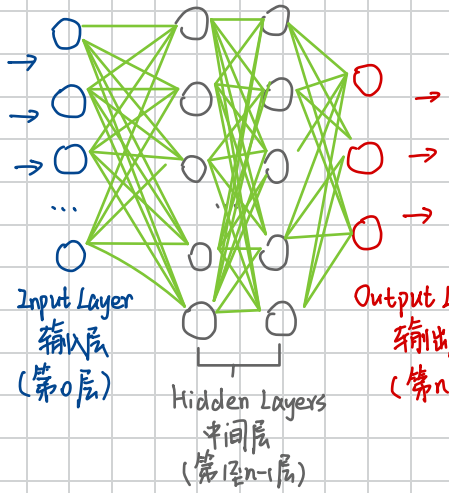
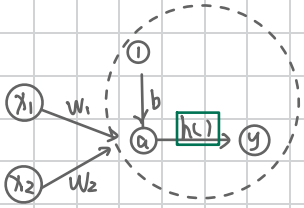
Neural Network：（n层）



激活函数：activation function



$$\begin{cases} a = w_1x_1 + w_2x_2 + b \\ y = h(a) \end{cases}$$

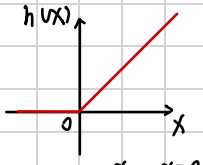
常用激活函数：

$$h(x) = \frac{1}{1+\exp(-x)}$$

Sigmoid 函数

$$h(x) = \begin{cases} 0 & x \le 0 \\ 1 & x > 0 \end{cases}$$

阶跃函数



$$h(x) = \begin{cases} x & x > 0 \\ 0 & x \le 0 \end{cases}$$

ReLU 函数

常用输出函数：恒等函数

softmax函数：$y_k = \frac{\exp(a_k + c')}{\sum_{i=1}^{n} \exp(a_n + c')}$

（加上常数c'仅便于计算，但没）

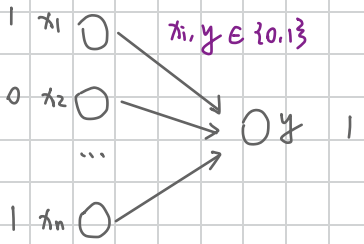
梯度法：$\begin{cases} x_0 = x_0 - \eta \frac{\partial f}{\partial x_0} \\ x_1 = x_1 - \eta \frac{\partial f}{\partial x_1} \end{cases}$

学习率，一般取0.01或0.001

神经网络梯度：$W = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix}$ $\frac{\partial L}{\partial W} = \begin{pmatrix} \frac{\partial L}{\partial w_{11}} & \frac{\partial L}{\partial w_{12}} & \frac{\partial L}{\partial w_{13}} \\ \frac{\partial L}{\partial w_{21}} & \frac{\partial L}{\partial w_{22}} & \frac{\partial L}{\partial w_{23}} \end{pmatrix}$ 损失函数

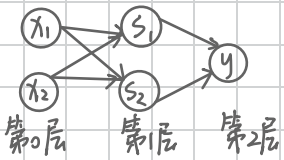
一般都为线性函数

感知机（perceptron）⇒ 可表现逻辑电路



$x_i, y \in \{0, 1\}$

多层感知机



常用损失函数

均方误差（mean squared error）

$$E = \frac{1}{2} \sum_k (y_k - t_k)^2$$

y_k：NN输出　t_k：监督数据

交叉熵误差（cross entropy error）

$$E = -\sum_k t_k \log y_k$$

mini-batch 学习：从大量（60000个）数据中随机选一些（100笔）

平均损失函数：$E = -\frac{1}{N} \sum_n \sum_k t_{nk} \log y_{nk}$　（以 cross entropy error 为例）

N：数据总数

machine learning 过程：给定一组参数 → 算其损失函数（及其导数）

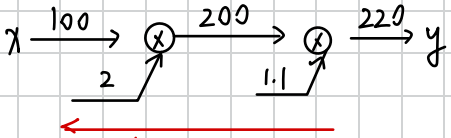
↑ 梯度法使得参数更新 ↓

通常为随机梯度下降法（Stochastic Gradient Descent）（SGD）

误差反向传播法

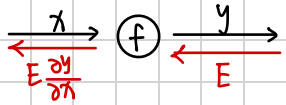
正向传播 (forward propagation) →

计算图：



← 反向传播 (backward propagation)

反向传播的计算：



目的：根据链式法则求 $\frac{\partial(\text{输出})}{\partial(\text{输入})}$

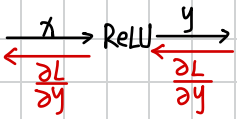
加法节点：仅将输入信号输出到下一节点

乘法节点：需要保存正向传播的输入信号

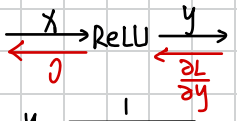
激活函数的 backward propagation：

ReLU 函数：$y=\begin{cases} x & (x>0) \\ 0 & (x\le 0) \end{cases}$

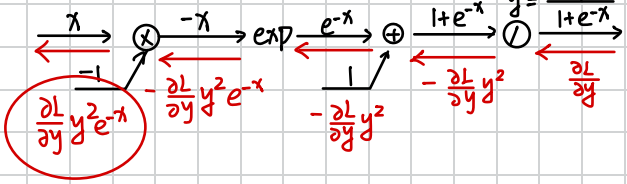
$x>0$ 时：



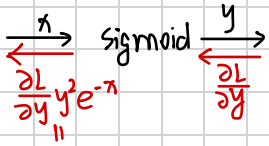
$x\le 0$ 时：



Sigmoid 函数：$y=\frac{1}{1+\exp(-x)}$

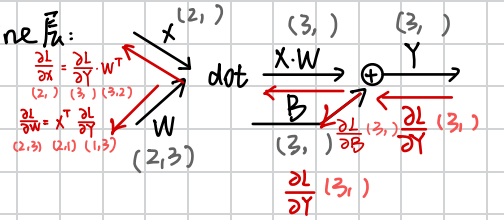


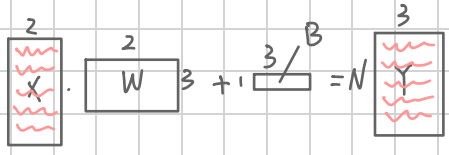
可简化为 Sigmoid 节点：



反向传播的值仅需正向传播的结果 $\frac{\partial L}{\partial y}y(1-y)$

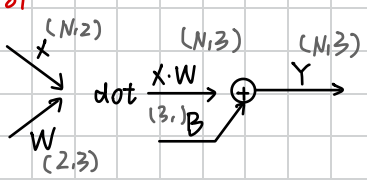
Affine 层：





批版本 Affine 层：

(N个数据一起正向传播)

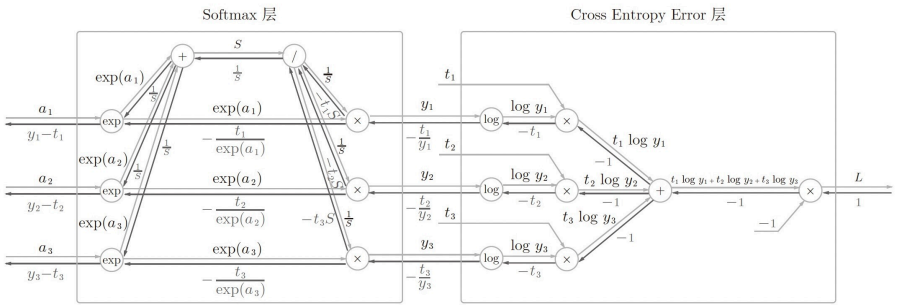


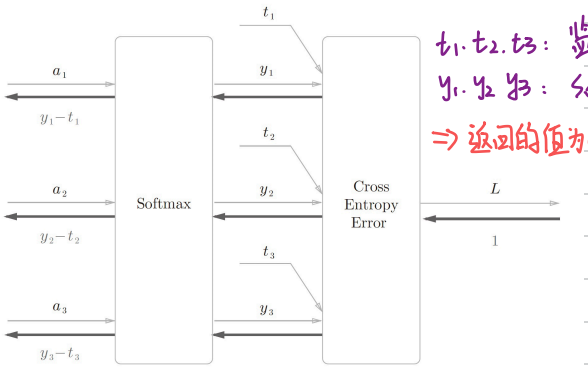
$\frac{\partial L}{\partial X}=\frac{\partial L}{\partial Y}\cdot W^T$

$\frac{\partial L}{\partial W}=X^T\frac{\partial L}{\partial Y}$

$\frac{\partial L}{\partial B}=\frac{\partial L}{\partial Y}$ 的第一个轴 (第0轴方向上的和)

Softmax-with-Loss 层：



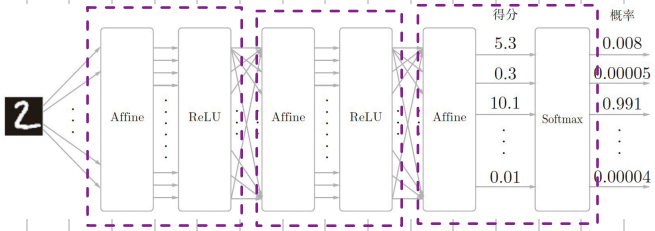


t_1, t_2, t_3：监督数据 (0,1,0)

y_1, y_2, y_3：Softmax 层输出 (0.3, 0.2, 0.5)

⇒ 返回的值为 (0.3, −0.8, 0.5)

分类问题：(e.g. 3层神经网络)



中间层1　中间层2　输出层　　合称 Softmax-with-Loss 层

学习阶段：Affine - Softmax - Cross Entropy Error

数据正规化　交叉熵误差 & 求和

推理阶段：仅 Affine 层求最大值选项