

第6章 神经语言模型

宗成庆

中国科学院自动化研究所

cqzong@nlpr.ia.ac.cn

本章内容



1. 问题提出
2. 神经网络概述
3. 前馈神经网络及语言模型
4. 循环神经网络及语言模型
5. 长时短时记忆网络
6. 注意力机制
7. GPT模型
8. 习题
9. 附录

1. 问题提出

◆ 回顾 n -gram 模型

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

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



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

1. 问题提出

● 问题

这本小说很枯燥，读起来很乏味。

$$p(\text{枯燥}|\text{很}) = \frac{\text{count}(\text{很枯燥})}{\text{count}(\text{很})} \quad p(\text{乏味}|\text{很}) = \frac{\text{count}(\text{很乏味})}{\text{count}(\text{很})}$$

```
graph TD; A[p("枯燥|很")]; B[p("乏味|很")]; C[①容易产生数据稀疏问题]; D[②忽略了语义相似性]; A --> C; A --> D; B --> C; B --> D;
```

①容易产生数据稀疏问题
 n -gram “很乏味”有可能未出现在训练样本中。

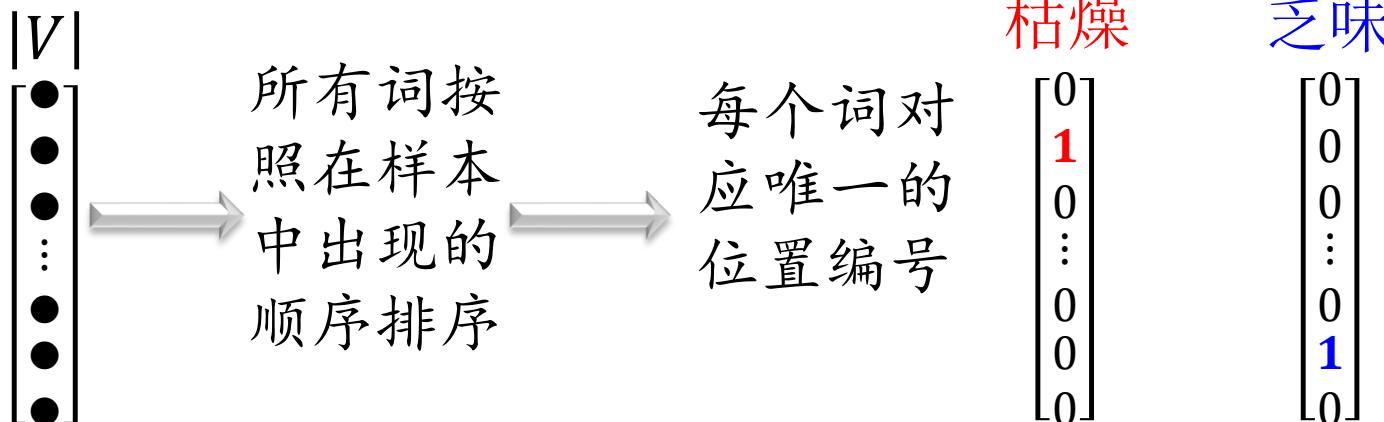
②忽略了语义相似性
“枯燥”与“乏味”虽然语义相似，但无法共享信息。

1. 问题提出

● 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

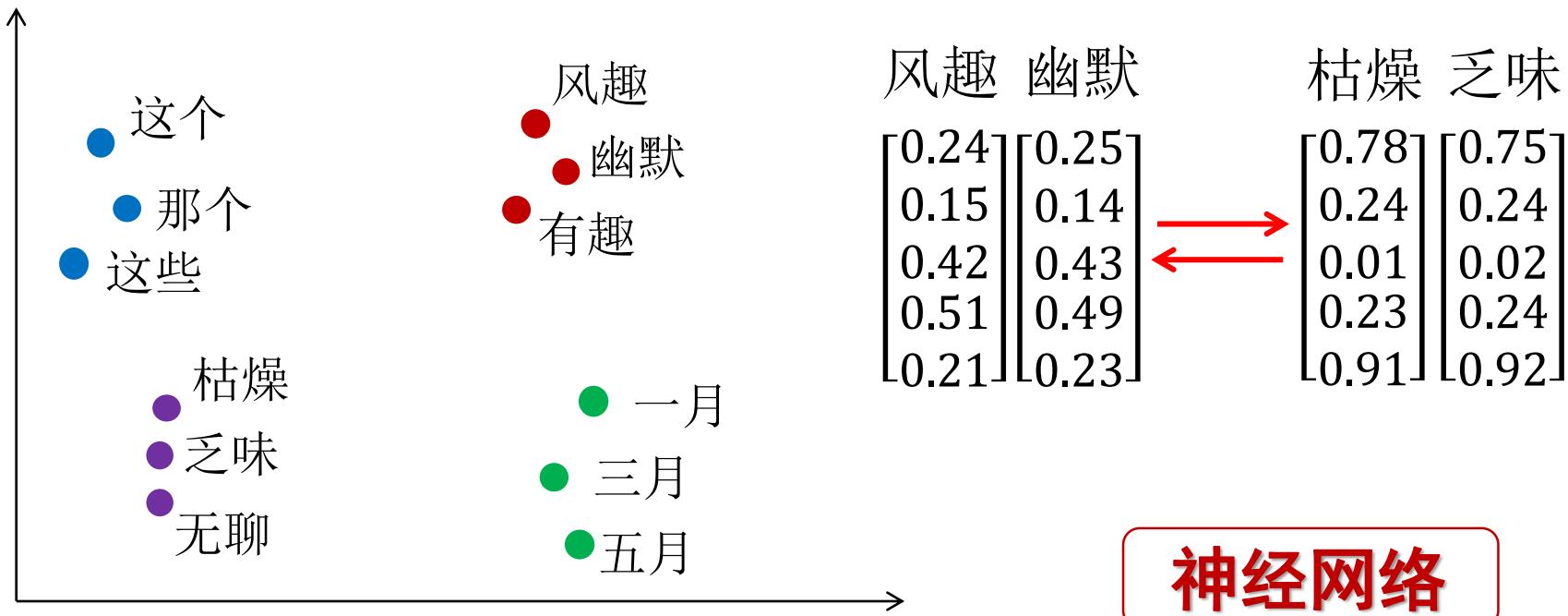


问题：
 $\left\{ \begin{array}{l} \text{枯燥} \otimes \text{乏味} = 0, \text{ 任意两个词之间的相似度都为0。} \\ \text{维度太高!} \end{array} \right.$

1. 问题提出

● 探索

是否可以用连续空间的分布式表示赋予词向量呢？



低维、稠密的连续实数空间

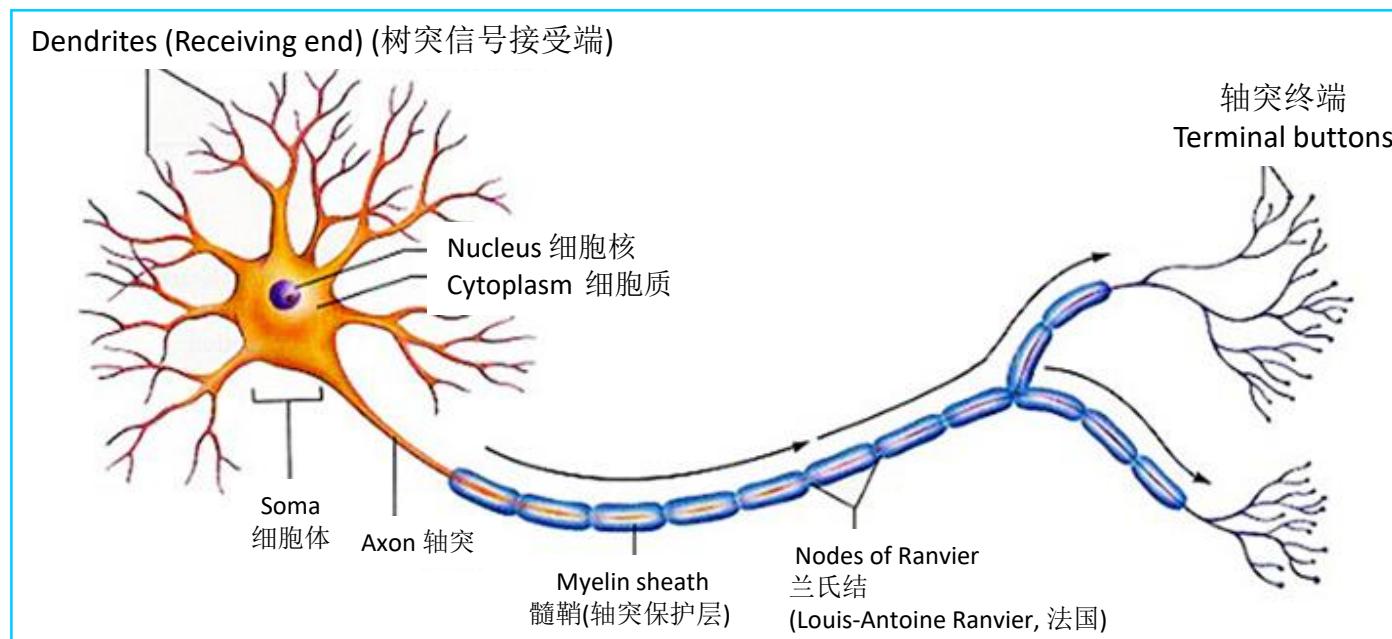
本章内容

1. 问题提出
2. 神经网络概述
3. 前馈神经网络及语言模型
4. 循环神经网络及语言模型
5. 长时短时记忆网络
6. 注意力机制
7. GPT模型
8. 习题
9. 附录

2. 神经网络概述

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

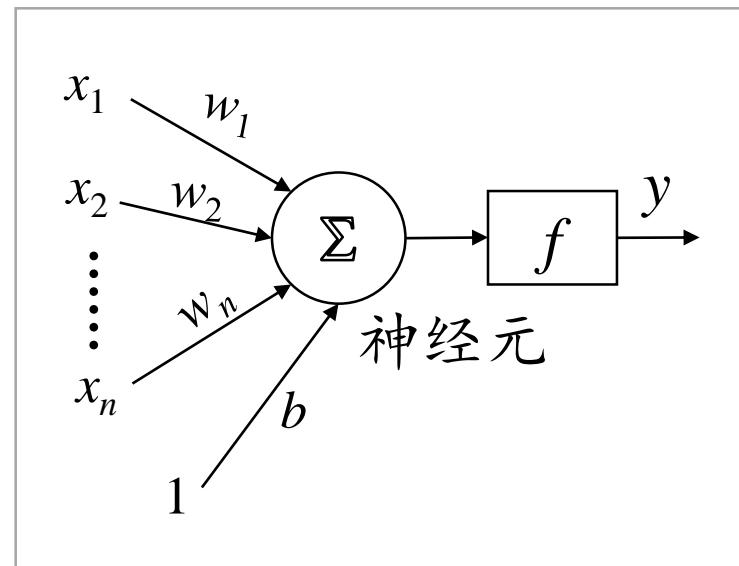
1943年心理学家 沃伦•麦卡洛克(W. McCulloch) 和数理逻辑学家 W. Pitts 建立了神经网络的数学模型，提出了神经元的形式化数学描述和网络结构方法。



2. 神经网络概述

● 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(激发/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： $y = f(\vec{W} \cdot \vec{X}^T + b)$
 \vec{W} 为权值向量； \vec{X} 为输入向量， \vec{X}^T 为 \vec{X} 的转置。



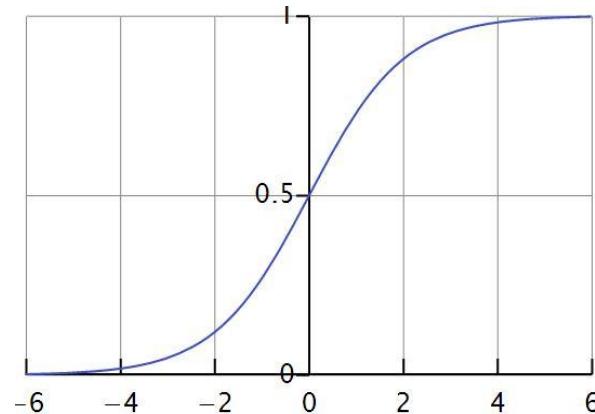
2. 神经网络概述

● 常用的激活函数

➤ Sigmoid 函数

① Logistic 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）。

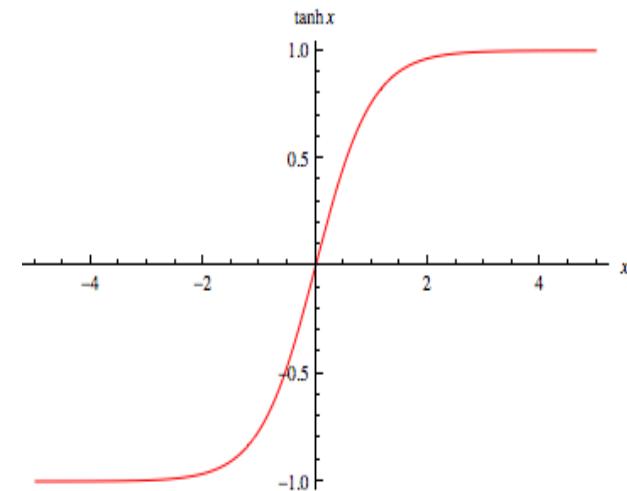
2. 神经网络概述

②Tanh 函数

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

或者：

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

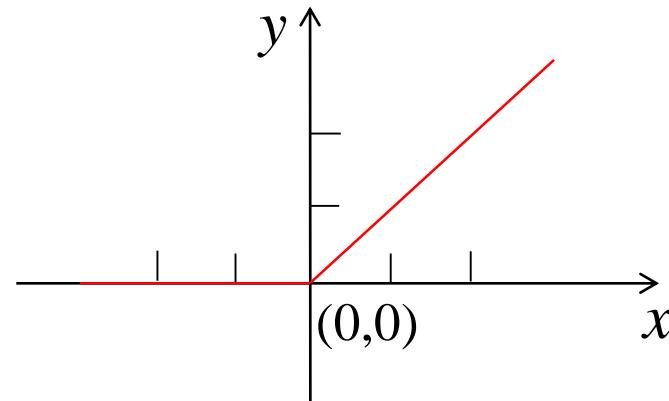


Tanh 函数可以看作是放大并平移的 Logistic 函数，其值域是 $(-1, 1)$ 。

2. 神经网络概述

► ReLU 函数 (rectified linear unit, 修正现象单元)[Nair et al., 2010]

$$\begin{aligned} \text{ReLU}(x) &= \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} \\ &= \max(0, x) \end{aligned}$$



优点：操作简单、高效。在优化方面，相对于Sigmoid 函数的两端饱和，ReLU为左饱和函数，且在 $x>0$ 时导数为1，在一定程度上缓解了神经网络的梯度消失问题，加速梯度下降的收敛速度。

弱点：输出是非0中心化的，给后一层的神经网络引入偏置偏移，影响梯度下降的效率。训练时容易导致神经元“死亡”。

2. 神经网络概述

常用的几种神经网络：

- ◆ 前馈神经网络 (Feed-forward Neural Network, FNN)
- ◆ 卷积神经网络 (Convolutional Neural Network, CNN)
- ◆ 循环神经网络 (Recurrent Neural Network, RNN)
- ◆ 长时短时记忆网络 (Long-Short Term Memory, LSTM)

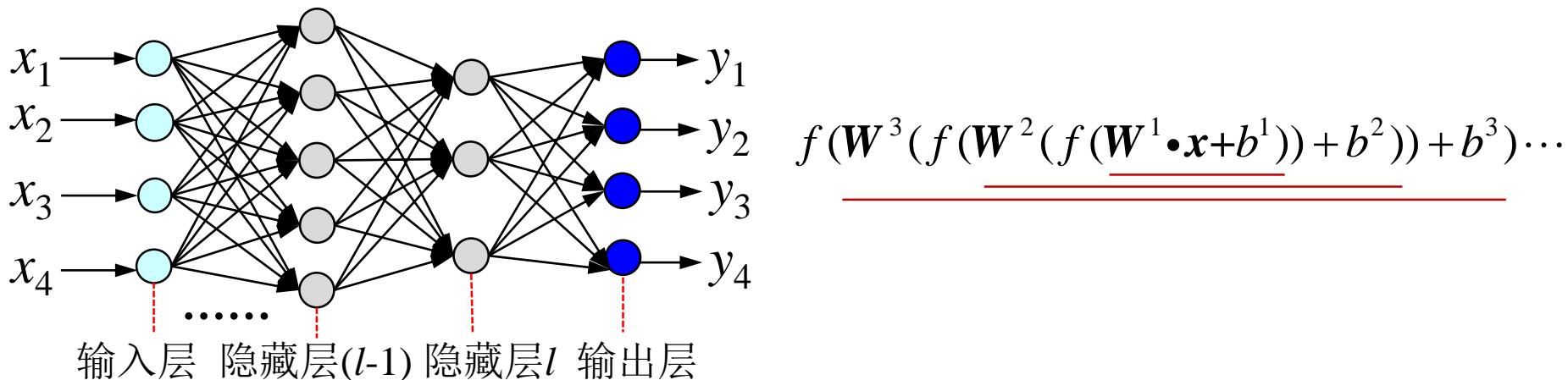
本章内容

1. 问题提出
2. 神经网络概述
- 3. 前馈神经网络及语言模型**
4. 循环神经网络及语言模型
5. 长时短时记忆网络
6. 注意力机制
7. GPT模型
8. 习题
9. 附录

3. 前馈神经网络及语言模型

◆ FNN描述

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)。前馈网络中各个神经元按接收信息的先后分为不同的组，每一组可以看作一个神经层，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，整个网络中的信息是朝一个方向传播，没有反向的信息传播，可以看作是一个有向的无环图，节点全连接。

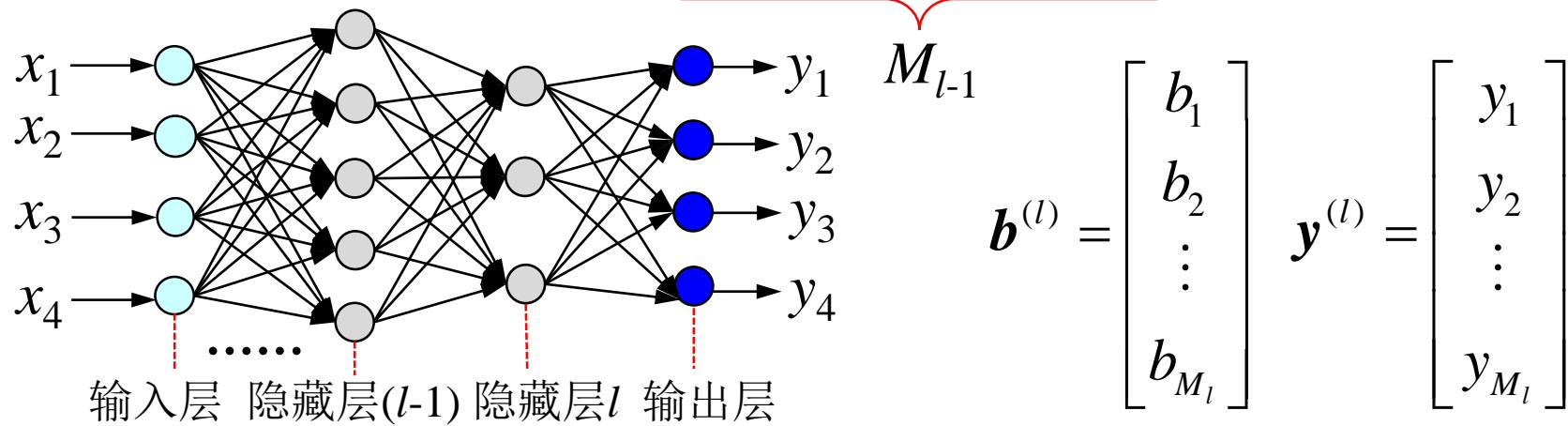


3. 前馈神经网络及语言模型

● 网络表示

L 为神经网络的层数；第1层有 M_1 个神经元，第2层有 M_2 个神经元……第 l 层有 M_l 个神经元； $f_l(\bullet)$ 为第 l 层神经元的激活函数；

$$\mathbf{x}^{(l-1)} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{M_{l-1}} \end{bmatrix} \quad \mathbf{W}^{(l)} = \begin{bmatrix} w_{11} & w_{21} & \cdots & w_{M_{l-1}1} \\ w_{12} & w_{22} & \ddots & w_{M_{l-1}2} \\ \vdots & \vdots & \vdots & \vdots \\ w_{1M_l} & w_{2M_l} & \cdots & w_{M_{l-1}M_l} \end{bmatrix} \in \mathbb{R}^{M_l \times M_{l-1}} \quad \left. M_l \right\}$$



3. 前馈神经网络及语言模型

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

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

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

根据第 $l-1$ 层神经元的活性值(Activation) $\alpha^{(l-1)}$ 计算出第 l 层神经元的净活性值(Net Activation) $\mathbf{z}^{(l)}$, 然后经过一个激活函数得到第 l 层神经元的活性值。上述两个式子可以合并为:

$$\alpha^{(l)} = f_l(\mathbf{W}^{(l)} \cdot \alpha^{(l-1)} + \mathbf{b}^{(l)}) \quad (6-1)$$

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

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

3. 前馈神经网络及语言模型

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

如果采用交叉熵作为损失函数，那么对于样本 (\mathbf{x}, \mathbf{y}) ，其损失函数为：

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = -\mathbf{y}^T \log \hat{\mathbf{y}}$$

第2章课件, P29: $H(X, q) = -\sum_{x \in X} p(x) \log q(x)$

给定训练集 $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^N$ ，对于每个输入样本 $\mathbf{x}^{(n)}$ 得到的网络输出为: $\hat{\mathbf{y}}^{(n)}$ ，那么在数据集 \mathcal{D} 上的结构化风险函数为:

$$\mathcal{R}(\mathbf{W}, \mathbf{b}) = \frac{1}{N} \sum_{n=1}^N \mathcal{L}(\mathbf{y}^{(n)}, \hat{\mathbf{y}}^{(n)}) + \frac{\varepsilon}{2} \|\mathbf{W}\|_F^2 \quad (6-2)$$

其中 \mathbf{W} 和 \mathbf{b} 分别表示网络中所有的权重矩阵和偏置向量。 $\|\mathbf{W}\|_F^2$ 是正则化项（只包括权重 \mathbf{W} ，不包含偏置 \mathbf{b} ），用以防止过拟合。 $0 < \varepsilon < 1$ 为超参数， ε 越大， \mathbf{W} 越接近于 0。

3. 前馈神经网络及语言模型

$\|W\|_F^2$ 一般采用 Frobenius 范数(弗罗宾尼斯范数, F-范数):

$$\|W\|_F^2 = \sum_{l=1}^L \sum_{i=1}^{M_l} \sum_{j=1}^{M_{l-1}} \left(w_{ij}^{(l)} \right)^2 \quad (6-3)$$

网络参数可以通过随机梯度下降法(stochastic gradient descent algorithm)进行学习。梯度表示某一函数在该点处的**方向导数**沿着该方向取得最大值，即函数在该点处沿着该梯度的方向变化最快，变化率最大。对于单变量的实值函数，梯度只是导数，或者说，对于一个线性函数，也就是线的斜率。对于曲面而言，梯度是曲面沿着给定方向的倾斜程度。

3. 前馈神经网络及语言模型

在梯度下降方法的每次迭代中，第 l 层的参数 $\mathbf{W}^{(l)}$ 和 $\mathbf{b}^{(l)}$ 参数更新方式为：

$$\begin{aligned}\mathbf{W}^{(l)} &\leftarrow \mathbf{W}^{(l)} - \ell \frac{\partial \mathcal{R}(\mathbf{W}, \mathbf{b})}{\partial \mathbf{W}^{(l)}} \\ &= \mathbf{W}^{(l)} - \ell \left(\frac{1}{N} \sum_{n=1}^N \left(\frac{\partial \mathcal{L}(\mathbf{y}^{(n)}, \hat{\mathbf{y}}^{(n)})}{\partial \mathbf{W}^{(l)}} \right) + \varepsilon \mathbf{W}^{(l)} \right)\end{aligned}\quad (6-4)$$

其中， ℓ 为学习率， $0 < \ell < 1$ 。

$$\begin{aligned}\mathbf{b}^{(l)} &\leftarrow \mathbf{b}^{(l)} - \ell \frac{\partial \mathcal{R}(\mathbf{W}, \mathbf{b})}{\partial \mathbf{b}^{(l)}} \\ &= \mathbf{b}^{(l)} - \ell \left(\frac{1}{N} \sum_{n=1}^N \frac{\partial \mathcal{L}(\mathbf{y}^{(n)}, \hat{\mathbf{y}}^{(n)})}{\partial \mathbf{b}^{(l)}} \right)\end{aligned}\quad (6-5)$$

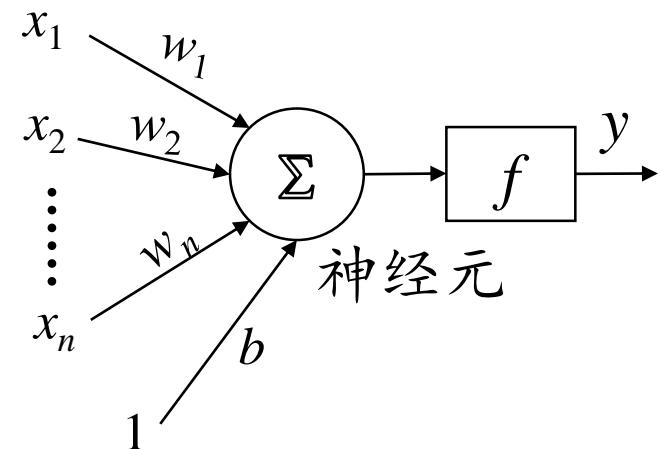
在神经网络的训练中通常使用反向传播(Back Propagation, BP) 算法高效地计算梯度。详见本章附录。

3. 前馈神经网络及语言模型

◆ 基于FNN的语言模型

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

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

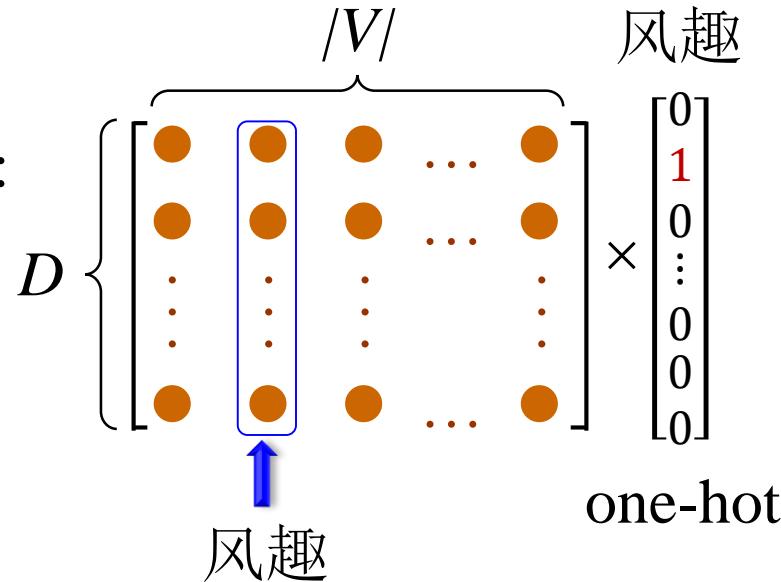


词向量 ?
 $W, b = ?$

3. 前馈神经网络及语言模型

● 词向量表示

Look-up
table(L_T):



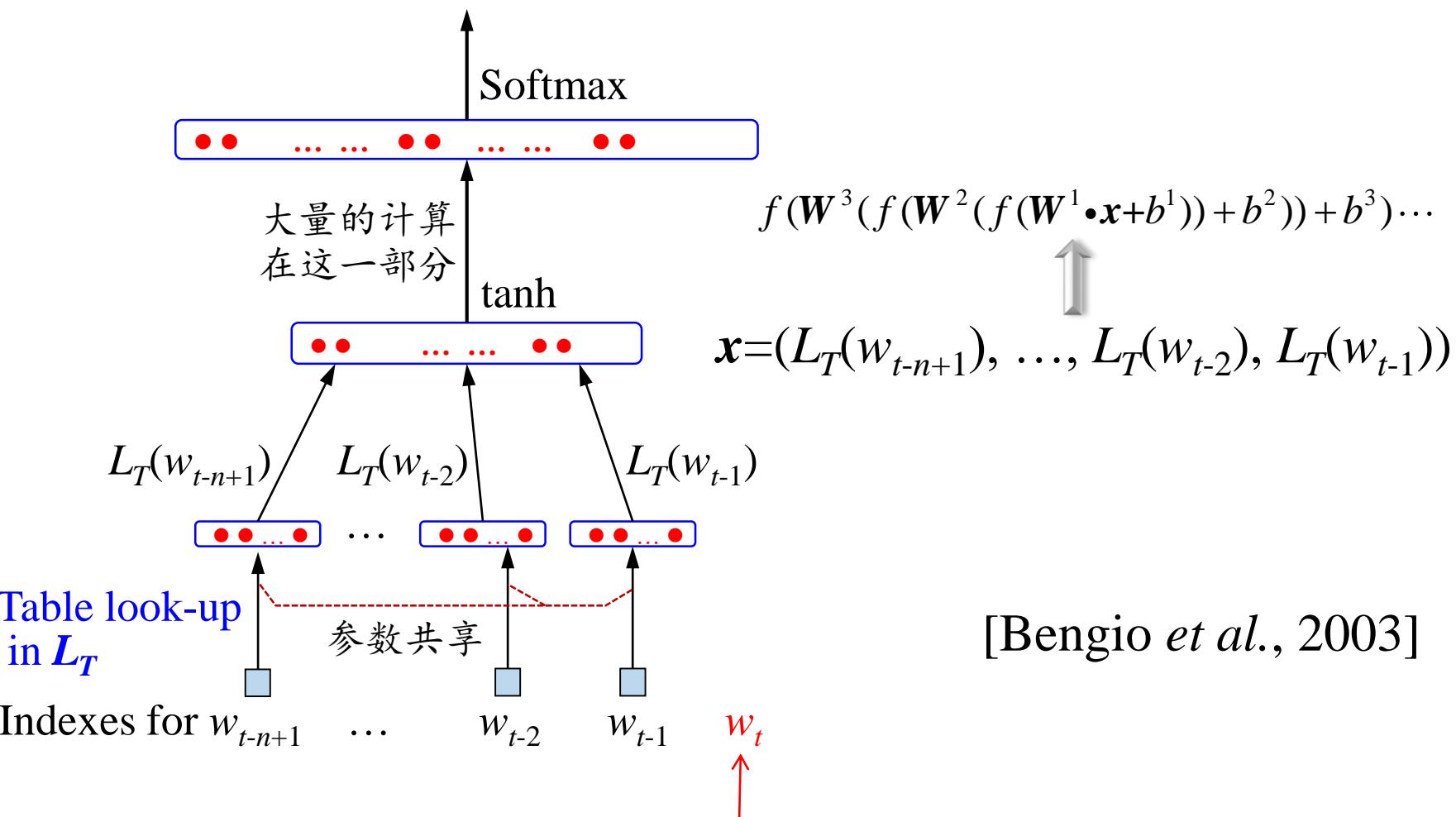
如何学习 L_T ?

通常先随机初始化，
然后通过目标函数
优化词的向量表示
(如最大化语言模
型似然度)。

- 词汇表 V 的三种确定方法: (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。
- 维度 D 的确定: 超参数, 人工设定, 一般从几十到几百。

3. 前馈神经网络及语言模型

输出: $w_t = p(w_t | \text{context})$



3. 前馈神经网络及语言模型

● Softmax 回归 (regression)

Softmax 回归也称为多项(Multinomial)或多类(Multi-Class)的 Logistic 回归, 是Logistic 回归在多分类问题上的推广, 它也可以看作是一种条件最大熵模型。

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

$$\begin{aligned} p(y = c | \mathbf{x}) &= \text{Softmax}(\mathbf{w}_c^T \mathbf{x}) \\ &= \frac{\exp(\mathbf{w}_c^T \mathbf{x})}{\sum_{c'=1}^C \exp(\mathbf{w}_{c'}^T \mathbf{x})} \quad \mathbf{w}_c = \begin{bmatrix} w_{c1} \\ w_{c2} \\ \vdots \\ w_{cn} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \end{aligned}$$

其中, \mathbf{w}_c 是第 c 类的权重向量。

决策函数: $\hat{y} = \arg \max_{c=1}^C p(y = c | \mathbf{x}) = \arg \max_{c=1}^C \mathbf{w}_c^T \mathbf{x}$



3. 前馈神经网络及语言模型

向量表示：用 $\hat{y} \in \mathbb{R}^C$ 表示所有类别预测的条件概率组成的向量：

$$\hat{y} = \text{Softmax}(W^T x) = \frac{\exp(W^T x)}{\mathbf{1}_C^T \exp(W^T x)}$$

其中， $W = [w_1, \dots, w_C]$ 是由 C 个类的权重向量组成的矩阵，即：

$$W = \begin{bmatrix} w_{11} & w_{21} & \cdots & w_{C1} \\ w_{12} & w_{22} & \cdots & w_{C2} \\ \vdots & \vdots & & \vdots \\ w_{1n} & w_{2n} & \cdots & w_{Cn} \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \mathbf{1}_C = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \quad C$$

$$W^T x = \begin{bmatrix} \bullet \\ \vdots \\ \bullet \end{bmatrix} \quad \exp(W^T x) = \begin{bmatrix} \bullet \\ \vdots \\ \bullet \end{bmatrix} \quad \mathbf{1}_C^T \exp(W^T x) = \Delta \quad \hat{y} = \frac{\exp(W^T x)}{\mathbf{1}_C^T \exp(W^T x)} = \frac{1}{\Delta} \begin{bmatrix} \bullet \\ \vdots \\ \bullet \end{bmatrix}$$

$$\sum_{c=1}^C \exp(w_c^T x)$$

3. 前馈神经网络及语言模型

例如：

$$\hat{y} = \begin{bmatrix} 0.03 \\ 0.16 \\ \vdots \\ 0.08 \end{bmatrix}$$

表示C个类别预测的概率。第 c 维的值样本 x 被预测为 c 类的概率。

3. 前馈神经网络及语言模型

◆ 举例说明

1. 假设已知 FNN，激活函数用 tanh 函数。

$$\mathbf{L}_T = \begin{bmatrix} \dots \mathbf{x}_i \dots \mathbf{x}_j \dots \mathbf{x}_k \dots \mathbf{x}_{\cdot} \dots \mathbf{x}_{\cdot} \dots \\ \dots 0.1 \dots 0.5 \dots 0.3 \dots 0.4 \dots 0.2 \dots \\ \dots 0.3 \dots 0.4 \dots 0.2 \dots 0.2 \dots 0.1 \dots \end{bmatrix} \quad 2 \times 5000$$

↑ ↑ ↑ ↑ ↑
本 ... 很...乏味...书 ... 这...

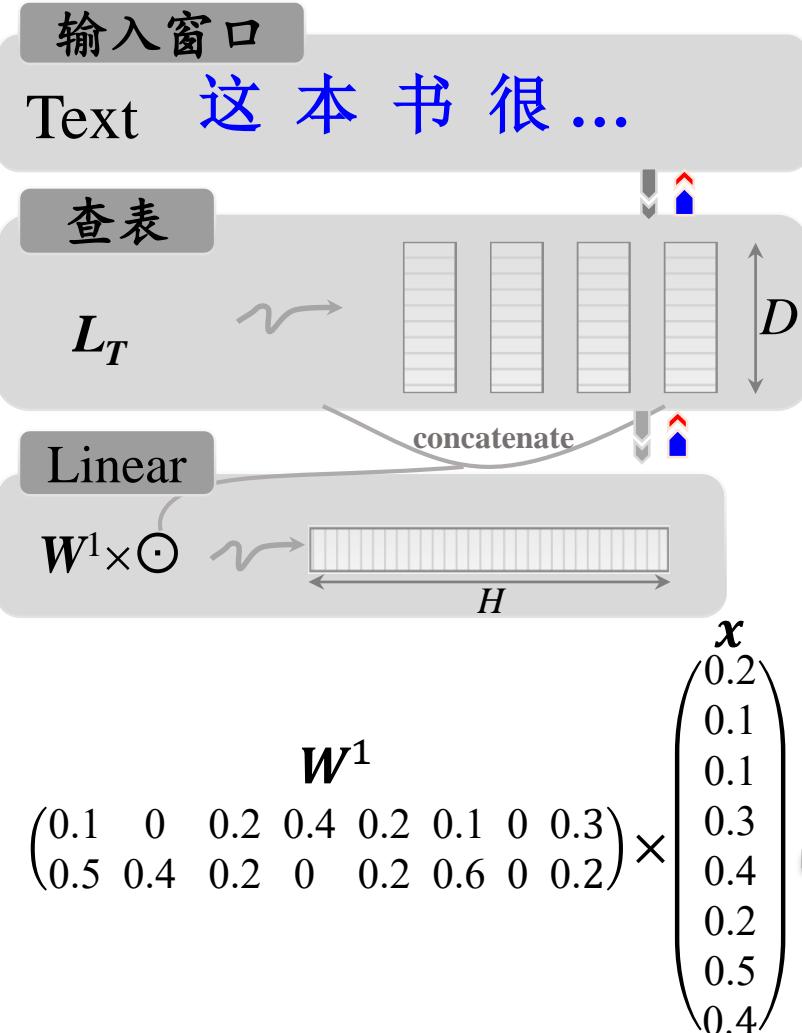
$$\mathbf{W} = \left(\dots \mathbf{w}_i \dots \mathbf{w}_j \dots \mathbf{w}_k \dots \mathbf{w}_{\cdot} \dots \mathbf{w}_{\cdot} \dots \mathbf{w}_{\cdot} \dots \mathbf{w}_{\cdot} \dots \right) \text{暂不考虑偏置 } b.$$

↑ ↑ ↑ ↑ ↑
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 ...

$$y = \tanh(\mathbf{w} \cdot \mathbf{x} + b) \qquad p(\text{乏味} | \text{这}, \text{本}, \text{书}, \text{很}) ?$$

→ Softmax(y)

3. 前馈神经网络及语言模型



$p(\text{乏味} | \text{这, 本, 书, 很})$

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。

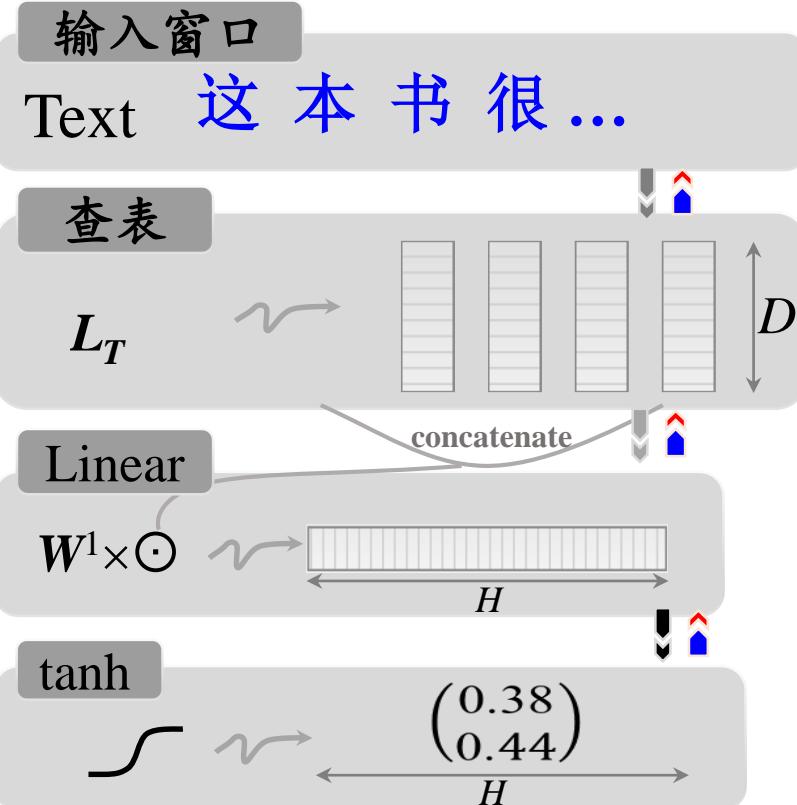
这 本 书 很

或者加和, 取平均

$$\rightarrow (0.2, 0.1, 0.1, 0.3, 0.4, 0.2, 0.5, 0.4)^T$$

③隐藏层: 线性映射+非线性变换。

3. 前馈神经网络及语言模型



$p(\text{乏味} | \text{这, 本, 书, 很})$

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

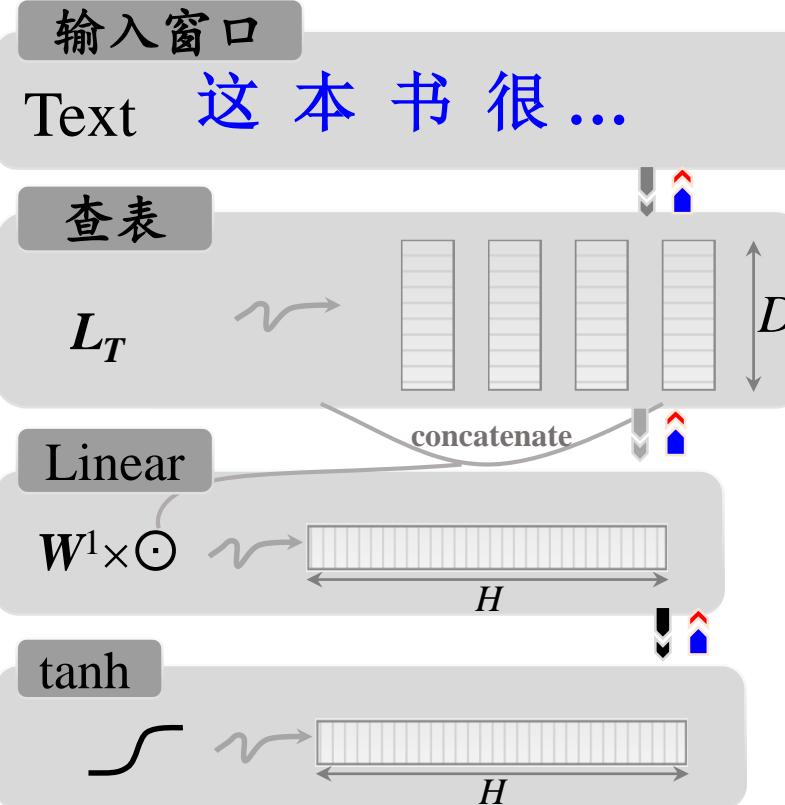
或者加和, 取平均

$$\rightarrow (0.2, 0.1, 0.1, 0.3, 0.4, 0.2, 0.5, 0.4)^T$$

③隐藏层: 线性映射+非线性变换。

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

3. 前馈神经网络及语言模型



$$\alpha = \begin{pmatrix} 0.36 \\ 0.41 \end{pmatrix}$$

内积, 相关性

$p(\text{乏味} | \text{这, 本, 书, 很})$

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

$$\rightarrow (0.2, 0.1, 0.1, 0.3, 0.4, 0.2, 0.5, 0.4)^T$$

③隐藏层: 线性映射+非线性变换。

$$L_T^T \times \alpha = \begin{pmatrix} 0.2 & 0.1 \\ 0.1 & 0.3 \\ 0.4 & 0.2 \\ 0.5 & 0.4 \\ 0.3 & 0.2 \\ \dots & \dots \end{pmatrix} \times \begin{pmatrix} 0.36 \\ 0.41 \end{pmatrix}$$

相当于计算给定历史 α 时, 词表中每个词出现在 α 后面时的情况。

3. 前馈神经网络及语言模型

$$\mathbf{L}_T^T \times \boldsymbol{\alpha} = \begin{pmatrix} 0.2 & 0.1 \\ 0.1 & 0.3 \\ 0.4 & 0.2 \\ 0.5 & 0.4 \\ \textcolor{red}{0.3} & \textcolor{red}{0.2} \\ \dots & \dots \end{pmatrix} \times \begin{pmatrix} 0.36 \\ 0.41 \end{pmatrix} = \begin{pmatrix} 0.113 \\ 0.159 \\ 0.226 \\ 0.344 \\ \textcolor{red}{0.190} \\ \dots \end{pmatrix}$$

乏味

$p'(\text{乏味} | \text{这, 本, 书, 很})$

计算 $\text{Softmax}(\bullet)$ 后对应位置上的值就是概率: $p(\text{乏味} | \text{这, 本, 书, 很})$ 。

$$\hat{\mathbf{y}} = \text{Softmax} \left(\begin{bmatrix} 0.113 \\ 0.159 \\ 0.226 \\ 0.344 \\ 0.190 \\ \dots \end{bmatrix} \right) = \frac{\exp(\bullet)}{\sum \exp(\cdot)} = \begin{bmatrix} 0.182 \\ 0.190 \\ 0.203 \\ 0.229 \\ \textcolor{red}{0.196} \\ \dots \end{bmatrix}$$



3. 前馈神经网络及语言模型

2. 如果 FNN 未知，需要通过训练样本集获得参数。

(1) 收集足够多的训练样本：

- ① 人间四月芳菲尽，山寺里的桃花才刚刚盛开...
- ② 人们都觉得这本书很乏味。
- ③ 近水楼台先得月，向阳的花木易获阳光和春风...
- ④ 他说这本书很有趣。
- ⑤ 这本书很真实地反映了当时的情况。
- ⑥ 他认为这本书很不适合小学生阅读。
- ⑦ 书架上面这本书很快将被卖完。

.....

3. 前馈神经网络及语言模型

- (2) 从训练样本中筛选词汇，确定词汇表 L_T ，根据词频或硬性规定词汇个数；
- (3) 确定表示词汇的向量的维数，初始化词汇表（每一维向量的赋值在 0~1 之间，用均匀分布或高斯分布）：

$$L_T = \begin{bmatrix} x_1 & x_2 & x_3 & \dots & x_i & \dots \dots \\ [0.3 & 0.5 & 0.3 & \dots & 0.3 & \dots \dots] \\ 0.3 & 0.4 & 0.2 & \dots & 0.2 & \dots \dots \\ 0.2 & 0.4 & 0.2 & \dots & 0.3 & \dots \dots \end{bmatrix} \quad 3 \times 30000$$

- (4) 初始化权重 W 和偏置 b （取值在 0~1 之间，均匀或高斯分布）；
- (5) 确定激活函数；确定 n -gram 的 n 的取值；
- (6) 在整个训练集上尽量准确地预测每一个 n 元文法，通过反向传播算法反复调整权重 W, b 和每一个词的向量表示 x_i ，直到在训练集上收敛，即风险损失最小。

3. 前馈神经网络及语言模型

◆开源工具

- 基于FNN的语言模型(feed-forward n-gram neural language model)
<http://nlg.isi.edu/software/nplm/>

◆代表论文

Yoshua Bengio, R éjean Ducharme, Pascal Vincent, Christian Jauvin. 2003. Neural Probabilistic Language Models[J]. Journal of Machine Learning Research 3 (2003) 1137–11556.



3. 前馈神经网络及语言模型

◆ 问题分析

- 在FNN中信息传播是单向的，网络模型的能力受到了限制；
- 在FNN中，每次输入是独立的，网络输出只依赖于当前的输入，而在很多任务中，当前时刻的输出不仅与当前的输入有关，还与之前一段时间内的输出都相关；
- FNN要求输入和输出的维数是固定的，不能任意改变，而时序数据（如语音、文本、视频等）的长度一般是不固定的，这样得到的向量维数（以句子为例）就不固定，这就为FNN带来了困难；
- 在 n -gram语言模型和基于FNN的语言模型中，仅对有限范围窗口内的历史信息进行建模，仅考虑前面 $n-1$ 个词的历史信息：

$$p(s) = \prod_{t=1}^m p(w_t | w_1 \dots w_{t-1}) = \prod_{t=1}^m p(w_t | w_{t-n+1}^{t-1})$$

能否对更长的甚至所有的历史信息进行建模呢？

本章内容

1. 问题提出
2. 神经网络概述
3. 前馈神经网络及语言模型
- 4. 循环神经网络及语言模型**
5. 长时短时记忆网络
6. 注意力机制
7. GPT模型
8. 习题
9. 附录

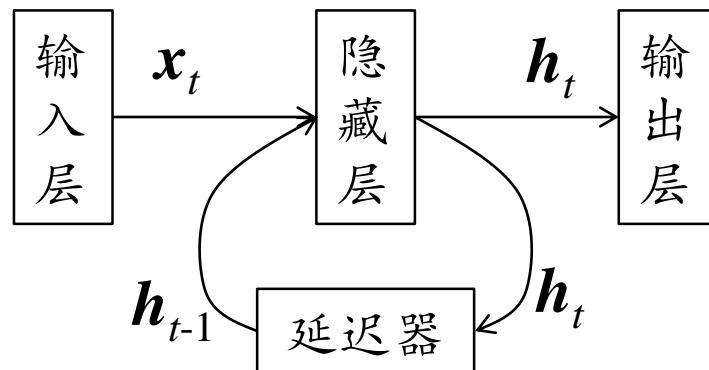
4. 循环神经网络及语言模型

◆RNN 模型描述

RNN通过使用带自反馈的神经元，能够处理任意长度的时序数据。给定一个输入序列： $\mathbf{x}_1^T = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T)$ ，RNN通过下面的公式更新带反馈边的隐藏层的活性值 \mathbf{h}_t ：

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t)$$

其中， $\mathbf{h}_0=0, f(\cdot)$ 为一个非线性函数，可以是一个前馈网络。



“延迟期”是一个虚拟单元，记录神经元最近一次（或几次）的活性值。

\mathbf{h}_t 也被称作“状态(state)”或“隐状态(hidden state)”。



4. 循环神经网络及语言模型

假设 $\mathbf{x}_t \in \mathbb{R}^M$ 是 t 时刻的网络输入, $\mathbf{h}_t \in \mathbb{R}^D$ 是隐藏层的状态(即隐藏层神经元的活性值), 那么, \mathbf{h}_t 不仅与当前时刻的输入 \mathbf{x}_t 有关, 而且与上一时刻的隐藏层状态 \mathbf{h}_{t-1} 有关, 因此, 一个简单的RNN在 t 时刻的更新公式为:

$$\mathbf{z}_t = \mathbf{U}\mathbf{h}_{t-1} + \mathbf{W}\mathbf{x}_t + \mathbf{b}$$

$$\mathbf{h}_t = f(\mathbf{z}_t)$$

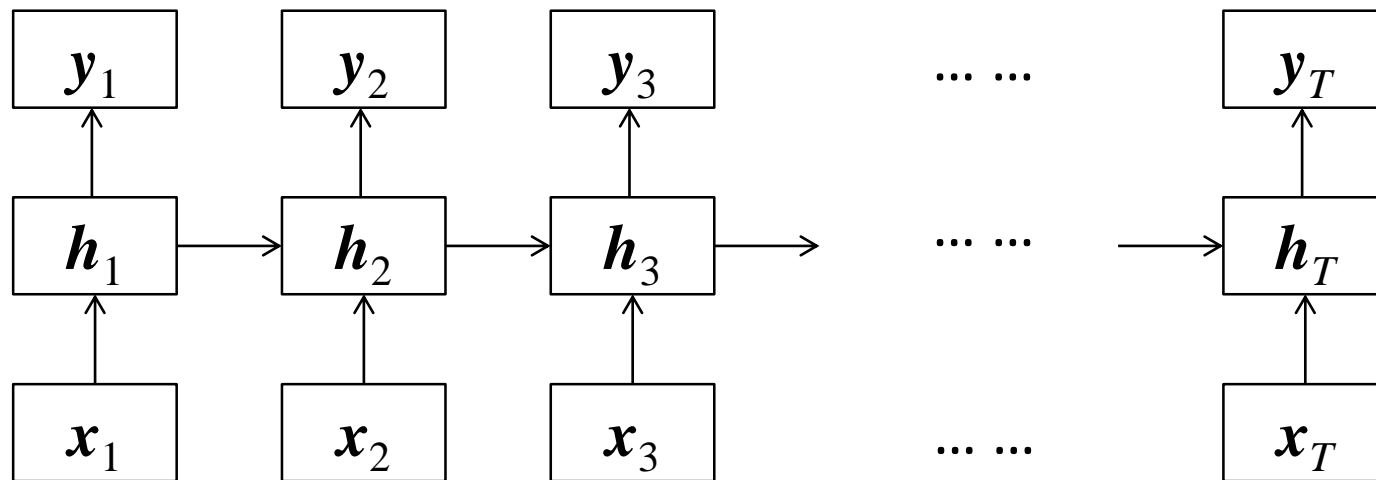
其中, \mathbf{z}_t 为隐藏层的净输入, $\mathbf{U}_t \in \mathbb{R}^{D \times D}$ 为状态-状态的权重矩阵; $\mathbf{W} \in \mathbb{R}^{D \times M}$ 为状态-输入的权重矩阵; $\mathbf{b} \in \mathbb{R}^D$ 为偏置向量。 $f(\cdot)$ 为非线性激活函数, 通常使用 Logistic 函数或者 \tanh 函数。

上面的两个公式可以直接写成:

$$\mathbf{h}_t = f(\mathbf{U}\mathbf{h}_{t-1} + \mathbf{W}\mathbf{x}_t + \mathbf{b})$$

4. 循环神经网络及语言模型

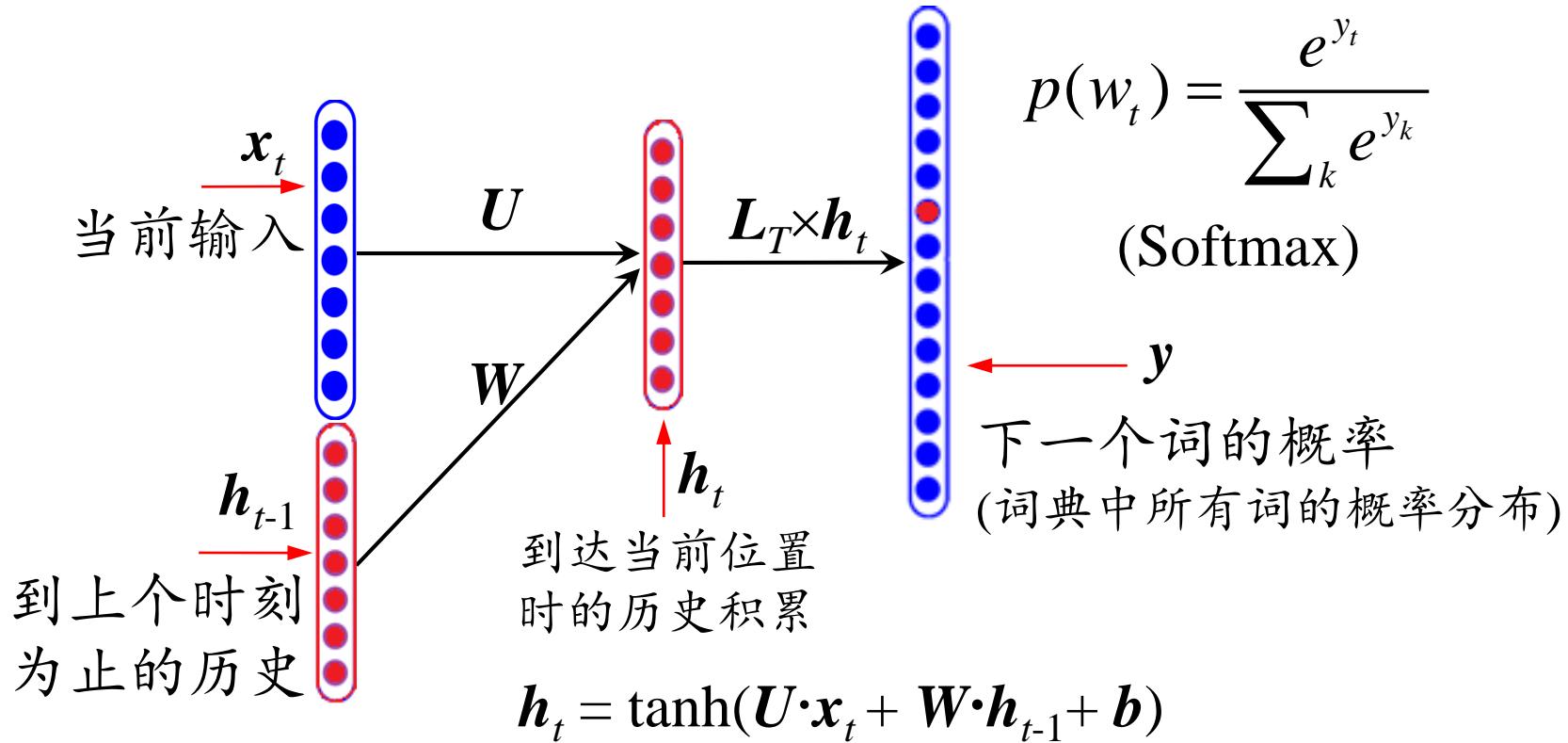
如果把每个时刻的状态都看作是FNN的一层，RNN可以看作是在时间维度上权值共享的神经网络。下面的图是按时间展开的RNN：



4. 循环神经网络及语言模型

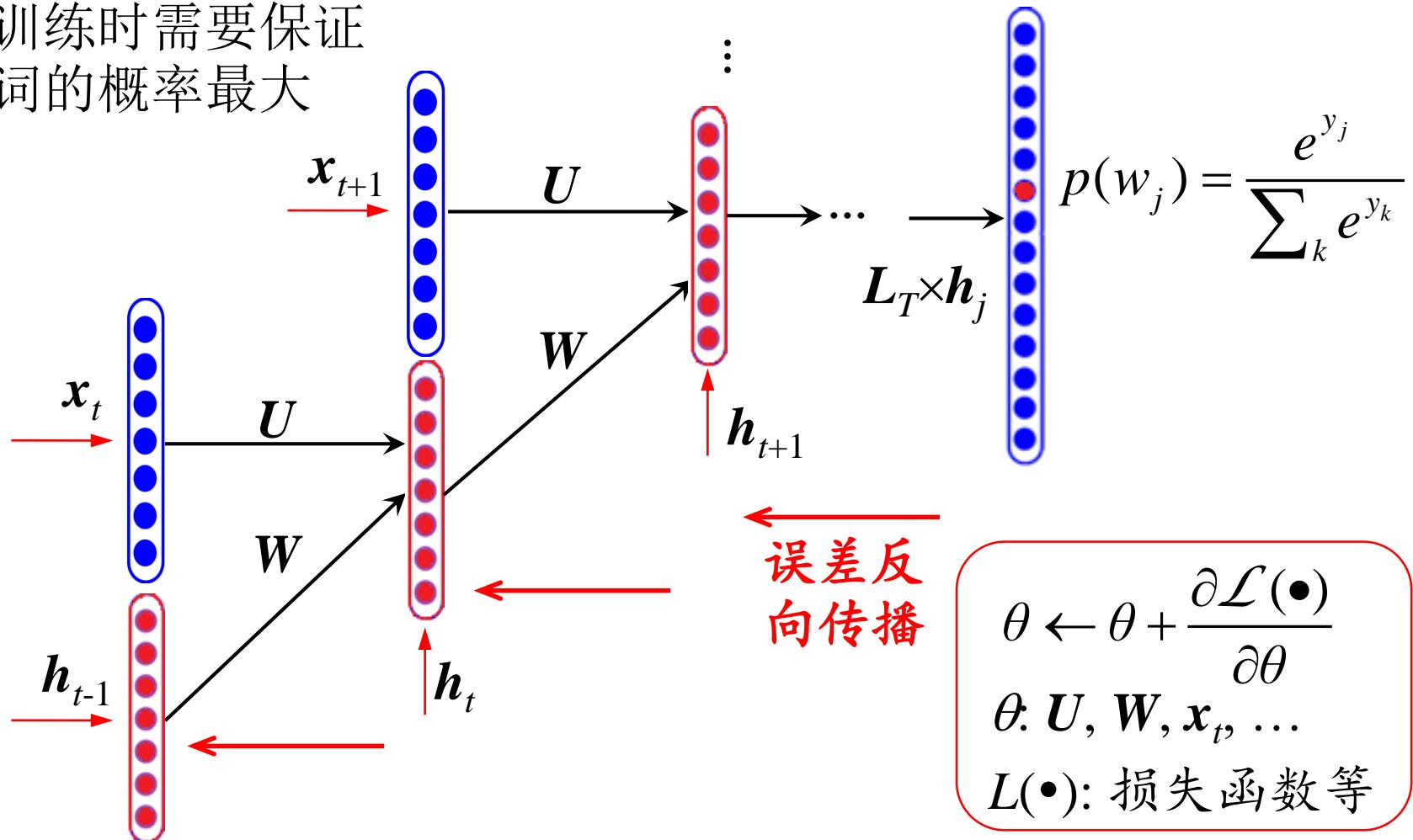
◆ 基于RNN的语言模型

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

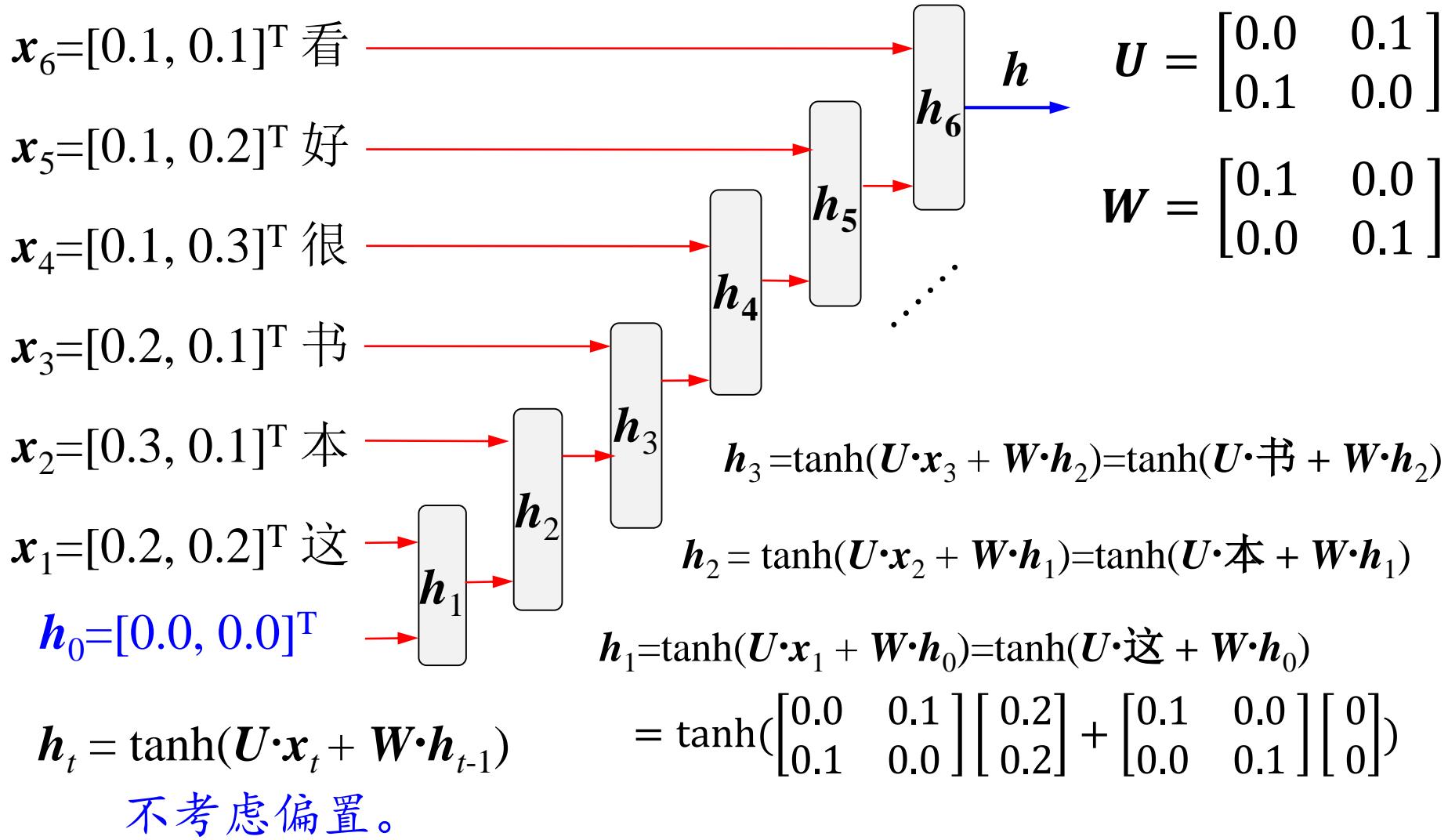


4. 循环神经网络及语言模型

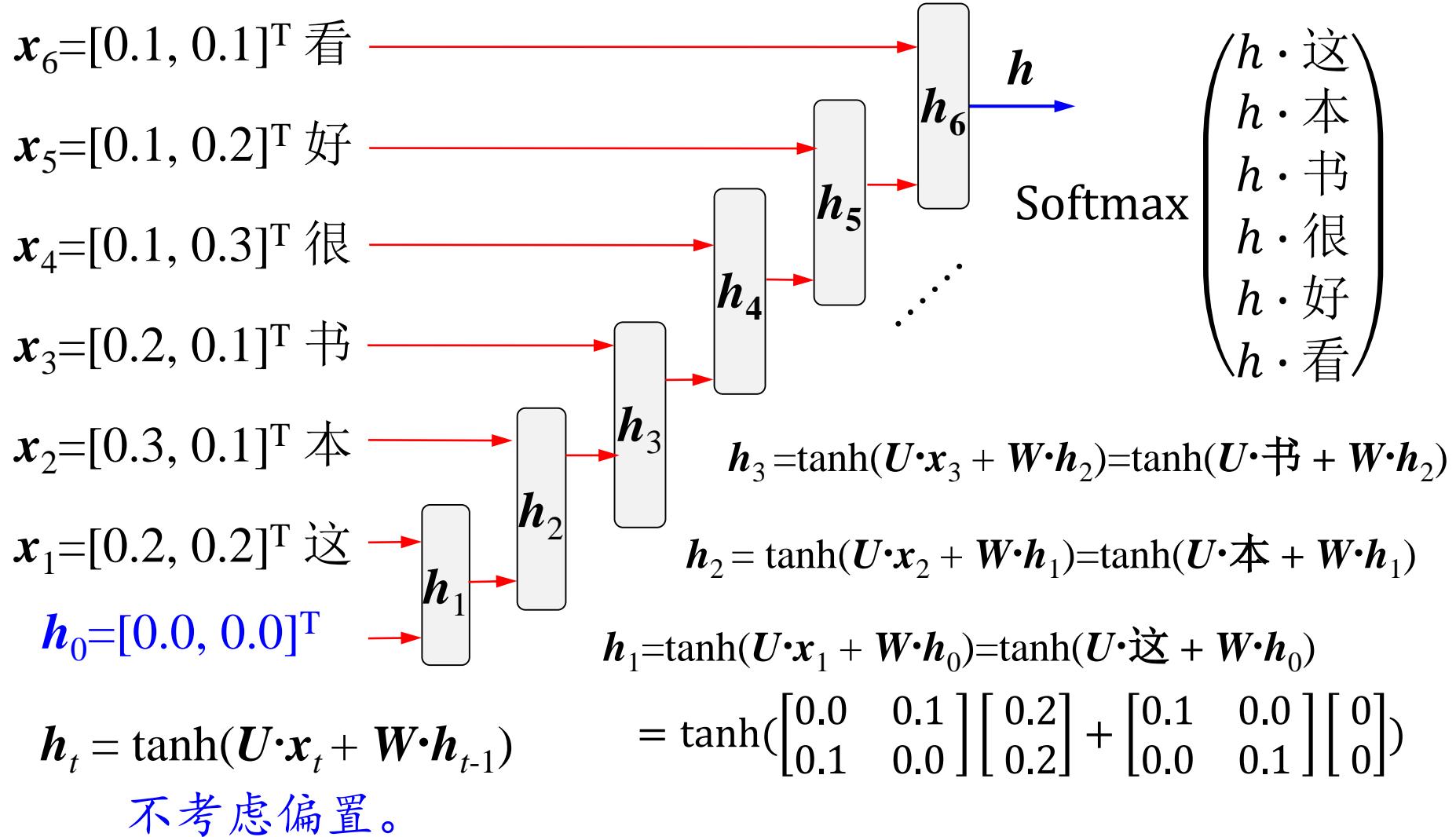
模型训练时需要保证
当前词的概率最大



4. 循环神经网络及语言模型



4. 循环神经网络及语言模型



4. 循环神经网络及语言模型

◆开源工具

- 基于循环神经网络的语言模型(recurrent neural language model)
<http://rnnlm.org/>

◆代表论文

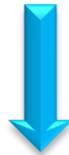
Mikolov, T. , M Karafi át, Burget, L. , Cernock, J. , and Khudanpur, S. 2010. Recurrent neural network based language model. In *Proceedings of the International Conference on Speech Communication Association* (Interspeech), Makuhari, Chiba, Japan. Pages 1045-1048.

4. 循环神经网络及语言模型

◆ 问题分析

梯度消失或爆炸：参数 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？



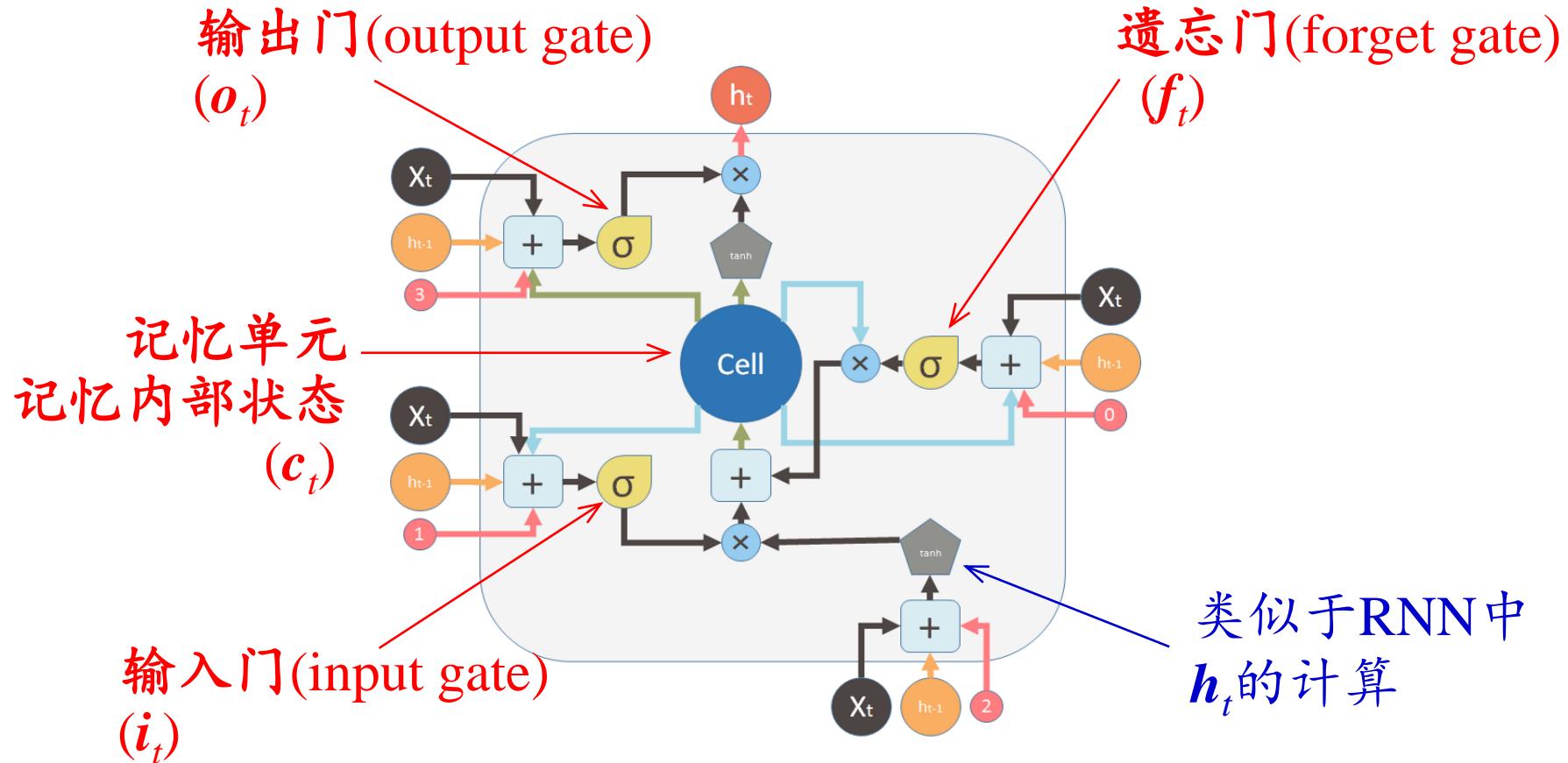
长短时记忆网络LSTM (Long-Short Term Memory)。

本章内容

1. 问题提出
2. 神经网络概述
3. 前馈神经网络及语言模型
4. 循环神经网络及语言模型
- 5. 长时短时记忆网络**
6. 注意力机制
7. GPT模型
8. 习题
9. 附录

5. 长时短时记忆网络

◆LSTM描述



LSTM 又称基于门控机制的循环神经网络。

5. 长时短时记忆网络

- 内部状态： $c_t = \mathbb{R}^D$ 进行线性的循环信息传递，同时（非线性地）输出信息给隐藏层的外部状态 $h_t = \mathbb{R}^D$ 。 c_t 的计算公式如下：

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

$$h_t = o_t \odot \tanh(c_t)$$

其中， $f_t \in [0, 1]^D$, $i_t \in [0, 1]^D$ 和 $o_t \in [0, 1]^D$ 控制传递信息的路径；
 \odot 表示向量元素的乘积； c_{t-1} 为上一时刻的记忆单元； $\tilde{c}_t \in \mathbb{R}^D$ 是通过非线性函数得到的候选状态：

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$$

RNN中的 h_t

在每一个时刻 t , LSTM 内部网络的内部状态 c_t 记录了到当前时刻为止的历史信息。



5. 长时短时记忆网络

● 门控机制 (0/1开关)

(1) 遗忘门 f_t : 控制上一时刻的内部状态 c_{t-1} 需要遗忘多少信息。

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

(2) 输入门 i_t : 控制当前时刻的候选状态 \tilde{c}_t 有多少信息需要保存。

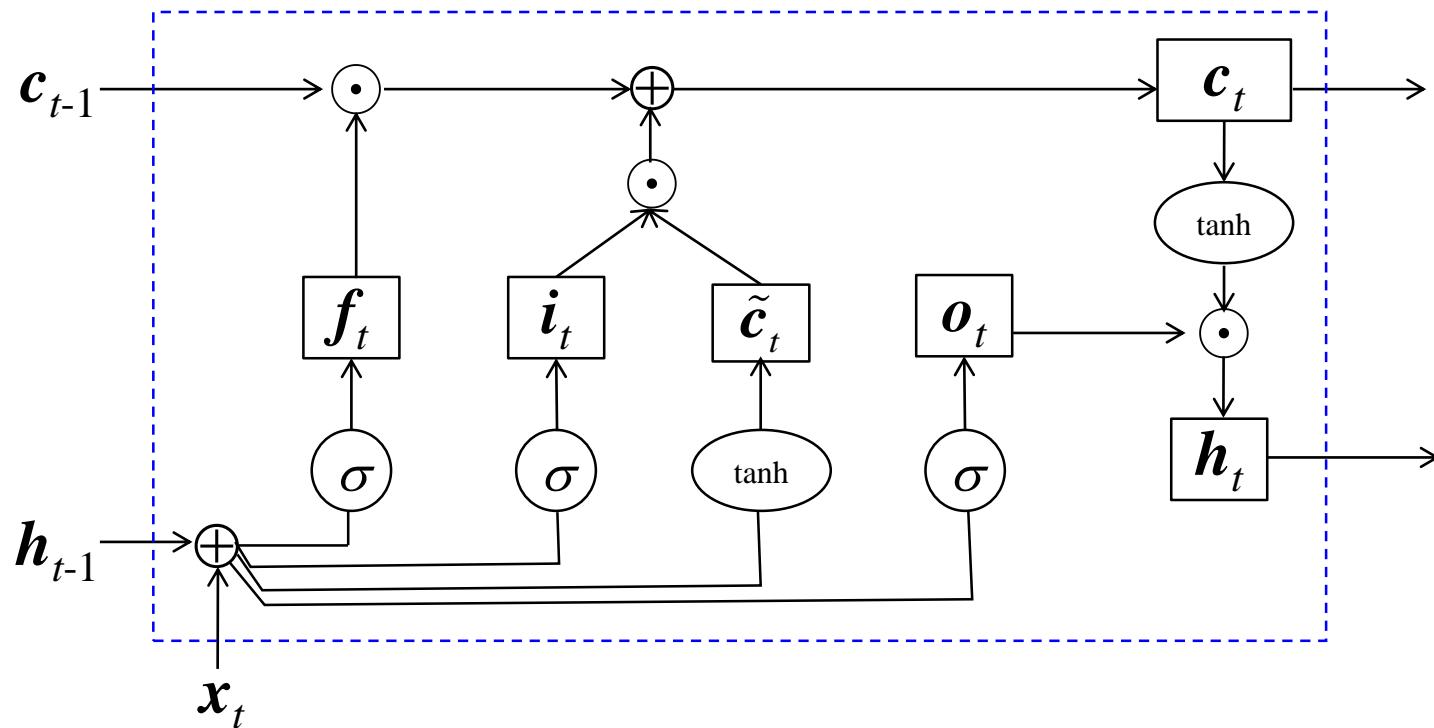
$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

(3) 输出门 o_t : 控制当前时刻的内部状态 c_t 有多少信息需要输出给外部状态 h_t 。

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

其中, $\sigma(\cdot)$ 是 Logistic 函数, 输出区间为 $(0,1)$, x_t 为当前时刻的输入, h_{t-1} 为上一时刻的外部状态。

5. 长时短时记忆网络

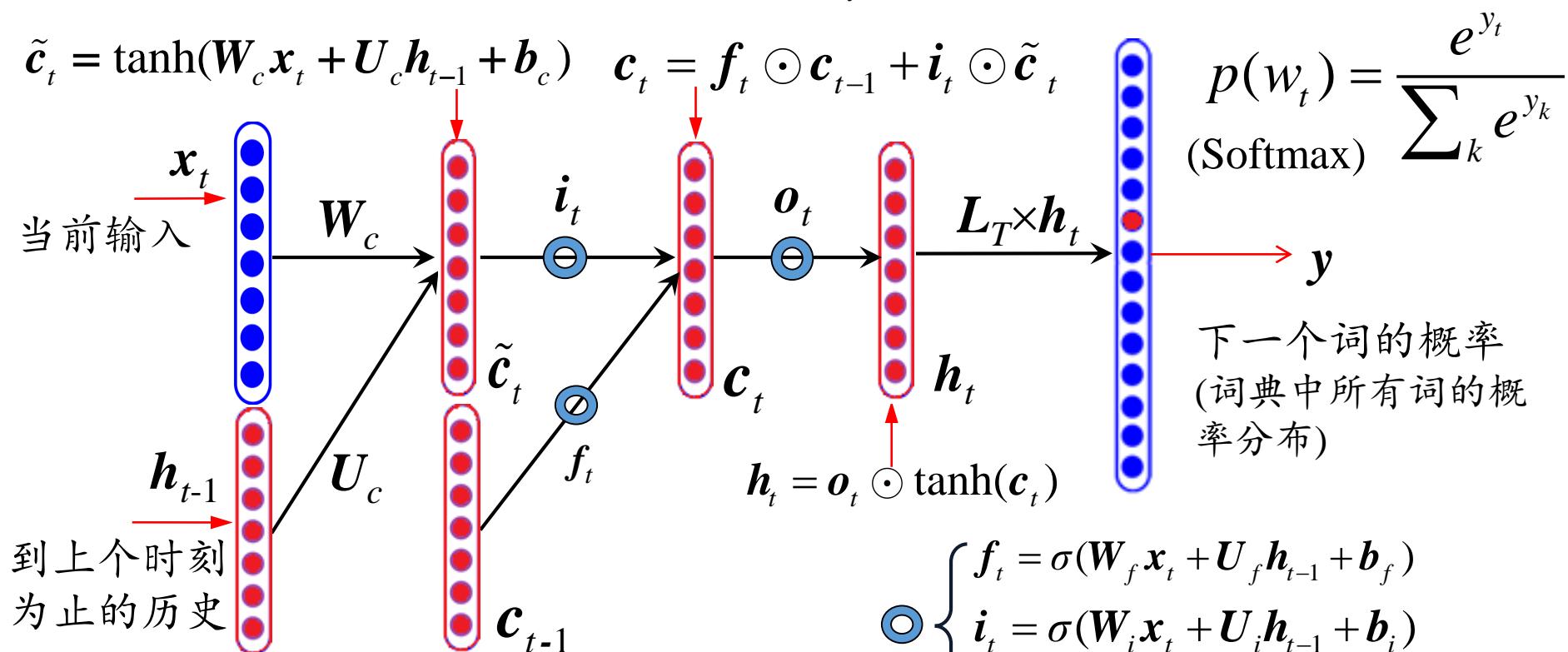


- (1) 利用上一时刻的外部状态 h_{t-1} 和当前时刻的输入 x_t , 计算出三个门 f_t, i_t, o_t 和候选状态 \tilde{c}_t ;
- (2) 结合遗忘门 f_t 和输入门 i_t , 更新记忆单元 c_t ;
- (3) 结合输出门 o_t , 将内部状态的信息传递给外部状态 h_t 。

5. 长时短时记忆网络

◆ 基于LSTM的语言模型

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



5. 长时短时记忆网络

(1) 计算三个门控值

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

(不考虑偏置 b 。)

$$x_3 = [0.2, 0.1]^T \text{ 书}$$

$$x_2 = [0.3, 0.1]^T \text{ 本}$$

$$x_1 = [0.2, 0.2]^T \text{ 这}$$

$$h_0 = [0.0, 0.0]^T$$

$$h_t = o_t \odot \tanh(c_t)$$

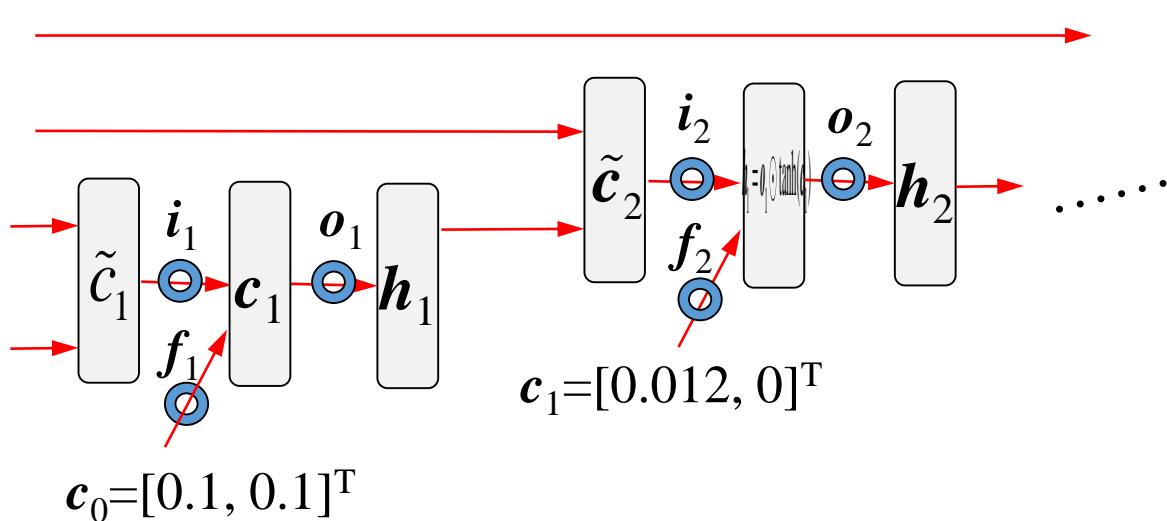
(2) 计算 \tilde{c}_1 , c_1 和 h_1

$$c_1 = f_1 \odot c_0 + i_1 \odot \tilde{c}_1$$

$$= \tanh\left(\begin{bmatrix} 0.0 & 0.1 \\ 0.1 & 0.0 \end{bmatrix} \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix} + \begin{bmatrix} 0.1 & 0.0 \\ 0.0 & 0.1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 0.02 \\ 0.02 \end{bmatrix}$$

$$c_1 = f_1 \odot c_0 + i_1 \odot \tilde{c}_1 = \begin{bmatrix} 0.1 \\ 0.0 \end{bmatrix} \odot \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0.0 \end{bmatrix} \odot \begin{bmatrix} 0.02 \\ 0.02 \end{bmatrix} = \begin{bmatrix} 0.012 \\ 0 \end{bmatrix}$$

$$\tilde{c}_1 = \tanh(W_x x_1 + W_h h_0) = \begin{bmatrix} 0.1 \\ 0.0 \end{bmatrix} \odot \tanh\left(\begin{bmatrix} 0.012 \\ 0 \end{bmatrix}\right)$$



工具与文献

◆开源工具

- LSTM语言模型(recurrent neural language model with LSTM unit)
<https://www-i6.informatik.rwth-aachen.de/web/Software/rwthlm.php>
- LSTM反向传播算法
<http://arunmallya.github.io/writeups/nn/lstm/index.html#/>

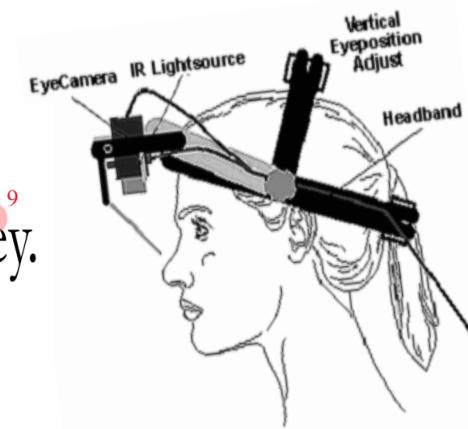
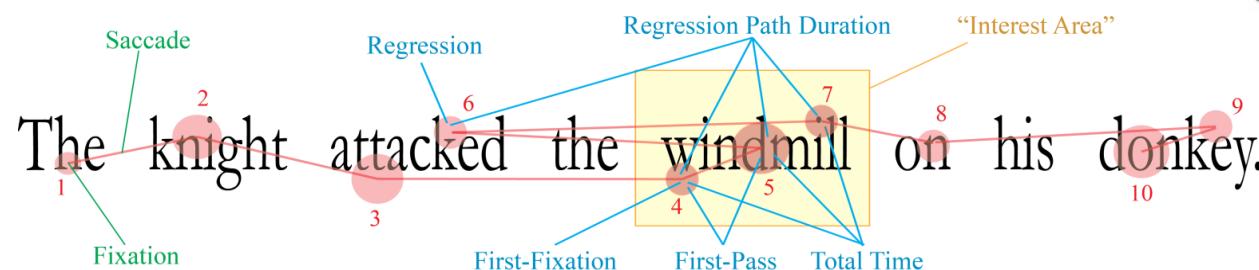
◆代表论文

- [1] Sepp Hochreiter, Jürgen Schmidhuber, 1997. Long short-term memory.
Neural computation, 1997, 9(8): 1735-1780.
- [2] Sepp Hochreiter, Jürgen Schmidhuber, 1997. LSTM can Solve Hard Long Time Lag Problems, *Advances in Neural Information Processing Systems*.
- [3] Felix A. Gers and Jürgen Schmidhuber, 2001. LSTM Recurrent Networks Learn Simple Context Free and Context Sensitive Languages. *IEEE Transactions on Neural Networks*. 12 (6): 1333–1340.

5. 长时短时记忆网络

◆ 问题分析

给定前面 $t-1$ 个单词(x_1, x_2, \dots, x_{t-1})预测当前单词 x_t 时，前面每个词的重要程度是一样的。而人的阅读语言句子时对于不同的词汇关注程度是不一样的。



例如：

Reading time/word	the	two	young	sea-lions	took
Rtfpass (第1遍时间)	27.2	138.7	155.5	314.8	169.3
Rtgopast (回看时间)	27.2	138.7	178.4	426.1	180.9
RTrb (右边界时间)	27.2	138.7	155.5	339.2	169.3

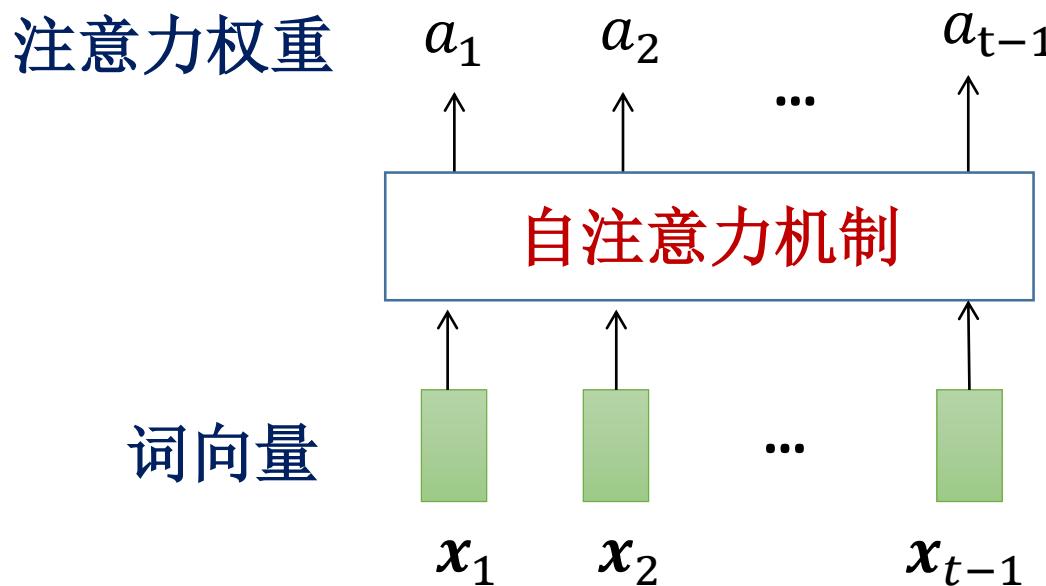
本章内容

1. 问题提出
2. 神经网络概述
3. 前馈神经网络及语言模型
4. 循环神经网络及语言模型
5. 长时短时记忆网络
-  6. 注意力机制
7. GPT模型
8. 习题
9. 附录

6. 自注意力机制

◆ 注意力机制的动机

对前 $t-1$ 个词(x_1, x_2, \dots, x_{t-1})的重要程度设置权重:
(a_1, a_2, \dots, a_{t-1})。



6. 自注意力机制

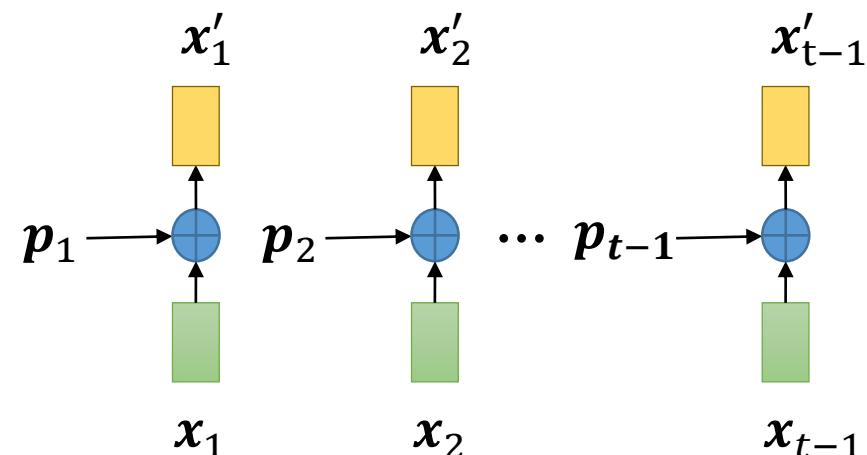
◆ 注意力机制的执行过程

Step 1: 查询 $t-1$ 个单词的词向量(x_1, x_2, \dots, x_{t-1})，添加位置向量 (p_1, p_2, \dots, p_{t-1})，计算方式（直接相加）：

$$(x'_1, x'_2, \dots, x'_{t-1}) = (x_1, x_2, \dots, x_{t-1}) + (p_1, p_2, \dots, p_{t-1})$$

向量位置如何确定？

- (1)按照规则确定；
- (2)看作参数，随机初始化，根据数据进行优化。



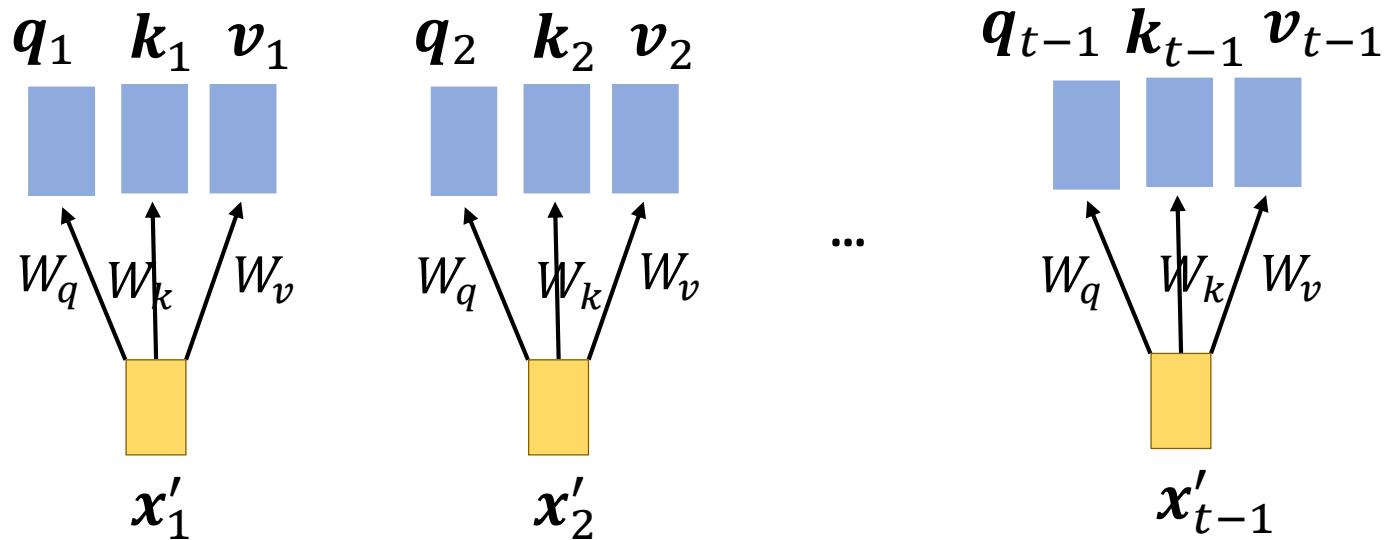
6. 自注意力机制

Step 2: 根据每个单词的表示($x'_1, x'_2, \dots, x'_{t-1}$)，计算对应的查询向量 q 、键向量 k 和值向量 v ，计算公式：

$$q_i = W_q x'_i$$

$$k_i = W_k x'_i$$

$$v_i = W_v x'_i$$



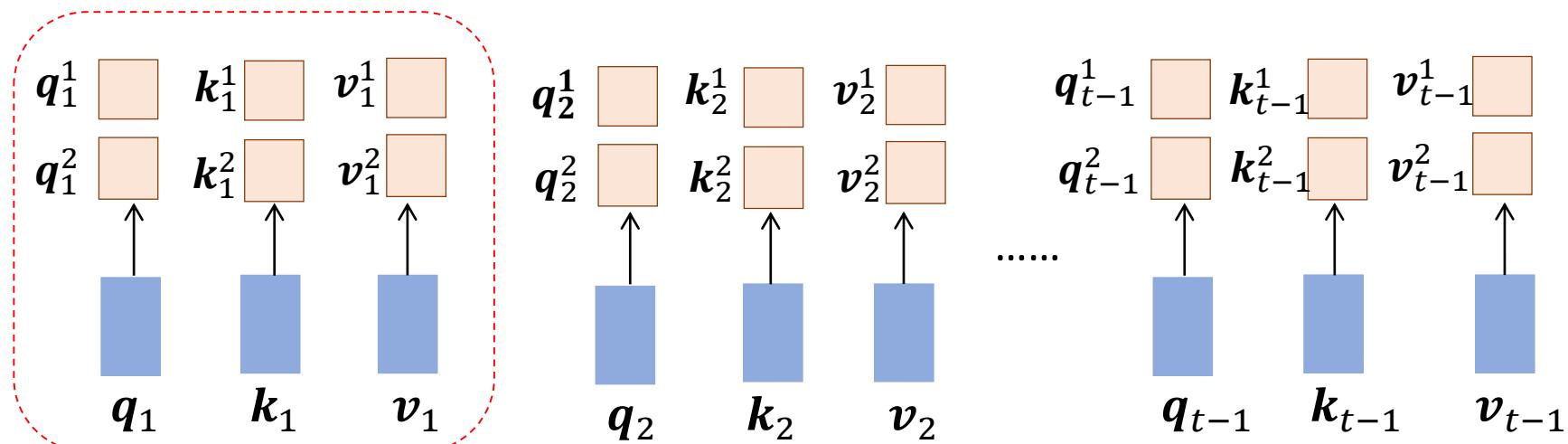
6. 自注意力机制

Step 3: 将查询向量 q 、键向量 k 和值向量 v 分解为多个向量（称为“多头”(multi-head)），分解过程如下(以两个头为例):

$$q_i = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.5 \end{bmatrix} \quad q_i^1 = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix} \quad q_i^2 = \begin{bmatrix} 0.2 \\ 0.5 \end{bmatrix}$$

$$k_i \quad k_i^1 \quad k_i^2$$

$$v_i \quad v_i^1 \quad v_i^2$$

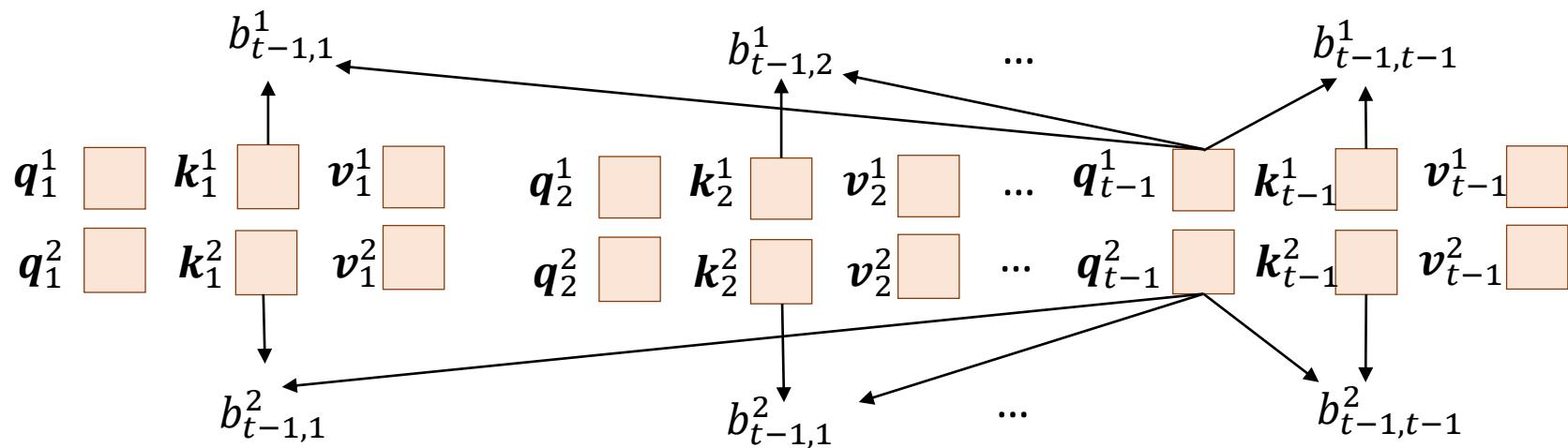


6. 自注意力机制

Step 4: 对每个头分别计算其注意力权重，计算方式为： $b = \frac{q^T k}{\sqrt{d_k}}$ 。

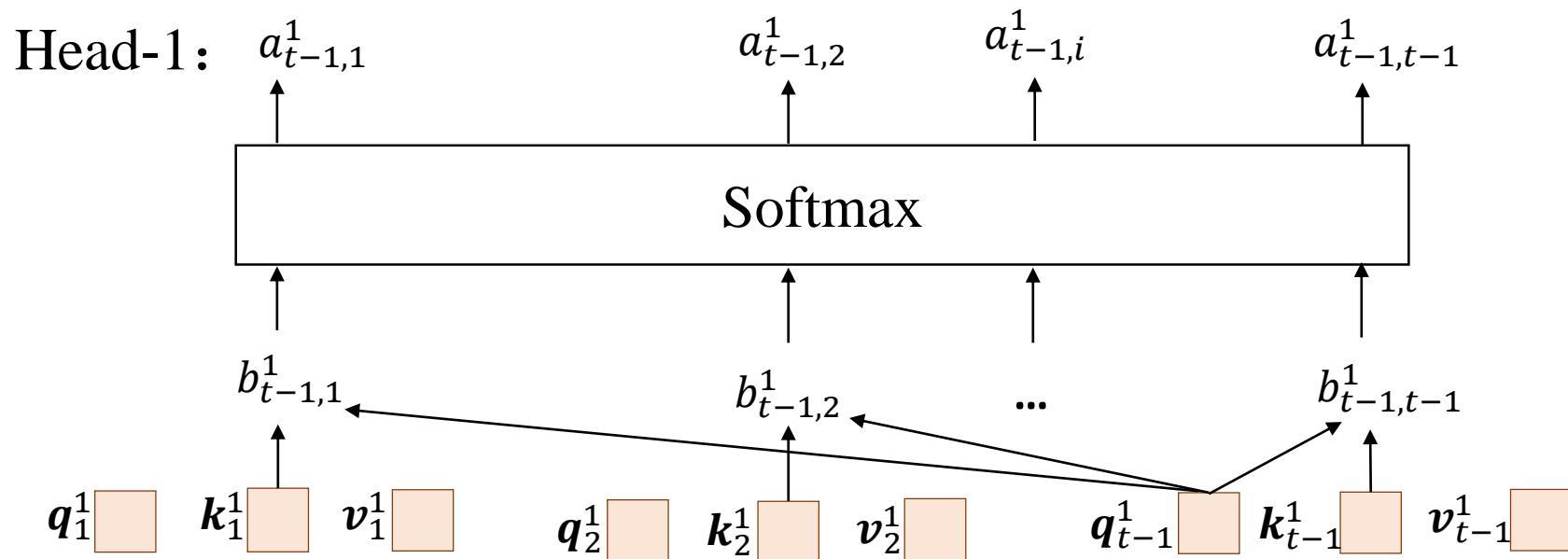
其中， d_k 为 q 和 k 的维度。在上一页PPT中，每个头的 q 和 k 为2维的向量，那么分母就是 $\sqrt{2}$ 。

在语言模型中， q 取 $t-1$ 时刻的查询向量， k 为前 $t-1$ 的所有键向量。



6. 自注意力机制

Step 5: 将所有注意力权重，归一化处理（Softmax函数）。

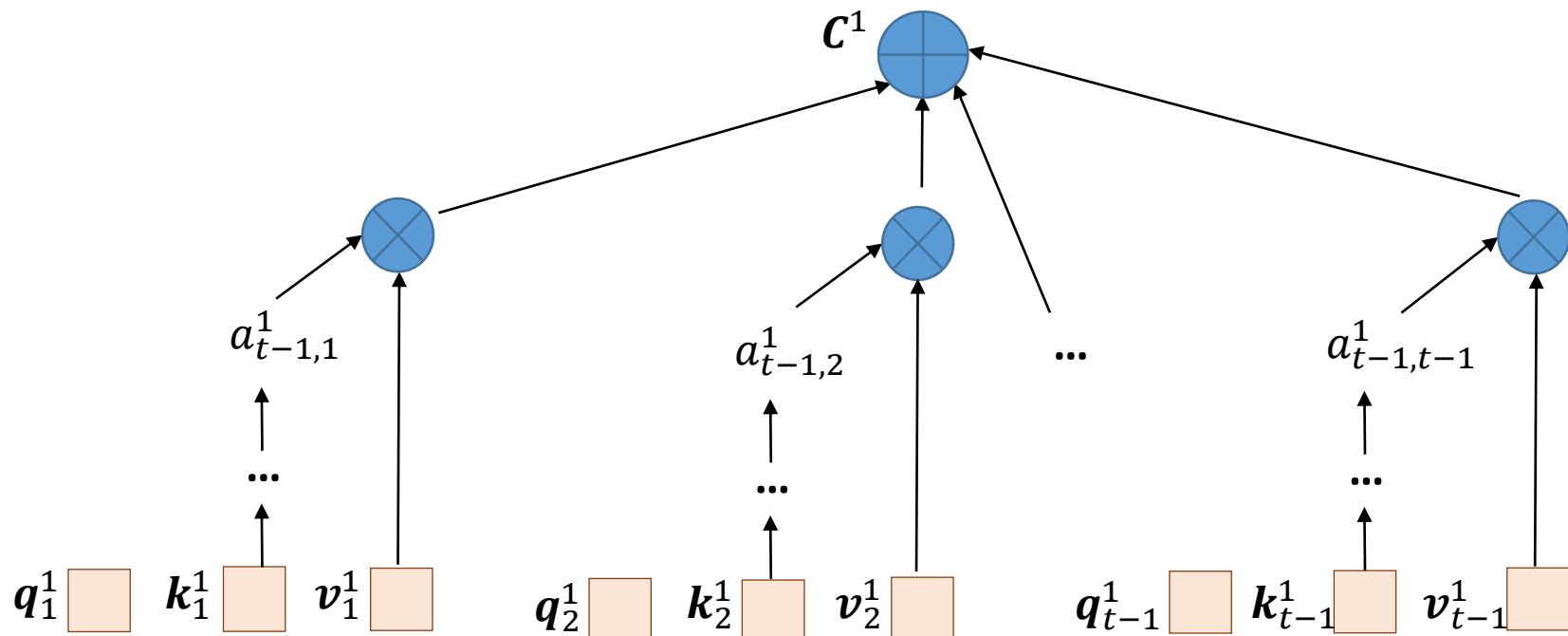


(图中仅画出头1的计算过程)

6. 自注意力机制

Step 6: 根据归一化的注意力权重 a , 聚合前 $t-1$ 个词的信息:

$$\text{Head-1: } c^1 = \sum_{i=1}^{t-1} a_{t-1,i} v_i = a_{t-1,1} v_1 + a_{t-1,2} v_2 + \cdots + a_{t-1,t-1} v_{t-1}$$



(图中仅画出头1的计算过程)

6. 自注意力机制

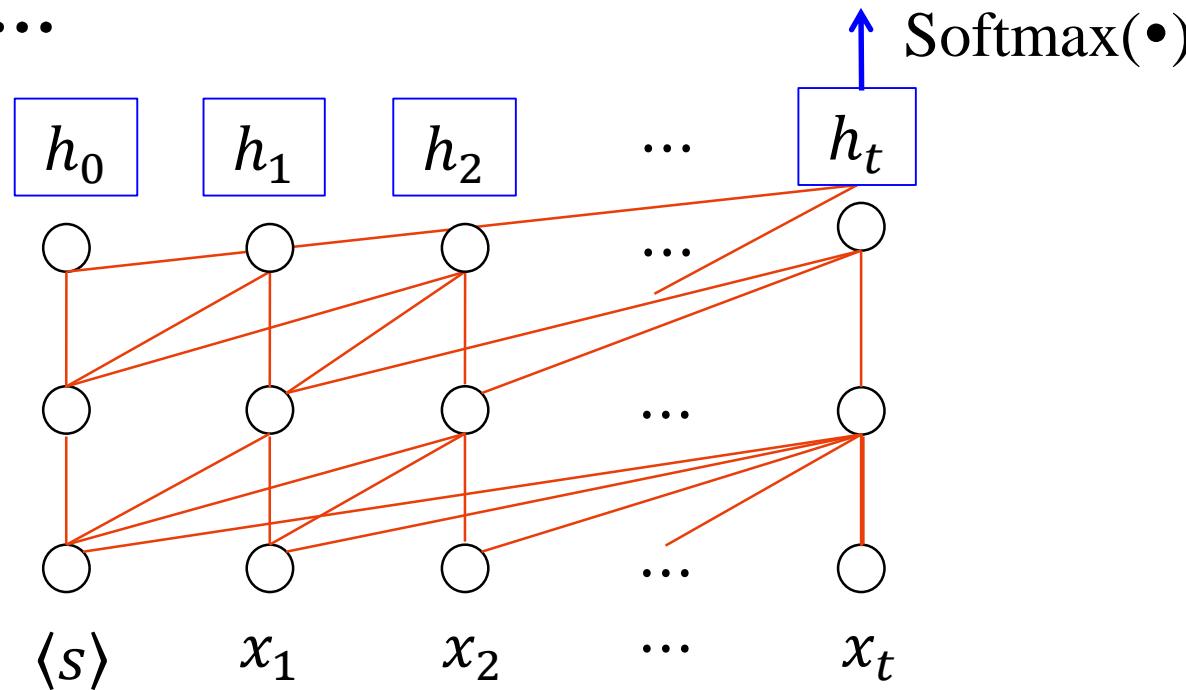
Step 7: 将不同头 (Head-1, Head-2 ...) 的语义向量进行拼接，拼接后的语义向量则包含了前 $t-1$ 个单词的信息。

$$\begin{array}{ccc} \mathbf{c}_{t-1}^1 & \xrightarrow{\quad} & \mathbf{c}_{t-1} \\ \mathbf{c}_{t-1}^2 & \xrightarrow{\quad} & \end{array}$$

Step 8: 利用 \mathbf{c}_{t-1} 对所有单词进行打分（内积和Softmax函数），得到最终的预测结果。

6. 自注意力机制

◆ 相当于…



◆ 代表论文：

- A. Vaswani et al. [Attention Is All You Need](#). In *Proceedings of the 31st Conference on Neural Information Processing Systems* (NIPS'2017), December 4-7, 2017, Long Beach, CA, USA, Pages 6000–6010

本章内容

1. 问题提出
2. 神经网络概述
3. 前馈神经网络及语言模型
4. 循环神经网络及语言模型
5. 长时短时记忆网络
6. 注意力机制
-  7. GPT模型
8. 习题
9. 附录



7. GPT模型

◆GPT模型的结构

GPT: Generative Pre-Training

预训练语言模型

- 模型结构: 自注意力机制
- 自监督学习任务: 给定前 $t-1$ 个单词, 预测第 t 个单词。
- 过程: 自注意力机制的 Step-1 ~ Step-8。

7. GPT模型

◆参考文献与开源工具

• GPT

论文: Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

网址: <https://www.cs.ubc.ca/~amuhamed/LING530/papers/radford2018improving.pdf>

• GPT-2

论文: Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.

网址: <https://github.com/openai/gpt-2>

• GPT-3

论文: Tom Brown, Benjamin Mann et al. 2020. Language models are few-shot learners. Advances in neural information processing systems, 33, 1877-1901.

网址: <https://github.com/openai/gpt-3>

TensorFlow 和 PyTorch 框架

- TensorFlow: <https://www.tensorflow.org/>

研发者: 谷歌人工智能团队谷歌大脑(Google Brain) 团队

- Pytorch: <https://pytorch.org/>

研发者: Facebook人工智能研究院(FAIR)

- 说明:

TensorFlow 和 Pytorch 等深度学习框架能够提供两方面功能:

- ① 提供GPU加速;
- ② 能够对神经网络的参数进行自动求导和优化。

本章内容

1. 问题提出
2. 神经网络概述
3. 前馈神经网络及语言模型
4. 循环神经网络及语言模型
5. 长时短时记忆网络
6. 注意力机制
7. GPT模型
-  8. 习题
9. 附录



8. 习题

请下载调试FNN、RNN和LSTM模型的开源工具。利用北京大学标注的《人民日报》1998年1月份的分词语料，或者利用网络爬虫自己从互联网上收集足够多的英文文本语料，借助FNN 或者 RNN/LSTM 开源工具，完成如下任务：

- (1) 获得汉语或英语词语的词向量。
- (2) 在任务(1)的基础上，随机选择20个单词，计算与其词向量最相似的前10个单词。
- (3) 对于同一批词汇，对比分别用FNN, RNN 或 LSTM获得的词向量的差异。

8. 习题

● 作业要求：

完成一份实验报告（3周时间）。

● 说明

- 由于同学们计算资源的限制，神经网络参数不用选择的过大，例如：词表选择1000个左右单词即可，其余单词用<UNK>代替；词向量的维度设置为10左右；神经网络的层数设置为1到2层；
- 可以使用某一种开放的深度学习框架，如TensorFlow 或者 PyTorch。
- 如果不借助开源工具和开放的深度学习框架，题目中的任务(3)可以不做。

本章小结

◆人工神经网络简介

基本概念； 网络描述； 激活函数； Softmax回归

◆FNN 及语言模型

FNN描述； 参数训练（反向传播算法）； 基于FNN的语言模型

◆RNN 及语言模型

RNN描述； 基于RNN的语言模型

◆LSTM 及语言模型

LSTM 描述； 基于LSTM的语言模型

◆注意力机制

提出注意力机制的动机； 执行过程

◆GPT模型的结构

本章内容

1. 问题提出
2. 神经网络概述
3. 前馈神经网络及语言模型
4. 循环神经网络及语言模型
5. 长时短时记忆网络
6. 注意力机制
7. GPT模型
8. 习题
9. 附录



9. 附录

◆ 反向传播算法的推导

不失一般性，对第 l 层中的参数 $\mathbf{W}^{(l)}$ 和 $\mathbf{b}^{(l)}$ 求偏导。由于 $\frac{\partial \mathcal{L}(\mathbf{y}^{(n)}, \hat{\mathbf{y}}^{(n)})}{\partial \mathbf{W}^{(l)}}$ 的计算涉及向量对矩阵的微分，十分繁琐，因此我们先计算 $\mathcal{L}(\mathbf{y}^{(n)}, \hat{\mathbf{y}}^{(n)})$ 关于参数矩阵中每个元素的偏导： $\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial w_{ij}^{(l)}}$ 。

$$\text{根据链式法则: } \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial w_{ij}^{(l)}} = \boxed{\frac{\partial \mathbf{z}^{(l)}}{\partial w_{ij}^{(l)}}} \boxed{\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}}} \quad (6-6)$$

$$\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{b}^{(l)}} = \boxed{\frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{b}^{(l)}}} \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}} \quad (6-7)$$

由于上面两个式子中第二项都是目标函数关于第 l 的神经元 $\mathbf{z}^{(l)}$ 的偏导数，称为**误差项**，可以一次计算得到，因此，只需要计算三个偏导数（红框所示）。

9. 附录

(1) 计算 $\frac{\partial \mathbf{z}^{(l)}}{\partial w_{ij}^{(l)}}$

回顾: $\mathbf{z}^{(l)} = \mathbf{W}^{(l)} \cdot \boldsymbol{\alpha}^{(l-1)} + \mathbf{b}^{(l)}$, 那么,

$$\begin{aligned}
 \frac{\partial \mathbf{z}^{(l)}}{\partial w_{ij}^{(l)}} &= \left[\frac{\partial z_1^{(l)}}{\partial w_{ij}^{(l)}}, \dots, \frac{\partial z_i^{(l)}}{\partial w_{ij}^{(l)}}, \dots, \frac{\partial z_{M_l}^{(l)}}{\partial w_{ij}^{(l)}} \right] \\
 &= \left[0, \dots, \frac{\partial (\mathbf{w}_{i:}^{(l)} \cdot \boldsymbol{\alpha}^{(l-1)} + b_i^{(l)})}{\partial w_{ij}^{(l)}}, \dots, 0 \right] \\
 &= \left[0, \dots, \alpha_j^{(l-1)}, \dots, 0 \right] \\
 &\triangleq \Delta_i(\boldsymbol{\alpha}_j^{(l-1)}) \tag{6-8}
 \end{aligned}$$

其中, $\mathbf{w}_{i:}^{(l)}$ 为权重矩阵 $\mathbf{W}^{(l)}$ 的第 i 行, $\Delta_i(\boldsymbol{\alpha}_j^{(l-1)})$ 表示第 i 个元素为 $\alpha_j^{(l-1)}$, 其余为 0 的行向量。

9. 附录

(2) 计算 $\frac{\partial z^{(l)}}{\partial b^{(l)}}$

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

所以 $\frac{\partial z^{(l)}}{\partial b^{(l)}} = I_{M_l} \in \mathbb{R}^{M_l \times M_l}$ ($M_l \times M_l$ 的单位矩阵) (6-9)



9. 附录

(3) 计算 $\frac{\partial \mathcal{L}(y, \hat{y})}{\partial z^{(l)}}$

该偏导数表示第 l 层神经元对最终损失的影响，也反映了最终损失对第 l 层神经元的敏感程度，因此一般将其称为第 l 层神经元的误差项，用 $\delta^{(l)}$ 表示。

$$\delta^{(l)} \triangleq \frac{\partial \mathcal{L}(y, \hat{y})}{\partial z^{(l)}} \in \mathbb{R}^{M_l} \quad (6-10)$$

误差项也间接地反映了不同神经元对网络能力的贡献程度，从而较好地解决了贡献度的分配问题(credit assignment problem, CAP)。

9. 附录

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

$$\frac{\partial z^{(l)}}{\partial \alpha^{(l-1)}} = (W^{(l)})^T \in \mathbb{R}^{M_{l-1} \times M_l} \quad (6-11)$$

根据 $\alpha^{(l)} = f_l(z^{(l)})$, 其中 $f_l(\bullet)$ 为按位计算的函数, 因此有

$$\begin{aligned} \frac{\partial \alpha^{(l)}}{\partial z^{(l)}} &= \frac{\partial f_l(z^{(l)})}{\partial z^{(l)}} \\ &= \text{diag}(f'_l(z^{(l)})) \in \mathbb{R}^{M_l \times M_l} \end{aligned} \quad (6-12)$$

9. 附录

因此，根据链式法则，第 l 层神经元的误差项为：

$$\begin{aligned}
 \delta^{(l)} &\triangleq \frac{\partial \mathcal{L}(y, \hat{y})}{\partial z^{(l)}} \\
 &= \frac{\partial a^{(l)}}{\partial z^{(l)}} \times \frac{\partial z^{(l+1)}}{\partial a^{(l)}} \times \frac{\partial \mathcal{L}(y, \hat{y})}{\partial z^{(l+1)}} \\
 &= \text{diag}(f_l'(z^{(l)})) \times (W^{(l+1)}) \times \delta^{(l+1)} \\
 &= f_l'(z^{(l)}) \odot ((W^{(l+1)})^T \cdot \delta^{(l+1)}) \quad \in \mathbb{R}^{M_l}
 \end{aligned} \tag{6-13}$$

其中， \odot 是向量的点积运算符，表示每个元素相乘。

通过上式可以看出，第 l 层的误差项可以通过第 $l+1$ 层的误差项计算得到，即误差反向传播，含义是：第 l 层的一个神经元的误差项(或敏感性)是所有与该神经元相连的第 $l+1$ 层的神经元的误差项的权重和，然后再乘上该神经元激活函数的梯度。

9. 附录

得到上面的三个偏导数之后，前面的(6-6)式可以做如下改写：

$$\begin{aligned}
 \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial w_{ij}^{(l)}} &= \frac{\partial \mathbf{z}^{(l)}}{\partial w_{ij}^{(l)}} \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}} \\
 &= \Delta_i(\alpha_j^{(l-1)}) \cdot \delta^{(l)} \\
 &= [0, \dots, \alpha_j^{(l-1)}, \dots, 0] \cdot [\delta_1^{(l)}, \dots, \delta_i^{(l)}, \dots, \delta_{M_l}^{(l)}] \\
 &= \delta_i^{(l)} \cdot \alpha_j^{(l-1)}
 \end{aligned} \tag{6-14}$$

其中， $\delta_i^{(l)} \cdot \alpha_j^{(l-1)}$ 相当于向量 $\delta_i^{(l)}$ 和向量 $\alpha^{(l-1)}$ 的外积的第*i,j*个元素。

9. 附录

上面的(6-14)式可以进一步改写为：

$$\left[\frac{\partial \mathcal{L}(y, \hat{y})}{\partial \mathbf{W}^{(l)}} \right]_{ij} = [\delta^{(l)}(\boldsymbol{\alpha}^{(l-1)})^T]_{ij} \quad (6-15)$$

因此, $\mathcal{L}(y, \hat{y})$ 关于第 l 层权重 $\mathbf{W}^{(l)}$ 的梯度为：

$$\frac{\partial \mathcal{L}(y, \hat{y})}{\partial \mathbf{W}^{(l)}} = \delta^{(l)}(\boldsymbol{\alpha}^{(l-1)})^T \in \mathbb{R}^{M_l \times M_{l-1}} \quad (6-16)$$

同理, $\mathcal{L}(y, \hat{y})$ 关于第 l 层偏置 $\mathbf{b}^{(l)}$ 的梯度为：

$$\frac{\partial \mathcal{L}(y, \hat{y})}{\partial \mathbf{b}^{(l)}} = \delta^{(l)} \in \mathbb{R}^{M_l} \quad (6-17)$$

9. 附录

计算出每一层的误差项之后，就可以得到每一层参数的梯度，因此，使用误差反向传播算法的前馈神经网络训练过程可以分为以下三步：

- (a) 前馈计算每一层的净输入 $z^{(l)}$ 和激活值 $\alpha^{(l)}$ ，直到最后一层；
- (b) 反向传播计算每一层的误差项 $\delta^{(l)}$ ；
- (c) 计算每一层参数的偏导数，更新参数。



9. 附录

● 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 $\mathcal{D} = \{(x^{(n)}, y^{(n)})\}_{n=1}^N$, 验证集 v, 学习率 ℓ , 正则化系数 ε , 网络层数 L , 神经元数量 M_l , $1 \leq l \leq L$ 。

输出: W, b

重复执行
该过程:

(1) 对训练集 D 中的样本随机重排序;
 (2) for $n=1 .. N$ do
 从训练集 D 中选取样本 $(x^{(n)}, y^{(n)})$;
 前馈计算每一层的净输入 $z^{(l)}$ 和激活值 $\alpha^{(l)}$, 直到最后一层;
 反向传播计算每一层的误差 $\delta^{(l)}$; 公式(6-13)

// 计算每一层参数的导数: $\left\{ \begin{array}{l} \forall l, \frac{\partial \mathcal{L}(y, \hat{y})}{\partial W^{(l)}} = \delta^{(l)} (\alpha^{(l-1)})^T \\ \forall l, \frac{\partial \mathcal{L}(y, \hat{y})}{\partial b^{(l)}} = \delta^{(l)} \end{array} \right.$ 公式(6-16)

 公式(6-17)

// 更新参数: $\left\{ \begin{array}{l} W^{(l)} \leftarrow W^{(l)} - \ell (\delta^{(l)} \cdot (\alpha^{(l-1)})^T + \varepsilon W^{(l)}) \\ b^{(l)} \leftarrow b^{(l)} - \delta^{(l)}; \end{array} \right.$

end

直到模型在验证集 v 上的损失函数值收敛, 结束过程, 输出 W, b 。

谢谢！

Thanks!

