简单神经网络
Vanilla Neural Network

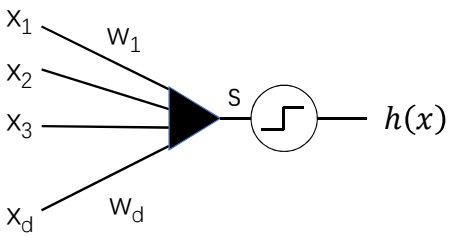
2019.09

林立晖

二分类模型回顾

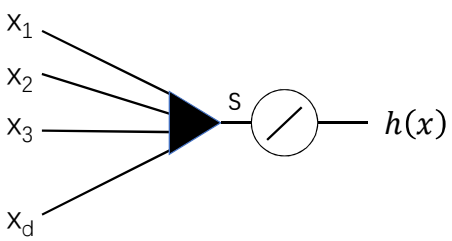
$$s = \sum_{i=1}^{d} w_i x_i + w_0 x_0 = \sum_{i=0}^{d} w_i x_i = \boldsymbol{W}\boldsymbol{x}$$

$h(x) = \text{sign}(s)$



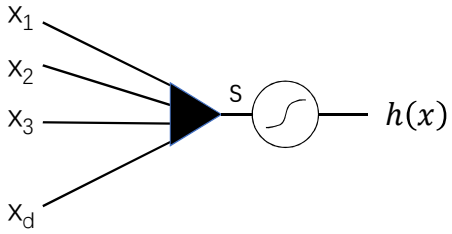
PLA: 0/1误差

$h(x) = s$



线性回归: 均方误差

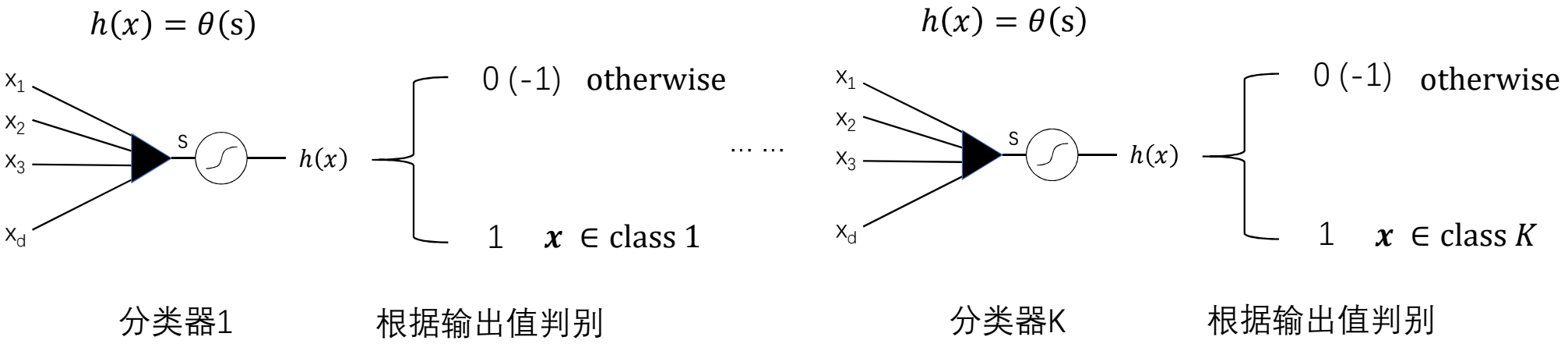
$h(x) = \theta(s)$



逻辑回归: 交叉熵

待解决问题：如果样本有多个类别（例如第一次实验的SemEval），那么应该如何应用上述三种模型?

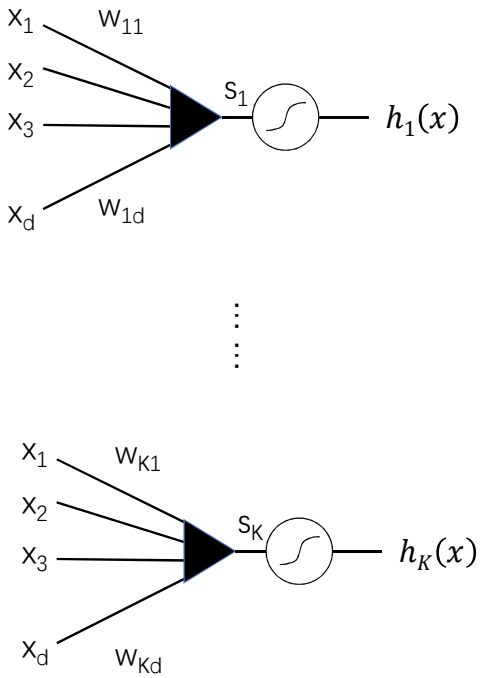
简单方法：对于K分类问题（K > 2），训练K个分类器，最后取输出值最大的分类器代表的类别



分类器1　　根据输出值判别　　分类器K　　根据输出值判别

思考：如果某些分类器输出的（判别为属于该类别）的概率值相等？

一般方法：对于K分类问题（K > 2），使用softmax回归。



$$s_j = \sum_{i=1}^{d} w_{ji}x_i + w_{j0}x_0 = \sum_{i=0}^{d} w_{ji}x_{ji} = \boldsymbol{W}_j\boldsymbol{x}$$

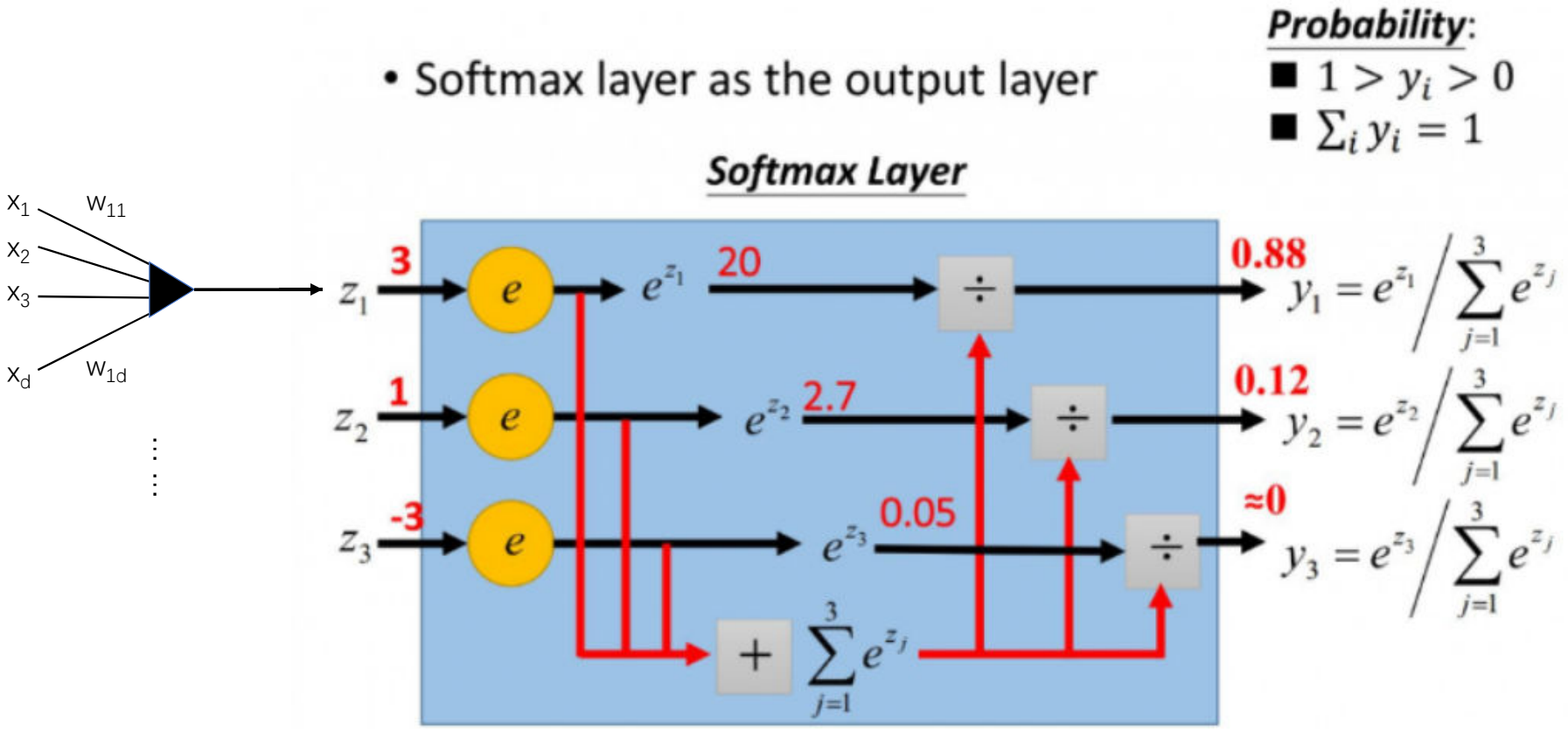
$$t_j(\boldsymbol{x}) = \exp(s_j) \quad , \quad j \in [1, K]$$

$$\mathrm{p}_j(y = j) = \frac{t_j(x)}{\sum_j t_j(x)}$$

简而言之：

$$p(y = j \mid x, \boldsymbol{W}_j) = h_j(\boldsymbol{x}) = softmax(\boldsymbol{W}_j\boldsymbol{x}) = \frac{\exp(\boldsymbol{W}_j\boldsymbol{x})}{\sum_{i=1}^{K}\exp(\boldsymbol{W}_j\boldsymbol{x})}$$

softmax回归示意图



模型函数对比

Given $\boldsymbol{x} = [x_1, x_2, \dots, x_i, \dots x_d], i \in [1, \mathrm{d}], j \in [1, K]$

逻辑回归: $h(\boldsymbol{x}) = \dfrac{1}{1 + \exp(-\sum_{i=0}^{d} w_{ji} x_i)} = \dfrac{1}{1 + \exp(-\boldsymbol{W}_j \boldsymbol{x})}$

接收一个标量，输出一个标量概率值

$p(\boldsymbol{x} \in class\ j \mid \boldsymbol{x}, \boldsymbol{W_j}) = h(\boldsymbol{x})$

softmax回归: $h(\boldsymbol{x}) = \dfrac{1}{\sum_{\mathrm{j}=1}^{K} \exp(\sum_{i=0}^{d} w_{ji} x_i)} \left[\exp\left(\sum_{i=0}^{d} w_{1i} x_i \right) \quad \dots \quad \exp\left(\sum_{i=0}^{d} w_{Ki} x_i \right) \right]$

$= \dfrac{1}{\sum_{\mathrm{j}=1}^{K} \exp(\sum_{i=0}^{d} w_{ji} x_i)} [\exp(\boldsymbol{W}_1 \boldsymbol{x}) \quad \dots \quad \exp(\boldsymbol{W}_K \boldsymbol{x})]^{\mathrm{T}}$

接收一个向量，输出一个归一化的概率分布向量:

$[\, p(\boldsymbol{x} \in class\ 1 \mid \boldsymbol{x}, \boldsymbol{W}_1, \dots \boldsymbol{W}_K), \dots, p(\boldsymbol{x} \in class\ K \mid \boldsymbol{x}, \boldsymbol{W}_1, \dots \boldsymbol{W}_K) \,] = h(\boldsymbol{x}) = softmax([\boldsymbol{W}_1 \boldsymbol{x}, \dots, \boldsymbol{W}_K \boldsymbol{x}])$

似然函数对比

假设数据集中有N个样本 $\{\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}\}$，对应的标签集合为 $\{\mathrm{y_1}, \ldots, \mathrm{y_n}\}$

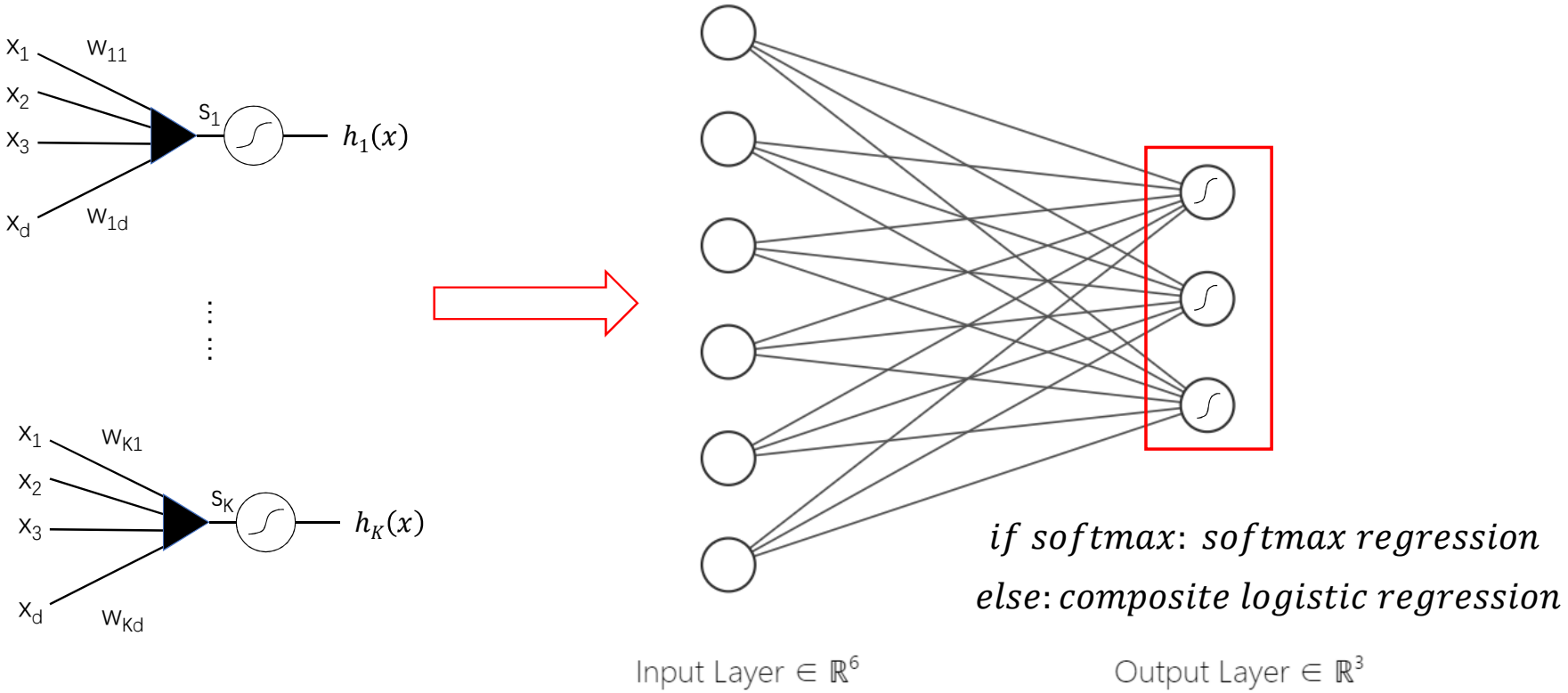
逻辑回归：$$likelihood = \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n) = \prod_{n=1}^{N} h(\boldsymbol{x}_n)^{y_n}\big(1 - h(\boldsymbol{x}_n)\big)^{1-y_n}$$

softmax回归：$$likelihood = \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n) = \prod_{n=1}^{N} h(\boldsymbol{x}_n)_{y_n}$$

更新过程相似，计算负对数似然，并对权重求导数后使用梯度下降法更新。

从逻辑/softmax回归到神经网络

假设给定的样本维度d是5，类别数为3。那么处理K分类任务的逻辑回归/softmax回归模型结构如下所示：



思考：这样的结构学习到的结果是什么？

现在我们考虑线性不可分集，如图 1.9 所示的线性不可分集

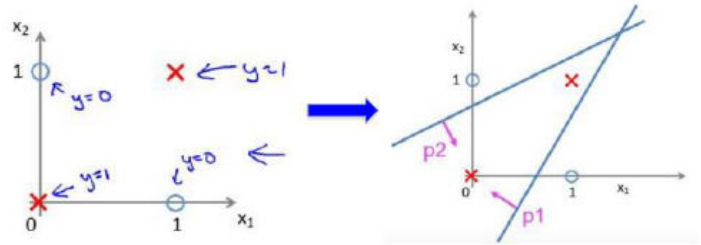


图 1.9 线性不可分集

假设我们定义落在直线一以下与直线二以上公共的区域的点记为一个类其余都当做另一个类，假设直线一与直线二有解析式

$$y_1 = k_1x_1 + k_2x_2 + b_1, y_2 = k_3x_1 + k_4x_2 + b_2$$

那么我们可以通过如下图 1.10 所示有一隐藏层的神经网络实现上述功能。

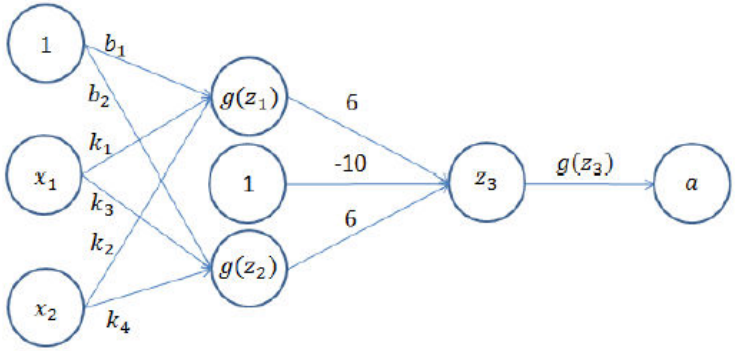
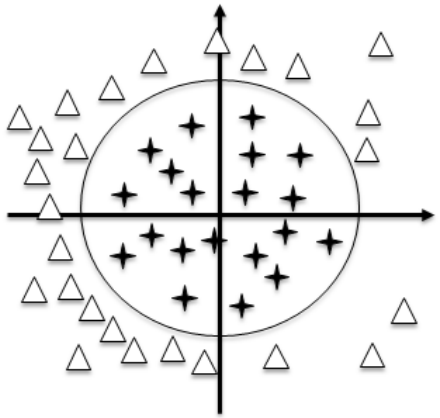


图 1.10 含有一层隐藏层的神经网络



上节课的思考：PLA应对非线性可分集的方法？

通过不断地增加隐藏层节点数目和神经网络的层数（深度），可以提高拟合能力（根据万能近似定理，即Universal Approximation Theorem，神经网络可以拟合任意形状的决策边界），但是抽象程度也随之增加，难以解释每个神经元输出的数值究竟有什么含义：

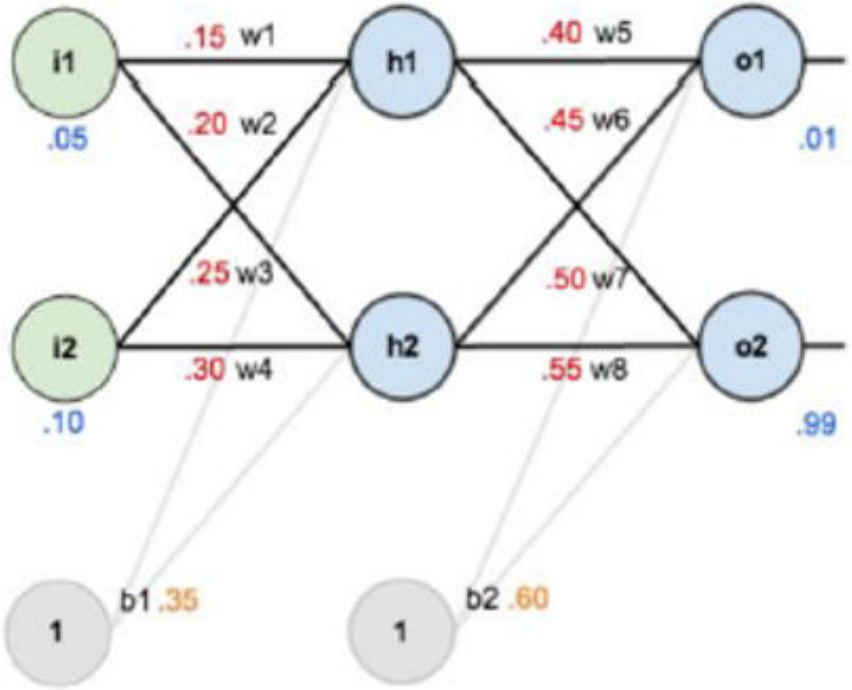
图 1.12 神经网络隐藏层的数量与其对应的能力

关于神经网络层数和节点数的作用：https://www.jianshu.com/p/91138ced2939

万能近似定理：https://blog.csdn.net/ningyanggege/article/details/84174312

神经网络可以通过基于链式求导的误差反向传播（Back Propagation）进行训练。此时我们的目标是最小化损失函数，我们也可以把损失函数的值称之为输出和真实值的误差，并通过它来计算一系列梯度。

下面以一个单隐层的神经网络为例，该网络接收的数据样本维度为2，隐层单元数目为2，输出单元为2（在分类任务中和类别数一致，回归任务可以直接设置为需要预测值的维度）。网络输入为**i**，隐层输出为**h**，最终输出为**o**，标签为0.01和0.99（类别的one hot编码或者回归的真实值），损失函数使用平方和误差（一般用于回归模型）。



向前（从输入层到输出层）计算输出和误差，使用sigmoid函数激活：

第一个隐层单元输入：

$$net_{h1} = w_1 i_1 + w_2 i_2 + b_1 * 1 = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

第一个隐层单元输出：

$$out_{h1} = \frac{1}{1 + e^{-net_{h1}}} = \frac{1}{1 + e^{-0.3775}} = 0.5933$$

$$\text{out}_{h2} = 0.5969$$

第一个输出单元输入：

$$net_{o1} = w_5 out_{h1} + w_6 out_{h2} + b_2 * 1 = 0.4 * 0.5933 + 0.45 * 0.5969 + 0.6 * 1 = 1.1060$$

第一个输出单元输出：

$$out_{o1} = \frac{1}{1 + e^{-net_{o1}}} = \frac{1}{1 + e^{-1.1059}} = 0.7514$$

$$\text{out}_{o2} = 0.7729$$

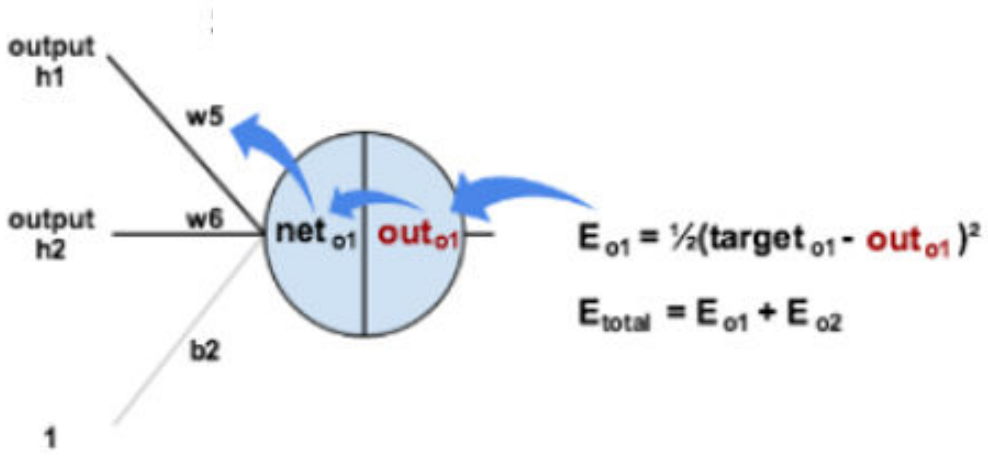
前向误差：

$$E_{\text{total}} = \sum \frac{1}{2}(\text{target - output})^2$$

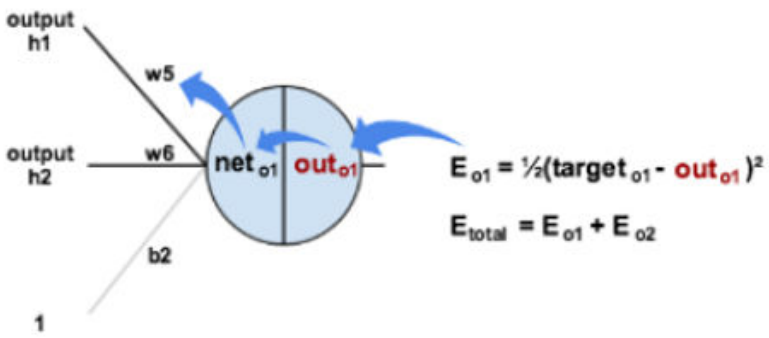
$$E_{\text{total}} = E_{o1} + E_{o2} = 0.2748 + 0.0236 = 0.2984$$

*前向误差的反向传播

以更新输出层权重w_5为例，和逻辑回归相同，我们需要求损失函数对w_5的偏导。但是w_5和误差函数之间没有直接的连接，它是通过参与输入加权，非线性激活这两层计算后对最终的误差产生贡献的。因此，我们在求导的时候需要使用链式法则。



$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{totol}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{totol}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$

在右边的计算过程中，误差从后往前通过out_{o1}和net_{o1}“反向传播”到了$\boldsymbol{w}_5$，使得我们可以通过梯度下降进行更新

思考：如果网络太深，会造成计算过程中出现什么情况？

通过链式法则求损失函数对$\boldsymbol{w}_5$的偏导

$$E_{\text{total}} = \frac{1}{2}(\text{target}_{o1} - \text{output}_{o1})^2 + \frac{1}{2}(\text{target}_{o2} - \text{output}_{o2})^2$$

$$\frac{\partial E_{\text{total}}}{\partial out_{o1}} = 2 * \frac{1}{2}(\text{target}_{o1} - \text{output}_{o1})^{2-1} * (-1) + 0 = \text{output}_{o1} - \text{target}_{o1}$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 0.7514 - 0.01 = 0.7414$$

$$out_{o1} = \frac{1}{1 + e^{-net_{o1}}}$$ Sigmoid导数性质

$$\frac{\partial out_{o1}}{\partial net_{o1}} = \boxed{out_{o1}(1 - out_{o1})} = 0.7514(1 - 0.7514) = 0.1868$$

$$net_{o1} = w_5 out_{h1} + w_6 out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial w_5} = 1 * out_{h1} * w_5^{1-1} + 0 + 0 = out_{h1} = 0.5933$$

$$\frac{\partial E_{\text{total}}}{\partial w_5} = (\text{output}_{o1} - \text{target}_{\text{ol}}) * \text{out}_{o1}(1 - \text{out}_{o1}) * \text{out}_{h1}$$

$$= 0.7414 * 0.1868 * 0.5933 = 0.0822$$

$$\text{new } w_5 = w_5 - \eta * \frac{\partial E_{\text{total}}}{\partial w_5} = 0.4 - 0.5 * 0.0822 = 0.3589$$

神经网络分类问题的损失函数——交叉熵（Cross Entropy）

交叉熵定义（从KL散度推导）：

$$H(p,q) = -\sum_{i=1}^{K} p(y_i)\log(q(y_i))$$

k改成大N，代表样本数

思考：为什么不像逻辑回归一样最小化负对数似然函数？

K分类问题的对数似然可以写成如下形式：

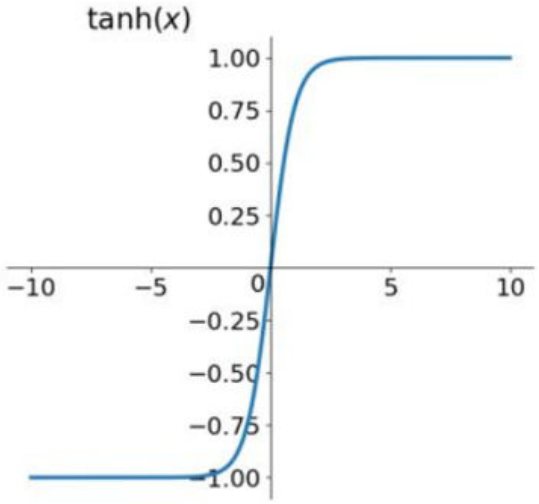
$$log(likelihood) = \log\prod_{i=1}^{N}\prod_{j=1}^{K} p_\theta(y_i = j|x_i)^{I(y_i=j)}$$

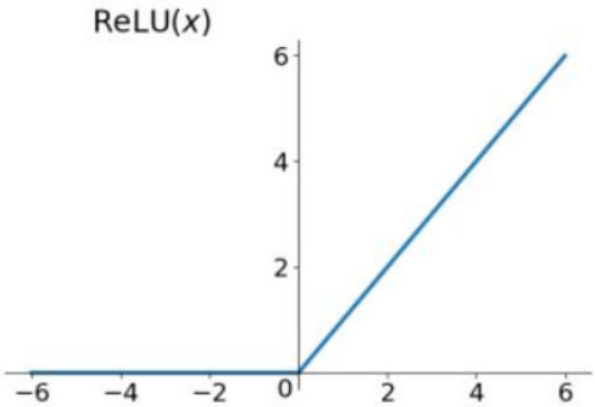
$$= \sum_{i=1}^{N}\sum_{j=1}^{K} I(y_i = j)\log p_\theta(y_i = j|x_i)$$

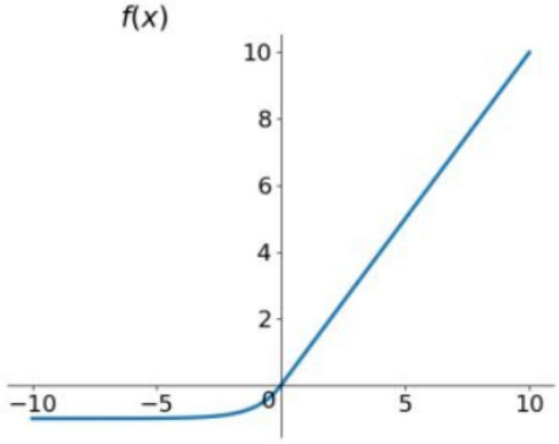
在分类问题中，标签一般是one hot向量，因此可以使用Identity函数来表示。这里的p_θ就是给定样本x_i和神经网络参数θ时，y的似然。

q是神经网络预测的类别分布（输入样本通过神经网络参数组合计算后输出的预测值），p是类别真实分布（one hot），只有一个类别的概率值为1，其他都是0。实际上两者形式是相同的。最小化交叉熵等价于最小化负对数似然。

其他常见神经网络激活函数



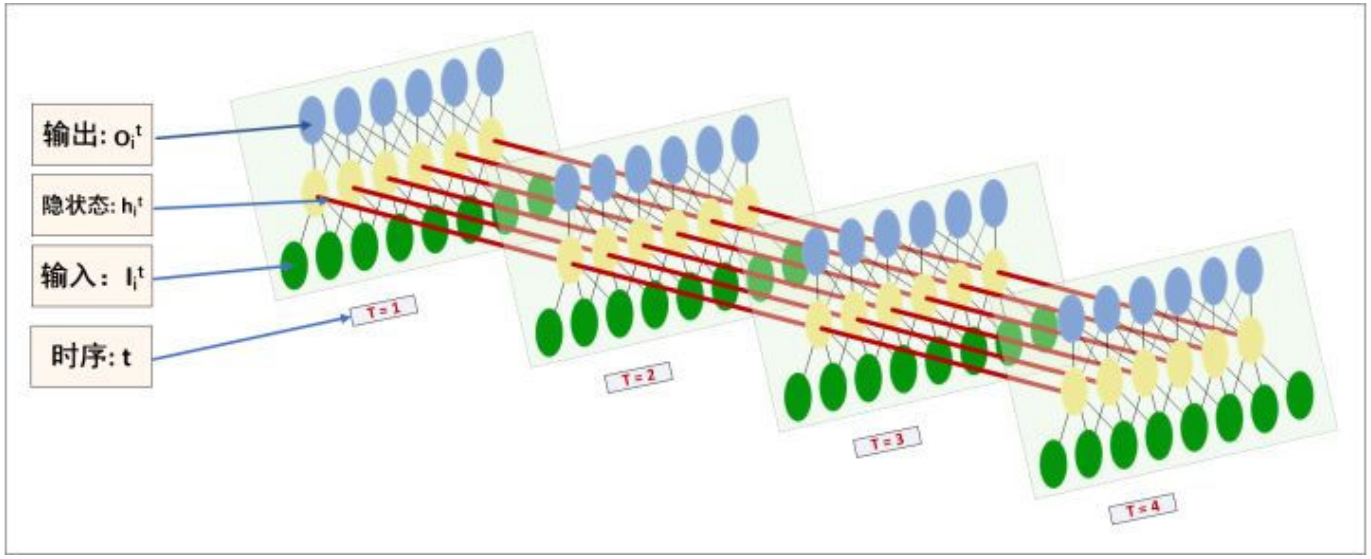




- 神经网络训练步骤：
  - 1. 给每一个样本的特征向量前加一维常数项1（初始偏置项）
  - 2. 随机初始化$(\mathrm{d}+1)$维的权重向量W_0（补充偏置项权重）
  - 3. 对每个样本前向计算误差（注意隐藏层可以自己决定是否加入偏置项）
  - 4. 根据误差反向传播梯度更新所有权重$w_{\mathrm{t+1}} \leftarrow w_t - \eta \nabla L(w_t)$
  - 5. 对所有训练样本重复步骤3步骤4（对所有样本都完成一次步骤3和步骤4称为一个epoch，即迭代多个epoch），直到满足一定的收敛条件（损失函数值不再下降）
  - 6. 根据模型最后的权重，预测样例

- 循环神经网络（Recurrent Neural Network）：

用于处理时间序列，即每个输入样本不再是一个向量，而是一系列向量（例如文本可以表示成一系列由它所包含单词的one hot向量组成的矩阵）：



RNN实际上只有一个网络，只不过把根据上一个输入向量计算出来的隐藏层输出保存下来，和当前输入的向量一起计算当前的隐藏层输出。

$$h_t = f(h_{t-1}, x_t)$$

思考题

- K个二元逻辑回归模型和softmax回归模型学习得到的是什么？
- 如果神经网络层数太多（网络太深），有可能导致什么现象？可以从数值计算的角度来回答。
- 如果神经网络中某个使用了非线性激活函数的隐层单元接收的输入值特别大，会造成什么现象？

- 基于python深度学习框架tensorflow或者pytorch实现简单神经网络。
- 本次数据为MNIST（手写数字识别数据集），每个样本格式为28x28大小图片的像素值矩阵（每个位置元素取值范围为0-255），相当于可以展开成28x28=784维的向量。标签为0到9的数字，需要处理成one hot。
- 训练集包含60000条样本，测试集包含10000条样本。
- 原始数据为bin格式，处理非常繁琐，可以直接通过框架导入（pytorch自带的导入比较繁琐，可以统一使用Keras框架来加载数据集）

```
from tensorflow.examples.tutorials.mnist import input_data
```

```
from keras.datasets import mnist
```

- 不需要验收，不用写实验报告，自行完成。建议提前安装tensorflow或pytorch，并熟悉两种框架各自的神经网络搭建方法，期中project会有相关内容。