第5章 语言模型

(2/3)

本章内容

5.1 传统语言模型

5.2 神经语言模型

5.3 文本表示

5.2 神经语言模型

- 5.2.1 问题的提出
- 5.2.2 前馈神经网络语言模型
- 5.2.3 循环神经网络语言模型

5.2.1 问题的提出

- 回顾n-gram

句子s的概率：$p(s)=\prod_{t=1}^{m} p(w_t \mid w_1 \cdots w_{t-1})$

$$=\prod_{t=1}^{m} p(w_t \mid w_{t-n+1}^{t-1})$$

$$p(w_t \mid w_{t-n+1}^{t-1}) = f(w_t \mid w_{t-n+1}^{t-1}) = \frac{c(w_{t-n+1}^{t})}{\sum_{w_t} c(w_{t-n+1}^{t})}$$

5.2.1 问题的提出

这 本 小说 很 枯燥 ， 读 起来 很 乏味 。

$$p(\text{枯燥}|\text{很}) = \frac{count(\text{很 枯燥})}{count(\text{很})} \qquad p(\text{乏味}|\text{很}) = \frac{count(\text{很 乏味})}{count(\text{很})}$$

问题①：数据稀疏

n-gram“很 乏味”有可能未出现在训练样本中。

问题②：忽略语义相似性

“枯燥”与“乏味”虽然语义相似，但无法共享信息。

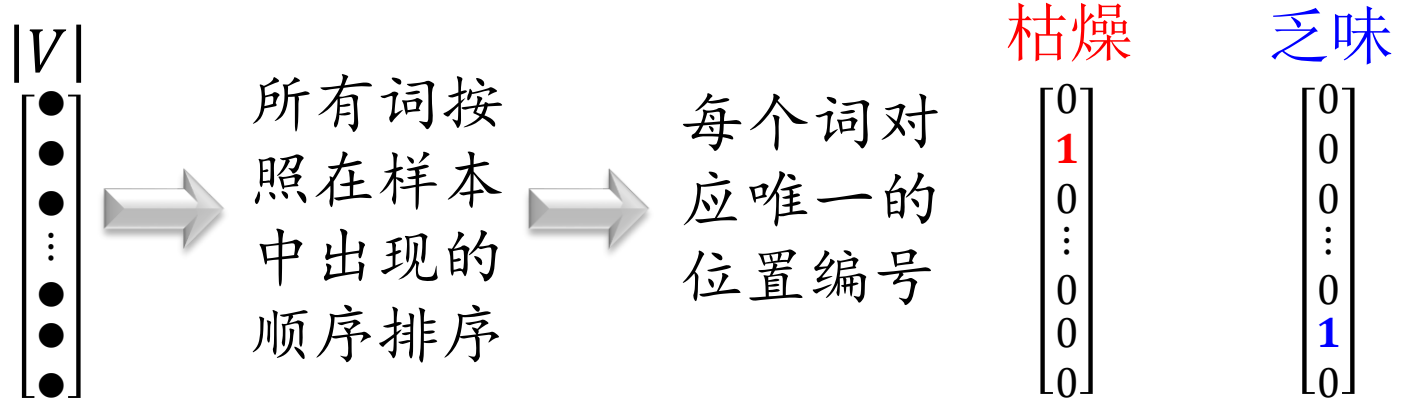
理想情况：同义替换后，$count$(很

5.2.1 问题的提出

- 分析原因

  “词”以词形本身表示，是离散的符号。

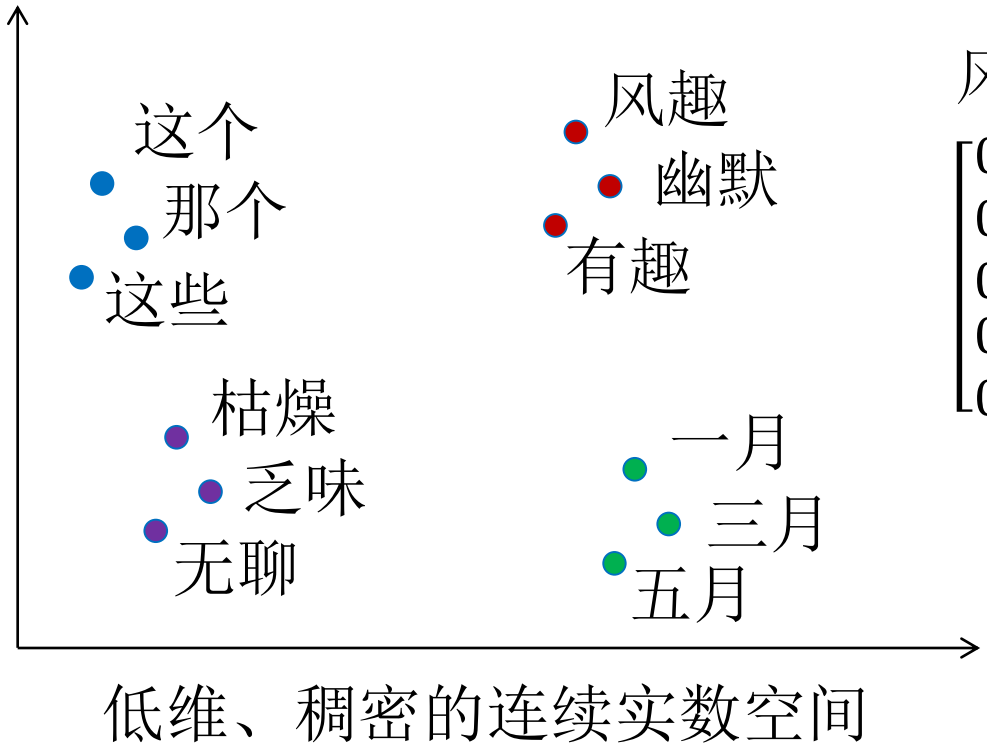
- 等价的表示：one-hot 向量

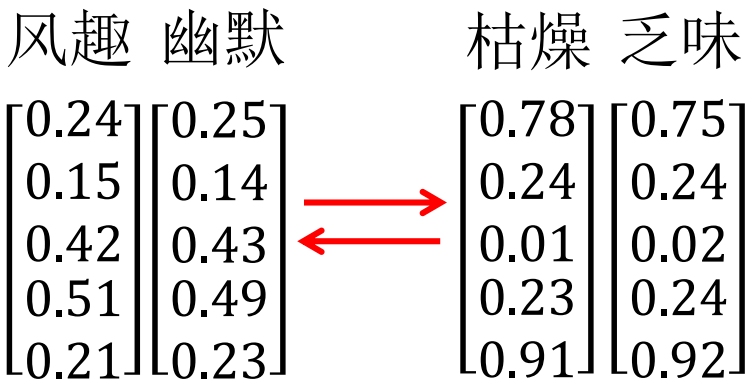


问题：
- 枯燥 ⊗ 乏味 = 0，任意两个词之间的相似度都为0。
- 维度太高！

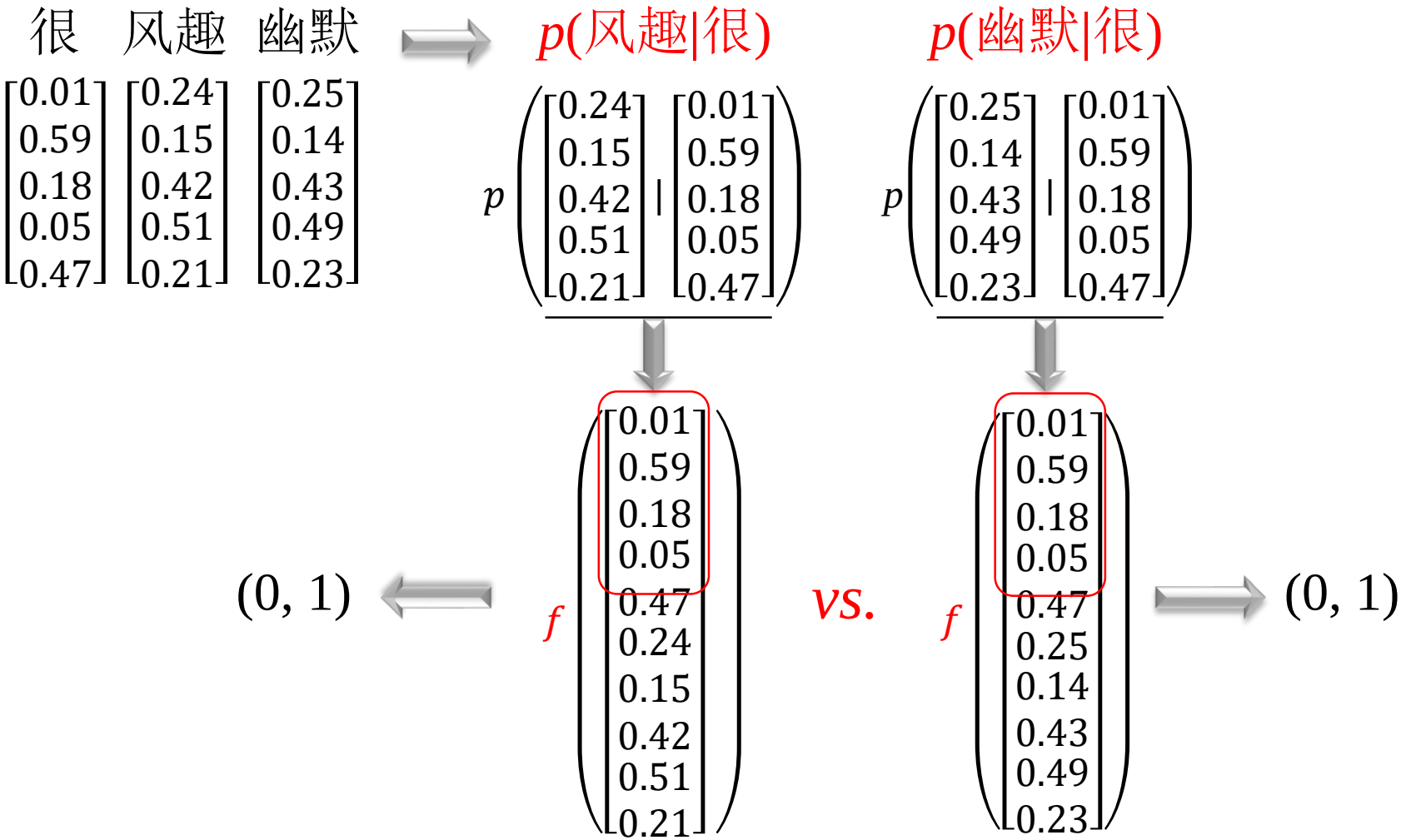
5.2.1 问题的提出

探索：是否可以用连续空间的分布式表示赋予词向量呢？





5.2.1 问题的提出


很
风趣
幽默
0.01
0.59
0.18
0.05
0.47
0.24
0.15
0.42
0.51
0.21
0.25
0.14
0.43
0.49
0.23
p(风趣|很)
p(幽默|很)
p
f
(0, 1)
vs.

5.2 神经语言模型

5.2.1 问题的提出

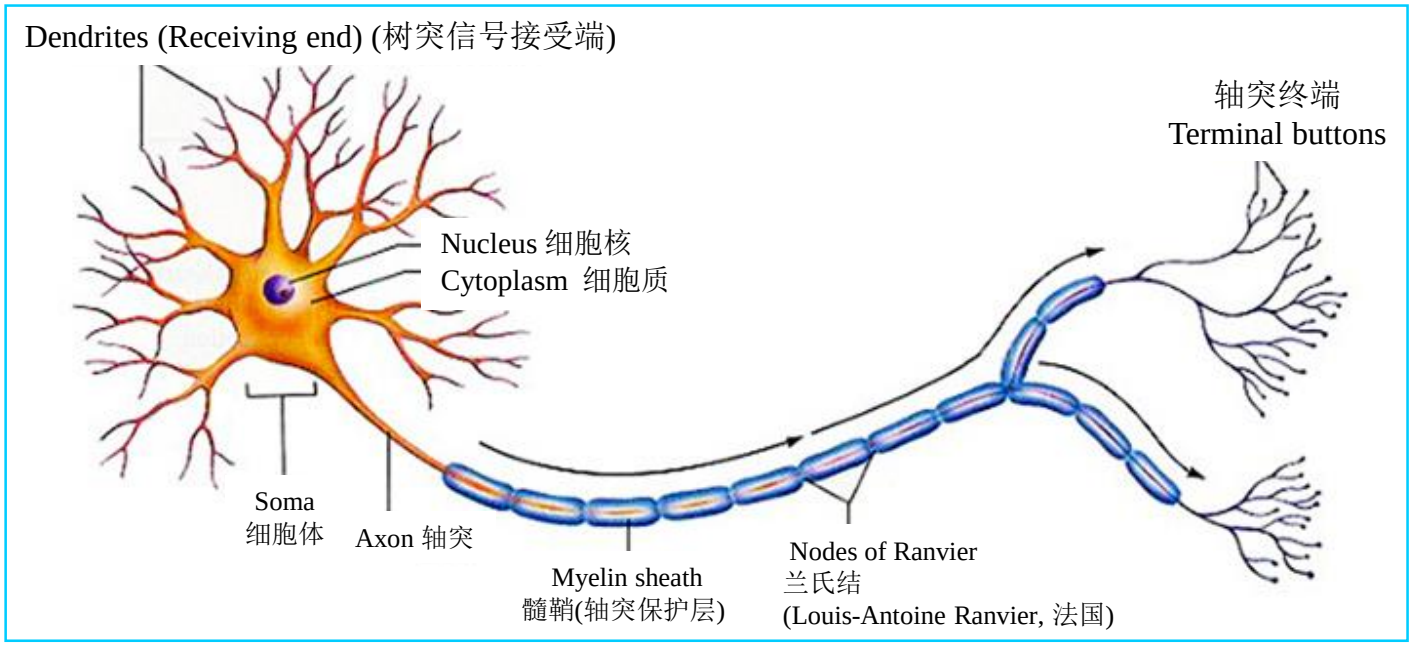
5.2.2 前馈神经网络语言模型

5.2.3 循环神经网络语言模型

5.2.2 前馈神经网络语言模型

◆人工神经网络 (artificial neural networks, ANN)

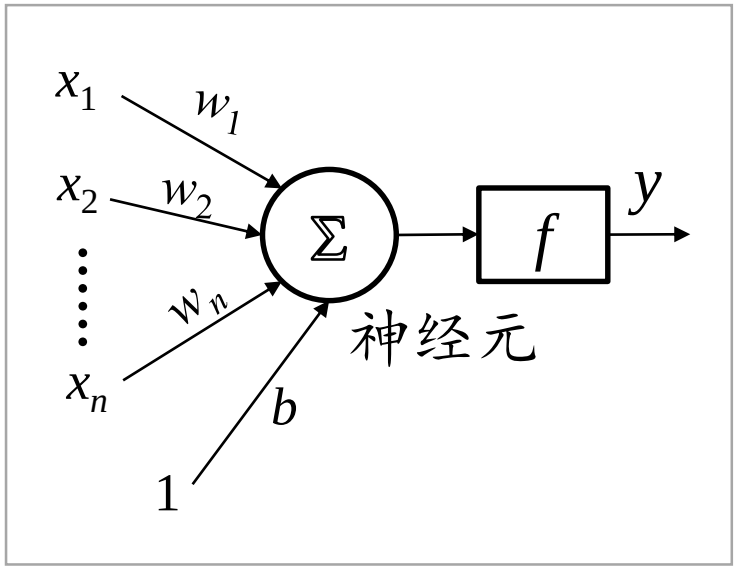
1943年心理学家 沃伦•麦卡洛克 和数理逻辑学家 W. Pitts 建立了神经网络的数学模型，提出了神经元的形式化数学描述和网络结构方法



5.2.2 前馈神经网络语言模型

- 神经元数学描述
  - $x_1 \sim x_n$ 为输入向量的各分量；
  - $w_1 \sim w_n$ 为权值系数(传递效率)；
  - b 为偏置；
  - f 为传递函数(激活函数)，
    通常是非线性的；
  - Σ是神经元的阈值；
  - y 为神经元的输出：$y = f(\vec{W} \bullet \vec{X}' + b)$

    $\vec{W}$ 为权值向量；$\vec{X}$ 为输入向量，$\vec{X}'$ 为 $\vec{X}$ 的转置。

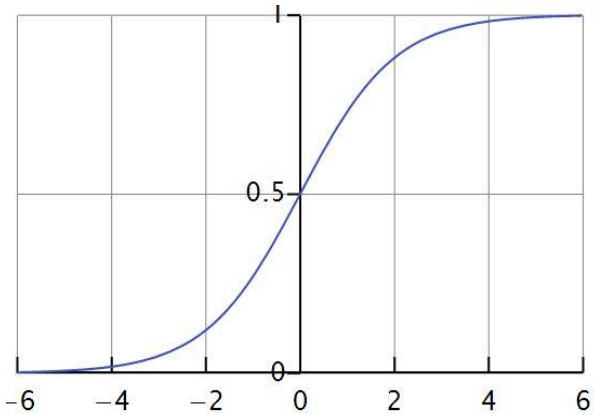


5.2.2 前馈神经网络语言模型

常用的激活函数：Sigmoid 型函数

①Logistic 函数：

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



Logistic 函数可以看成是一个“挤压”函数，把一个实数域的输入“挤压”到(0, 1)。当输入值在0附近时，似为线性函数；当输入值靠近两端时，对输入进行抑制。输入越小，越接近于0；输入越大，越接近于1。这种特点与生物神经元类似，对某些输入会产生兴奋（输出为1），对另一些输入产生抑制（输出为0）。

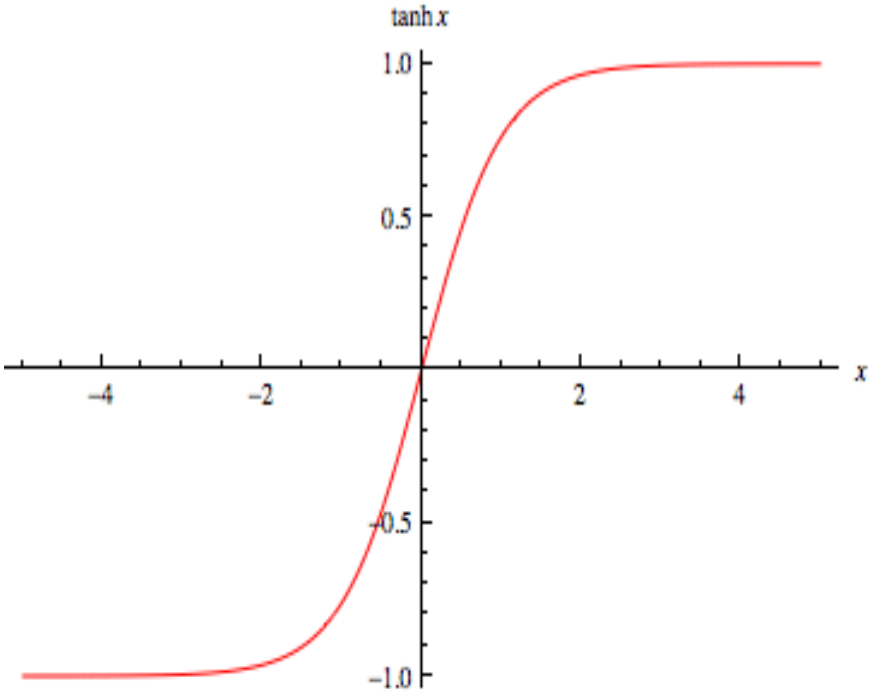
5.2.2 前馈神经网络语言模型

②Tanh 函数

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

或者：

$$\tanh(x) = 2\sigma(2x) - 1$$



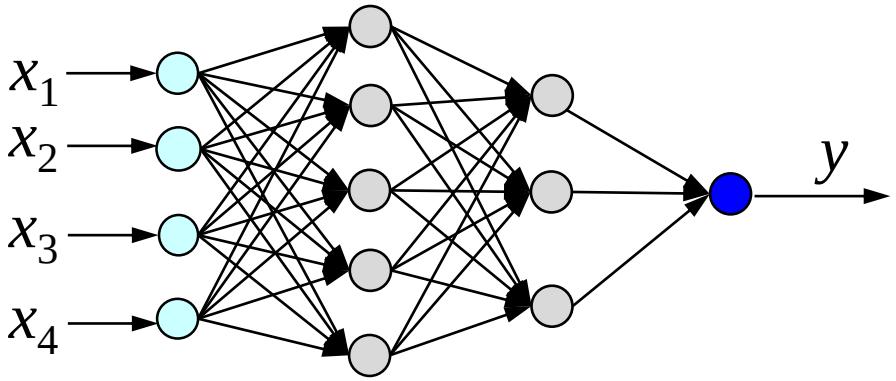
Tanh 函数可以看作是放大并平移的Logistic 函数，其值域是$(-1, 1)$。

5.2.2 前馈神经网络语言模型

◆前馈神经网络(Feedforward Neural Network，FNN)

FNN是最早发明的简单人工神经网络，也经常被称为多层感知器(Multi-Layer Perceptron, MLP)。

每一层中的神经元接收前一层神经元的输出，并输出到下一层神经元，整个网络中的信息朝一个方向传播，可以看作是一个有向的无环图。



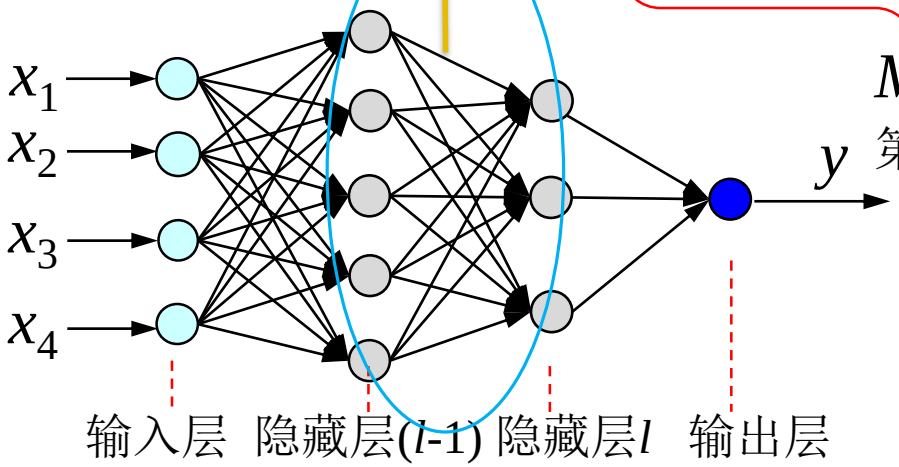
5.2.2 前馈神经网络语言模型

- **网络表示**：L 为神经网络的层数；$f_l(\bullet)$ 为第 l 层神经元的激活函数；

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \mathrm{M} \\ x_l \end{bmatrix} \quad \boldsymbol{W}^{(l)} = \begin{bmatrix} w_{11} & w_{21} & & w_{(l-1)1} \\ w_{12} & w_{22} & \mathrm{M} & w_{(l-1)2} \\ \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M} \\ w_{1M_l} & w_{2M_l} & & w_{(l-1)M_l} \end{bmatrix} \in \mathbb{R}^{M_l \times M_{l-1}}$$

M_l：第 l 层神经元个数

M_{l-1}：第 $l-1$ 层神经元个数



$$\boldsymbol{b}^{(l)} = \begin{bmatrix} b_1 \\ b_2 \\ \mathrm{M} \\ b_l \end{bmatrix}$$

5.2.2 前馈神经网络语言模型

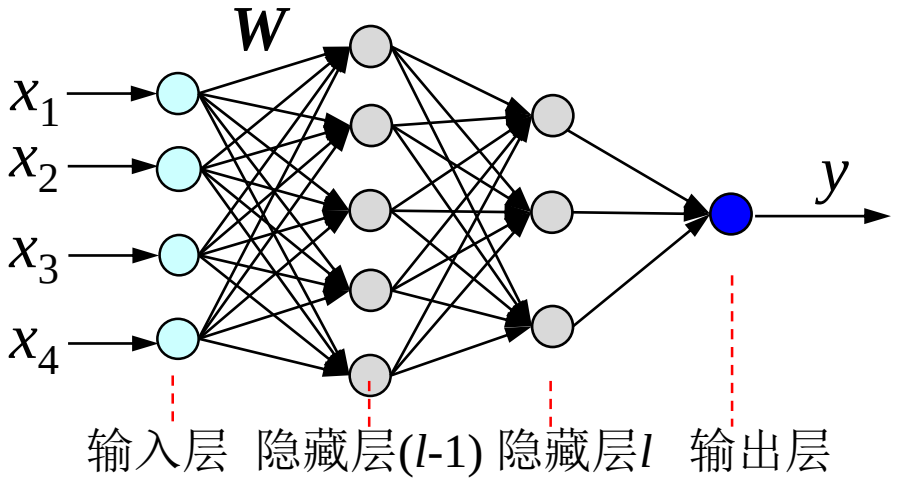
令 $\boldsymbol{\alpha}^{(0)} = \boldsymbol{x}$，前馈神经网络通过迭代进行信息传播：

$$\boldsymbol{z}^{(l)}=\boldsymbol{W}^{(l)}\cdot\boldsymbol{\alpha}^{(l-1)} + \boldsymbol{b}^{(l)}$$

$$\boldsymbol{\alpha}^{(l)}=f_l(\boldsymbol{z}^{(l)})$$ 激活

上述两个式子可以合并为：

$$\boldsymbol{\alpha}^{(l)}=f_l(\boldsymbol{W}^{(l)}\cdot\boldsymbol{\alpha}^{(l-1)} + \boldsymbol{b}^{(l)})$$



整个网络可以看作一个复合函数$\phi(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b})$，将向量$\boldsymbol{x}$ 作为第 1 层的输入$\boldsymbol{\alpha}^{(0)}$，将第$L$ 层的输出$\boldsymbol{\alpha}^{(L)}$ 作为整个函数的输出：

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

5.2.2 前馈神经网络语言模型

- **参数学习**：确定网络中所有的 $\boldsymbol{W}$ 和 $\boldsymbol{b}$

如果采用交叉熵作为损失函数：

$$\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\boldsymbol{y} \log \hat{\boldsymbol{y}}$$

给定训练集 $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)})\}_{n=1}^{N}$，结构化风险函数为：

$$\mathcal{R}(\boldsymbol{W}, \boldsymbol{b}) = \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}(\boldsymbol{y}^{(n)}, \hat{\boldsymbol{y}}^{(n)}) + \frac{1}{2} \lambda \| \boldsymbol{W} \|_F^2 \qquad (5.2\text{-}2)$$

$\boldsymbol{W}$ 和 $\boldsymbol{b}$ 分别表示网络中所有的权重矩阵和偏置向量。$\| \boldsymbol{W} \|_F^2$ 是正则化项，用以防止过拟合。$0<\lambda<1$ 为超参数

5.2.2 前馈神经网络语言模型

$\|\boldsymbol{W}\|_F^2$ 一般采用 Frobenius 范数(F-范数)：

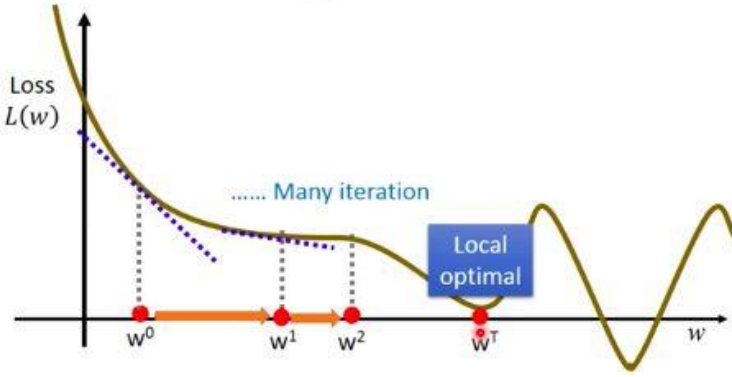
$$\|\boldsymbol{W}\|_F^2 = \sum_{l=1}^{L}\sum_{i=1}^{M_l}\sum_{j=1}^{M_{l-1}}\left(w_{ij}^{(l)}\right)^2 \qquad (5.2\text{-}3)$$

网络参数可以通过随机梯度下降法(Stochastic Gradient Descent Algorithm)进行学习。每次迭代中，第 l 层的参数 $\boldsymbol{W}^{(l)}$ 和 $\boldsymbol{b}^{(l)}$ 参数更新方式为：

$$\boldsymbol{W}^{(l)} \leftarrow \boldsymbol{W}^{(l)} - \eta\frac{\partial \mathcal{R}(\boldsymbol{W},\boldsymbol{b})}{\partial \boldsymbol{W}^{(l)}}$$

$$\mathcal{R}(\boldsymbol{W},\boldsymbol{b}) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}(\boldsymbol{y}^{(n)},\hat{\boldsymbol{y}}^{(n)}) + \frac{1}{2}\lambda\|\boldsymbol{W}\|_F^2$$

$$= \boldsymbol{W}^{(l)} - \eta\left(\frac{1}{N}\sum_{n=1}^{N}\left(\frac{\partial \mathcal{L}(\boldsymbol{y}^{(n)},\hat{\boldsymbol{y}}^{(n)})}{\partial \boldsymbol{W}^{(l)}}\right) + \lambda\boldsymbol{W}^{(l)}\right)$$



η为学习率， $0<\eta<1$。

5.2.2 前馈神经网络语言模型

同理，

$$\boldsymbol{b}^{(l)} \leftarrow \boldsymbol{b}^{(l)} - \eta \frac{\partial \mathcal{R}(\boldsymbol{W}, \boldsymbol{b})}{\partial \boldsymbol{b}^{(l)}}$$

$$= \boldsymbol{b}^{(l)} - \eta \left(\frac{1}{N} \sum_{n=1}^{N} \frac{\partial \mathcal{L}(\boldsymbol{y}^{(n)}, \hat{\boldsymbol{y}}^{(n)})}{\partial \boldsymbol{b}^{(l)}} \right) \quad (5.2\text{-}5)$$

在神经网络的训练中通常使用反向传播(Back Propagation, BP)算法高效地计算梯度。

5.2.2 前馈神经网络语言模型

- 基于反向传播算法的随机梯度下降参数训练过程

输入: 训练集 $\mathcal{D}=\{(\boldsymbol{x}^{(n)},\boldsymbol{y}^{(n)})\}_{n=1}^{N}$，验证集 $\mathcal{V}$，学习率 η，正则化系数 λ，网络层数 L，神经元数量 M_l，$1\le l\le L$。

输出： $\boldsymbol{W},\boldsymbol{b}$

重复执行该过程：

(1) 对训练集 $\mathcal{D}$ 中的样本随机重排序；

(2) for n=1 .. N do

正向

从训练集 $\mathcal{D}$ 中选取样本 $(\boldsymbol{x}^{(n)},\boldsymbol{y}^{(n)})$；

前馈计算每一层的净输入 $\boldsymbol{z}^{(l)}$ 和激活值 $\boldsymbol{\alpha}^{(l)}$，直到最后一层；

反向传播计算每一层的误差 $\delta^{(l)}$；

// 计算每一层参数的导数：

$$\begin{cases}\forall l,\ \dfrac{\partial \mathcal{L}(\boldsymbol{y},\hat{\boldsymbol{y}})}{\partial \boldsymbol{W}^{(l)}}=\delta^{(l)}(\boldsymbol{\alpha}^{(l-1)})^{\mathrm{T}}\\ \forall l,\ \dfrac{\partial \mathcal{L}(\boldsymbol{y},\hat{\boldsymbol{y}})}{\partial \boldsymbol{b}^{(l)}}=\delta^{(l)}\end{cases}$$

// 更新参数：

$$\begin{cases}\boldsymbol{W}^{(l)}\leftarrow \boldsymbol{W}^{(l)}-\eta(\delta^{(l)}\cdot(\boldsymbol{\alpha}^{(l-1)})^{\mathrm{T}}+\lambda\boldsymbol{W}^{(l)});\\ \boldsymbol{b}^{(l)}\leftarrow \boldsymbol{b}^{(l)}-\eta\delta^{(l)};\end{cases}$$

end

直到模型在验证集 $\mathcal{V}$ 上的损失函数值收敛，结束过程，输出 $\boldsymbol{W},\boldsymbol{b}$。

5.2.2 前馈神经网络语言模型

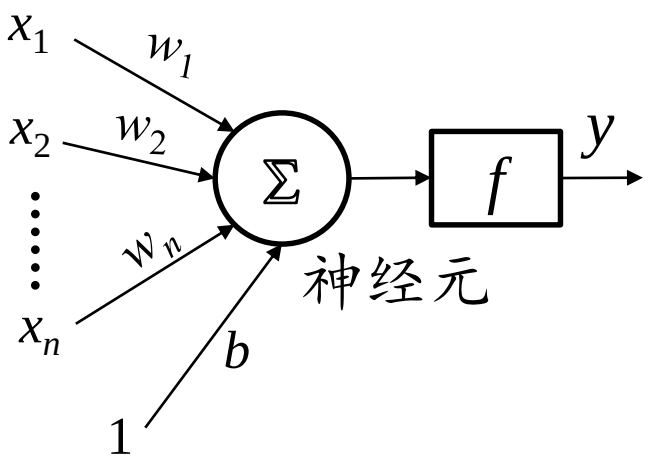
◆基于 FNN 的语言模型

$$p(\text{风趣}|\text{很}) = f\left(\begin{bmatrix}0.01\\0.59\\0.18\\0.05\\0.47\\0.24\\0.15\\0.42\\0.51\\0.21\end{bmatrix}\right) \overset{?}{\Rightarrow} (0, 1)$$

$$p(\text{幽默}|\text{很}) = f\left(\begin{bmatrix}0.01\\0.59\\0.18\\0.05\\0.47\\0.25\\0.14\\0.43\\0.49\\0.23\end{bmatrix}\right) \overset{?}{\Rightarrow} (0, 1)$$

语言模型目标：

给定前1~n个词，预测n+1个词出现的概率



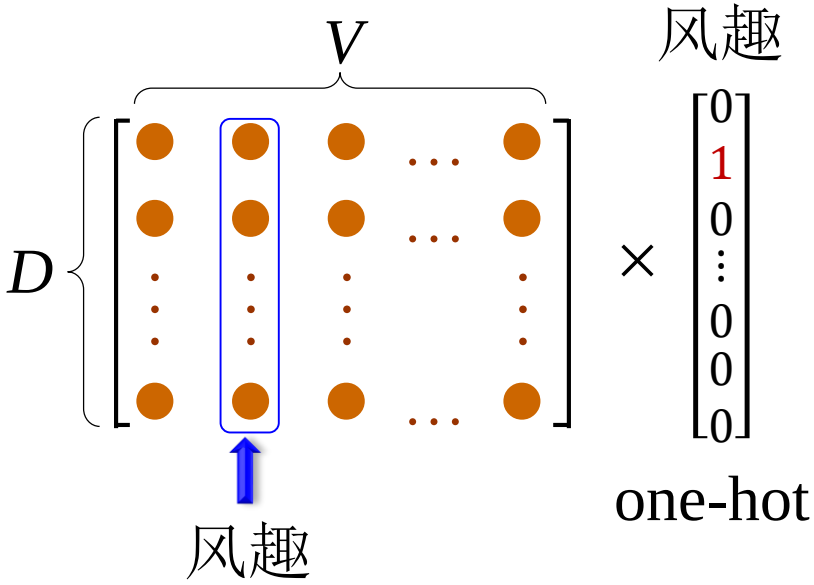
词向量？

$W, b = ?$

5.2.2 前馈神经网络语言模型

- 词向量表示

Look-up table(L_T):

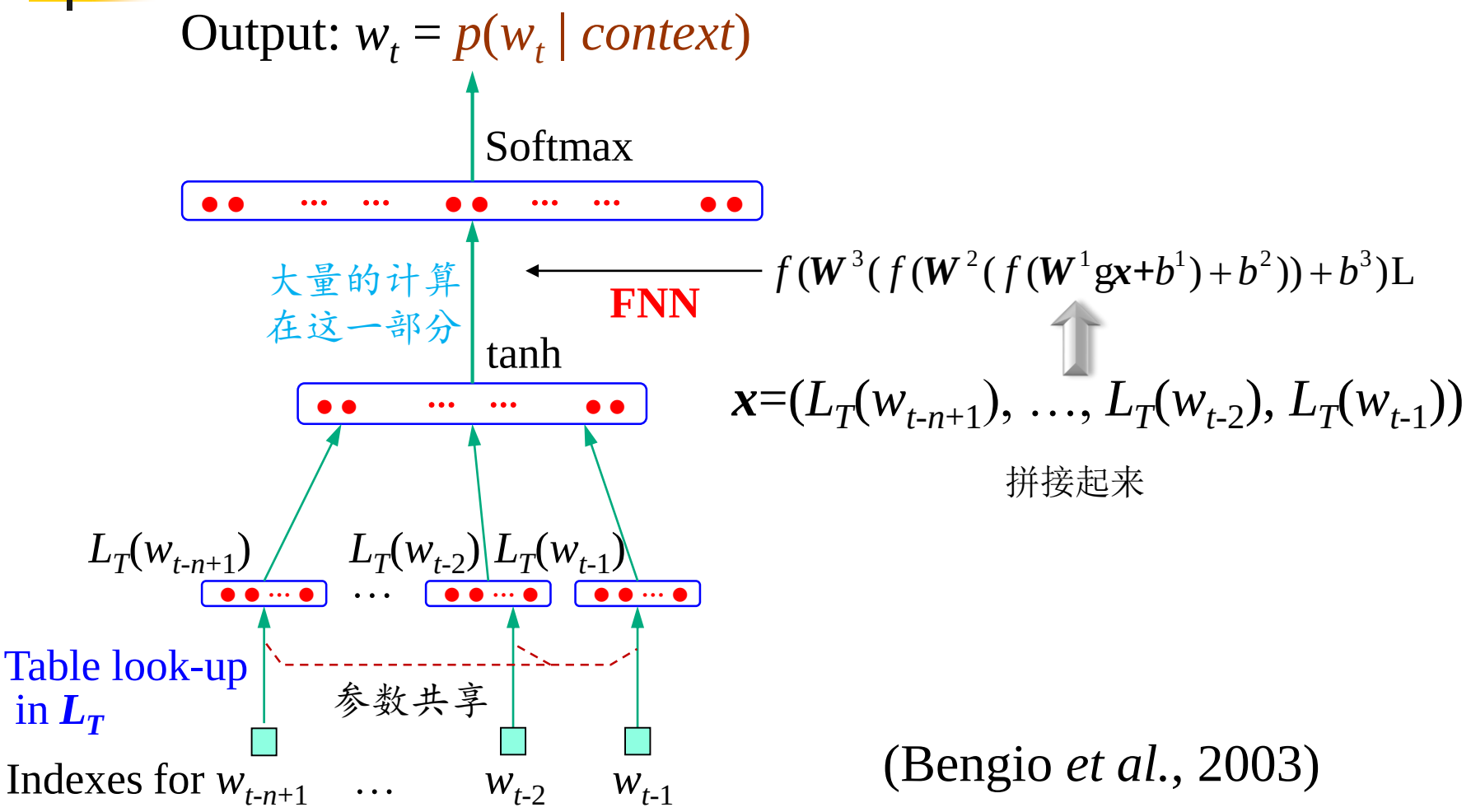


如何学习L_T？

下一章解决

- ***V*(词表大小）**：(1)训练数据中所有词；(2)频率高于某个阈值的所有词；(3)前*V*个频率最高的词。
- ***D*(向量维数)**：超参数，人工设定，一般从几十到几百。

5.2.2 前馈神经网络语言模型



(Bengio *et al.*, 2003)

5.2.2 前馈神经网络语言模型

- Softmax 回归 (regression)

Softmax 回归也称多类(Multi-Class)Logistic 回归，是Logistic回归在多分类问题上的推广。

对于多类问题，类别标签 $y \in \{1, 2, \cdots, C\}$可以有$C$个取值，给定一个样本$\boldsymbol{x}$, Softmax 回归预测的属于类别$c$ 的条件概率为:

$$p(y=c \mid \boldsymbol{x}) = \text{Softmax}(\boldsymbol{w}_c^{\mathrm{T}} \boldsymbol{x})$$

$$= \frac{\exp(\boldsymbol{w}_c^{\mathrm{T}} \boldsymbol{x})}{\sum_{c'=1}^{C} \exp(\boldsymbol{w}_{c'}^{\mathrm{T}} \boldsymbol{x})}$$

其中，$\boldsymbol{w}_c$ 是第 c 类的权重向量。

5.2.2 前馈神经网络语言模型

- Softmax 回归 (regression)

softmax函数可以把神经网络的输出值转换为概率分布，同时放大输出值的影响力

例如：神经网络的原输出为 $\begin{bmatrix} 8 \\ 5 \\ 0 \end{bmatrix}$

$$\begin{cases} e^{z_1} = e^8 = 2981.0 \\ e^{z_2} = e^5 = 148.4 \\ e^{z_3} = e^0 = 1.0 \end{cases}$$

softmax函数转换后：

$$\begin{cases} \sigma(\vec{z})1 = \frac{2981.0}{3130.4} = 0.9523 \\ \sigma(\vec{z})2 = \frac{148.4}{3130.4} = 0.0474 \\ \sigma(\vec{z})3 = \frac{1.0}{3130.4} = 0.0003 \end{cases}$$

5.2.2 前馈神经网络语言模型

- 举例说明

$$L_T = \begin{bmatrix} \ldots\ 0.1\ \ldots\ 0.5\ \ldots\ 0.3\ \ldots\ 0.4\ \ldots\ 0.2\ \ldots \\ \ldots\ 0.3\ \ldots\ 0.4\ \ldots\ 0.2\ \ldots\ 0.2\ \ldots\ 0.1\ \ldots \end{bmatrix}$$

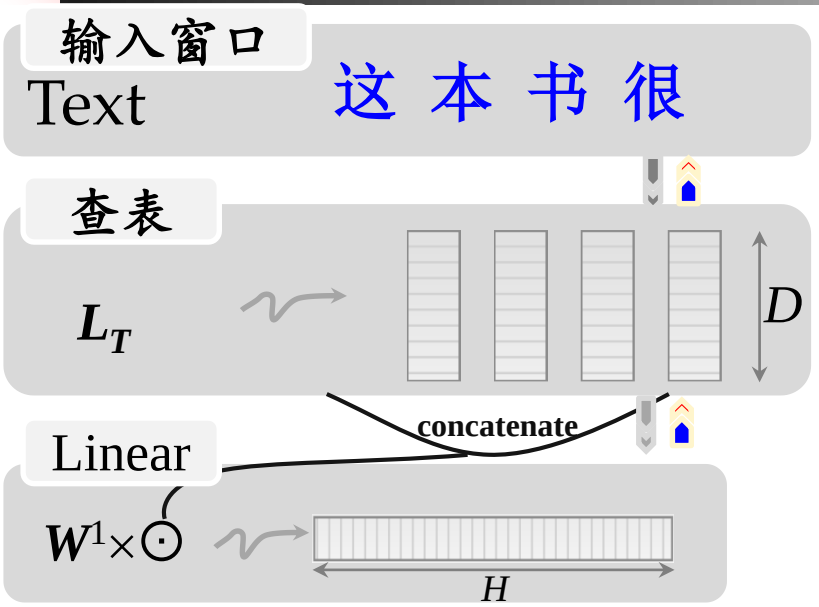
本 … 很…乏味…书 … 这…

2维 × 5000个

输入词串：这 本 书 很 乏味？

$\Longrightarrow$ p(乏味 | 这, 本, 书, 很)

5.2.2 前馈神经网络语言模型



p(乏味 | 这, 本, 书, 很)

①查词表

0.2	0.1	0.4	0.5
0.1	0.3	0.2	0.4
↑	↑	↑	↑
这	本	书	很

②拼接所有的条件词的向量，形成一个向量。

这 本 书 很

→ $(0.2, 0.1, 0.1, 0.3, 0.4, 0.2, 0.5, 0.4)^{\mathrm{T}}$

或者加和，取平均

③隐藏层：线性映射+非线性变换。

$$\boldsymbol{W}^1 = \begin{pmatrix} 0.1 & 0 & 0.2 & 0.4 & 0.2 & 0.1 & 0 & 0.3 \\ 0.5 & 0.4 & 0.2 & 0 & 0.2 & 0.6 & 0 & 0.2 \end{pmatrix} \times \boldsymbol{x} = \begin{pmatrix} 0.2 \\ 0.1 \\ 0.1 \\ 0.3 \\ 0.4 \\ 0.2 \\ 0.5 \\ 0.4 \end{pmatrix}$$

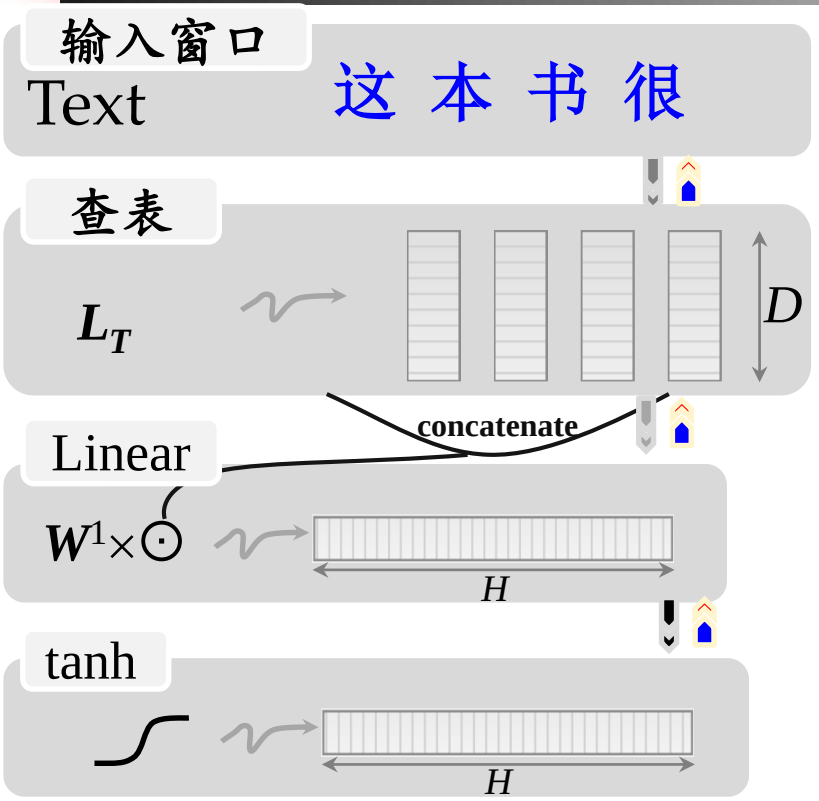
任设的初始值(0~1之间，均匀或高斯分布)

$$\boldsymbol{W}^1 \times \boldsymbol{x} \Rightarrow \begin{pmatrix} 0.38 \\ 0.44 \end{pmatrix}$$

暂不考虑偏置b

得到一个预测的2维向量

5.2.2 前馈神经网络语言模型



p(乏味 | 这, 本, 书, 很)

①查词表

0.2	0.1	0.4	0.5
0.1	0.3	0.2	0.4
↑	↑	↑	↑
这	本	书	很

②拼接所有的条件词的向量，形成一个向量。

这 本 书 很

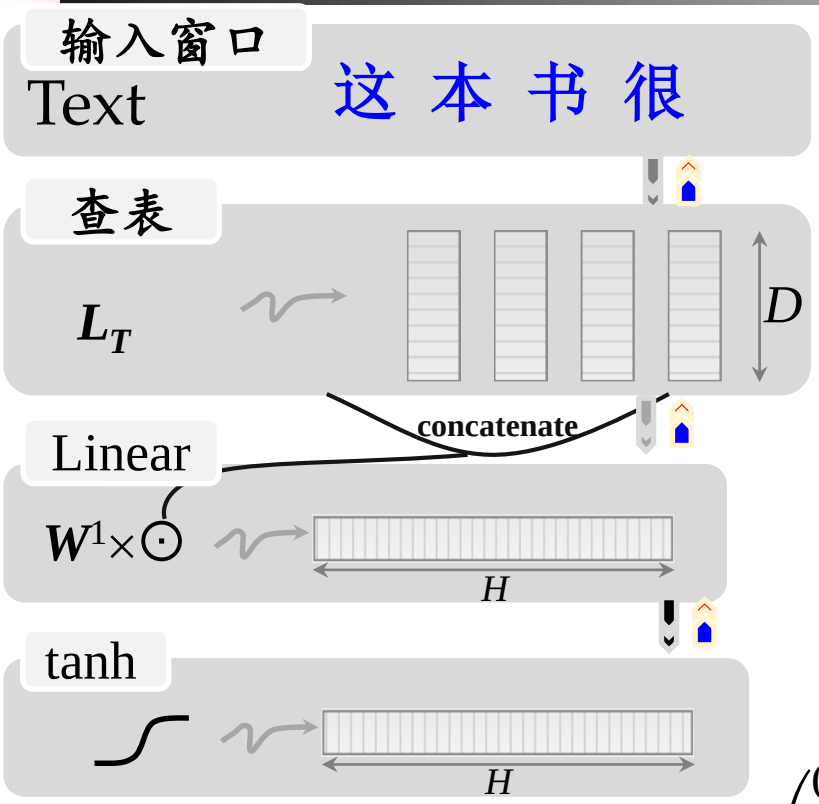
或者加和，取平均

→ $(0.2, 0.1, 0.1, 0.3, 0.4, 0.2, 0.5, 0.4)^{\mathrm{T}}$

③隐藏层：线性映射+非线性变换。

$$\boldsymbol{W}^1 \times \boldsymbol{x} \rightarrow \begin{pmatrix} 0.38 \\ 0.44 \end{pmatrix} \xrightarrow{\tanh(\bullet)} \begin{pmatrix} 0.36 \\ 0.41 \end{pmatrix}$$

5.2.2 前馈神经网络语言模型



p(乏味 | 这, 本, 书, 很)

①查词表

0.2	0.1	0.4	0.5
0.1	0.3	0.2	0.4
↑	↑	↑	↑
这	本	书	很

②拼接所有的条件词的向量，形成一个向量。

这 本 书 很

或者加和，取平均

→ $(0.2, 0.1, 0.1, 0.3, 0.4, 0.2, 0.5, 0.4)^{\mathrm{T}}$

③隐藏层：线性映射+非线性变换。

$$\boldsymbol{h}=\begin{pmatrix}0.36\\0.41\end{pmatrix}\Longrightarrow \boldsymbol{L_T}^{\mathrm{T}}\times\boldsymbol{h}=\begin{pmatrix}0.2&0.1\\0.1&0.3\\0.4&0.2\\0.5&0.4\\0.3&0.2\\\cdots&\cdots\end{pmatrix}\times\begin{pmatrix}0.36\\0.41\end{pmatrix}=\begin{pmatrix}0.113\\0.159\\0.226\\0.344\\0.190\\\cdots\end{pmatrix}$$

乏味

相当于查表

Softmax(•)

5.2.2 前馈神经网络语言模型

◆问题分析

- 仅对小窗口的历史信息进行建模

n-gram语言模型仅考虑前面*n*-1个词的历史信息：

$$p(s)=\prod_{t=1}^{m} p(w_t \mid w_1 \cdots w_{t-1})$$

$$=\prod_{t=1}^{m} p(w_t \mid w_{t-n+1}^{t-1})$$

能否对所有的历史信息进行建模?

5.2 神经语言模型

5.2.1 问题的提出

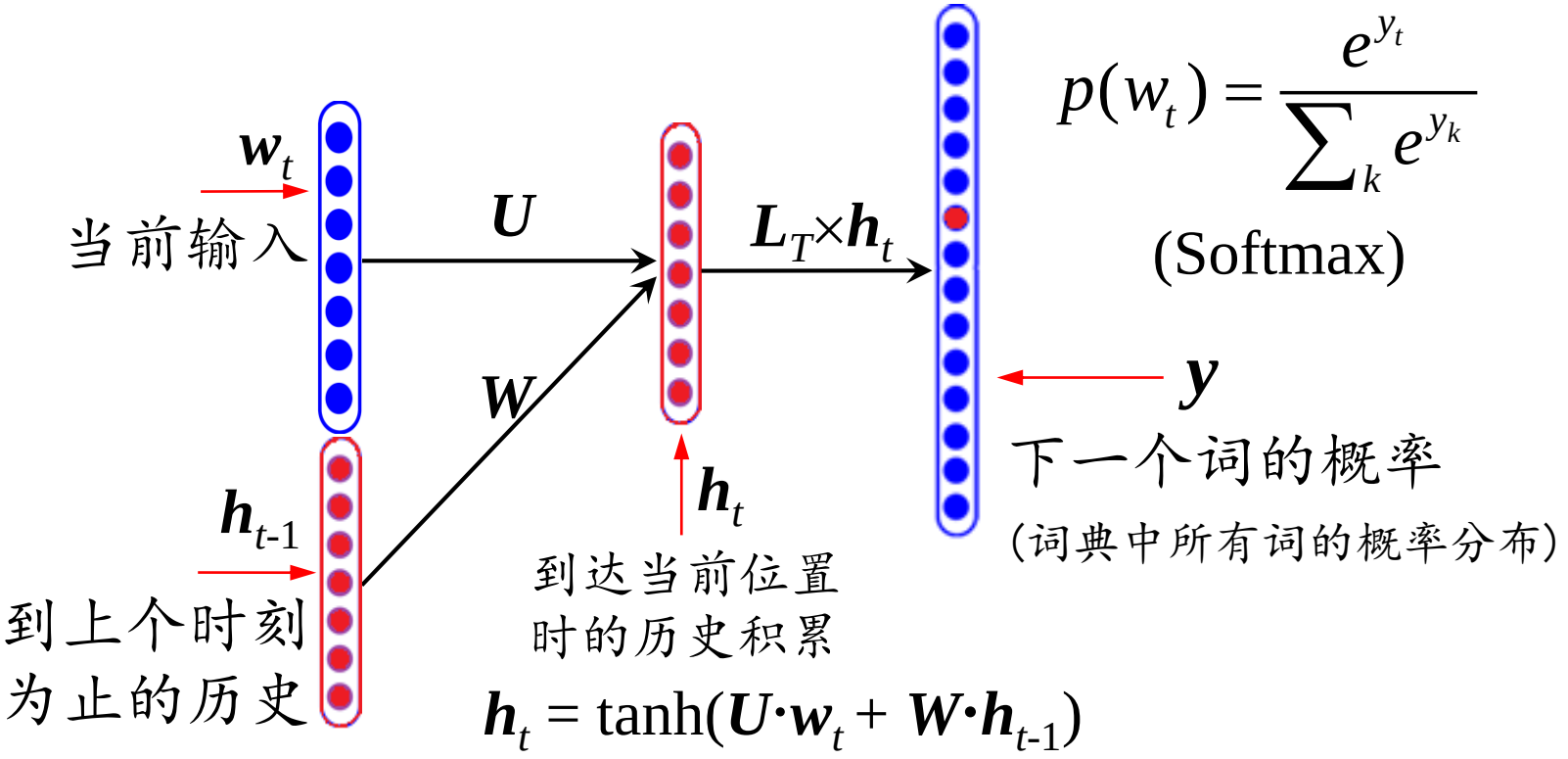
5.2.2 前馈神经网络语言模型

5.2.3 循环神经网络语言模型

5.2.3 循环神经网络语言模型

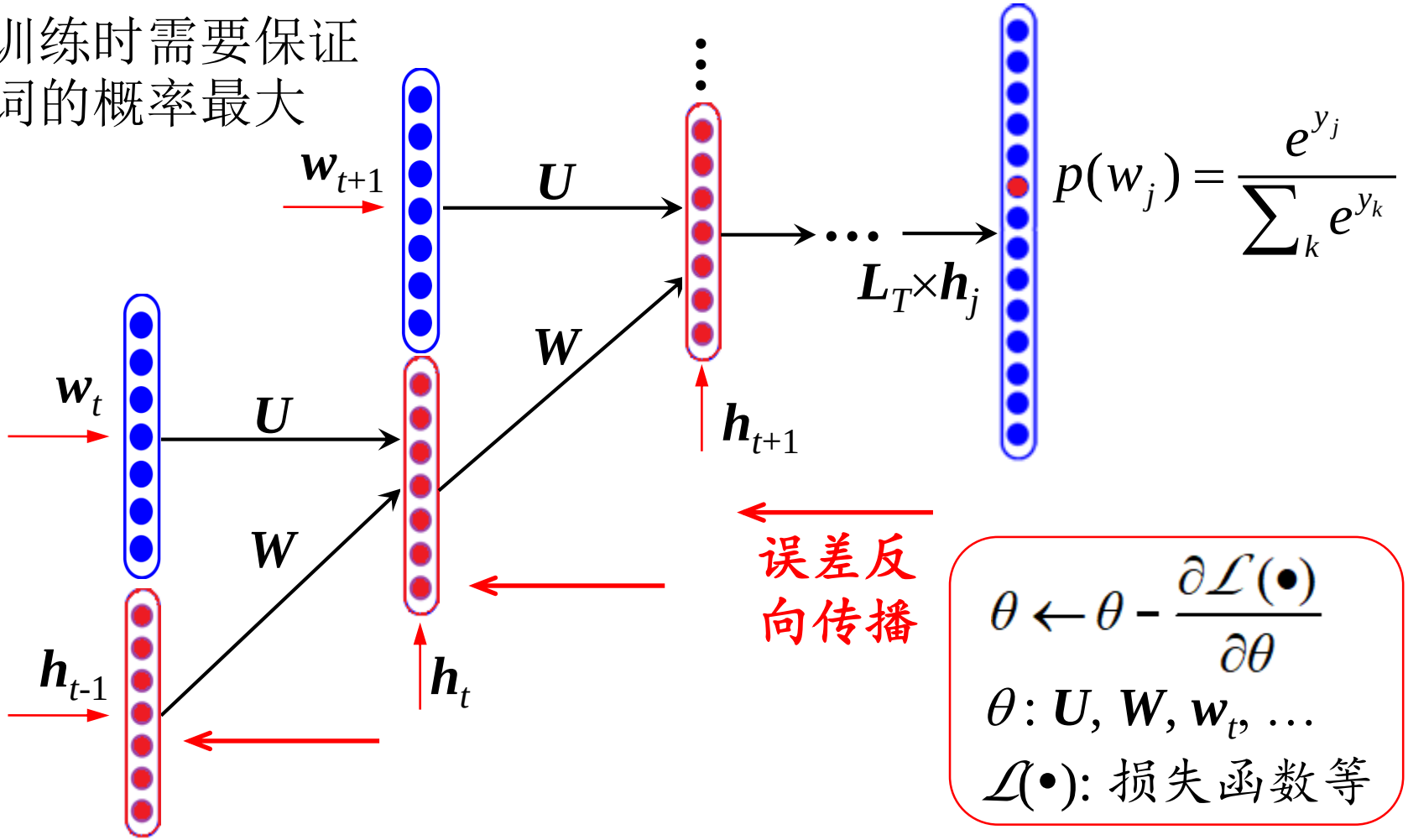
◆循环神经网络

- 输入: 从开始到 t-1 时刻的历史 $\boldsymbol{h}_{t-1}$; 当前位置t 的词向量 $\boldsymbol{w}_t$;
- 输出: 到 t 位置时的历史积累 $\boldsymbol{h}_t$ 及其该位置上词的概率。

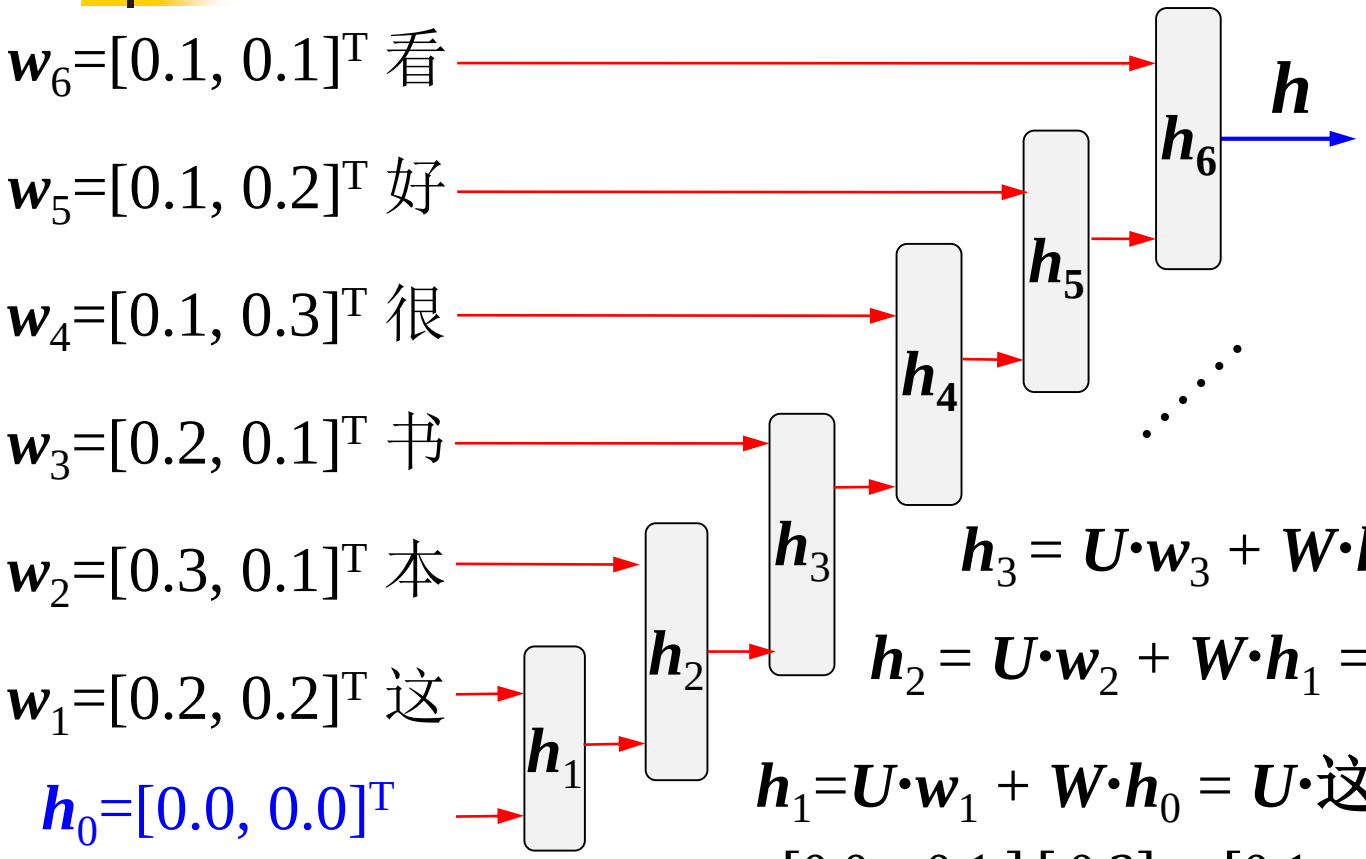


5.2.3 循环神经网络语言模型

模型训练时需要保证
当前词的概率最大



5.2.3 循环神经网络语言模型



$$U = \begin{bmatrix} 0.0 & 0.1 \\ 0.1 & 0.0 \end{bmatrix}$$

$$W = \begin{bmatrix} 0.1 & 0.0 \\ 0.0 & 0.1 \end{bmatrix}$$

$$h_3 = U \cdot w_3 + W \cdot h_2 = U \cdot 书 + W \cdot h_2$$

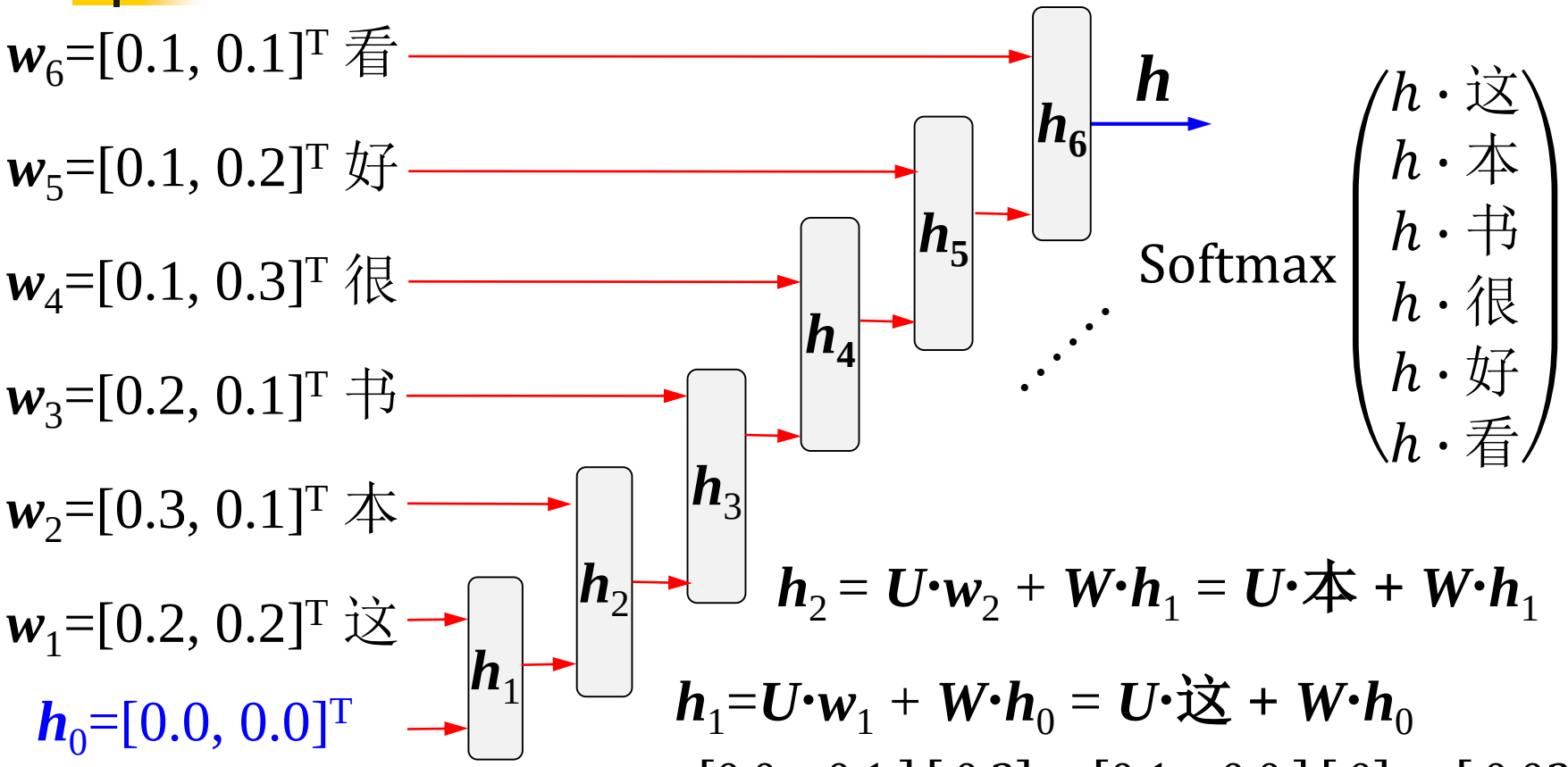
$$h_2 = U \cdot w_2 + W \cdot h_1 = U \cdot 本 + W \cdot h_1$$

$$h_1 = U \cdot w_1 + W \cdot h_0 = U \cdot 这 + W \cdot h_0$$

$$= \begin{bmatrix} 0.0 & 0.1 \\ 0.1 & 0.0 \end{bmatrix} \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix} + \begin{bmatrix} 0.1 & 0.0 \\ 0.0 & 0.1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.02 \\ 0.02 \end{bmatrix}$$

$$h_t = \tanh(U \cdot w_t + W \cdot h_{t-1})$$

5.2.3 循环神经网络语言模型



$h_2 = U \cdot w_2 + W \cdot h_1 = U \cdot$本 $+ W \cdot h_1$

$h_1 = U \cdot w_1 + W \cdot h_0 = U \cdot$这 $+ W \cdot h_0$

$h_t = \tanh(U \cdot w_t + W \cdot h_{t-1})$

$$= \begin{bmatrix} 0.0 & 0.1 \\ 0.1 & 0.0 \end{bmatrix} \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix} + \begin{bmatrix} 0.1 & 0.0 \\ 0.0 & 0.1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.02 \\ 0.02 \end{bmatrix}$$

5.2.3 循环神经网络语言模型

◆问题分析

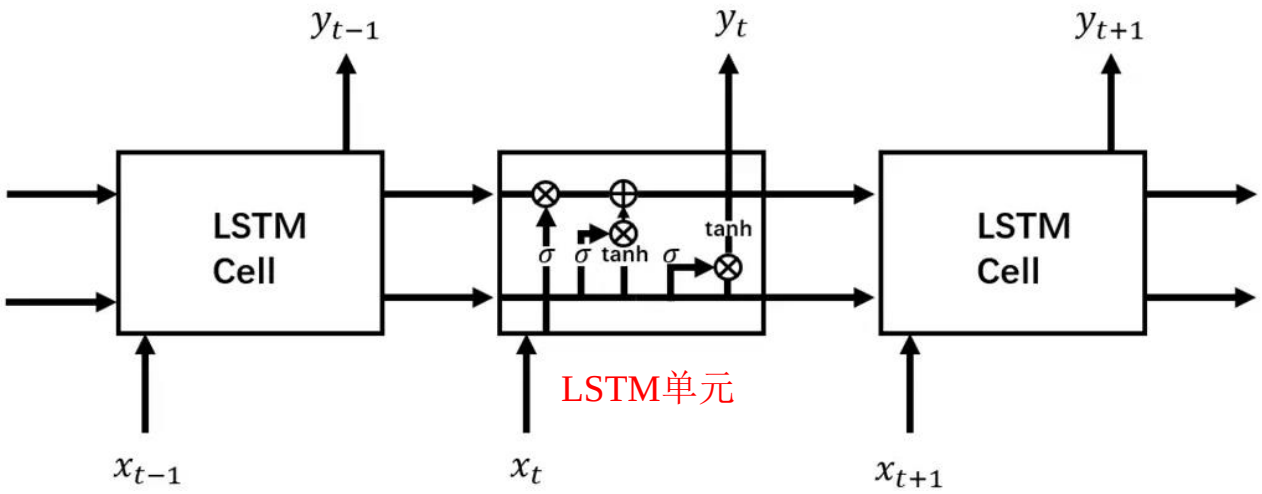
- **梯度消失或爆炸：**参数$\boldsymbol{W}$经过多次传递后有可能导致梯度消失(小于1时)或者爆炸(大于1)。

是否能够通过某种策略选择性地保留或者遗忘某些信息？

长短时记忆网络LSTM (Long-Short Term Memory)。

5.2.3 循环神经网络语言模型

◆LSTM 网络



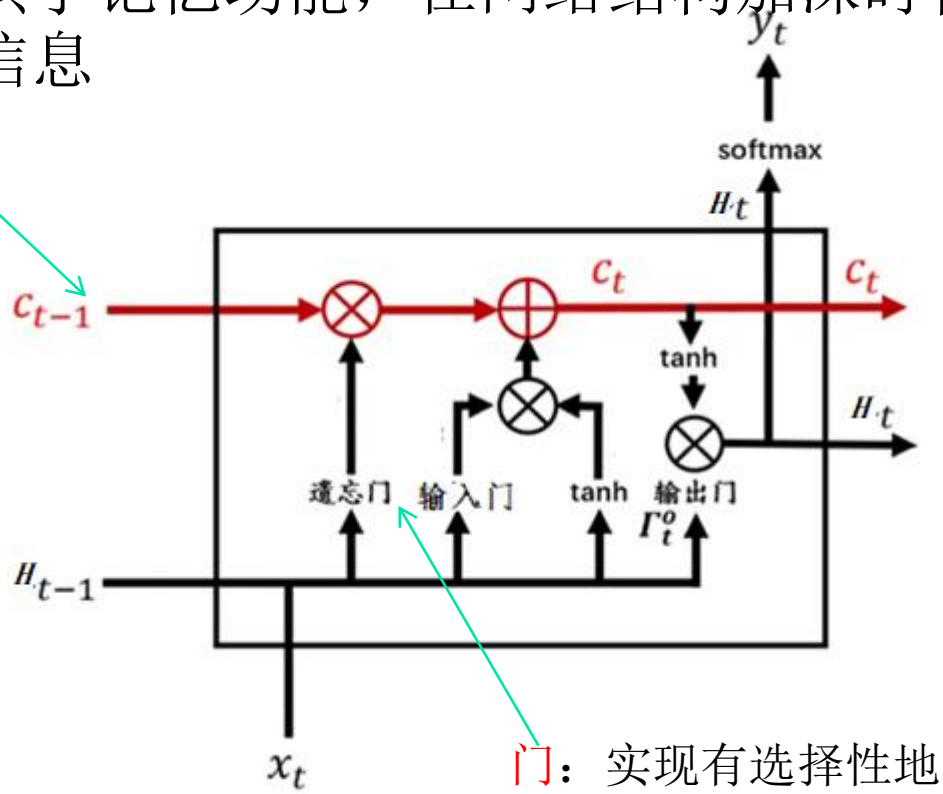

```
The cat, which already ate a bunch of food, was full.
 |   |     |      |     |  |   |   |   |     |   |
t0  t1    t2     t3    t4 t5  t6  t7  t8    t9 t10
```

LSTM是RNN的变体，可以解决RNN中梯度消失问题，从而可以处理long-term sequences。

5.2.3 循环神经网络语言模型

◆LSTM 网络结构

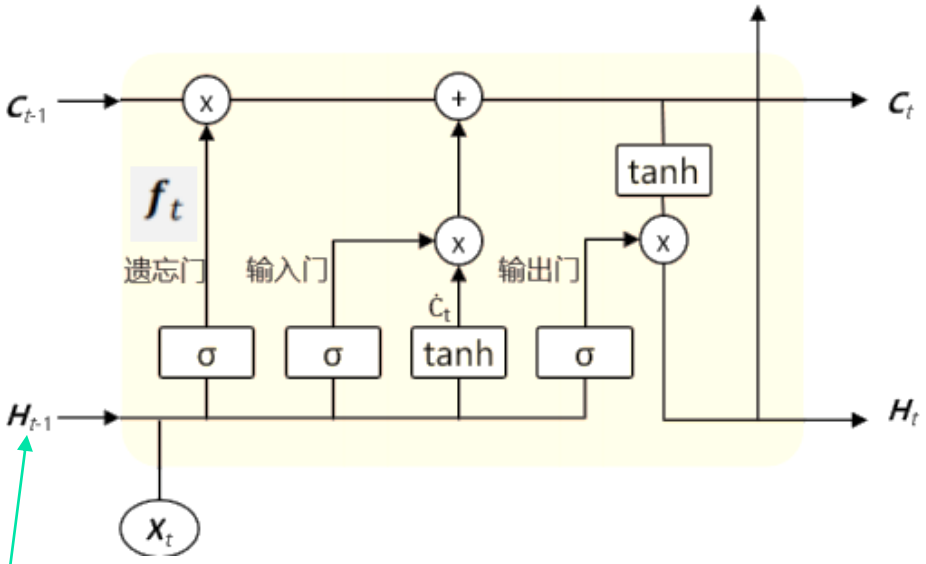
细胞状态：提供了记忆功能，在网络结构加深时仍能传递前后层的网络信息



门：实现有选择性地让信息通过

5.2.3 循环神经网络语言模型

- 遗忘门：



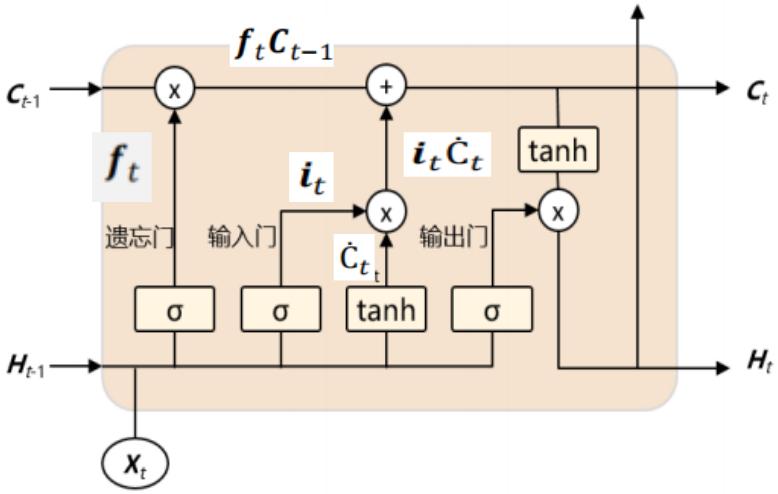
遗忘门控制上一步细胞信息$\boldsymbol{C}_{t-1}$的保存或丢弃。其策略是从上一个隐藏状态$\boldsymbol{H}_{t-1}$和当前输入$\boldsymbol{X}_t$的信息经过sigmoid函数计算，输出范围为[0, 1]。0意味完全遗忘，1意味完全保留。计算公式如下:

$$\boldsymbol{f}_t = \text{sigmoid}(\boldsymbol{W}_f \boldsymbol{X}_t + \boldsymbol{W}_{\text{hf}} \boldsymbol{H}_{t-1} + \boldsymbol{b}_f)$$

$\boldsymbol{f}_t \boldsymbol{C}_{t-1}$ 表示保留下的细胞信息

5.2.3 循环神经网络语言模型

- 输入门：



输入门控制哪些新的信息需要增加到细胞状态中。分为两步，首先是输入门决定哪些值需要更新，然后tanh层创建一个新的C_t值（候选细胞）。

$$i_t = \text{sigmoid}(W_i X_t + W_{\text{hi}} H_{t-1} + b_i)$$

$$\dot{C}_t = \tanh(W_C X_t + W_{hC} H_{t-1} + b_C)$$

当前单元输入加上前一单元状态

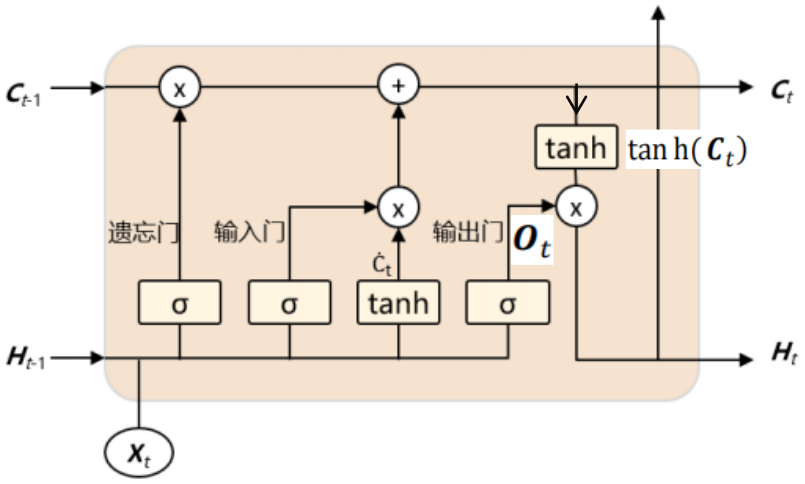
$i_t \dot{C}_t$ 表示更新后的细胞状态

结合遗忘门、输入门、上一个单元记忆细胞状态共同决定和更新当前细胞状态

$$C_t = f_t C_{t-1} + i_t \dot{C}_t$$

5.2.3 循环神经网络语言模型

- 输出门：



输出基于当前的细胞状态的过滤版本。首先通过sigmoid函数计算输出门，决定哪些细胞状态需要输出。然后通过tanh计算细胞状态，乘以输出门。

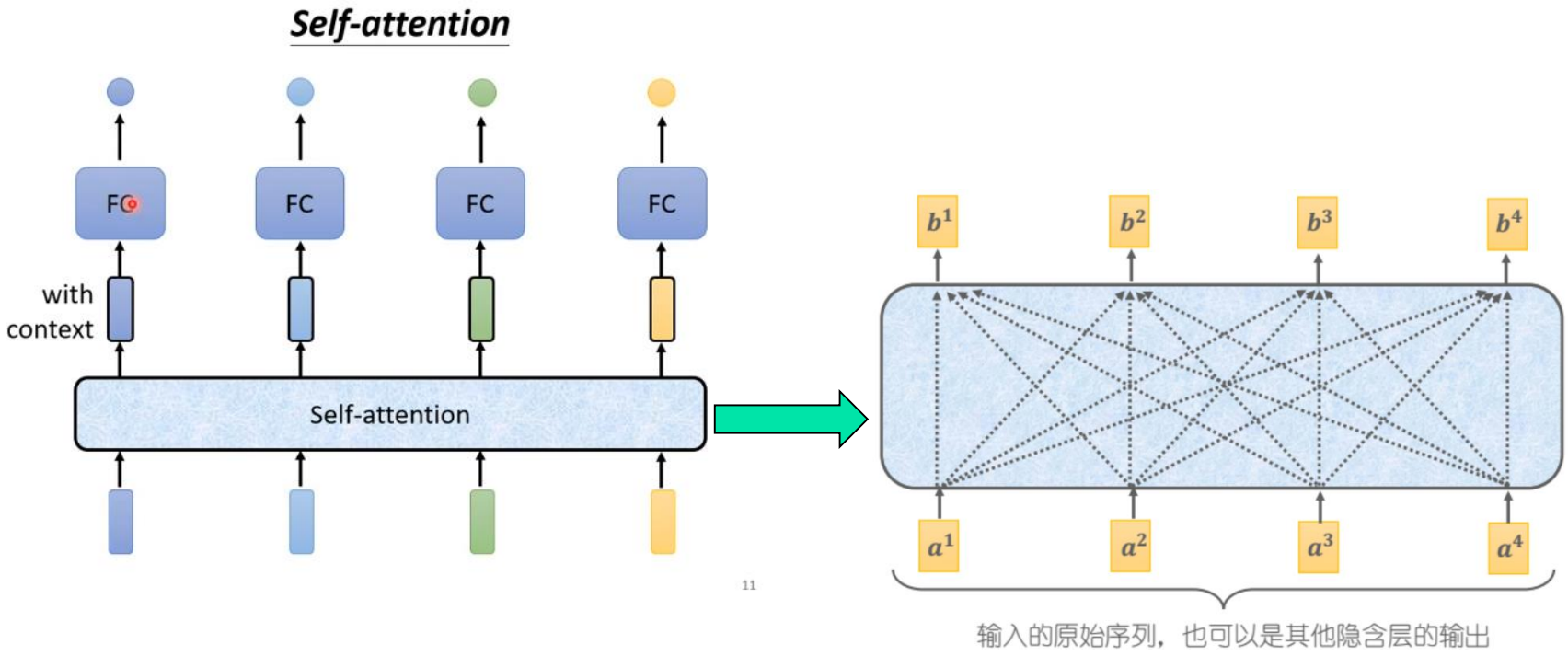
$$\boldsymbol{O}_t = \text{sigmoid}(\boldsymbol{W}_o\boldsymbol{X}_t + \boldsymbol{W}_{ho}\boldsymbol{H}_{t-1} + \boldsymbol{b}_o)$$

$$\boldsymbol{h}_t = \boldsymbol{O}_t \times \tanh(\boldsymbol{C}_t)$$

5.2.3 循环神经网络语言模型

◆自注意力机制

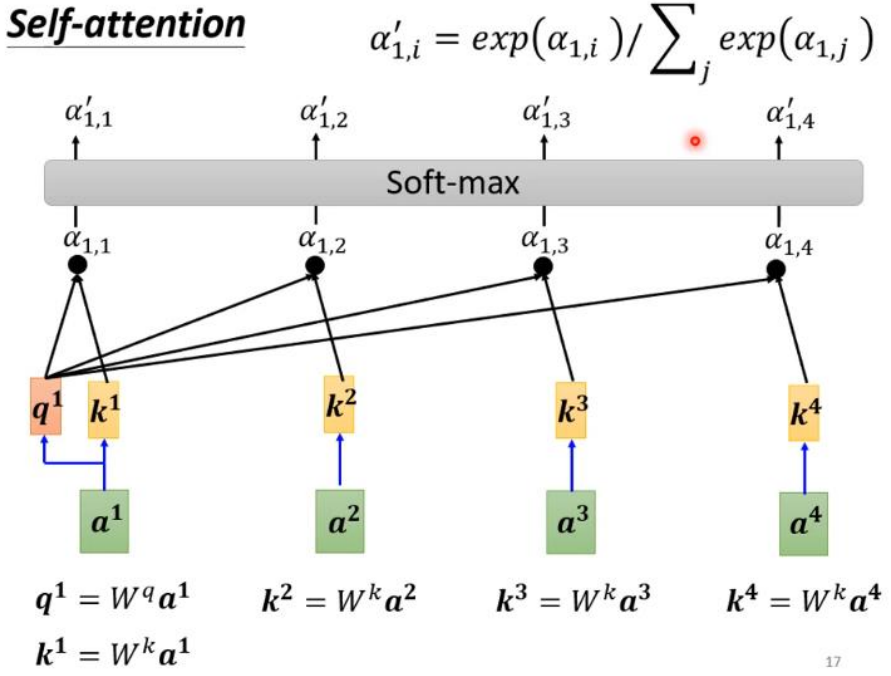
根据当前词与历史中各个词之间的相关性大小设置权重。



5.2.3 循环神经网络语言模型

◆自注意力机制

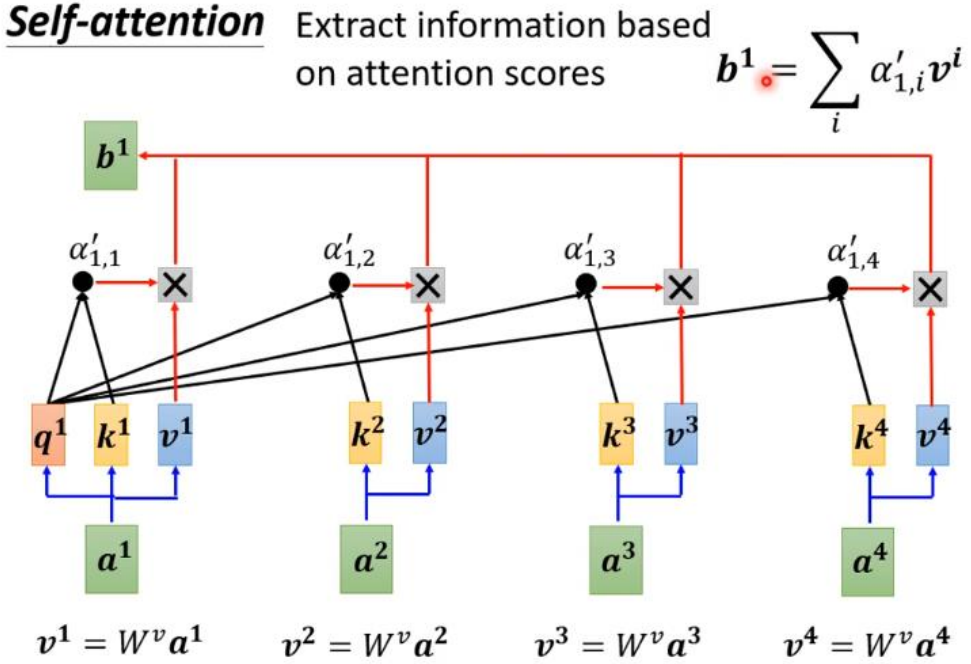
1. 为每个单词创建Query、Key、Value。
2. 对于每个输入token，使用其Query向量对其他所有的token的Key向量进行评分，获得注意力分数。
3. 将Value向量乘以上一步得到的注意力分数，之后加起来。



5.2.3 循环神经网络语言模型

◆自注意力机制

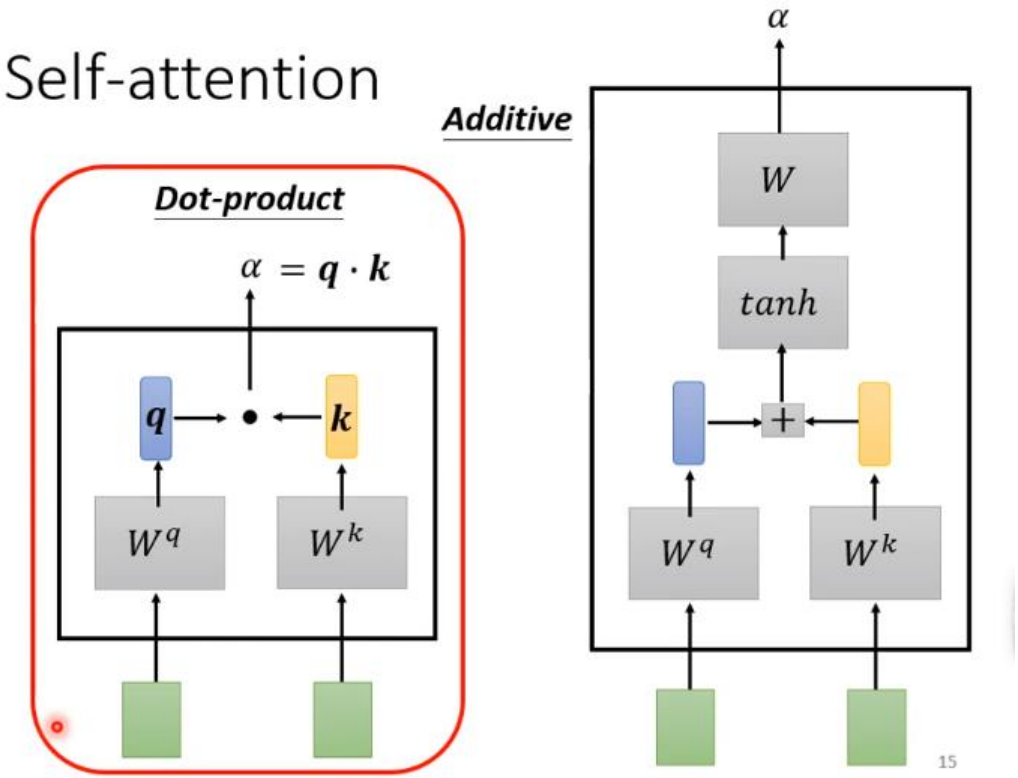
1. 为每个单词创建Query、Key、Value。
2. 对于每个输入token，使用其Query向量对其他所有的token的Key向量进行评分，获得注意力分数。
3. 将Value向量乘以上一步得到的注意力分数，之后加起来。



5.2.3 循环神经网络语言模型

◆自注意力机制

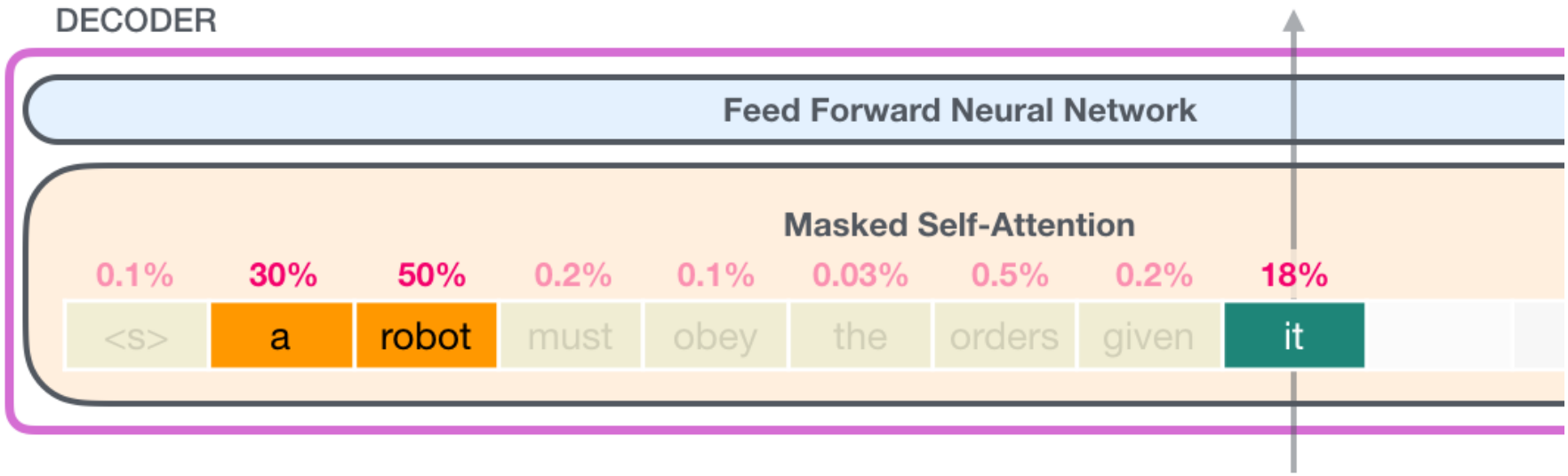
计算相关性的两种常见方式：　点积　拼接



5.2.3 循环神经网络语言模型

◆自注意力机制

效果展示

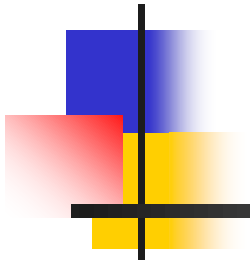


部分开源工具

- 前馈神经网络语言模型(feed-forward n-gram neural language model)
  http://nlg.isi.edu/software/nplm/
- 循环神经网络语言模型(recurrent neural language model)
  http://rnnlm.org/
- LSTM语言模型(recurrent neural language model with LSTM unit)
  https://www-i6.informatik.rwth-aachen.de/web/Software/rwthlm.php
- 自我注意机制语言模型GPT(Language Model with Generative Pre-training)
  https://github.com/openai/gpt-2
- LSTM反向传播算法
  http://arunmallya.github.io/writeups/nn/lstm/index.html#/
- Google Word2Vec: http://code.google.com/p/word2vec/

本部分小结

- 问题的提出
  离散符号表示 *vs.* 分布式连续向量表示
- 前馈神经网络语言模型
- 循环神经网络语言模型
  - 循环神经网络
  - 长短时记忆网络LSTM
  - 自注意力语言模型



Thanks

謝謝！