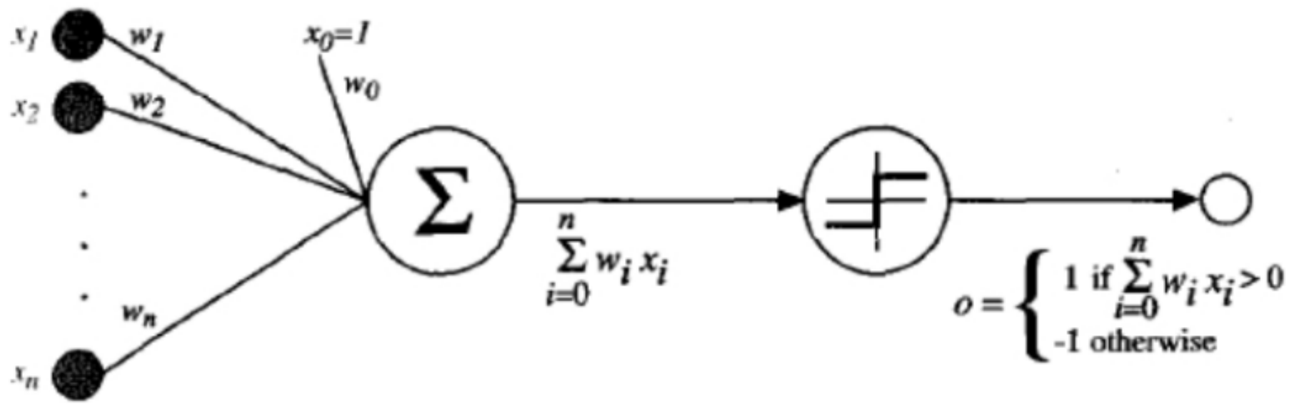
神经元



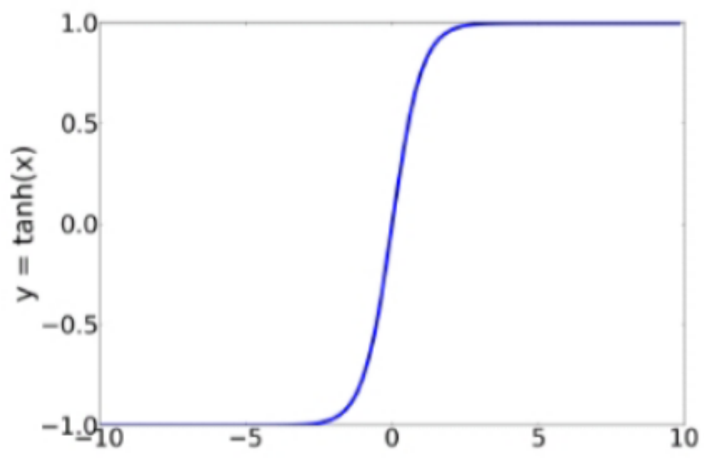
输入：$x_1, x_2, \cdots, x_n$

求和：$z = b + \sum_{i=1}^{n} w_i x_i$

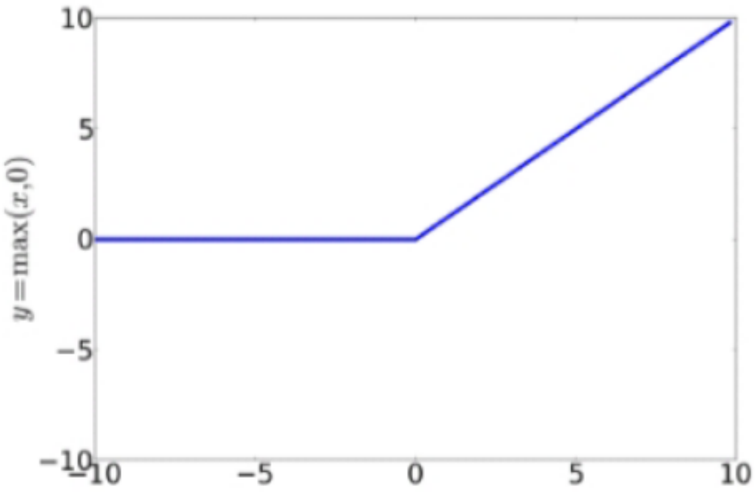
激活：$y = f(z) = \sigma(z) = \frac{1}{1+e^{-z}}$

激活函数不只sigmoid一种，还有

Tanh：$f(z) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$



ReLU：$f(z) = \max(0, z)$



实际运用中ReLU及其变种用得更多.

单个神经元只是一个二分类决策器，它本质上在做一件事：在特征空间中画一条直线(或超平面)，将数据分成两类，一类能激活它而另一类不能

激活函数是神经网络的灵魂，没有它，网络只是一堆线性变换的叠加，而利用多层激活函数，就可以逼近任何复杂函数，赋予网络非线性表达能力

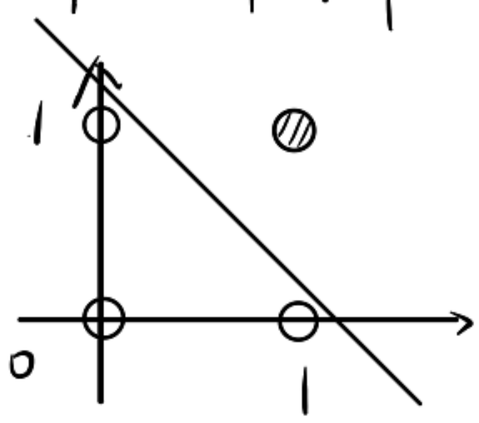
XOR问题

前面提到，单个神经元能处理线性可分问题，如：

$$y=\begin{cases}0 & if\ w\cdot x+b\le 0\\ 1 & if\ w\cdot x+b>0\end{cases} \qquad x=[x_1,x_2]$$

对于AND问题

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1



令 $w=[1,1]$ $b=-1$

只有 $x_1=1$ $x_2=1$ 时

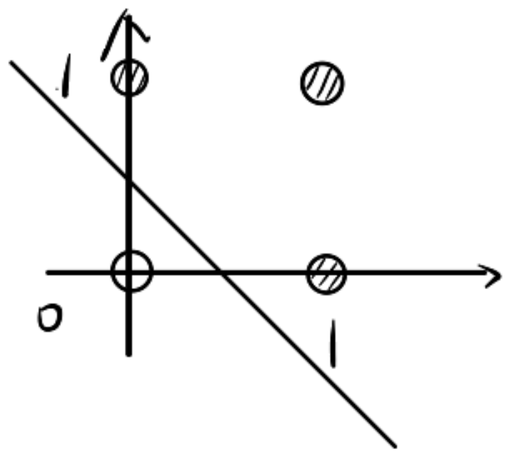
$y=1$

○ 表示输出 0

◍ 表示输出 1

对于OR问题

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1



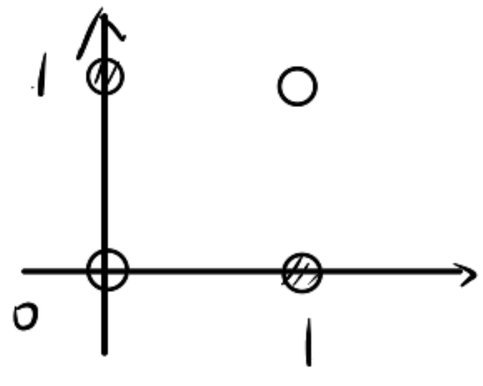
令 $w=[1,1]$ $b=0$

只有 $x_1=x_2=0$ 时

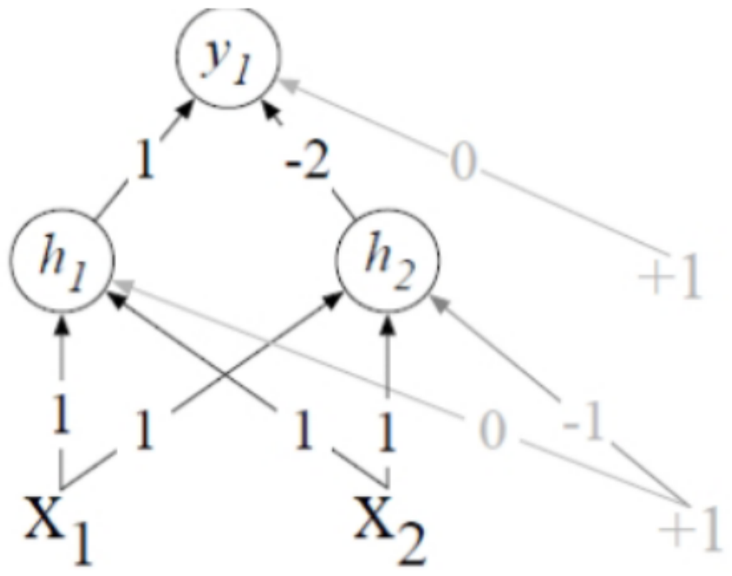
$y=0$

但对XOR问题

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



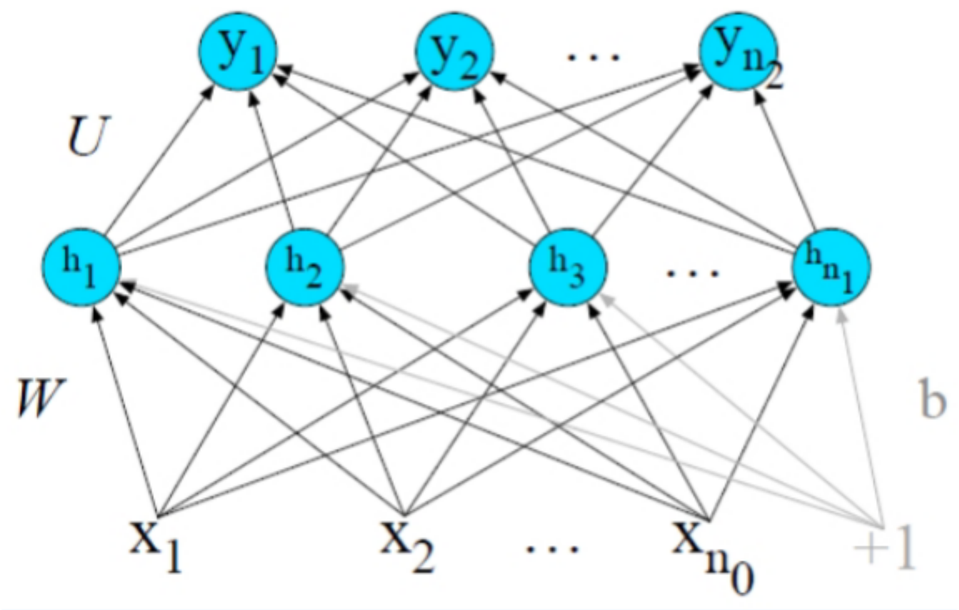
无法用单个神经元解决



h_1：或门
h_2：与门

前向神经网络

即不含环的多层神经网络，又称多层感知器



多元 Logistics 回归：类别 K，类别集合 C，文档向量 x

$$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{\sum_j P(x|C_j)P(C_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

其中 $a_k = \ln P(x|C_k)P(C_k)$

Softmax

假设每个类别的特征 x 服从高斯分布，且所有类别共享同一个协方差矩阵时，a_k 可以被简化为线性函数 $a_k(x) = w_k^T x + w_{k0}$

其中 $w_k = \Sigma^{-1}\mu_k$　μ_k 是 C_k 的特征均值向量，Σ^{-1} 是协方差矩阵的逆

$$w_{k0} = -\frac{1}{2}\mu_k^T \Sigma^{-1}\mu_k + \ln P(C_k)$$

Softmax 函数：

$$softmax(z) = \left[\frac{exp(z_1)}{\sum_{i=1}^{k} exp(z_i)}, \frac{exp(z_2)}{\sum_{i=1}^{k} exp(z_i)}, \cdots, \frac{exp(z_k)}{\sum_{i=1}^{k} exp(z_i)}\right]$$

$$Softmax(z_i) = \frac{exp(z_i)}{\sum_{j=1}^{k} exp(z_j)}$$

Softmax 的作用：将任意实数向量 $Z = [z_1, \cdots, z_k]$ 映射为和为 1 的概率分布

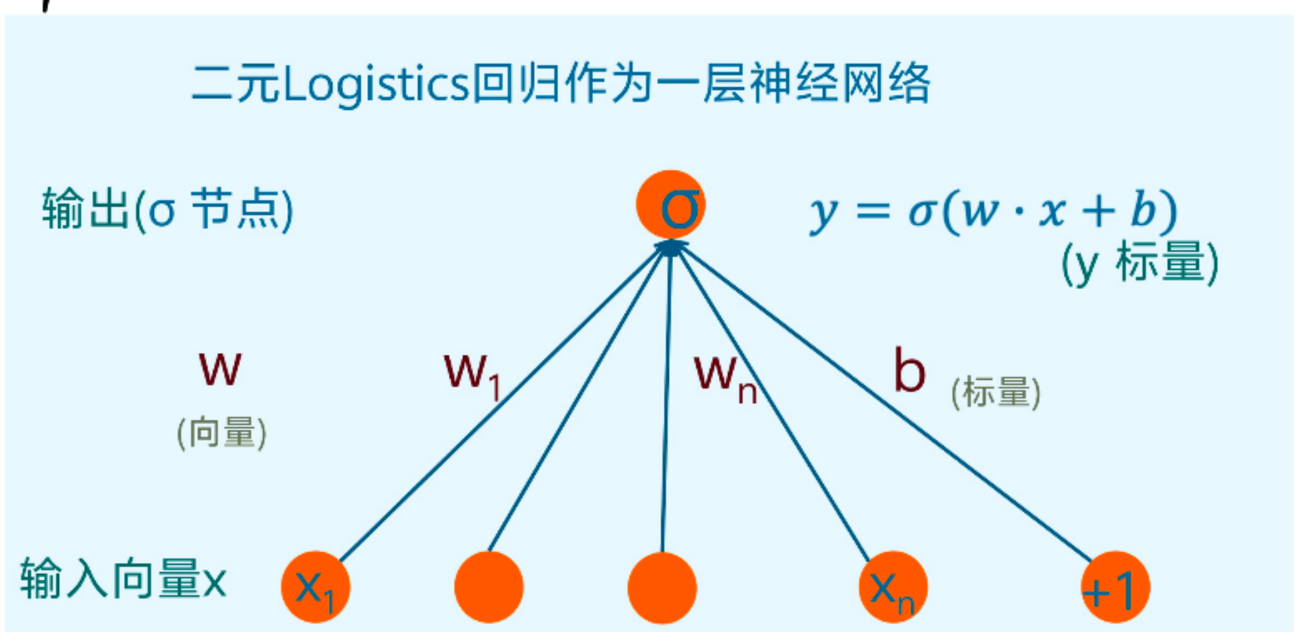
在神经网络中，Softmax 就是输出层的激活函数。隐藏层的输出（特征向量 h）经过线性变换得到向量 a_K（$a_K = W_K^T \cdot h + W_{K0}$），再由 Softmax 函数将 a_K 转化为每个类别的概率 $P(C_K | x)$

前向神经网络结构

1、单层网络 = 逻辑回归（输入层不计入层数）

① 二元 logistic 回归（二分类）

$y = \sigma(w \cdot x + b)$

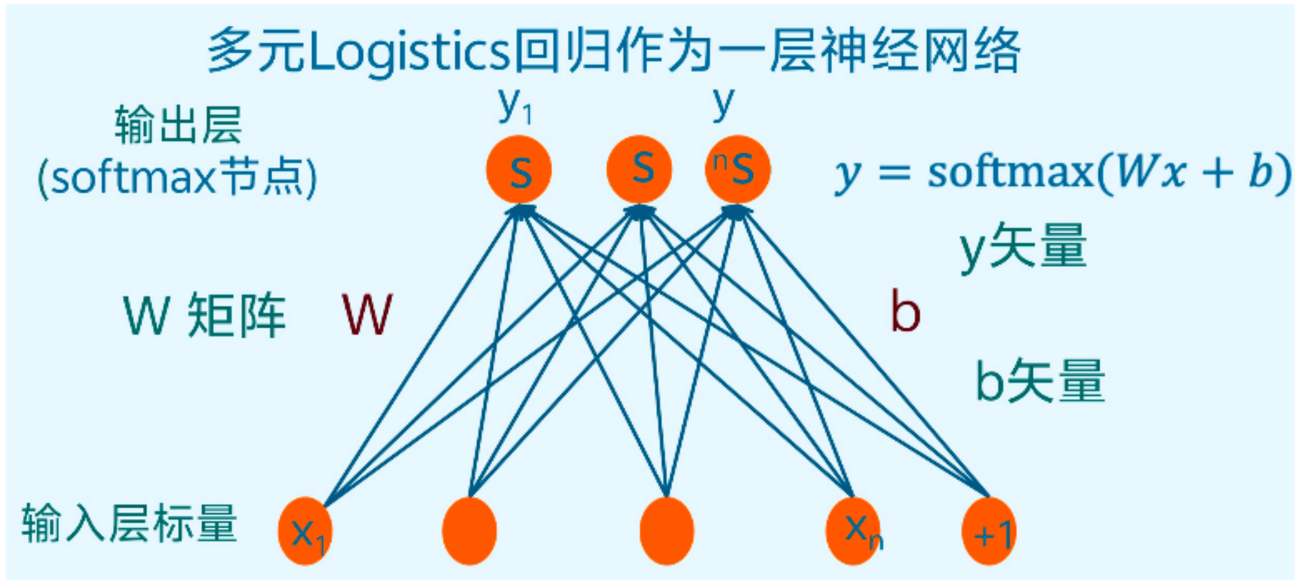


X：输入向量　W：权重向量　b：标量偏置

只能解决线性可分问题

② 多元 logistic 回归（多分类）

$y = softmax(Wx + b)$

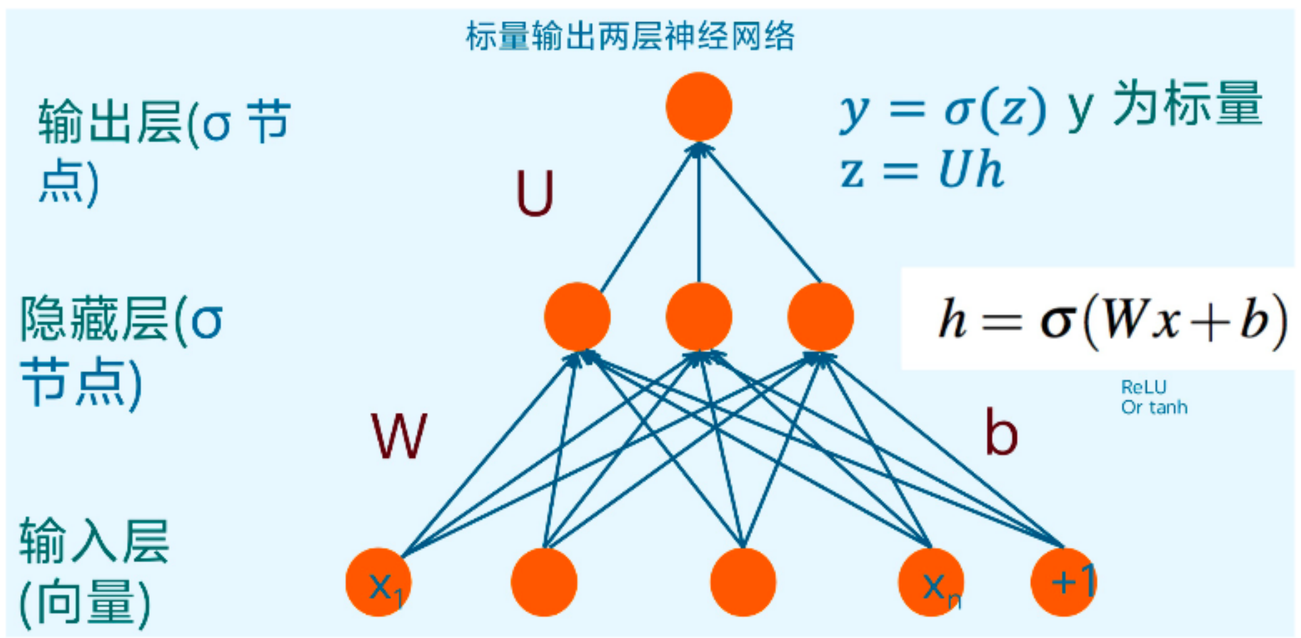


W：权重矩阵 每一行权重对应一个类别

b：偏置向量

2、多层神经网络

① 二分类（标量输出）



- $h = \sigma(Wx + b)$

  W：输入→隐藏层权重矩阵

  W_{ji}表示第i个输入单元x_i→第j个隐藏单元h_j的连接权重

  b：隐藏层偏置向量

  σ：隐藏层激活函数（Tanh 或 ReLU）

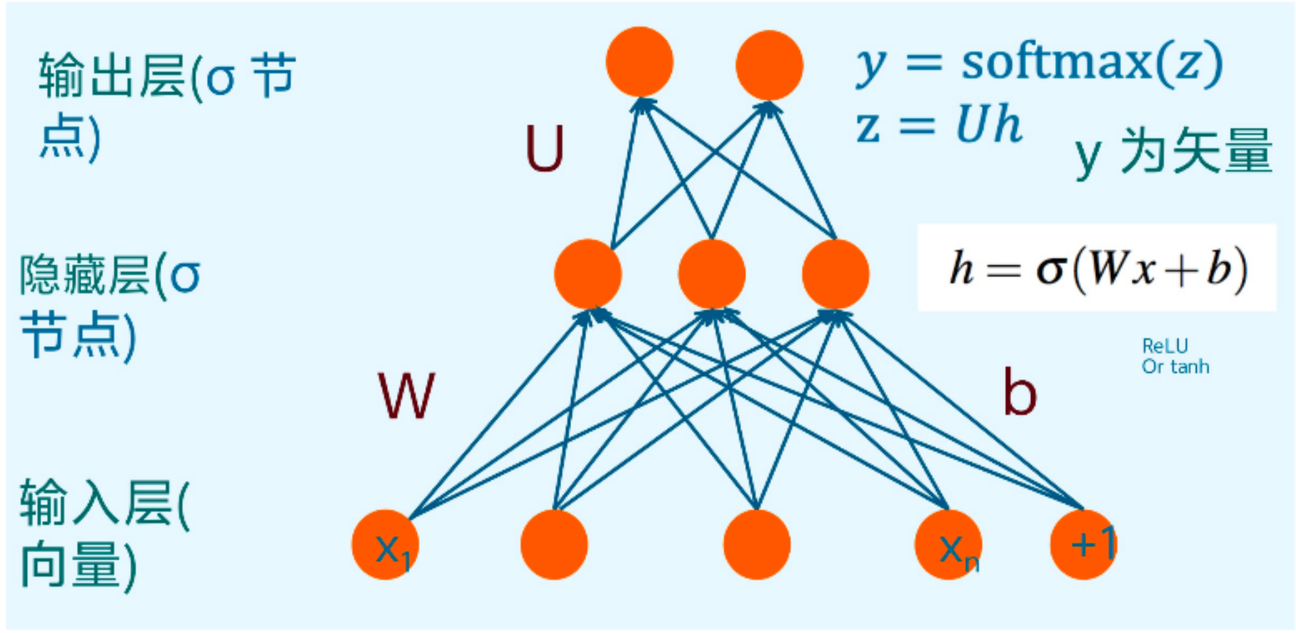
  h：隐藏层输出向量

- $z = Uh \quad y = \sigma(z)$

  U：隐藏层→输出层权重矩阵

  y：标量输出

② 多分类（向量输出）



$h = \sigma(Wx+b)$ $z = Uh$ $y = \text{softmax}(z)$

带隐藏层的神经网络 = 先自己从输入 x 中学习特征 h.

再对 h 做逻辑回归，替代了传统手动设计特征的方式

深层网络符号表示

$a^{[l]}$：第 l 层的输出（激活值） $a^{[0]} = x$（输入层）

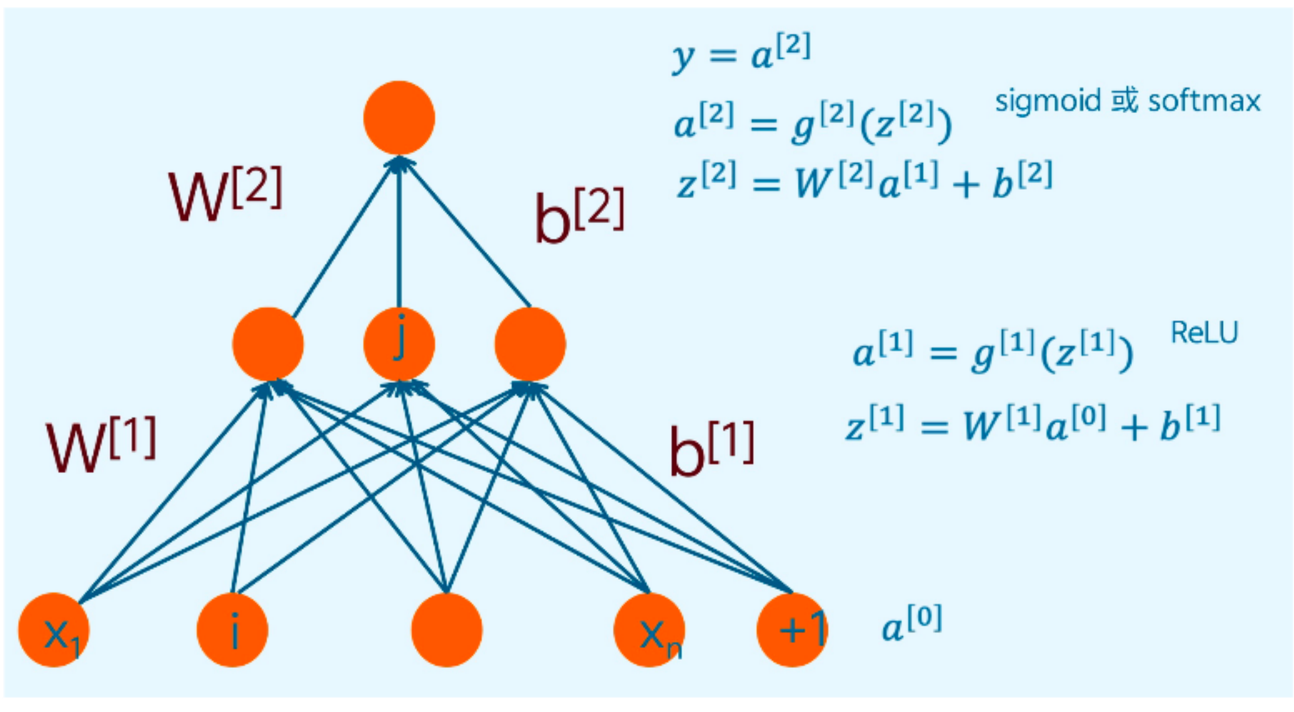
$z^{[l]}$：第 l 层的线性组合值（激活前）

$W^{[l]}$：第 l-1 层 → 第 l 层的权重矩阵

$b^{[l]}$：第 l 层的偏置向量

$g^{[l]}(\cdot)$：第 l 层的激活函数

如：



对于任意深度 n 的前向网络，每一层的计算逻辑完全一致

形成循环递推：

for i **in** 1..n

$$z^{[i]} = W^{[i]}\, a^{[i-1]} + b^{[i]}$$

$$a^{[i]} = g^{[i]}(z^{[i]})$$

$$\hat{y} = a^{[n]}$$

总结、

通用表示

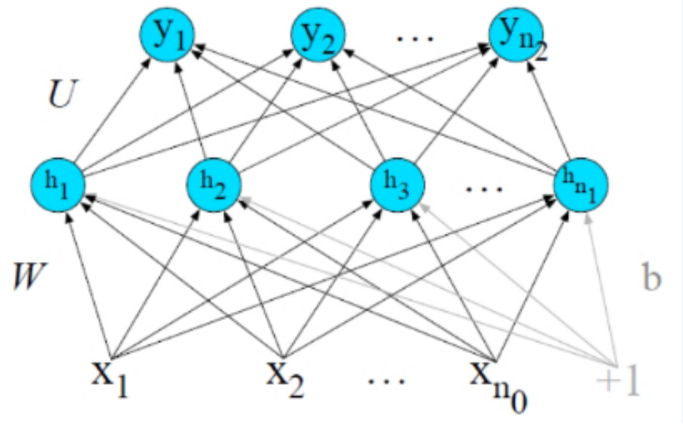
$$
\begin{aligned}
z^{[1]} &= W^{[1]}a^{[0]} + b^{[1]} \\
a^{[1]} &= g^{[1]}(z^{[1]}) \\
z^{[2]} &= W^{[2]}a^{[1]} + b^{[2]} \\
a^{[2]} &= g^{[2]}(z^{[2]}) \\
\hat{y} &= a^{[2]}
\end{aligned}
$$

$$
\begin{aligned}
h &= \sigma(Wx+b) \\
z &= Uh \\
y &= \text{softmax}(z)
\end{aligned}
$$

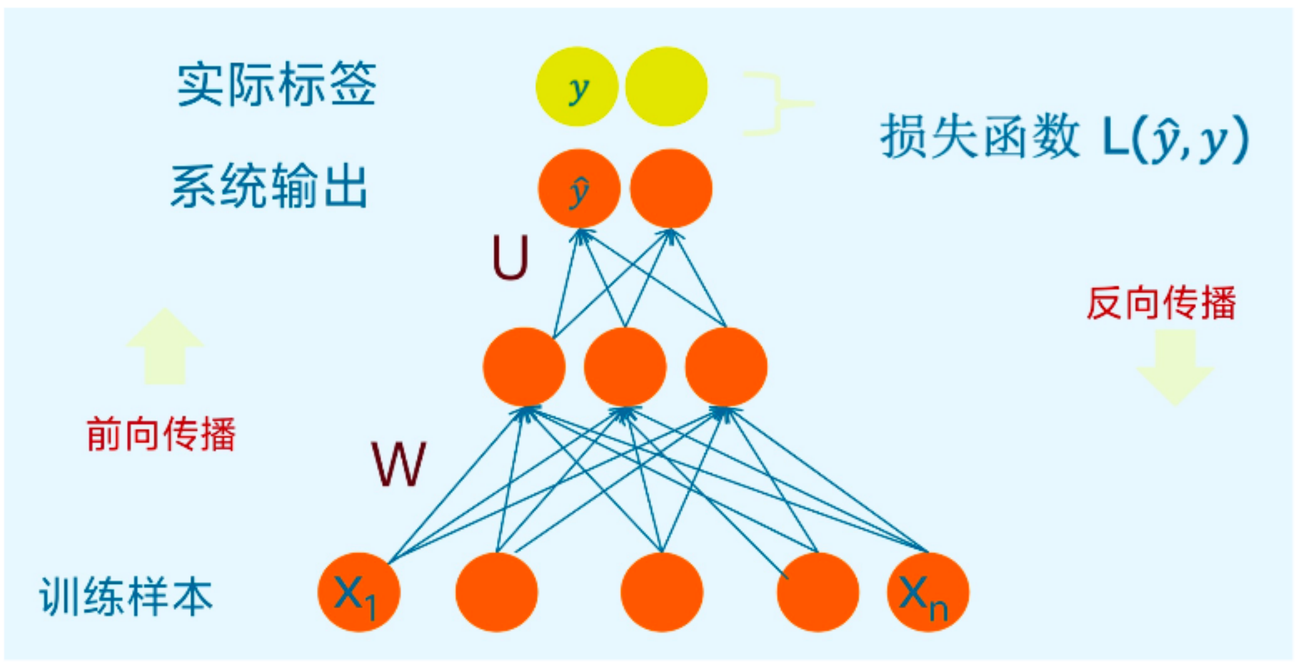
前向算法

for i **in** 1..n

$$
\begin{aligned}
z^{[i]} &= W^{[i]}\,a^{[i-1]} + b^{[i]} \\
a^{[i]} &= g^{[i]}(z^{[i]}) \\
\hat{y} &= a^{[n]}
\end{aligned}
$$



神经网络训练



对于每一个样本(x, y):

① 执行前向计算预测 $\hat{y}$

② 执行后向计算更新权重

- 对于每一个输出层节点，计算 y 与 $\hat{y}$ 之间的损失 L
  计算损失函数对最后一层权重的梯度，根据梯度更新权重
- 对于每一个隐藏节点，计算当前层输出的损失
  计算当前损失对于当前层权重的梯度，根据梯度更新权重

损失函数

二元分类 $L_{CE}(\hat{y}, y) = -\log P(y|x) = -[y\log\hat{y} + (1-y)\log(1-\hat{y})]$

多元分类 $L_{CE}(\hat{y}, y) = -\sum_{i=1}^{C} y_i \log \hat{y}_i$

硬判决分类 $(y_i = 1, y_j = 0 \ \forall j \neq i)$ $L_{CE}(\hat{y}, y) = -\log \hat{y}_i$

Softmax $Lce(\hat{y}, y) = -\log \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$ $z = Uh$

单层网络：梯度下降

$$Lce(\hat{y}, y) = \log p(y|x) = -[y \log \hat{y} + (1-y)\log(1-\hat{y})]$$

$$= -\{y \log \sigma(w \cdot x + b) - (1-y)\log[1-\sigma(wx+b)]\}$$

$$\theta = \arg\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} Lce(y^{(i)}, x^{(i)}, \theta)$$

$$w^{t+1} = w^{t} - \eta \frac{d}{dw} f(x; w)$$

其中 $\frac{\partial Lce(w,b)}{\partial w_j} = (\hat{y} - y) x_j$

$$= [\sigma(w \cdot x + b) - y] x_j$$

注：对多元分类 $\frac{\partial Lce}{\partial w_k} = -[1\{y=k\} - p(y=k|x)] x_k$

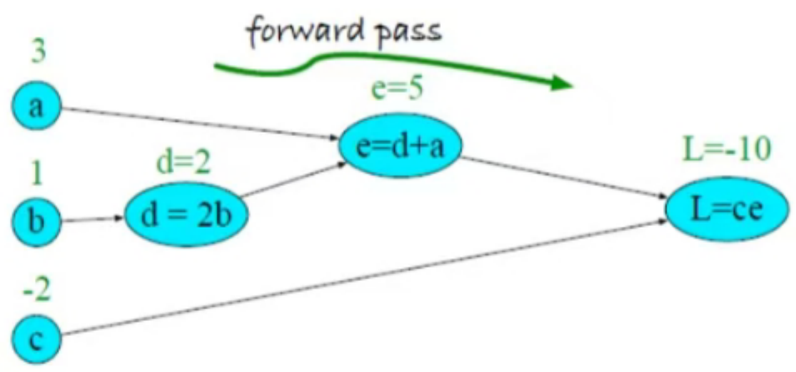
$$= -[1\{y=k\} - \frac{e^{w_k \cdot x + b_k}}{\sum_{j=1}^{k} e^{w_j \cdot x + b_j}}] x_k$$

$$1\{y=k\} = \begin{cases} 1 & y=k \\ 0 & y \neq k \end{cases}$$

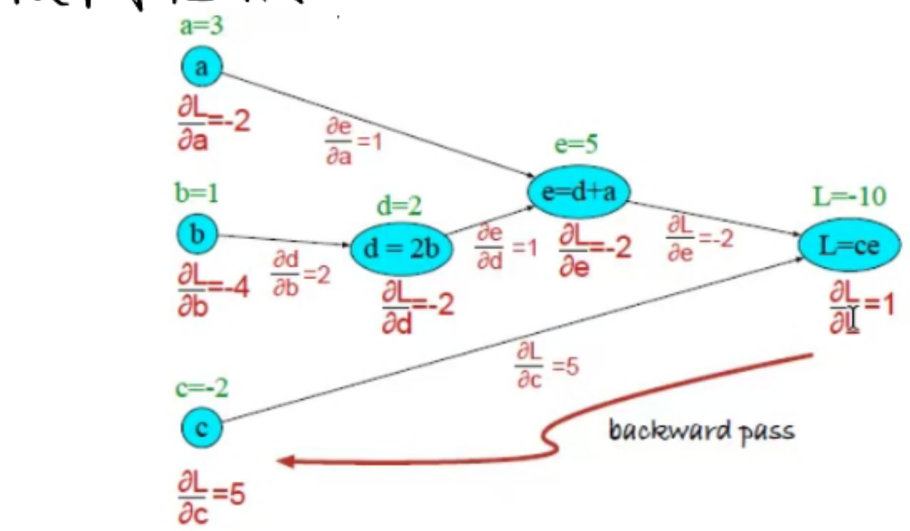
多层网络：

计算图：一种数学表达式计算过程的表示方法

如：$L(a,b,c) = c(a+2b)$



反向传播



$$
\begin{aligned}
L = ce: \quad & \frac{\partial L}{\partial e} = c, \frac{\partial L}{\partial c} = e \\
e = a + d: \quad & \frac{\partial e}{\partial a} = 1, \frac{\partial e}{\partial d} = 1 \\
d = 2b: \quad & \frac{\partial d}{\partial b} = 2
\end{aligned}
$$

链式法则：$\frac{\partial L}{\partial c} = e$

$$\frac{\partial L}{\partial a} = \frac{\partial L}{\partial e}\frac{\partial e}{\partial a}$$

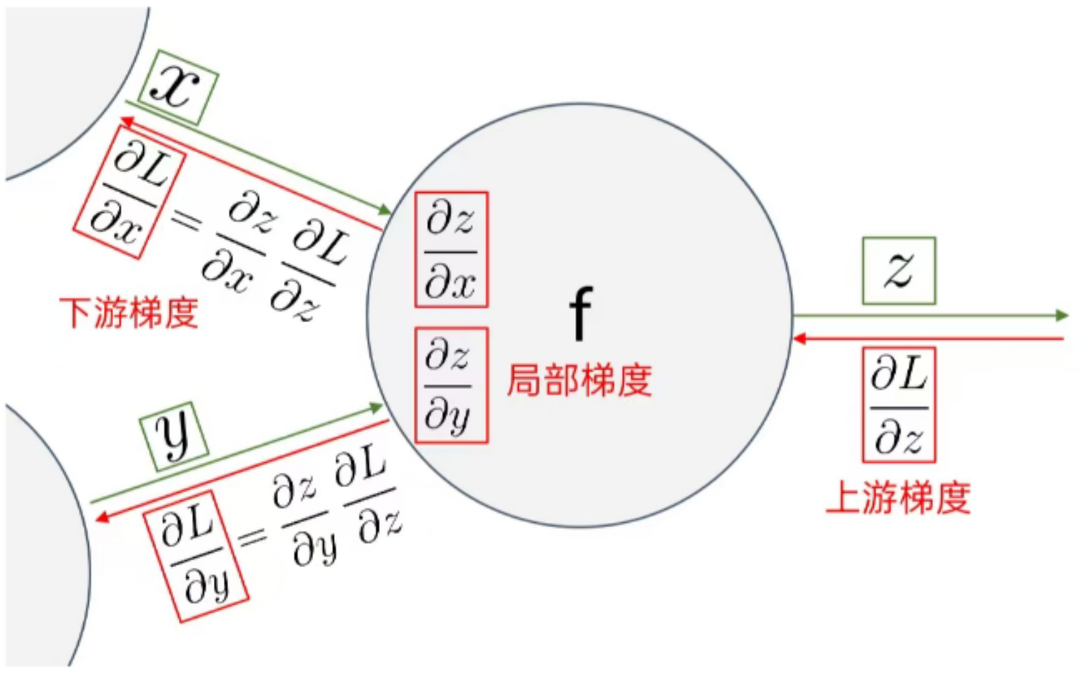
$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial e}\frac{\partial e}{\partial d}\frac{\partial d}{\partial b}$$

以 $\frac{\partial L}{\partial a} = \frac{\partial L}{\partial e}\frac{\partial e}{\partial a}$ 为例

$\frac{\partial L}{\partial a}$ 称为下游梯度

$\frac{\partial L}{\partial e}$ 称为上游梯度

$\frac{\partial e}{\partial a}$ 称为局部梯度



例、

网络

$$
\begin{aligned}
z^{[1]} &= W^{[1]}\mathbf{x} + b^{[1]} \\
a^{[1]} &= \mathrm{ReLU}(z^{[1]}) \\
z^{[2]} &= W^{[2]}a^{[1]} + b^{[2]} \\
a^{[2]} &= \sigma(z^{[2]}) \\
\hat{y} &= a^{[2]}
\end{aligned}
$$

计算图

激活函数梯度

$$\text{sigmoid}: \frac{d\sigma(z)}{dz} = \sigma(z)(1-\sigma(z))$$

$$\tanh: \frac{d\tanh(z)}{dz} = 1-\tanh^2(z)$$

$$ReLU: \frac{d\,ReLU(z)}{dz} = \begin{cases} 0 & for\ x<0 \\ 1 & for\ x\geq 0 \end{cases}$$

避免过拟合：

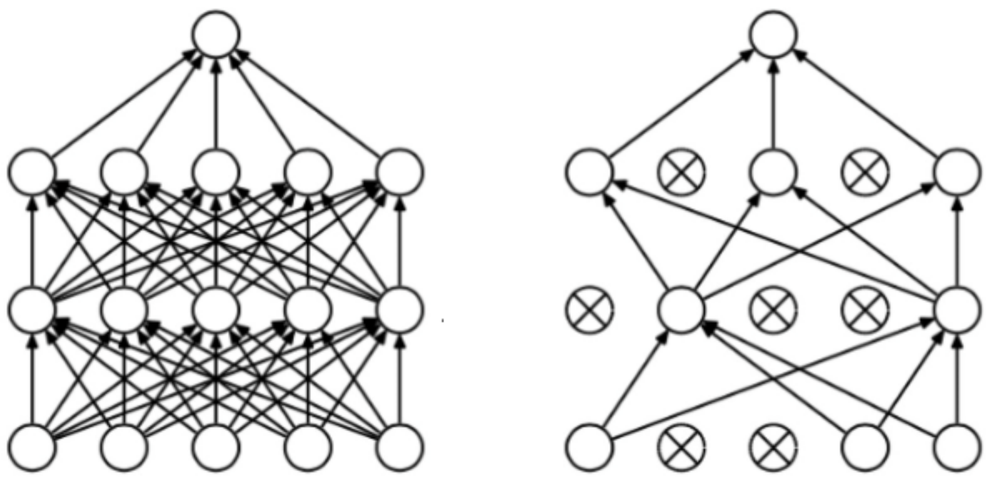
1、正则化：$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{m} \log P(y^{(i)}|x^{(i)}) - \alpha R(\theta)$

$\alpha R(\theta)$：正则化项（惩罚项），α是正则化强度
$R(\theta)$是对θ的约束函数，用于保持
θ简单

原理：过拟合的根本是θ过大、过于复杂而将噪
声、个别样本、特殊特征等误判为通用规律
在公式中加入"$-\alpha R(\theta)$"，是在最大化"拟合
度-惩罚项"，即让模型权衡"拟合数据"
与"保持θ简单"

2、dropout：又称随机失活，在每一轮训练的前向传播中
以一定概率P（称为dropout率）随机让隐藏
层的部分神经元临时失活（本次输出为0）
而在测试阶段激活全部神经元

原理：部分神经元会过度依赖某些神经元的输
出，只学习到训练集的细节特征、局部噪声
因此用随机失活来减少这种行为的发生



3、降低网络参数规模

4、数据增强

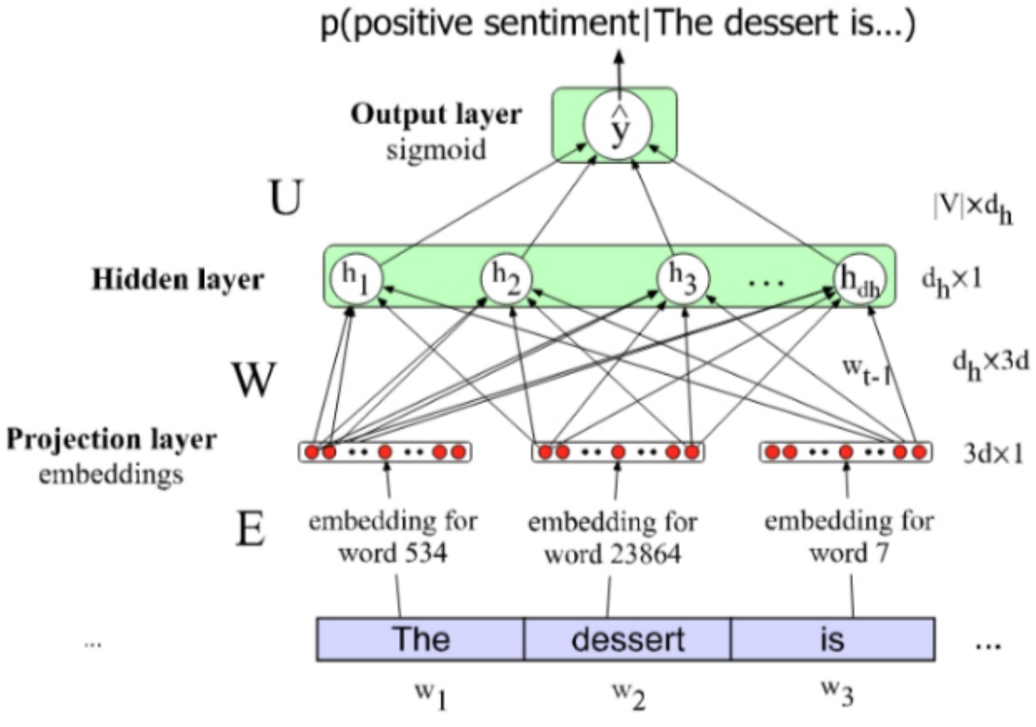
5、训练测试数据划分

神经网络文本应用

深层神经网络的好处是机器不依赖于手动特征的构建，能够自己学习数据的深层特征

文本分类：可以增加文本的嵌入表示，如使用词向量、句子向量等

如以词向量作为输入：



问题：文本长度不固定

简单解决方法：1、输入长度设置足够的大。文本短则填0长则截断

2、使用句子向量表示文本。以所有词向量的平均值作为文本嵌入向量，或以词向量的各个维度的最大值作为文本嵌入向量

神经语言模型(LMS)

典型：Transformers

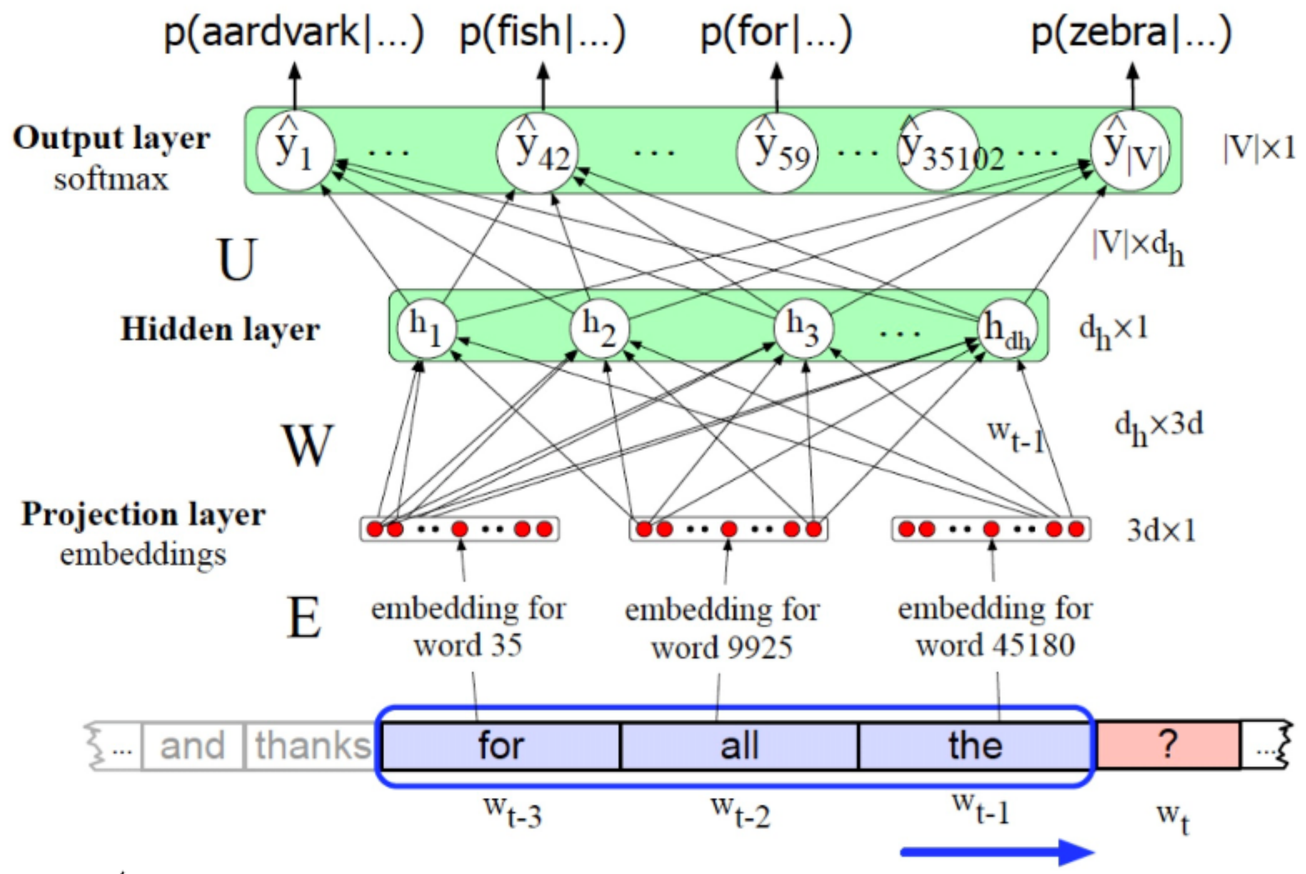
简单的前向LMS (Bengio)：给定前词 $w_{t-1}, w_{t-2}, w_{t-3}, \cdots$

预测下一个词：w_t

使用一个窗口长度固定的滑动窗口来处理任意长度的序列

$$P(w_t \mid w_1^{t-1}) \approx P(w_t \mid w_{t-N+1}^{t-1})$$

N为窗口长度



① 输入层

提取 w_t 前 N 个词，

② 投影层（Projection layer）

也即是神经网络输入层

将这N个词通过查找矩阵E投影为向量

③ 隐藏层（Hidden layer）

将 $w_{t-1} \cdots w_{t-N+1}$ 词向量拼在一起组成长向量 x，并进行计算

④ 输出层

输出一个 |V|×1 的概率向量（|V|是词表大小），选最大概率对应的词作为 w_t