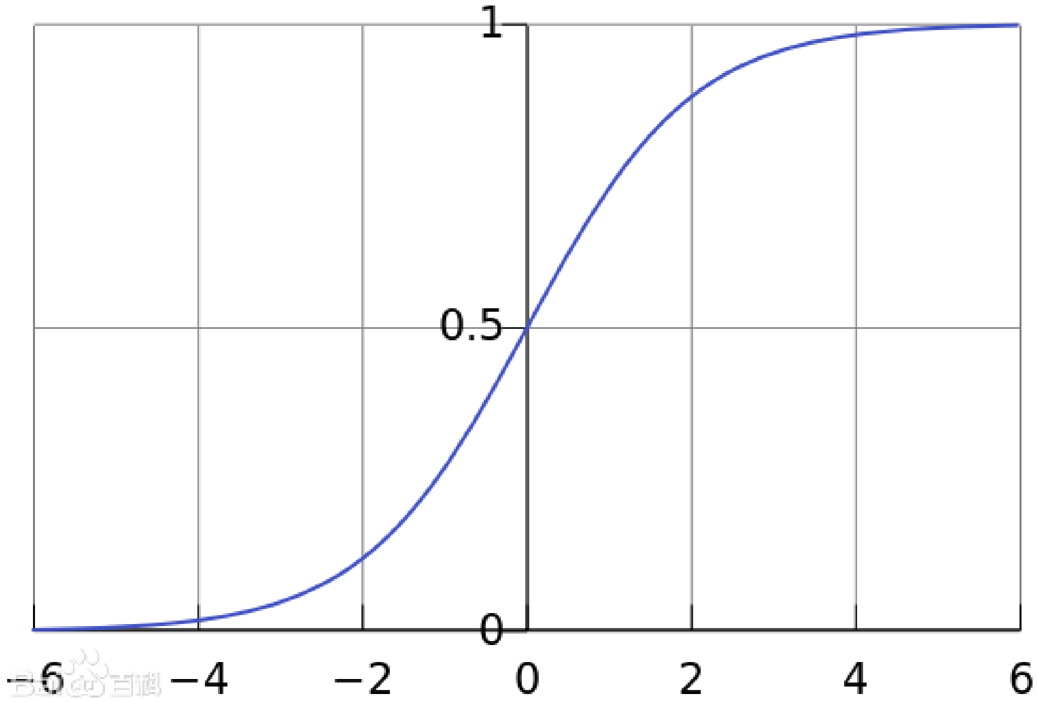
激活函数:

> 性质:
>
> 1. 连续可导的非线性函数(数值优化学习网络参数)
> 2. 函数及其导函数尽可能简单(提高计算效率)
> 3. 导函数的值域空间要在合适区间(训练的效率和稳定性)

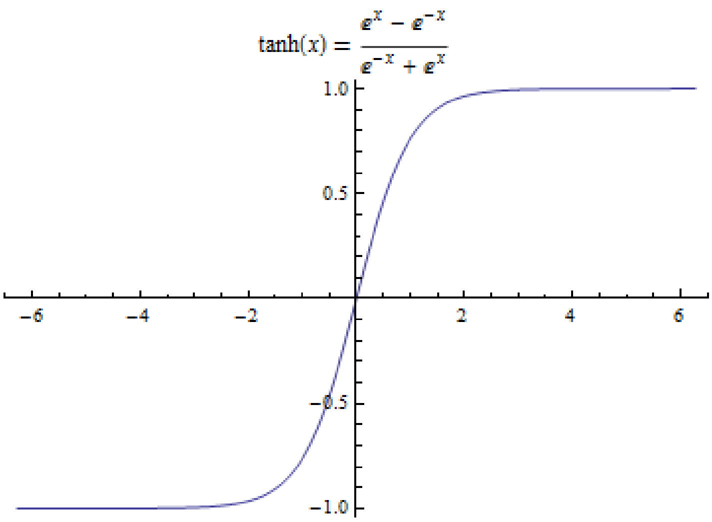
sigmoid函数

- Logistic函数

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



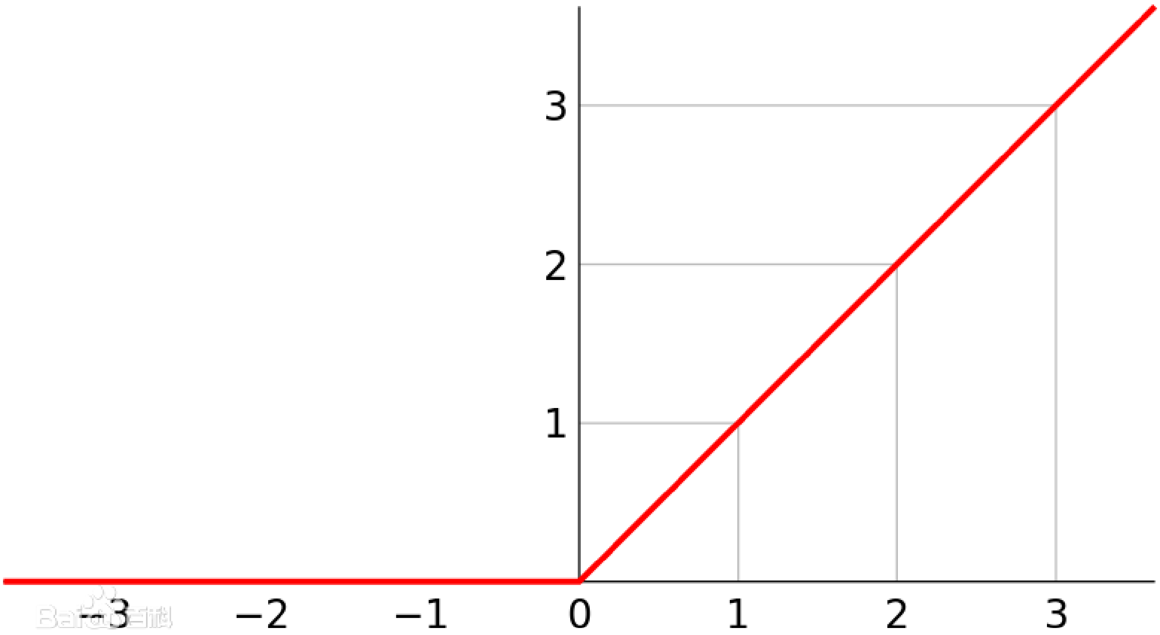
- Tanh函数



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

RELU函数

> **线性整流函数**（Rectified Linear Unit, **ReLU**），又称**修正线性单元,** 是一种人工神经网络中常用的激活函数（activation function），通常指代以斜坡函数及其变种为代表的非线性函数



5.2 感知机perceptron

> 线性分类器,可以看作最简单的神经网络,只有一个神经元

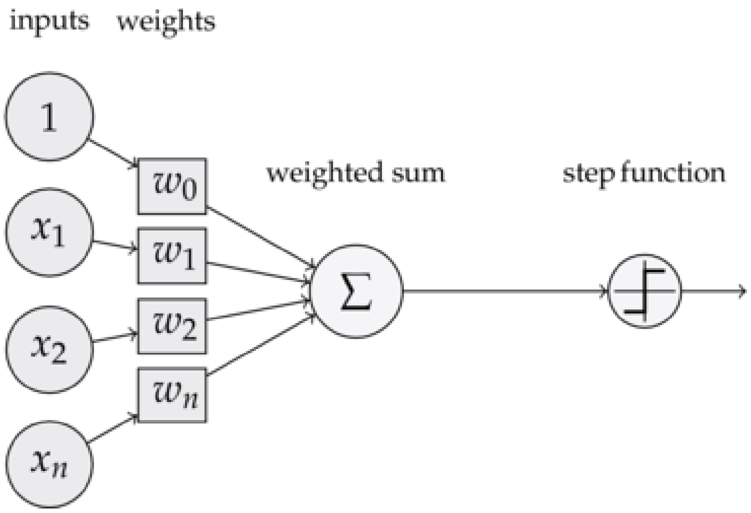
$$y=f(\sum_{i=1}^{n}w_ix_i-\theta)=f(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}-\theta)$$

其中，$\boldsymbol{x}\in\mathbb{R}^n$为样本的特征向量，是感知机模型的输入；$\boldsymbol{w},\theta$是感知机模型的参数，$\boldsymbol{w}\in\mathbb{R}^n$为权重，$\theta$为阈值。上式中的$f$通常设为**符号函数**(单位阶跃函数)，那么感知机模型的公式可进一步表示为

$$\begin{aligned}h(x)&=sign((\sum_{i}^{d}w_ix_i)-threshold) &\qquad (-threshold)(+1)=w_0*x_0\\&=sign(\sum_{0}^{d}w_ix_i)=sign(w^Tx) &\qquad (w,x\text{为向量})\end{aligned}$$

> 由于n维空间中的超平面方程为
> $w_1x_1+w_2x_2+\cdots+w_nx_n+b=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b=0$
> 所以此时感知机模型公式中的$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}-\theta$可以看作是$n$维空间中的一个超平面，通过它将$n$维空间划分为$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}-\theta\geq 0$和$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}-\theta<0$两个子空间，落在前一个

> 子空间的样本对应的模型输出值为1，落在后一个子空间的样本对应的模型输出值为0，以此来实现分类功能。



5.2.1 PLA(Perceptron Learning Algorithm)

1. **目标:**给定一个线性可分的数据集T（参见附录①），感知机的学习目标是求得能对数据集T中的正负样本完全正确划分的分离超平面:
   $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}-\theta=0$

   假设此时误分类样本集合为$M\subseteq T$，对任意一个误分类样本$(\boldsymbol{x},y)\in M$来说，当$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}-\theta\geq 0$时，模型输出值为$\hat{y}=1$，样本真实标记为$y=0$；反之，当$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}-\theta<0$时，模型输出值为$\hat{y}=0$，样本真实标记为$y=1$。综合两种情形可知，以下公式恒成立
   $(\hat{y}-y)(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}-\theta)\geq 0$

2. **损失函数:**

   所以，给定数据集T，其损失函数可以定义为:
   $L(\boldsymbol{w},\theta)=\sum_{\boldsymbol{x}\in M}(\hat{y}-y)(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}-\theta)$
   显然，此损失函数是非负的。如果没有误分类点，损失函数值是0。而且，误分类点越少，误分类点离超平面越近，损失函数值就越小。因此，给定数据集T，损失函数$L(\boldsymbol{w},\theta)$是关于$\boldsymbol{w},\theta$的连续可导函数

3. **学习算法**

   1. 感知机模型的学习问题可以转化为求解损失函数的最优化问题，具体地，给定数据集
      $T=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\ldots,(\boldsymbol{x}_N,y_N)\}$
      其中$\boldsymbol{x}_i\in\mathbb{R}^n,y_i\in\{0,1\}$，求参数$\boldsymbol{w},\theta$，使其为极小化损失函数的解:

      $$\min_{\boldsymbol{w},\theta}L(\boldsymbol{w},\theta)=\min_{\boldsymbol{w},\theta}\sum_{\boldsymbol{x}_i\in M}(\hat{y}_i-y_i)(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i-\theta)$$

      其中$M\subseteq T$为误分类样本集合。若将阈值θ看作一个固定输入为-1的“哑节点”，即
      $-\theta=-1\cdot w_{n+1}=x_{n+1}\cdot w_{n+1}$
      那么$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i-\theta$可化简为

$$\begin{aligned}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x_i}-\theta &= \sum_{j=1}^{n} w_j x_j + x_{n+1}\cdot w_{n+1} \\ &= \sum_{j=1}^{n+1} w_j x_j \\ &= \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x_i}\end{aligned}$$

其中$\boldsymbol{x_i}\in\mathbb{R}^{n+1},\boldsymbol{w}\in\mathbb{R}^{n+1}$。根据该式，可将要求解的极小化问题进一步简化为

$\min\limits_{\boldsymbol{w}} L(\boldsymbol{w})=\min\limits_{\boldsymbol{w}}\sum_{\boldsymbol{x_i}\in M}(\hat{y}_i-y_i)\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x_i}$

2. 假设误分类样本集合M固定，那么可以求得损失函数$L(\boldsymbol{w})$的梯度为：
   $\nabla_{\boldsymbol{w}}L(\boldsymbol{w})=\sum_{\boldsymbol{x_i}\in M}(\hat{y}_i-y_i)\boldsymbol{x_i}$
   感知机的学习算法具体采用的是随机梯度下降法，也就是极小化过程中不是一次使M中所有误分类点的梯度下降，而是一次随机选取一个误分类点使其梯度下降。所以权重$\boldsymbol{w}$的更新公式为
   $\boldsymbol{w}\leftarrow\boldsymbol{w}+\Delta\boldsymbol{w}$
   $\Delta\boldsymbol{w}=-\eta(\hat{y}_i-y_i)\boldsymbol{x}_i=\eta(y_i-\hat{y}_i)\boldsymbol{x}_i$
   相应地，$\boldsymbol{w}$中的某个分量w_i的更新公式即为公式(5.2)。

> 在w中的一个错误记为$(x_{n(t)},y_{n(t)})$ 有$sign(w^Tx_{n(t)}\neq y_{n(t)})$
>
> 纠错:$w_{t+1}:=w_t+y_{n(t)}x_{n(t)}$
> 对于二元分类问题,y的取值为-1和1

左侧详见《神经网络与深度学习》丘锡鹏 p65 3.4.1参数学习李算法3.1

目标:找到超平面	$y^{(n)}w^{*T}x^{(n)}>0,N\in(1,2,\cdots,N)$	$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}-\theta=0$
损失函数	$\mathcal{L}(w;x,y)=max(0,-yw^Tx)$	$L(\boldsymbol{w},\theta)=\sum_{\boldsymbol{x}\in M}(\hat{y}-y)(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}-\theta)$
梯度 $\nabla_{\boldsymbol{w}}L(\boldsymbol{w})$	0 if $yw^Tx>0$ $-yx$ if $yw^Tx<0$	$\nabla_{\boldsymbol{w}}L(\boldsymbol{w})=\sum_{\boldsymbol{x_i}\in M}(\hat{y}_i-y_i)\boldsymbol{x_i}$
更新	$w_{t+1}:=w_t+y_{n(t)}x_{n(t)}$	$\boldsymbol{w}\leftarrow\boldsymbol{w}+\Delta\boldsymbol{w}$ $\Delta\boldsymbol{w}=-\eta(\hat{y}_i-y_i)\boldsymbol{x}_i=\eta(y_i-\hat{y}_i)\boldsymbol{x}_i$

5.2.2收敛问题

定理 2.1（Novikoff） 设训练数据集 $T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$ 是线性可分的，其中 $x_i\in\mathcal{X}=\mathbf{R}^n$， $y_i\in\mathcal{Y}=\{-1,+1\}$， $i=1,2,\cdots,N$， 则

（1）存在满足条件 $\|\hat{w}_{\mathrm{opt}}\|=1$ 的超平面 $\hat{w}_{\mathrm{opt}}\cdot\hat{x}=w_{\mathrm{opt}}\cdot x+b_{\mathrm{opt}}=0$ 将训练数据集完全正确分开；且存在 $\gamma>0$，对所有 $i=1,2,\cdots,N$

$$y_i(\hat{w}_{\mathrm{opt}}\cdot\hat{x}_i)=y_i(w_{\mathrm{opt}}\cdot x_i+b_{\mathrm{opt}})\geqslant\gamma \tag{2.8}$$

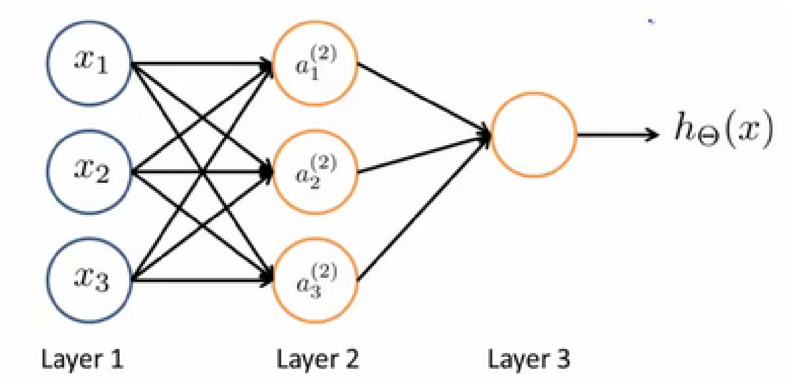
（2）令 $R=\max\limits_{1\leqslant i\leqslant N}\|\hat{x}_i\|$，则感知机算法 2.1 在训练数据集上的误分类次数 k 满足不等式

$$k\leqslant\left(\frac{R}{\gamma}\right)^2 \tag{2.9}$$

5.3网络结构

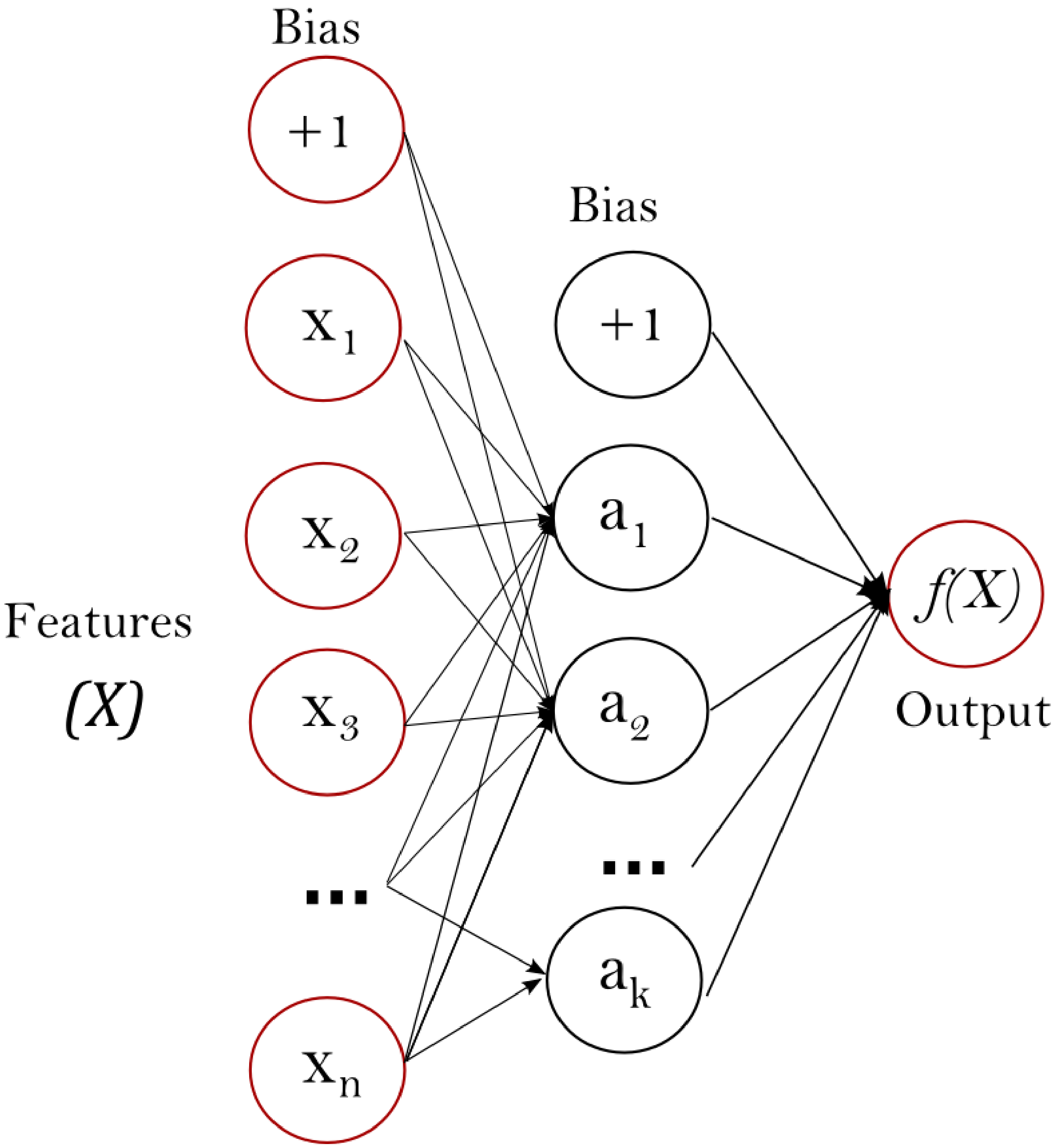
> end to end learning / black box model

- **前馈神经网络（多层感知器）**
- 记忆网络(反馈网络)
- 图网络



其中x_1,x_2,x_3 是输入单元（**input units**），我们将原始数据输入给它们。a_1,a_2,a_3是中间单元，它们负责将数据进行处理，然后呈递到下一层。 最后是输出单元，它负责计算$h_\theta(x)$。

神经网络模型是许多逻辑单元按照不同层级组织起来的网络，每一层的输出变量都是下一层的输入变量。下图为一个3层的神经网络，第一层成为输入层（**Input Layer**），最后一层称为输出层（**Output Layer**），中间一层成为隐藏层（**Hidden Layers**）。我们为每一层都增加一个偏差单位（**bias unit**）：



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

我们把这样从左到右的算法称为前向传播算法(**FORWARD PROPAGATION**)

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

其中W, b表示为网络中所以层的连接权重和偏置.(b = bias)

神经网络的学习过程就是根据features来调整神经元之间的weight以及每个功能神经元的阈值

Now by the definition of matrix vector multiplication, we can write $z = [z_1, \ldots, z_m]^\top \in \mathbb{R}^m$ as

$$\underbrace{\begin{bmatrix} z_1 \\ \vdots \\ \vdots \\ z_m \end{bmatrix}}_{z \in \mathbb{R}^{m\times 1}} = \underbrace{\begin{bmatrix} - & {w_1^{[1]}}^\top & - \\ - & {w_2^{[1]}}^\top & - \\ & \vdots & \\ - & {w_m^{[1]}}^\top & - \end{bmatrix}}_{W^{[1]} \in \mathbb{R}^{m\times d}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}}_{x \in \mathbb{R}^{d\times 1}} + \underbrace{\begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ \vdots \\ b_m^{[1]} \end{bmatrix}}_{b^{[1]} \in \mathbb{R}^{m\times 1}} \tag{2.8}$$

Or succinctly,

$$z = W^{[1]}x + b^{[1]} \tag{2.9}$$

	多层感知器
优点	学习非线性模型的能力。 (在线学习模型的能力。
缺点	具有隐藏层的 MLP 具有非凸的损失函数，它有不止一个的局部最小值。 因此不同的随机初始化权重会导致不同的验证集准确率。 MLP 需要调试一些超参数，例如隐藏层神经元的数量、层数和迭代轮数。 MLP对**特征缩放**是敏感的。

cost function	$J^{(i)}(\theta) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$	**对应第i个数据的lost**
	$J(\theta) = \frac{1}{n}\sum_{i=1}^n J^{(i)}(\theta)$	对应整个数据集的cost
优化(SGD)	$\theta := \theta - \alpha\nabla_\theta J(\theta)$	
	$\theta := \theta - \frac{\alpha}{B}\sum_{k=1}^B \nabla_\theta J(\theta)$	batch SGD B:batch size

5.4 Backward propagation 反向传播 BP

> 链式求导法则,梯度下降

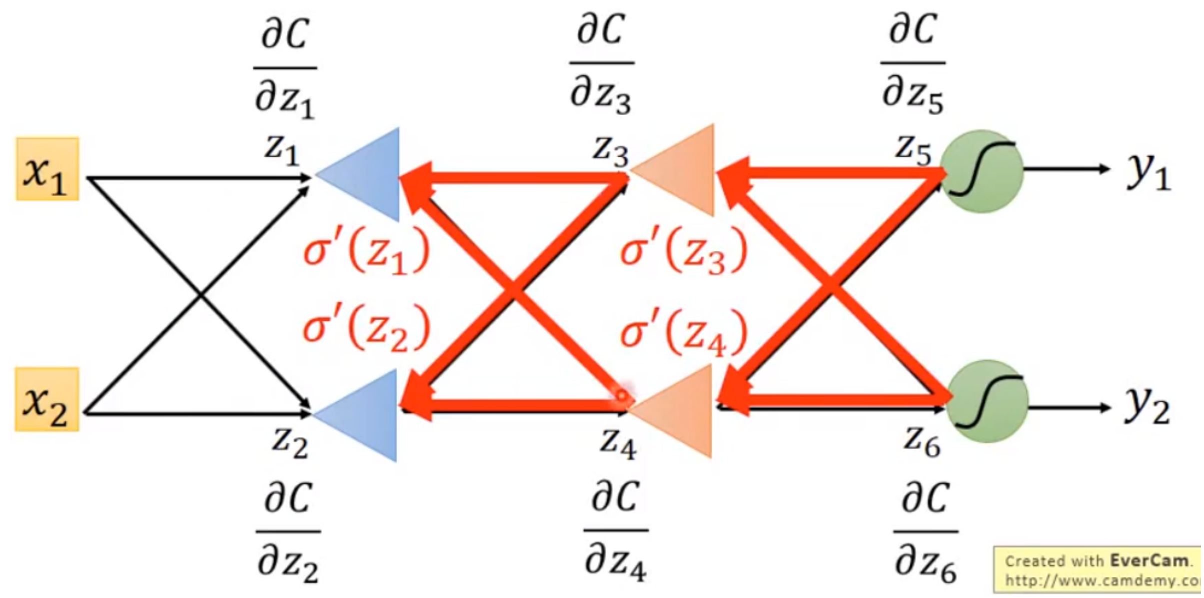
链式求导法则:

$$y = g(x), z = h(y)$$

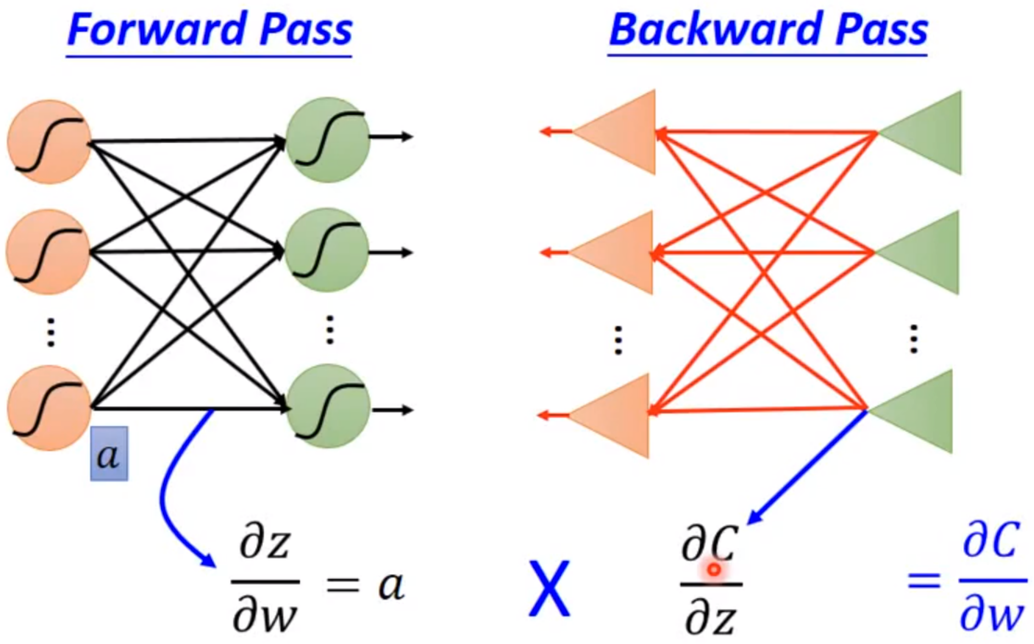
$$\Delta x \rightarrow \Delta y \rightarrow \Delta z$$

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

$$x = g(s), y = h(s), z = k(x, y)$$
$$\frac{dz}{ds} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial s} + \frac{\partial z}{\partial x}\frac{\partial x}{\partial s}$$



Backpropagation – Summary



Algorithm 3 Back-propagation for two-layer neural networks

1: Compute the values of $z \in \mathbb{R}^m$, $a \in \mathbb{R}^m$, and $o \in \mathbb{R}$

2: Compute

$$\delta^{[2]} \triangleq \frac{\partial J}{\partial o} = (o - y) \in \mathbb{R}$$
$$\delta^{[1]} \triangleq \frac{\partial J}{\partial z} = (W^{[2]^\top}(o - y)) \odot 1\{z \geq 0\} \in \mathbb{R}^{m \times 1}$$

(by eqn. (3.12) and (3.13))

3: Compute

$$\frac{\partial J}{\partial W^{[2]}} = \delta^{[2]} a^\top \in \mathbb{R}^{1 \times m} \qquad \text{(by eqn. (3.5))}$$
$$\frac{\partial J}{\partial b^{[2]}} = \delta^{[2]} \in \mathbb{R} \qquad \text{(by eqn. (3.6))}$$
$$\frac{\partial J}{\partial W^{[1]}} = \delta^{[1]} x^\top \in \mathbb{R}^{m \times d} \qquad \text{(by eqn. (3.7))}$$
$$\frac{\partial J}{\partial b^{[1]}} = \delta^{[1]} \in \mathbb{R}^m \qquad \text{(as an exercise)}$$

Algorithm 4 Back-propagation for multi-layer neural networks.

Compute and store the values of $a^{[k]}$'s and $z^{[k]}$'s for $k = 1, \ldots, r$, and J. ▷ This is often called the "forward pass"
.

.

for $k = r$ to 1 **do** ▷ This is often called the "backward pass"

  if $k = r$ **then**

    compute $\delta^{[r]} \triangleq \frac{\partial J}{\partial z^{[r]}}$

  else

    compute

$$\delta^{[k]} \triangleq \frac{\partial J}{\partial z^{[k]}} = \left(W^{[k+1]^{\top}} \delta^{[k+1]}\right) \odot \text{ReLU}'(z^{[k]})$$

  Compute

$$\frac{\partial J}{\partial W^{[k]}} = \delta^{[k]} a^{[k-1]^{\top}}$$
$$\frac{\partial J}{\partial b^{[k]}} = \delta^{[k]}$$

5.5问题

优化问题

泛化问题

神经网络可以调整的部分

- critical point
  - localminimum&saddlepoint
- batch\momentum动量
  - Small vs Lagre (size)
- learning rate
  - training stuck != small gradient
- **cost function**
- batch normalization
- 激活函数
- 初始化参数(随机,但不能全0)
- ...
- 梯度消失问题

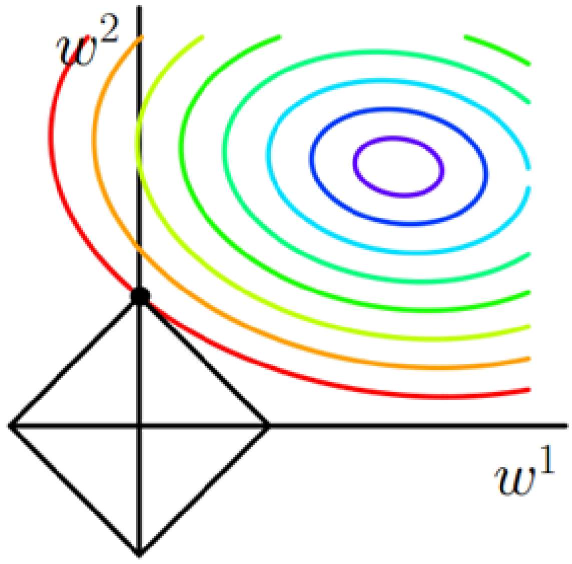
5.5.1 正则化项

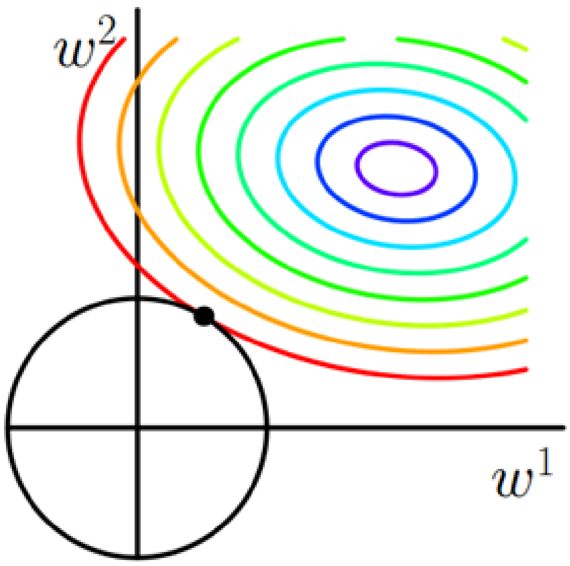
> Alpha是正则化项（也称为惩罚项）的参数，它通过限制权重的大小来防止过度拟合。 增加alpha可能会通过鼓励较小的权重来解决大方差（过度拟合的迹象），从而导致决策边界图以较小的曲率出现。 同样，降低alpha值可能会通过鼓励更大的权重来解决高偏差（欠拟合的迹象），从而可能导致决策边界更加复杂

神经网络中损失函数后一般会加一个额外的正则项L1或L2,也称为L1范数和L2范数。正则项可以看做是损失函数的惩罚项，用来对损失函数中的系数做一些限制。

- L1正则化是指权值向量w中各个元素的绝对值之和； $||W||_1 = \sum_{n=1}^{N} |w_n|$
- L2正则化是指权值向量w中各个元素的平方和然后再求平方根； $||w||_2 = \sqrt{\sum_{n=1}^{N} w_n^2} = \sqrt{w^T w}$

一般都会在正则化项之前添加一个**系数**，这个系数需要用户设定，系数越大，正则化作用越明显。



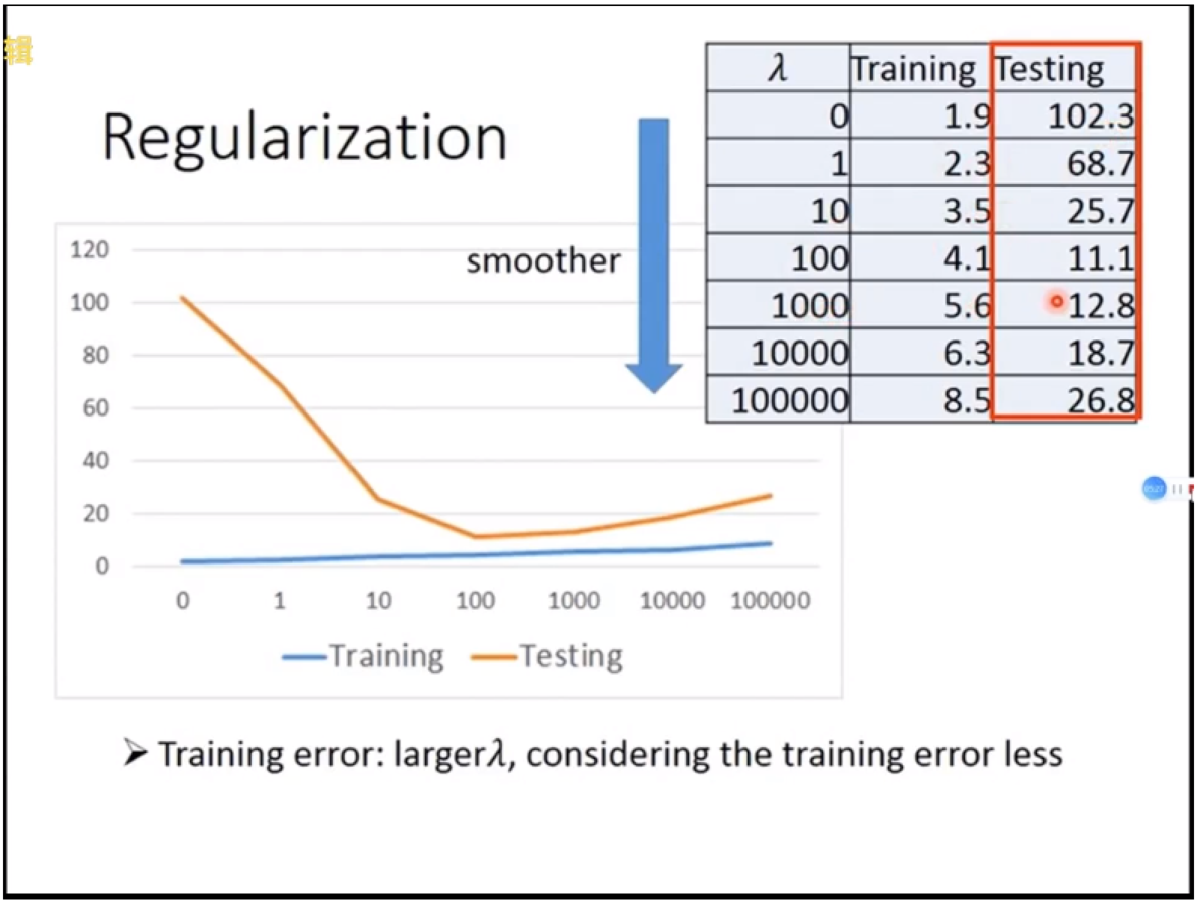


(假设只有两个参数 w_1, w_2)

$$Loss = Loss_0 + \alpha L1$$

or（$L2$）

L2正则化，又称权重衰减（weight decay）关注的是权重平方和的平方根，是要网络中的权重接近0但不等于0，而在L1正则中，要关注的是权重的绝对值，权重可能被压缩成0。在深度学习中，L1会趋向于产生少量的特征，而其他的特征都是0，而L2会选择更多的特征，这些特征都会接近于0。神经网络需要每一层的神经元尽可能的提取出有意义的特征，而这些特征不能是无源之水，因此L2正则用的多一些。



5.6 具体问题

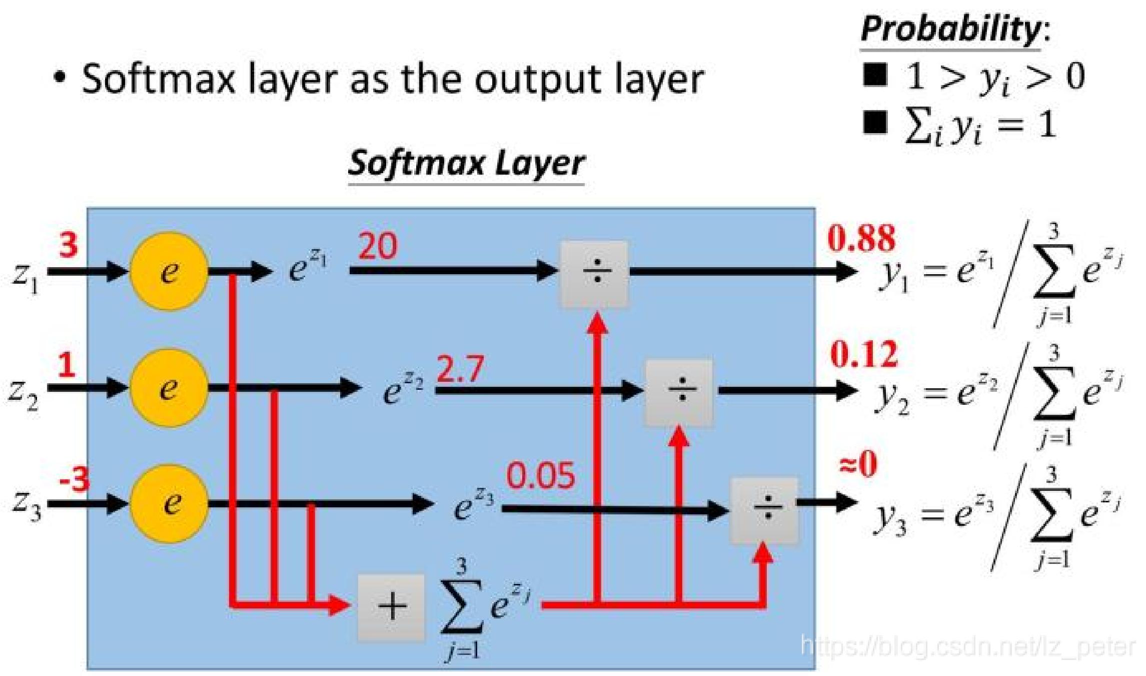
5.6.1 二分类问题(逻辑回归)

输出层为sigmoid函数,

5.6.2 多分类问题

利用**softmax函数**(归一化指数函数)作为输出函数,

> 1）预测的概率为非负数；2）各种预测结果概率之和等于1。



总结一下softmax如何将多分类输出转换为概率，可以分为两步：

1）分子：通过指数函数，将实数输出映射到零到正无穷。

2）分母：将所有结果相加，进行归一化。

Details of the softmax classifier

$$p(y|x) = \frac{\exp(W_y.x)}{\sum_{c=1}^{C} \exp(W_c.x)}$$

We can tease apart the prediction function into two steps:

1. Take the y'th row of W and multiply that row with x:

$$W_y.x = \sum_{i=1}^{d} W_{yi} x_i = f_y$$

Compute all f_c for c=1,…,C

2. Apply softmax function to get normalized probability:

$$p(y|x) = \frac{\exp(f_y)}{\sum_{c=1}^{C} \exp(f_c)} = softmax(f)_y$$

5.6.3 回归问题

输出层中没有激活函数，这也可以看作是使用identity function作为激活函数。因此，它使用平方误差作为损失函数，输出是一组连续值。

5.7其他神经网络

- RBF网络
- ART网络
- SOM网络
- 级连相关网络
- Elman网络
- 伯努利受限玻尔兹曼机
- ...

补充

①数据集的线性可分

给定一个数据集
$T=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\ldots,(\boldsymbol{x}_N,y_N)\}$
其中，$\boldsymbol{x}_i\in\mathbb{R}^n,y_i\in\{0,1\},i=1,2,\ldots,N$，如果存在某个超平面
$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b=0$
能将数据集T中的正样本和负样本完全正确地划分到超平面两侧，即对所有$y_i=1$的样本$\boldsymbol{x}_i$，有$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b\geq 0$，对所有$y_i=0$的样本$\boldsymbol{x}_i$，有$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b<0$，则称数据集$T$线性可分，否则称数据集$T$线性不可分。

②最大似然估计(Maximum Likelihood Estimation,MLE)

$P(Y=c_k)$和$P(X^{(i)}=x^{(i)}|Y=c_k)$ 可用最大似然估计法估计相应概率

$$P(Y=C_K)=\frac{\sum_{i=1}^{N}I(y_i=c_k)}{N},k=1,2,\cdots,K$$

设第j个特征$x^{(j)}$可能取值的集合为{$a_{j1},a_{j2},\cdots,a_{js_j}$},条件概率
$P(X^{(i)}=a_{ij}|Y=c_k)$的最大似然估计为:

$$P(X^{(i)}=a_{ij}|Y=c_k)=\frac{\sum_{i=1}^{N}I(y_i=c_k)I(x_i^{(j)}=a_{jl},y_i=c_k)}{\sum_{i=1}^{N}I(y_i=c_k)}$$

式中,$x_i^{(j)}$是第i个样本的第j个特征;a_{jl}是第j个特征可能取的第l个值;$I()$为指示函数.

③逻辑回归

目标:一个分类问题

	logistic	
step1初始化w,b	$f_{w,b} = \sigma(\sum_i w_i x_i + b)$ output between 0 and 1	$\sigma(\cdot) = \frac{1}{1-exp(\cdot)}$
step2优化w,b (loss function 最大似然估计) / 梯度下降法更新	训练集:$(x^n, \hat{y}^n)$ $\hat{y}^n$:1 for class 1 ; 0 for class 2 $L(f) = \sum_n C(f(x^n), \hat{y}^n)$ $\#f = f(w,b)$	
	cross entropy $C(f(x^n), \hat{y}^n) = -\frac{1}{N}\sum_{i=1}^{m} p(y_i \| x_i, \theta) = -[\hat{y}^n lnf(x^n) + (1-\hat{y}^n) ln(1-f(x^n))]$	x_i,\theta)为什么不用均方误差? N为类别数
step3 find the best function 利用新的 $\hat{y} = \sigma(wx+b)$ 去预测	偏微分: $\frac{-lnL(w,b)}{\partial w_i}$ = ...=$\sum_n -(\hat{y}^n - f_{w,b}(x^n))x_i^n$ update: $w_i := w_i - \eta \sum_n -(\hat{y}^n - f_{w,b}(x^n))x_i^n$	

逻辑回归与感知机的异同:

> 两类都是线性分类器;
>
> **损失函数**两者不同：逻辑斯蒂回归使用极大似然（对数损失函数），感知机使用的是均方损失函数（即错误点到分离平面的距离，最小化这个值）
>
> 逻辑斯蒂比感知机的优点在于对于**激活函数**的改进。
>
> 前者为sigmoid function，后者为阶跃函数。这就导致LR是连续可导，而阶跃函数则没有这个性质。
>
> LR使得最终结果有了概率解释的能力（将结果限制在0-1之间），sigmoid为平滑函数，能够得到更好的分类结果，而step function为分段函数，对于分类的结果处理比较粗糙，非0即1，而不是返回一个分类的概率。

逻辑回归的损失函数推导:

对于二分类问题:

$$p_1 = p(y=1|x,\theta) = \frac{e^{x_\theta}}{1+e^{x_\theta}}, y=1$$

$$p_0 = p(y=0|x,\theta) = \frac{1}{1+e^{x_\theta}}, y=0$$

统一写法：

$$p = p(y|x,\theta) = p_1^{y_i} * p_0^{1-y_i}$$

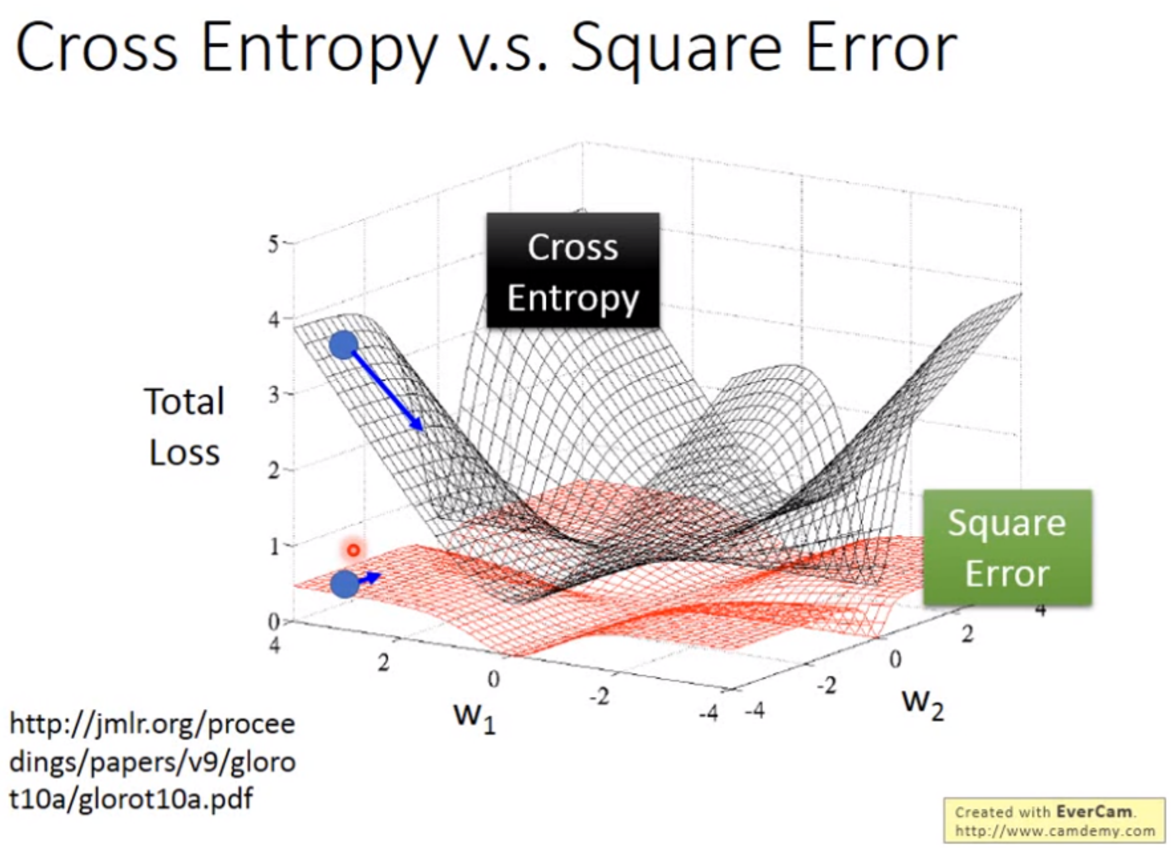
令$p_1 = f(x^n), p_0 = 1 - f(x^n)$，带入目标函数再取对数$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{m} p(y_i|x_i,\theta)$

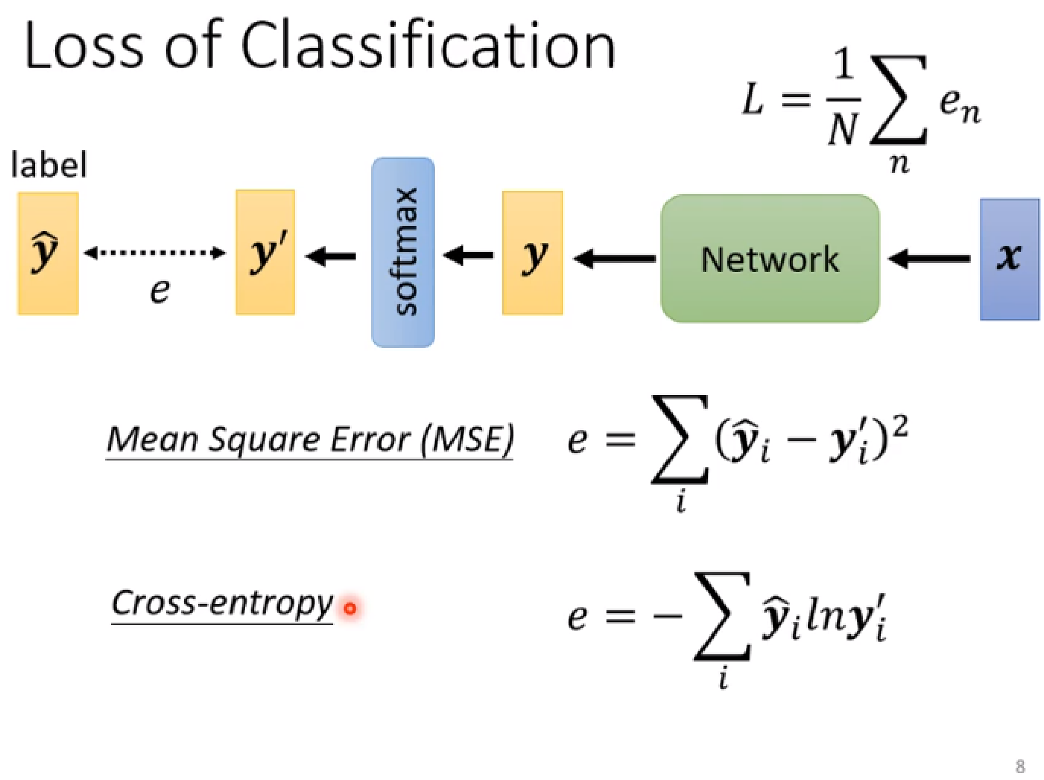
$$C(f(x^n), \hat{y}^n) = -[\hat{y}^n lnf(x^n) + (1-\hat{y}^n)ln(1-f(x^n))]$$

感知器

逻辑回归模型

④交叉熵与均方误差





[直观理解为什么分类问题用交叉熵损失而不用均方误差损失?](#)

[为什么不用平方误差（MSE）作为Logistic回归的损失函数?](#)

⑤范数

范数是具有“长度”概念的函数。在[线性代数](#)、[泛函分析](#)及相关的数学领域，范数是一个函数，是[矢量空间](#)内的所有矢量赋予非零的正**长度**或**大小**。半范数可以为非零的矢量赋予零长度。

⑥[神经网络损失函数中的正则化项L1和L2](#)

⑦特征缩放

> 将输入向量 X 的每个属性放缩到到 [0, 1] 或 [-1，+1]，或者将其标准化使它具有 0 均值和方差 1;注意，为了得到有意义的结果，您必须对测试集也应用*相同的*尺度缩放。 您可以使用 `StandardScaler` 进行标准化。