上海海事大学
SHANGHAI MARITIME UNIVERSITY

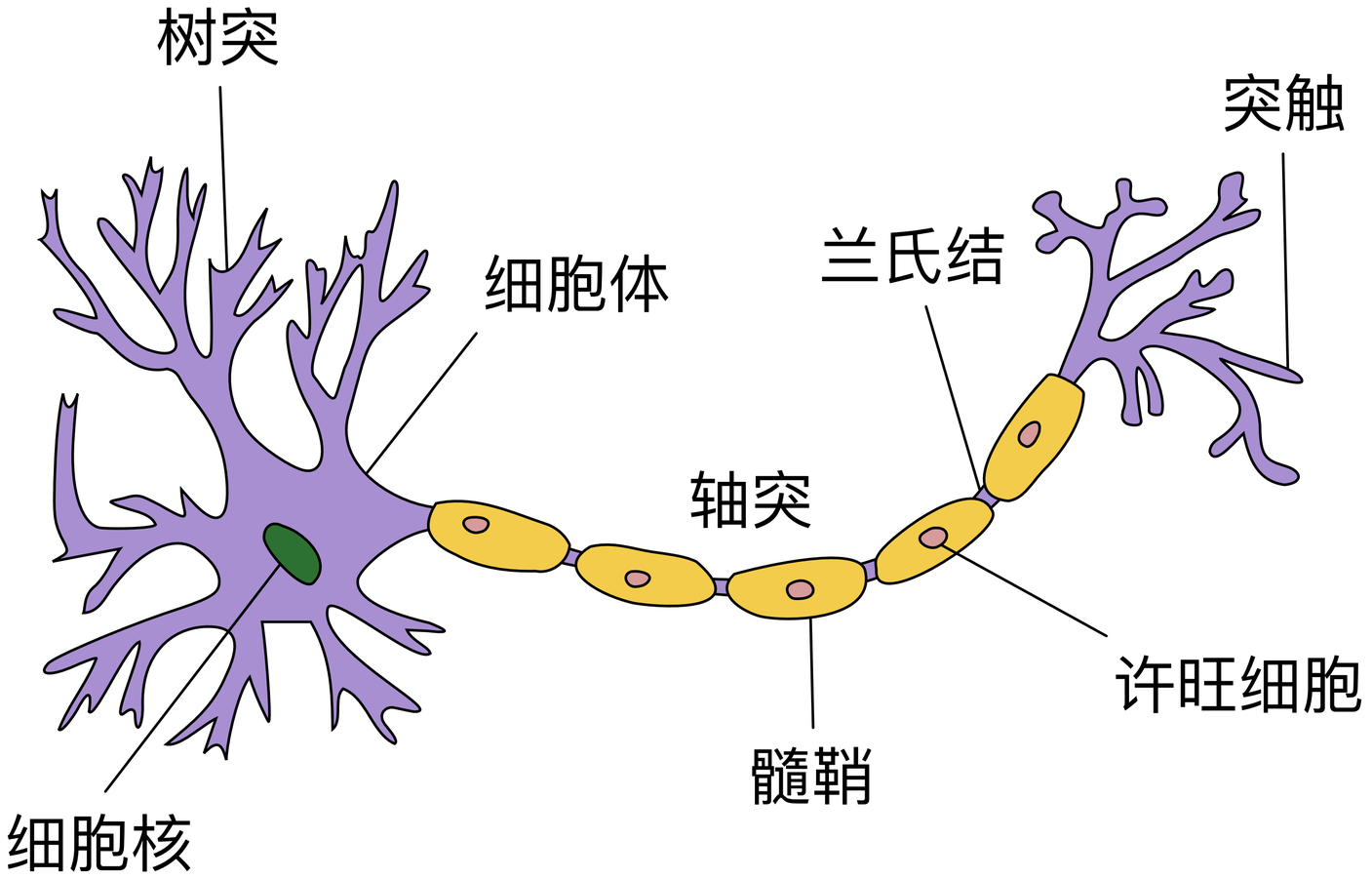
自然语言处理

2024–2025 学年第 2 学期

信息工程学院　谢雨波

神经网络

（你脑子里的）神经元



（神经网络里的）神经元

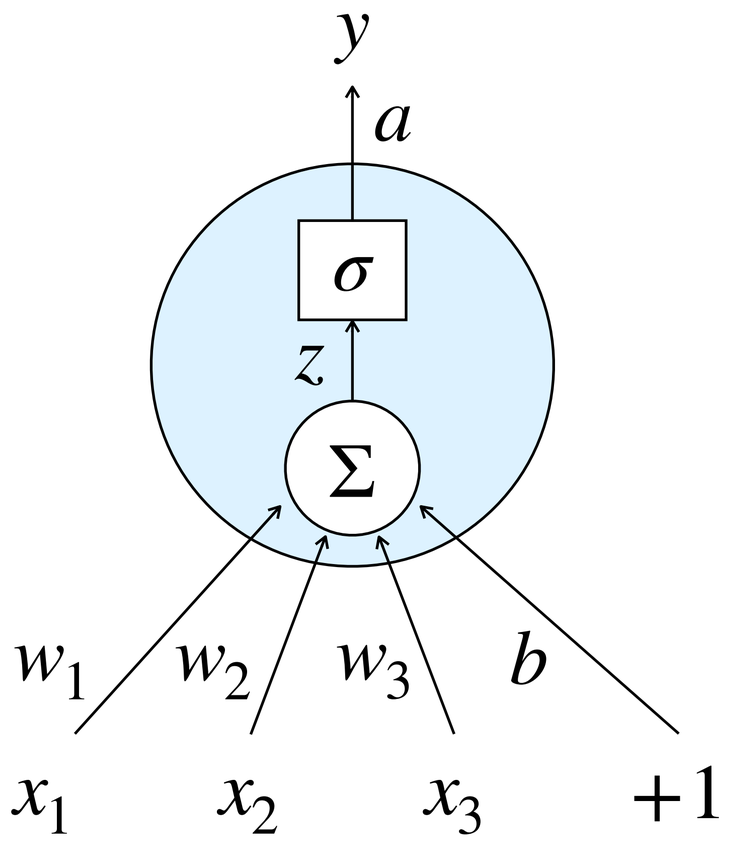
输出

非线性转换

加权和

权重

输入层



神经元

- 计算输入的加权和（加上一个偏置）

$$z = b + \sum_i w_i x_i$$

$$z = w \cdot x + b$$

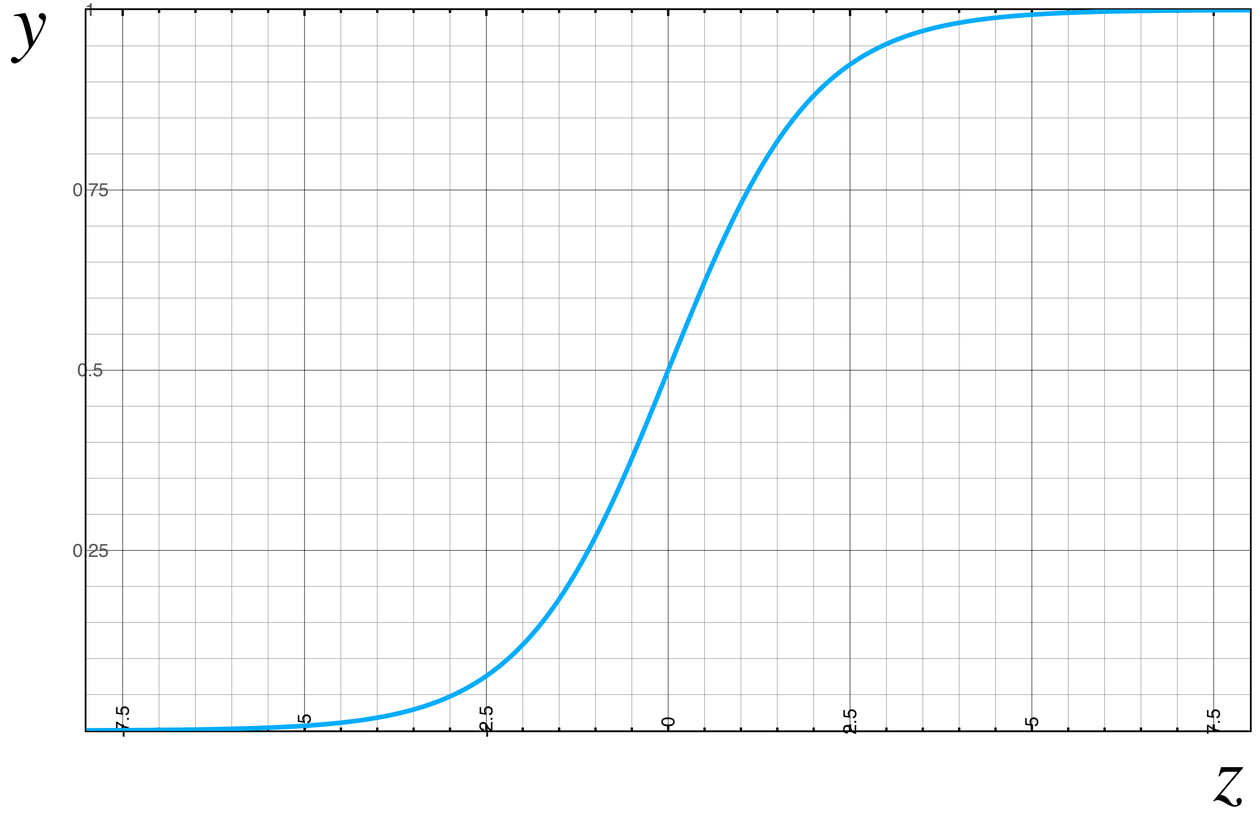
- 然后在 z 的基础上进行非线性转换（激活函数）

$$y = a = f(z)$$

非线性激活函数

- 例如 sigmoid 函数

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



神经元

输出

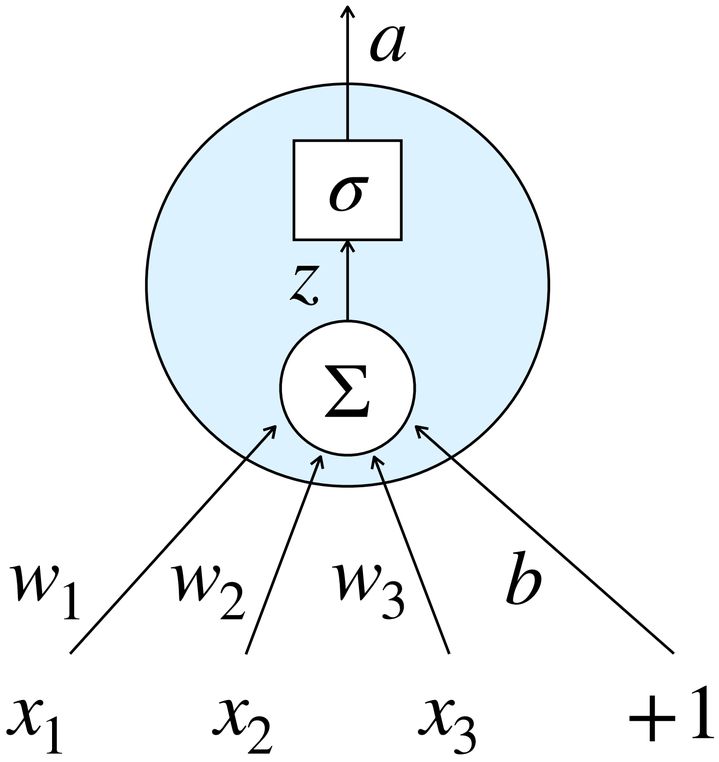
非线性转换

加权和

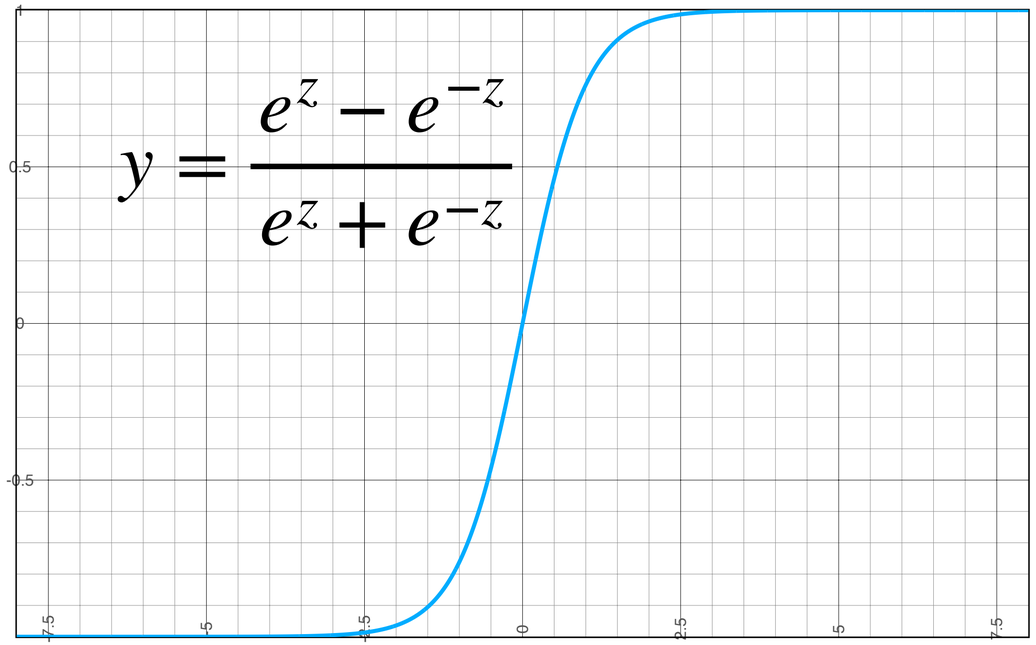
权重

输入层

$$y = \sigma(w \cdot x + b) = \frac{1}{1 + e^{-(w \cdot x + b)}}$$

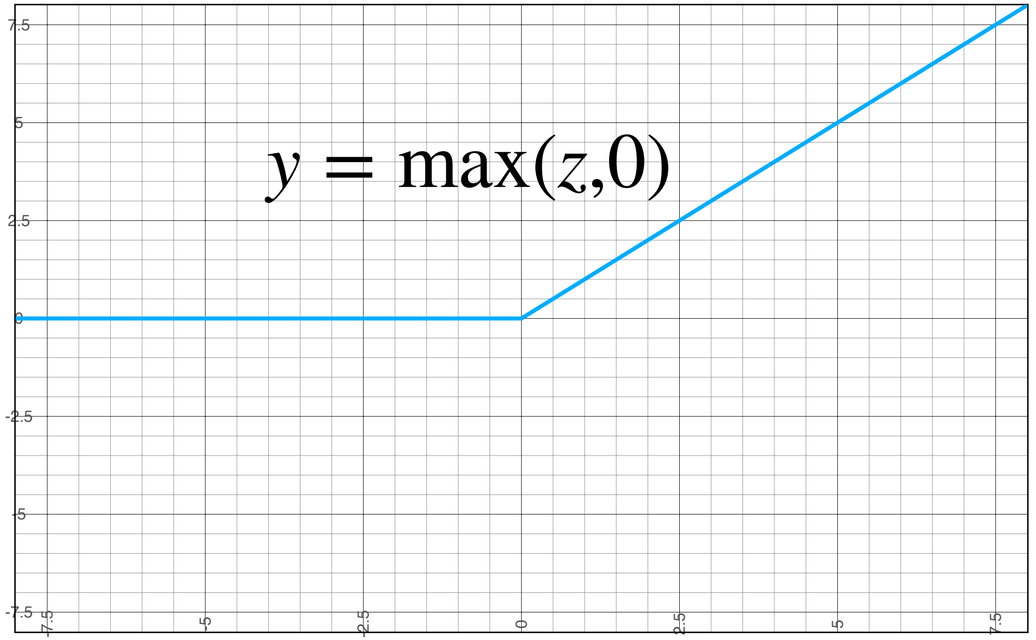


其他非线性激活函数



tanh

(最常用)



ReLU: Rectified Linear Unit

XOR 问题

- 神经元可以拟合输入的简单函数吗?

AND			OR			XOR		
x1	x2	y	x1	x2	y	x1	x2	y
0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	1	1
1	0	0	1	0	1	1	0	1
1	1	1	1	1	1	1	1	0

感知器

- **感知器（Perceptron）**：一个非常简单的神经元
  - 二元输出
  - 没有非线性激活函数

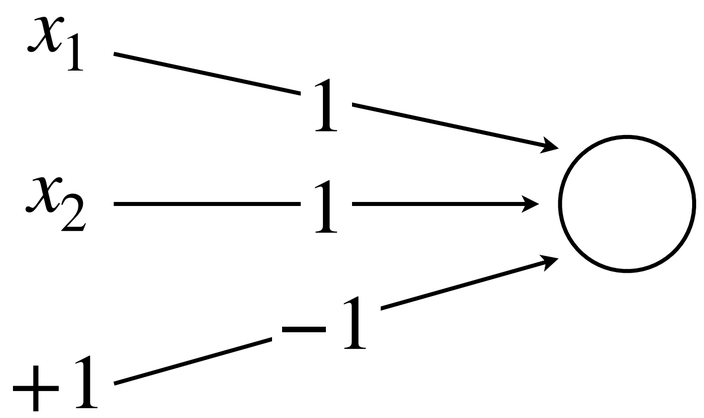
$$y = \begin{cases} 0, & \text{if } w \cdot x + b \leq 0 \\ 1, & \text{if } w \cdot x + b > 0 \end{cases}$$

使用感知器解决 AND 问题

$$y = \begin{cases} 0, & \text{if } w \cdot x + b \leq 0 \\ 1, & \text{if } w \cdot x + b > 0 \end{cases}$$

AND

x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1

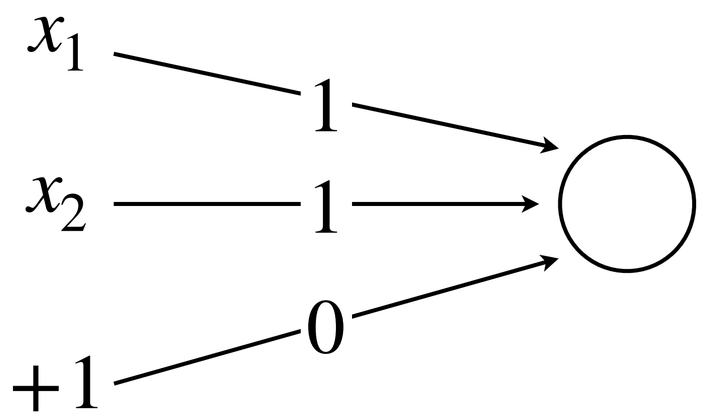


使用感知器解决 OR 问题

$$y = \begin{cases} 0, & \text{if } w \cdot x + b \leq 0 \\ 1, & \text{if } w \cdot x + b > 0 \end{cases}$$

OR

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1



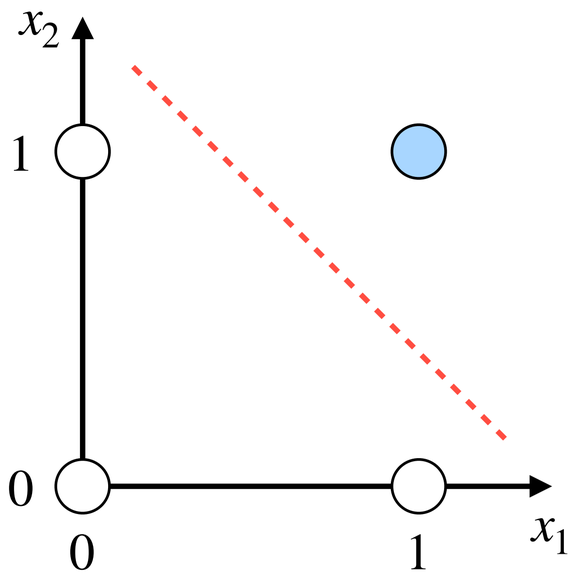
感知器无法解决 XOR 问题

- 感知器是线性分类器
  - 给定 x_1 和 x_2，决策边界是一条直线：

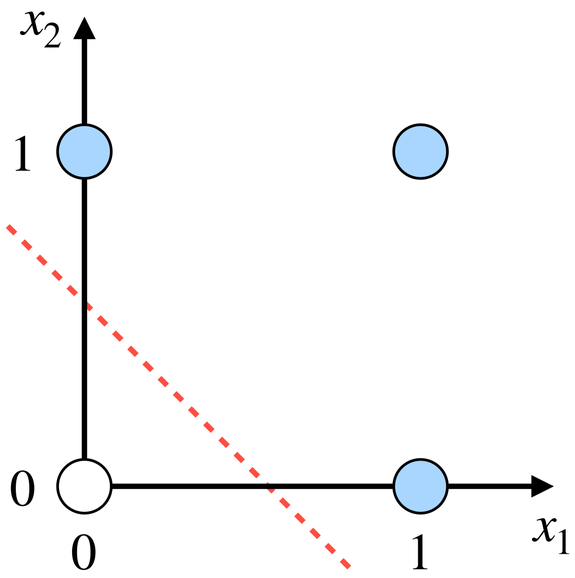
$$w_1 x_1 + w_2 x_2 + b = 0$$

    - 如果输入在直线的一边，则输出 0
    - 如果输入在直线的另一边，则输出 1

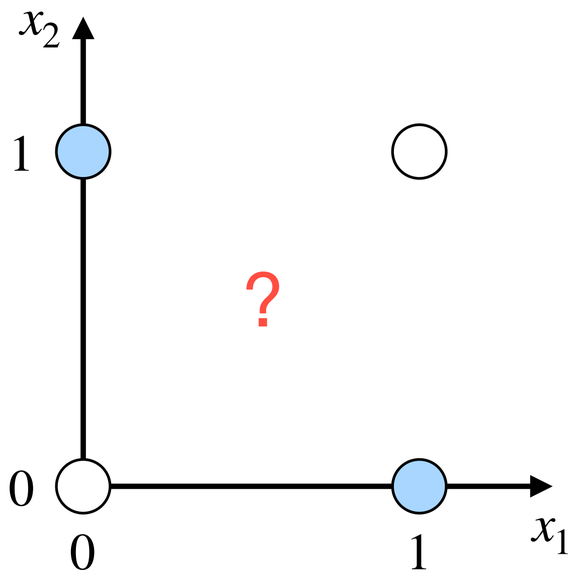
决策边界



x_1 AND x_2



x_1 OR x_2

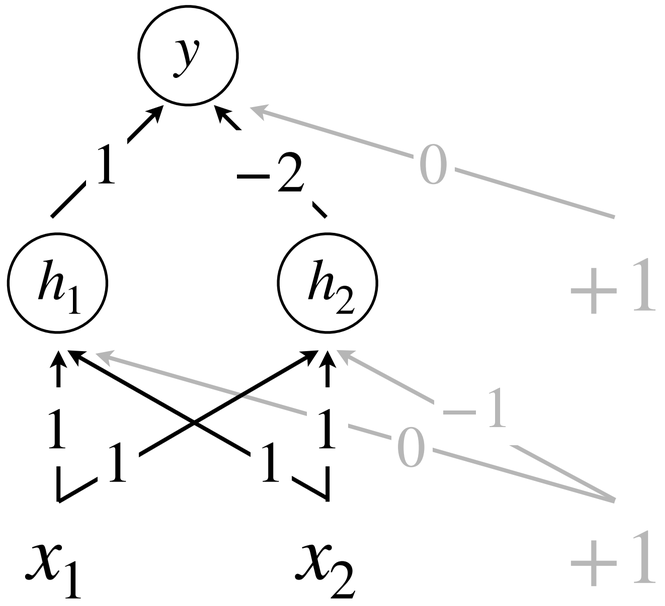


x_1 XOR x_2

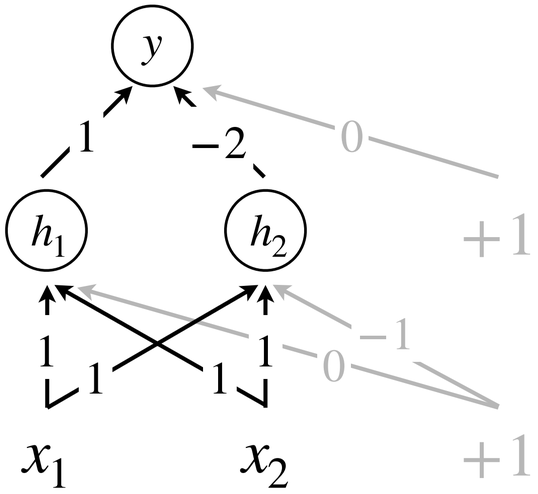
XOR 问题

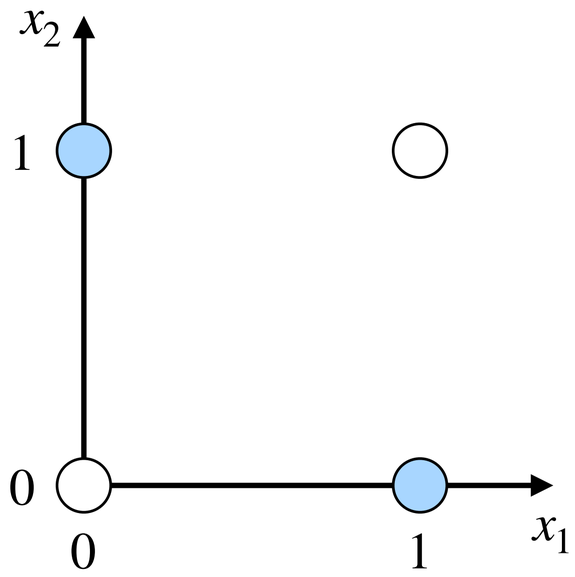
- XOR 问题不能用单个感知器解决
- 但是可以用多层网络结果的感知器解决

XOR		
x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

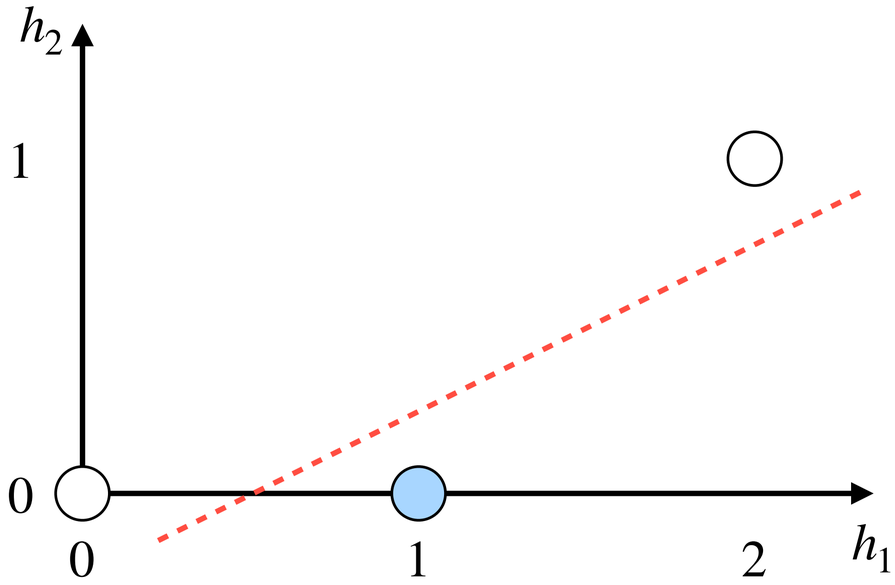


隐藏层表示 h





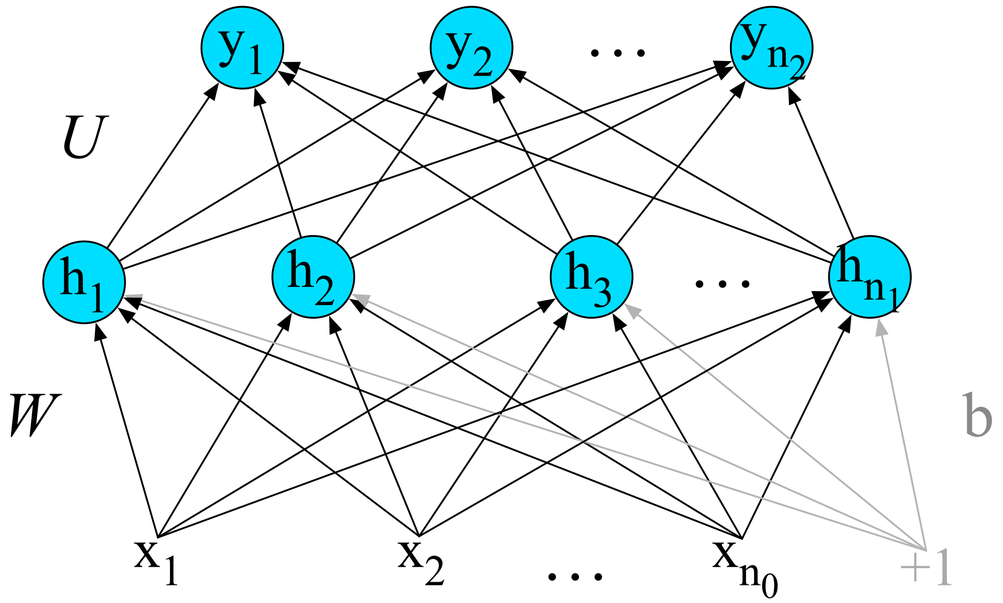
x 空间



h 空间

前馈神经网络

- **前馈神经网络（Feedforward Neural Network）**
  - 也被称为**多层感知器（Multi-Layer Perceptron, MLP）**

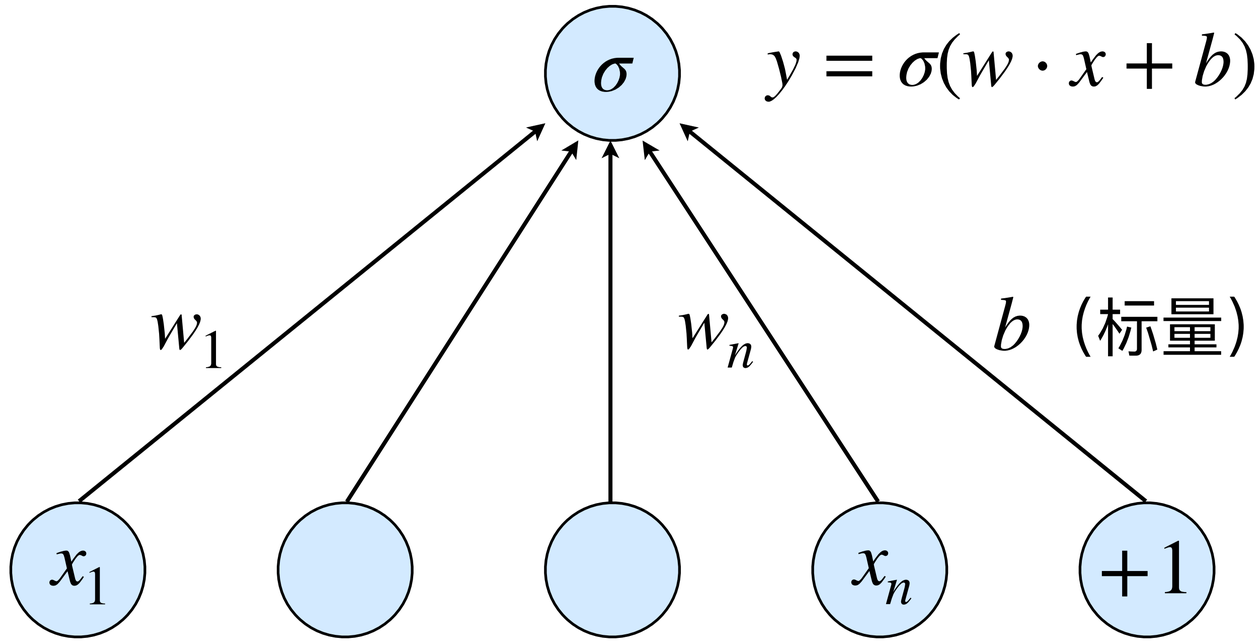


二元逻辑回归

输出层

权重 w（向量）

输入层

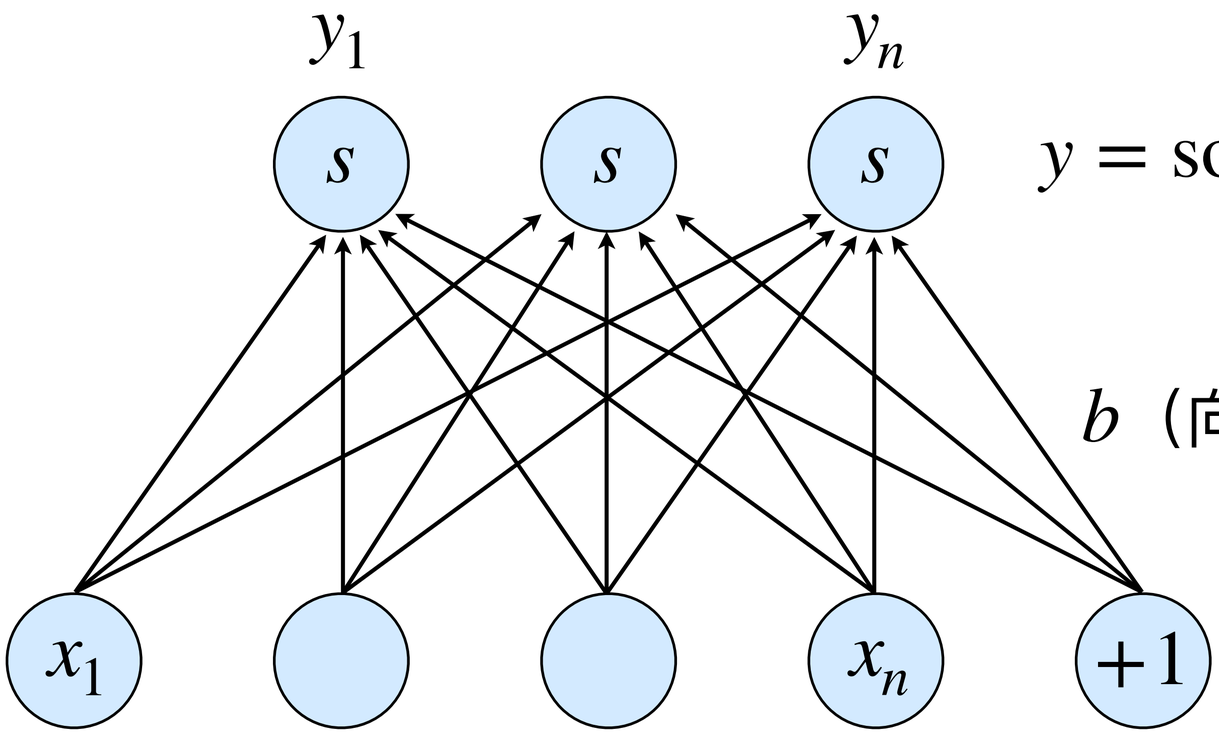


多元逻辑回归

输出层

权重 W（矩阵）

输入层



$y = \mathrm{softmax}(Wx + b)$

b（向量）

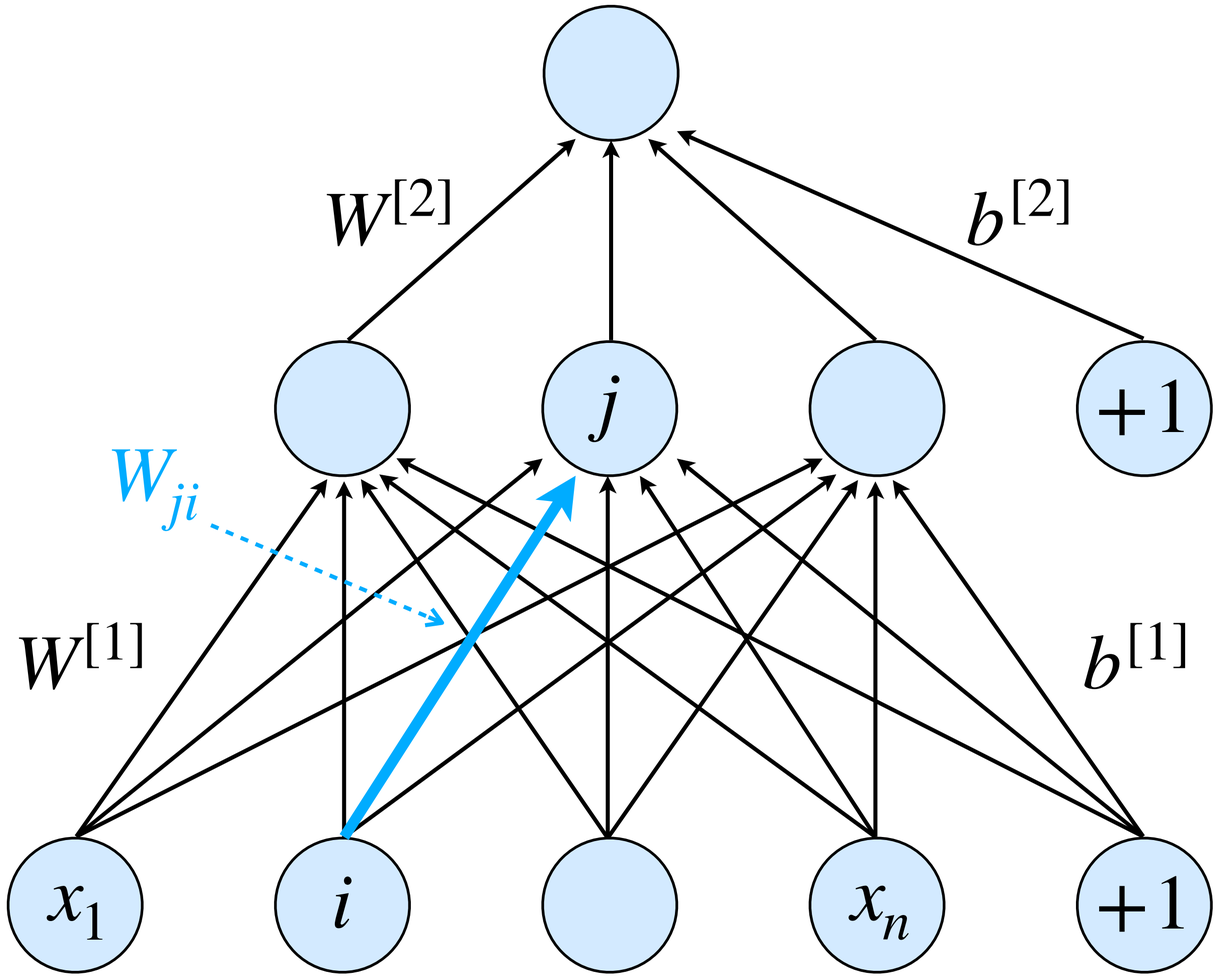
softmax 函数

- sigmoid 函数的一般情况
- 对于 $z \in \mathbb{R}^k$

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

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

多层神经网络



$$y = a^{[2]}$$

$$a^{[2]} = g^{[2]}\left(z^{[2]}\right) \quad \text{sigmoid/softmax}$$

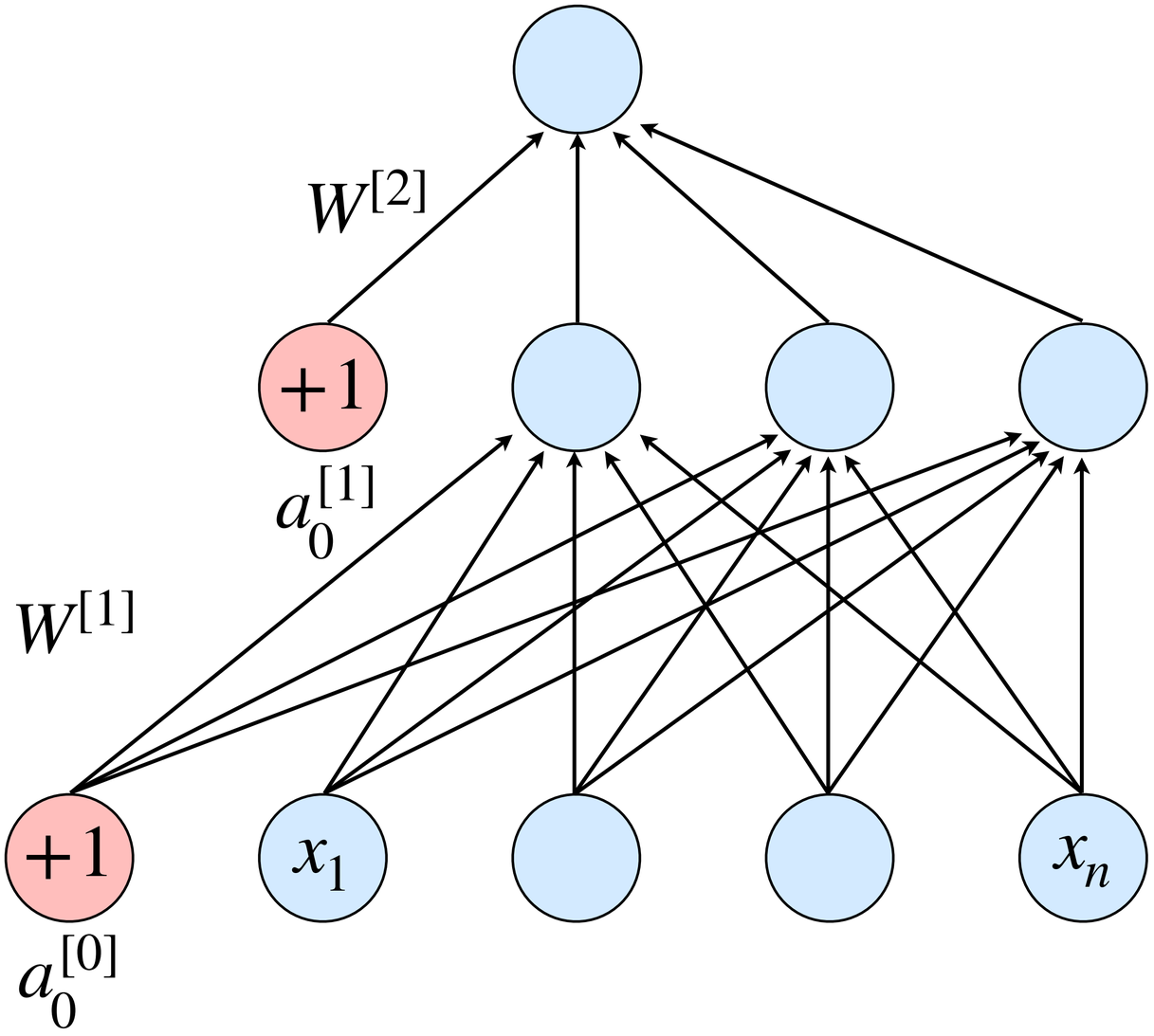
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

$$a^{[1]} = g^{[1]}\left(z^{[1]}\right) \quad \text{ReLU}$$

$$z^{[1]} = W^{[1]}a^{[0]} + b^{[1]}$$

$$a^{[0]} = x$$

多层神经网络



$$y = a^{[2]}$$

$$a^{[2]} = g^{[2]}\left(z^{[2]}\right) \quad \text{sigmoid/softmax}$$

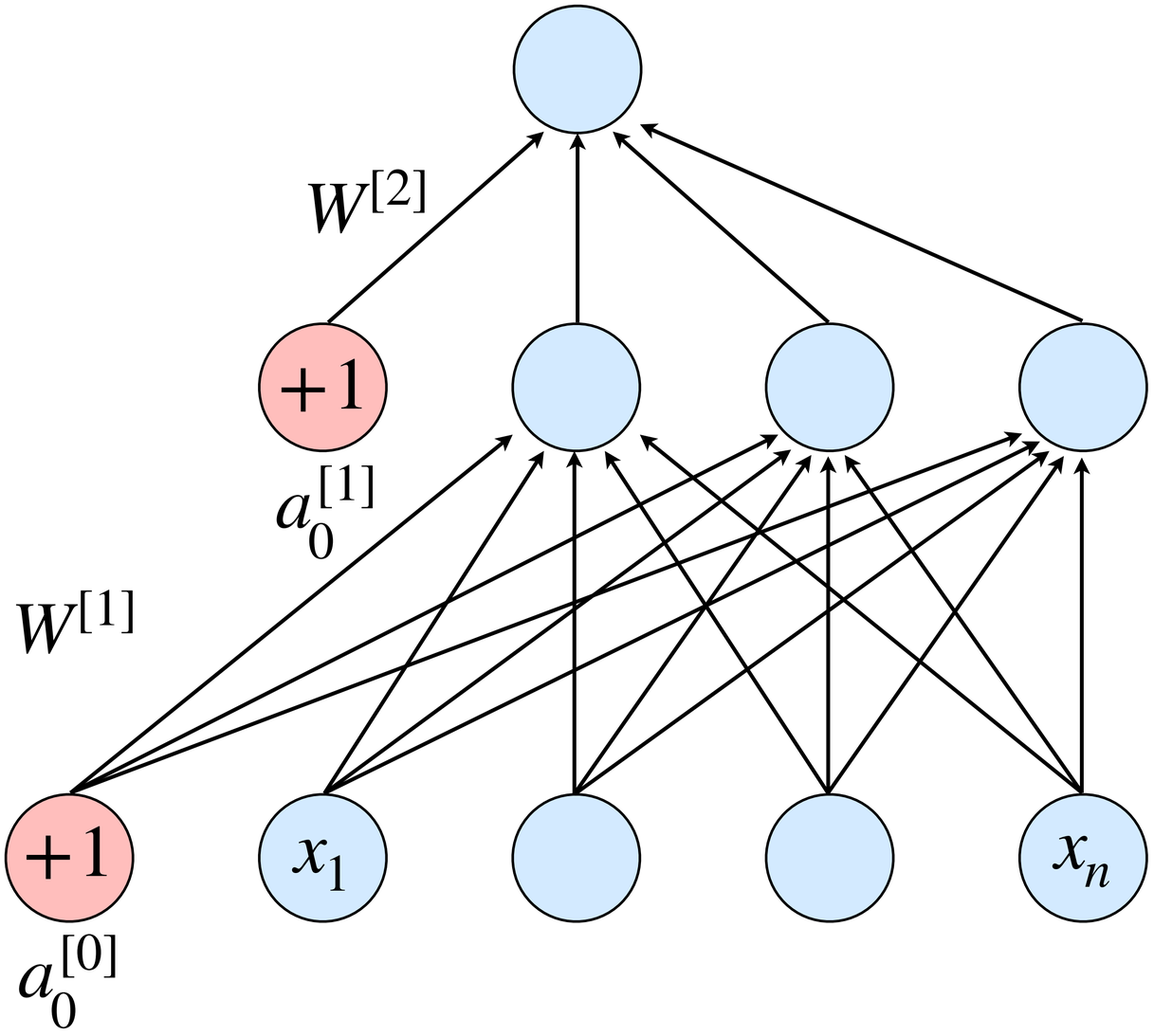
$$z^{[2]} = W^{[2]}a^{[1]}$$

$$a^{[1]} = g^{[1]}\left(z^{[1]}\right) \quad \text{ReLU}$$

$$z^{[1]} = W^{[1]}a^{[0]}$$

$$a^{[0]} = x$$

多层神经网络



$$a^{[0]} = x$$

for l **in** $1..n$:

$$z^{[l]} = W^{[l]} a^{[l-1]}$$

$$a^{[l]} = g^{[l]}\left(z^{[l]}\right)$$

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

神经网络的训练

梯度下降

- 梯度下降（Gradient Descent）
  - **选择初始参数**
  - **计算梯度**：在当前参数值处计算目标函数的梯度
  - **更新参数**：沿着负梯度方向更新参数

$$\theta^{t+1} = \theta^t - \eta \nabla L(\theta^t)$$

  - **重复迭代**

反向传播

- **反向传播（Backpropagation）**
  - 核心：**链式法则（Chain Rule）**
  - 反向传播算法的步骤：
    1. 前向传播：输入数据，计算 $z^{[l]}, a^{[l]}$，以及损失函数 L
    2. 反向传播：使用链式法则计算参数 W 的梯度
    3. 参数更新

反向传播

- 计算参数 $W_{ji}^{[l]}$ 的梯度

$$\frac{\partial L}{\partial W_{ji}^{[l]}} = \frac{\partial L}{\partial z_j^{[l]}} \frac{\partial z_j^{[l]}}{\partial W_{ji}^{[l]}}$$

$$= \frac{\partial L}{\partial z_j^{[l]}} a_i^{[l-1]}$$

令 $\delta_j^{[l]} = \frac{\partial L}{\partial z_j^{[l]}}$ $\quad$ $\frac{\partial L}{\partial W_{ji}^{[l]}} = \delta_j^{[l]} a_i^{[l-1]}$

$a^{[0]} = x$

for l **in** $1..n$:

$z^{[l]} = W^{[l]} a^{[l-1]}$

$a^{[l]} = g^{[l]}\left(z^{[l]}\right)$

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

反向传播

$$\delta_j^{[l]} = \frac{\partial L}{\partial z_j^{[l]}}$$

$$\delta_j^{[n]} = \frac{\partial L}{\partial a_j^{[n]}} \frac{\partial a_j^{[n]}}{\partial z_j^{[n]}} = \frac{\partial L}{\partial a_j^{[n]}} g^{[n]\prime}\left(z_j^{[n]}\right)$$

$$\delta_j^{[l]} = \sum_k \frac{\partial L}{\partial z_k^{[l+1]}} \frac{\partial z_k^{[l+1]}}{\partial z_j^{[l]}} = \sum_k \delta_k^{[l+1]} W_{kj}^{[l+1]} g^{[l]\prime}\left(z_j^{[l]}\right)$$

（偏导的链式法则）

$a^{[0]} = x$

for l **in** $1..n$:

$z^{[l]} = W^{[l]} a^{[l-1]}$

$a^{[l]} = g^{[l]}\left(z^{[l]}\right)$

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

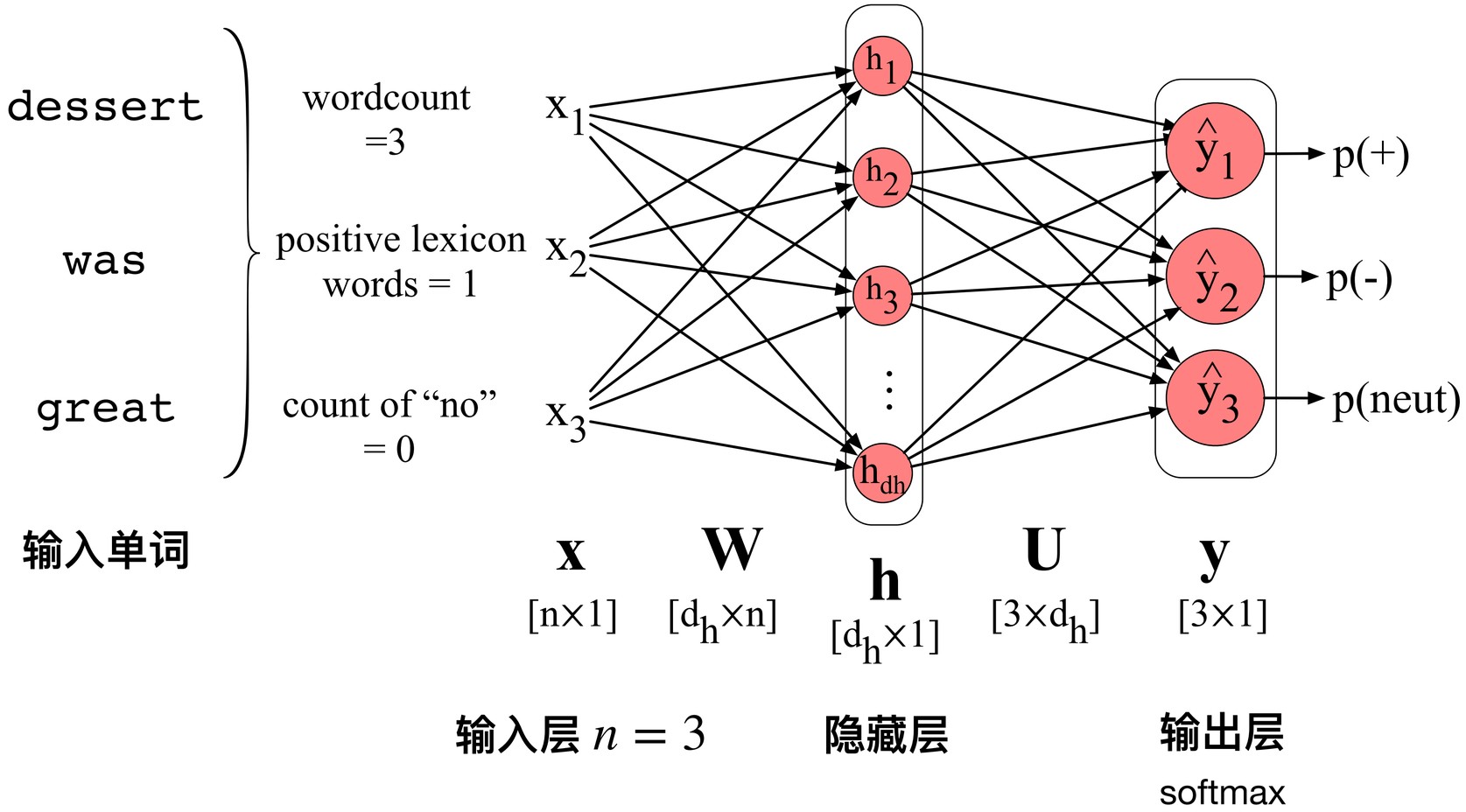
前馈神经网络的 NLP 应用

文本分类：情感分析

- 输入 x 为手动设计的特征，例如：

$$
\begin{aligned}
x_1 &\quad \text{count(positive lexicon)} \in \text{doc}) \\
x_2 &\quad \text{count(negative lexicon)} \in \text{doc}) \\
x_3 &\quad \begin{cases} 1 & \text{if } \text{“no”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases} \\
x_4 &\quad \text{count}(\text{1st and 2nd pronouns} \in \text{doc}) \\
x_5 &\quad \begin{cases} 1 & \text{if } \text{“!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases} \\
x_6 &\quad \log(\text{word count of doc})
\end{aligned}
$$

文本分类：情感分析

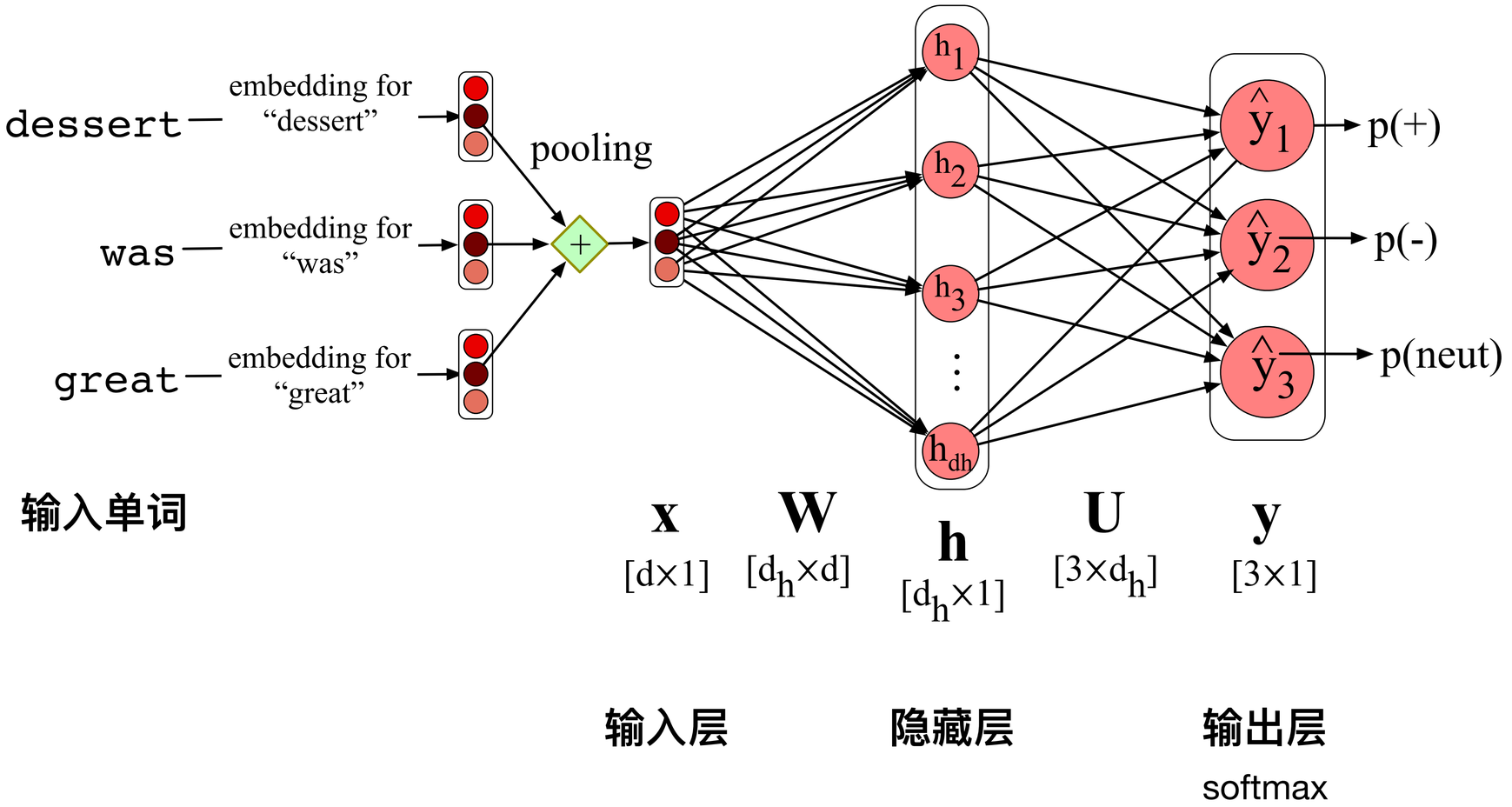

dessert
was
great
wordcount =3
positive lexicon words = 1
count of “no” = 0
x1
x2
x3
h1
h2
h3
hdh
ŷ1
ŷ2
ŷ3
p(+)
p(-)
p(neut)
输入单词
x
[n×1]
W
[dh×n]
h
[dh×1]
U
[3×dh]
y
[3×1]
输入层 n = 3
隐藏层
输出层
softmax

表征学习

- 类似于 word2vec 模型，从训练集中学习单词的表征：
  - 输入为 n 个单词的文本 $w_1, \cdots, w_n$
  - 将文本表示为 n 个词嵌入 $e(w_1), \cdots, e(w_n)$（神经网络的参数）
  - 平均之后作为神经网络的输入：

$$x_{\text{mean}} = \frac{1}{n} \sum_{i=1}^{n} e(w_i)$$

表征学习


dessert
embedding for "dessert"
was
embedding for "was"
great
embedding for "great"
pooling
+
h1
h2
h3
hdh
ŷ1
ŷ2
ŷ3
p(+)
p(-)
p(neut)
输入单词
x
[d×1]
W
[dh×d]
h
[dh×1]
U
[3×dh]
y
[3×1]
输入层
隐藏层
输出层
softmax

预训练

- $e(w_i)$ 可以随机初始化
- $e(w_i)$ 也可以初始化为 word2vec 或 GloVe 词嵌入
  - **预训练（Pre-training）**
    - 在大量数据上训练模型，学习通用特征
  - 是深度学习与大语言模型的核心思想

神经语言模型

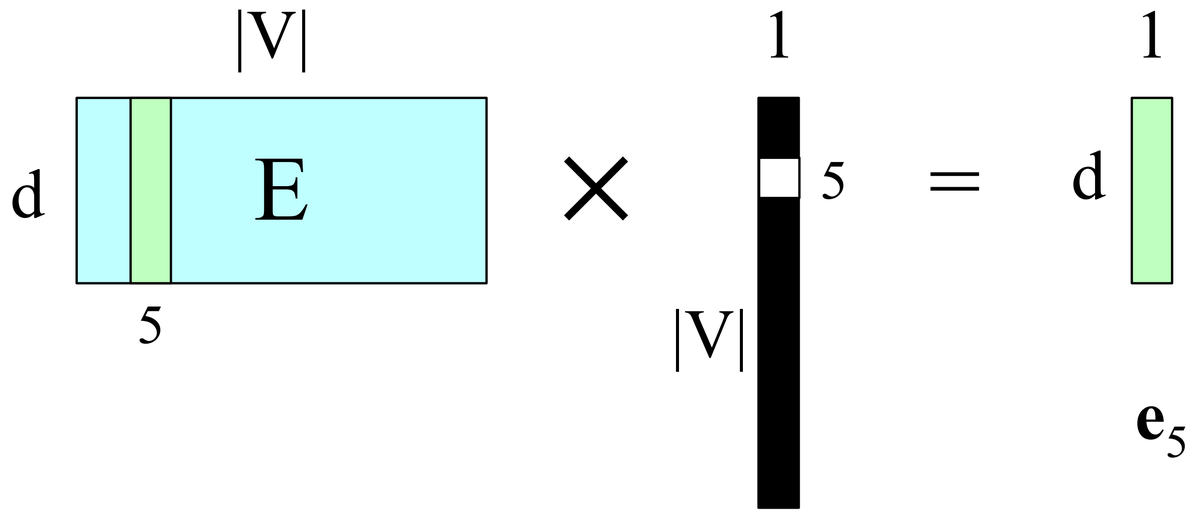
- **神经语言模型（Neural Language Model）**

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

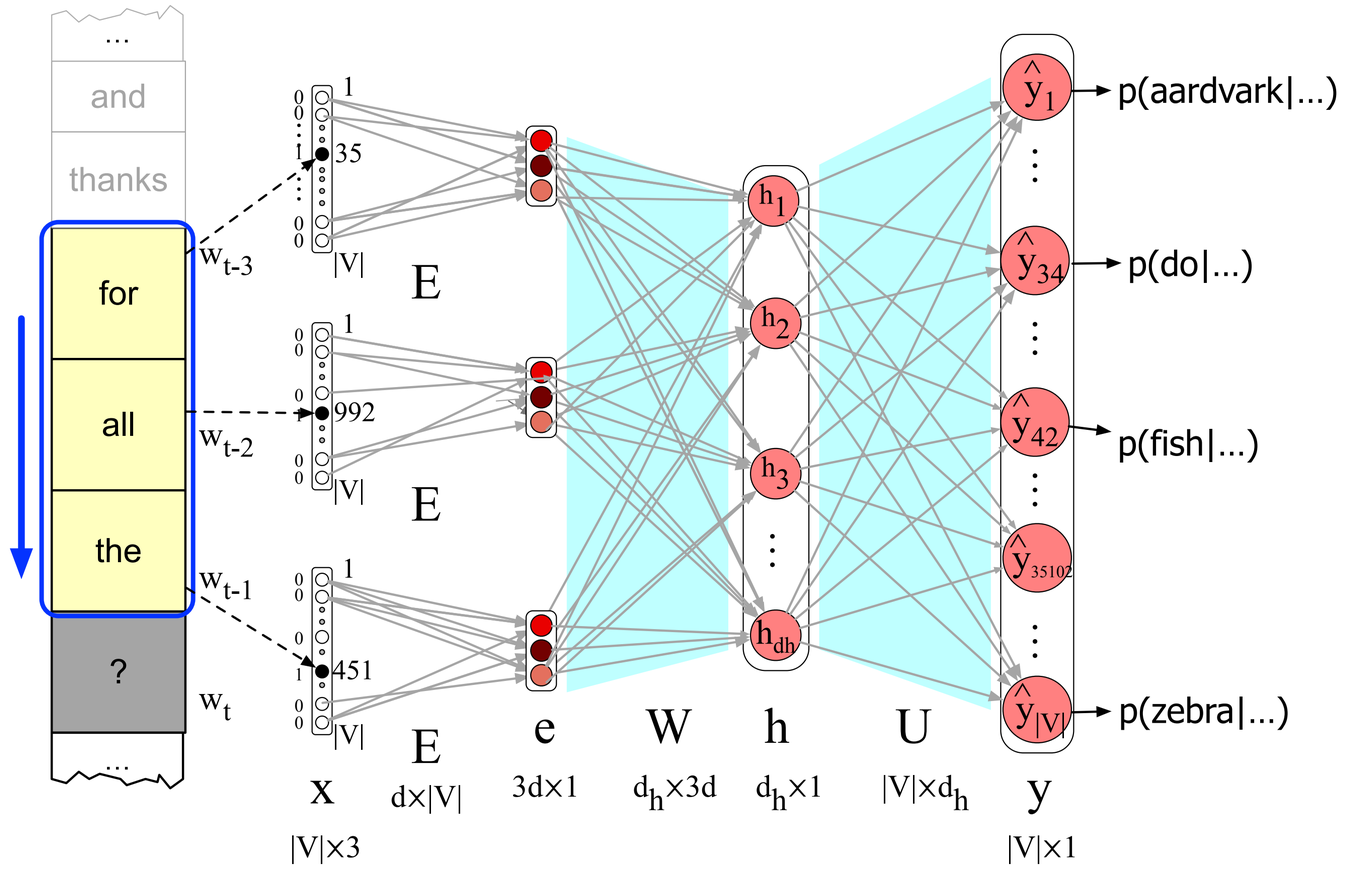
- 相较于 N 元语法模型：
  - 能处理更长的历史文本
  - 泛化能力更强（能处理相似单词）
  - 单词的预测能力更强

神经语言模型

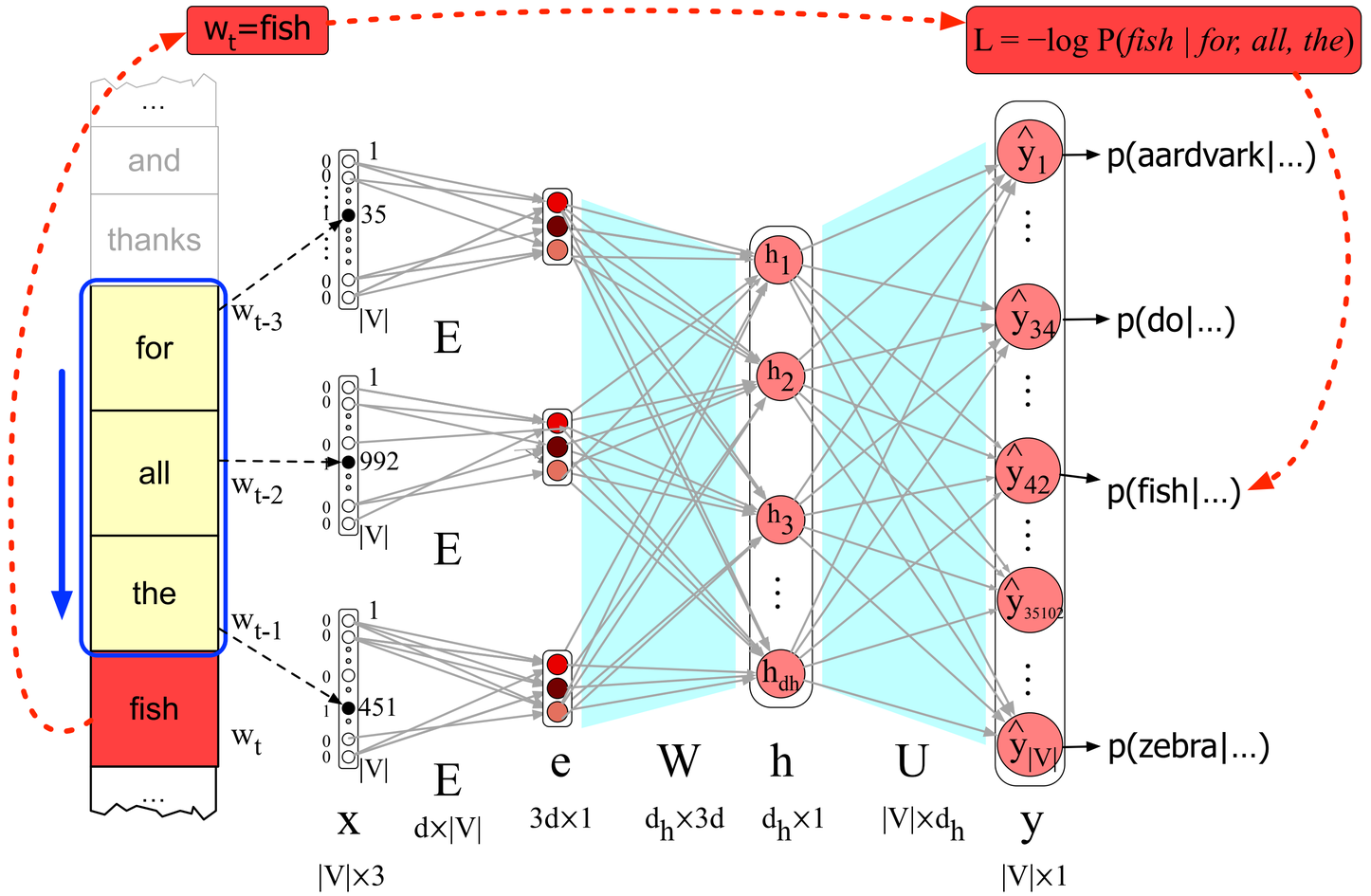
- 将前 $N-1$ 个单词表示为**独热向量（One-hot Vector）**
  - 向量中，单词的词典索引位置为 1，其他元素都是 0
- 将独热向量与嵌入矩阵 E 相乘，得到词的向量表示



神经语言模型


...
and
thanks
for
all
the
?
...
w_{t-3}
w_{t-2}
w_{t-1}
w_t
35
992
451
|V|
E
E
E
x
d×|V|
e
3d×1
W
d_h×3d
h
d_h×1
U
|V|×d_h
y
|V|×1
h_1
h_2
h_3
h_{dh}
$\hat{y}_1$
$\hat{y}_{34}$
$\hat{y}_{42}$
$\hat{y}_{35102}$
$\hat{y}_{|V|}$
p(aardvark|...)
p(do|...)
p(fish|...)
p(zebra|...)
|V|×3
输入层
独热向量
嵌入层
隐藏层
输出层
softmax

神经语言模型的训练



输入层 **嵌入层** **隐藏层** **输出层**

独热向量 softmax