3.2 神经网络语言模型
(RNN/LSTM/GRU)

林洲汉

上海交通大学2023年秋季学期

目录

目录

日常生活中的语言模型

语言模型基本概念

基本任务：已知前文预测下一个字

- 输入：单词序列$w_1, w_2, \cdots, w_{k-1}$
- 输出：对下一个单词预测的概率$P(w_k|w_1, \cdots, w_{k-1})$

语言模型基本概念

基本任务：已知前文预测下一个字

- 输入：单词序列$w_1, w_2, \cdots, w_{k-1}$
- 输出：对下一个单词预测的概率$P(w_k|w_1, \cdots, w_{k-1})$

另一个角度：推测句子的合理程度

- $\prod_{k=1}^{K} P(w_k|w_1, \cdots, w_{k-1}) = P(w_1, w_2, \cdots, w_K)$

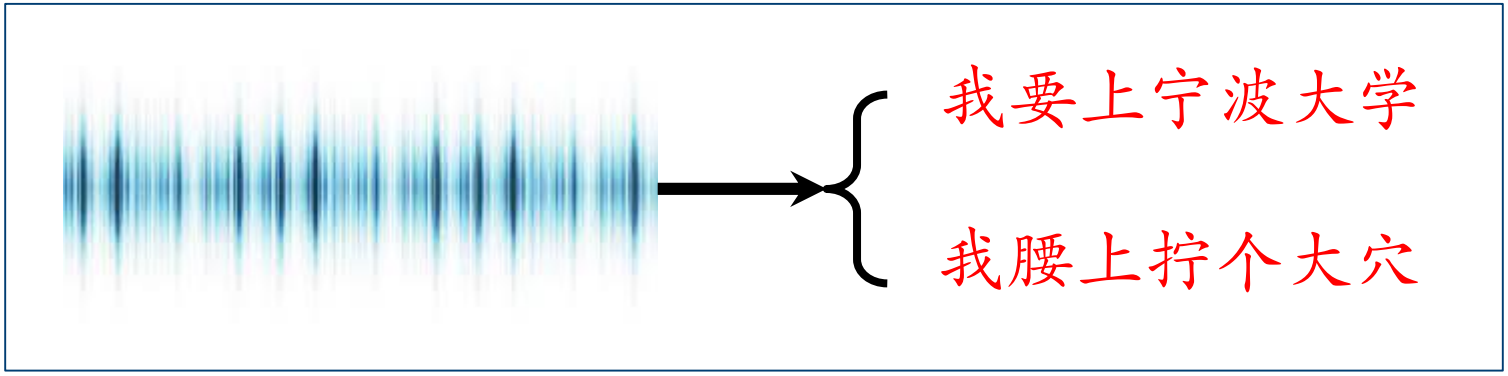
语言模型基本概念

基本任务：已知前文预测下一个字

- 输入：单词序列$w_1, w_2, \cdots, w_{k-1}$
- 输出：对下一个单词预测的概率$P(w_k|w_1, \cdots, w_{k-1})$

另一个角度：推测句子的合理程度

- $\prod_{k=1}^{K} P(w_k|w_1, \cdots, w_{k-1}) = P(w_1, w_2, \cdots, w_K)$



目录

统计语言模型回顾：n-gram

- 马尔科夫假设：每个字出现的概率仅与其前面出现的$n-1$个字有关。
- 基本思想：用出现频率代替出现概率
- 公式：

$$P(w_k|w_1,\cdots,w_{k-1}) \approx P(w_k|w_{k-n+1},\cdots,w_{k-1}) = \frac{P(w_{k-n+1},\cdots,w_{k-1},w_k)}{P(w_{k-n+1},\cdots,w_{k-1})}$$

- 实现：
  - 对数据集中的所有n-gram, (n-1)-gram计数并存储
  - $\frac{P(w_{k-n+1},\cdots,w_{k-1},w_k)}{P(w_{k-n+1},\cdots,w_{k-1})} = \frac{\text{Count}(w_{k-n+1},\cdots,w_{k-1},w_k)}{\text{Count}(w_{k-n+1},\cdots,w_{k-1})}$

n-gram模型的不足

- Storage Problem
  - 模型需要存储的数据规模为$|\mathcal{V}|^n$量级
  - 不能建模远距依赖关系

n-gram模型的不足

- Storage Problem
  - 模型需要存储的数据规模为$|\mathcal{V}|^n$量级
  - 不能建模远距依赖关系

He grew up in France, so he speaks French fluently.

二元n-gram语言模型：

p(French | He grew up in France, so he speaks) = p(French | speaks)

p(Chinese | He grew up in France, so he speaks) = p(Chinese | speaks)

n-gram模型的不足

- Storage Problem
  - 模型需要存储的数据规模为$|\mathcal{V}|^n$量级
  - 不能建模远距依赖关系
- Sparsity Problem
  - 低频次
  - 零概率
  - Discounting & Backing-off

目录

前馈神经网络语言模型

- 直接参考n-gram模型的方法，将历史词的词向量拼接起来送入前馈神经网络并计算下一个词的概率

前馈神经网络语言模型的计算流程

我　　超　　级　　喜　　欢　　这　　部

前馈神经网络语言模型的计算流程

喜　　欢　　这　　部

前馈神经网络语言模型的计算流程

各个字的单热向量

$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$

喜 $\boldsymbol{x}^{(1)}$　欢 $\boldsymbol{x}^{(2)}$　这 $\boldsymbol{x}^{(3)}$　部 $\boldsymbol{x}^{(4)}$

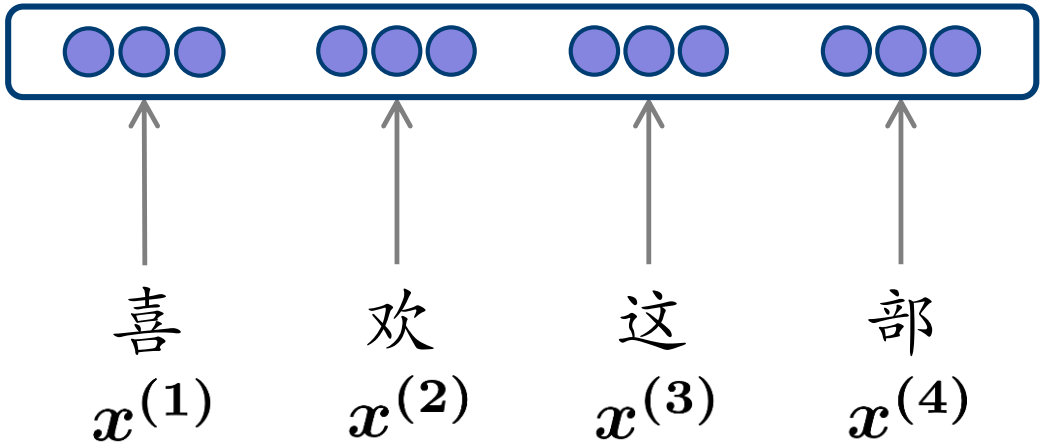
前馈神经网络语言模型的计算流程

查询拼接相应词向量

$$e = [e^{(1)}, e^{(2)}, e^{(3)}, e^{(4)}]$$

各个字的单热向量

$$x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}$$



前馈神经网络语言模型的计算流程

隐藏层

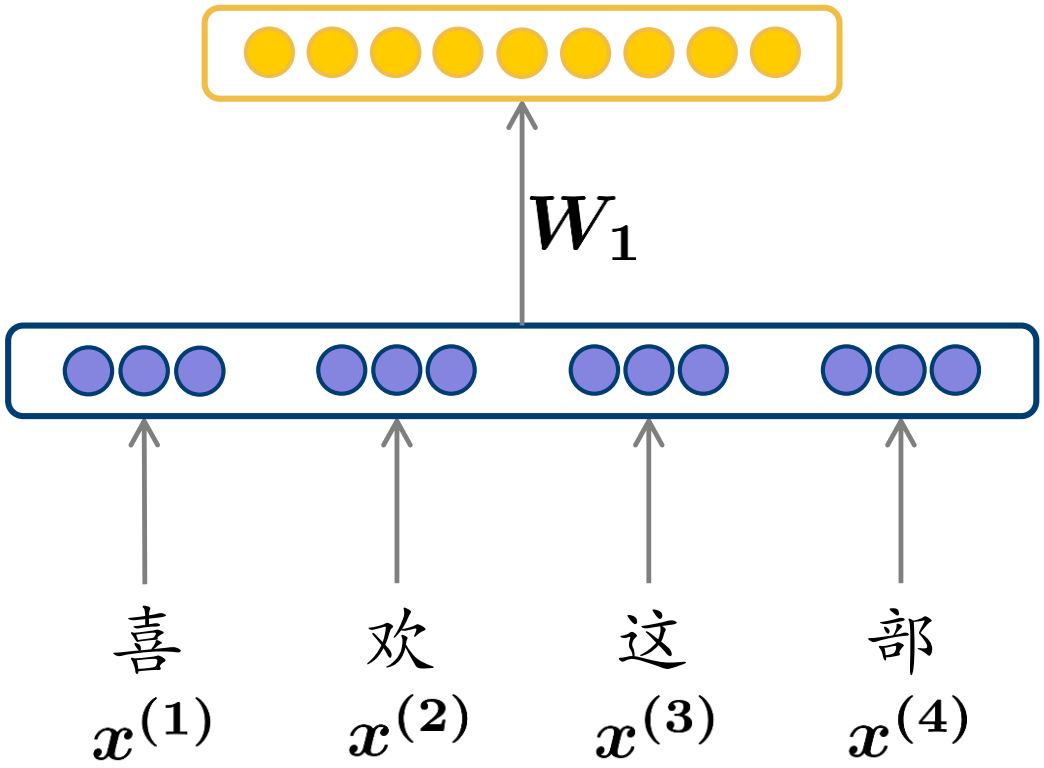
$$\boldsymbol{h} = f(\boldsymbol{W_1 e} + \boldsymbol{b_1})$$

查询拼接相应词向量

$$\boldsymbol{e} = [\boldsymbol{e}^{(1)}, \boldsymbol{e}^{(2)}, \boldsymbol{e}^{(3)}, \boldsymbol{e}^{(4)}]$$

各个字的单热向量

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$$



前馈神经网络语言模型的计算流程

输出概率分布

$$\hat{\boldsymbol{y}} = \text{Softmax}(\boldsymbol{W_2 h} + \boldsymbol{b_2})$$

隐藏层

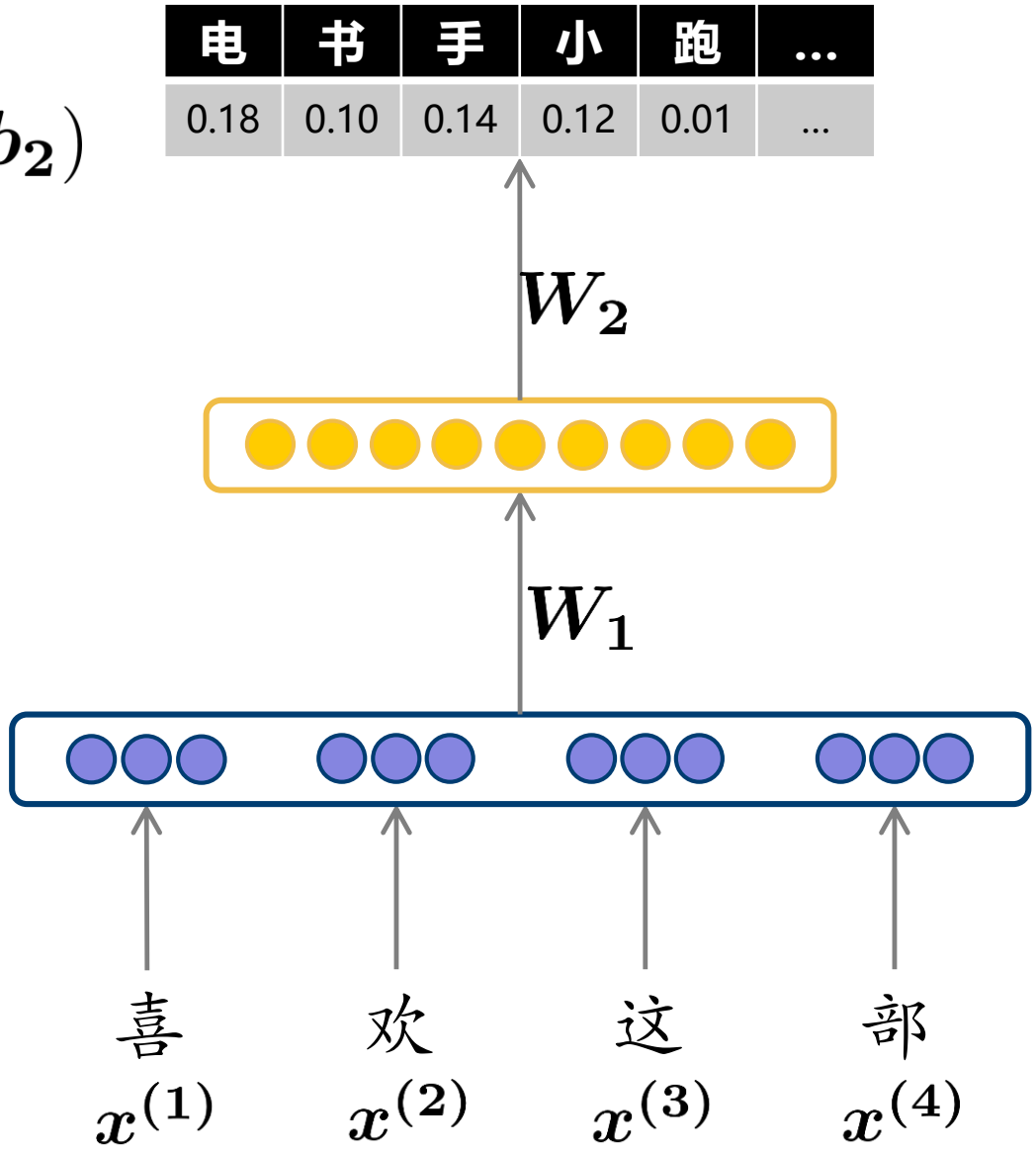
$$\boldsymbol{h} = f(\boldsymbol{W_1 e} + \boldsymbol{b_1})$$

查询拼接相应词向量

$$\boldsymbol{e} = [\boldsymbol{e}^{(1)}, \boldsymbol{e}^{(2)}, \boldsymbol{e}^{(3)}, \boldsymbol{e}^{(4)}]$$

各个字的单热向量

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$$



前馈神经网络语言模型

输出概率分布

$$\hat{\boldsymbol{y}} = \text{Softmax}(\boldsymbol{W_2 h} + \boldsymbol{b_2})$$

隐藏层

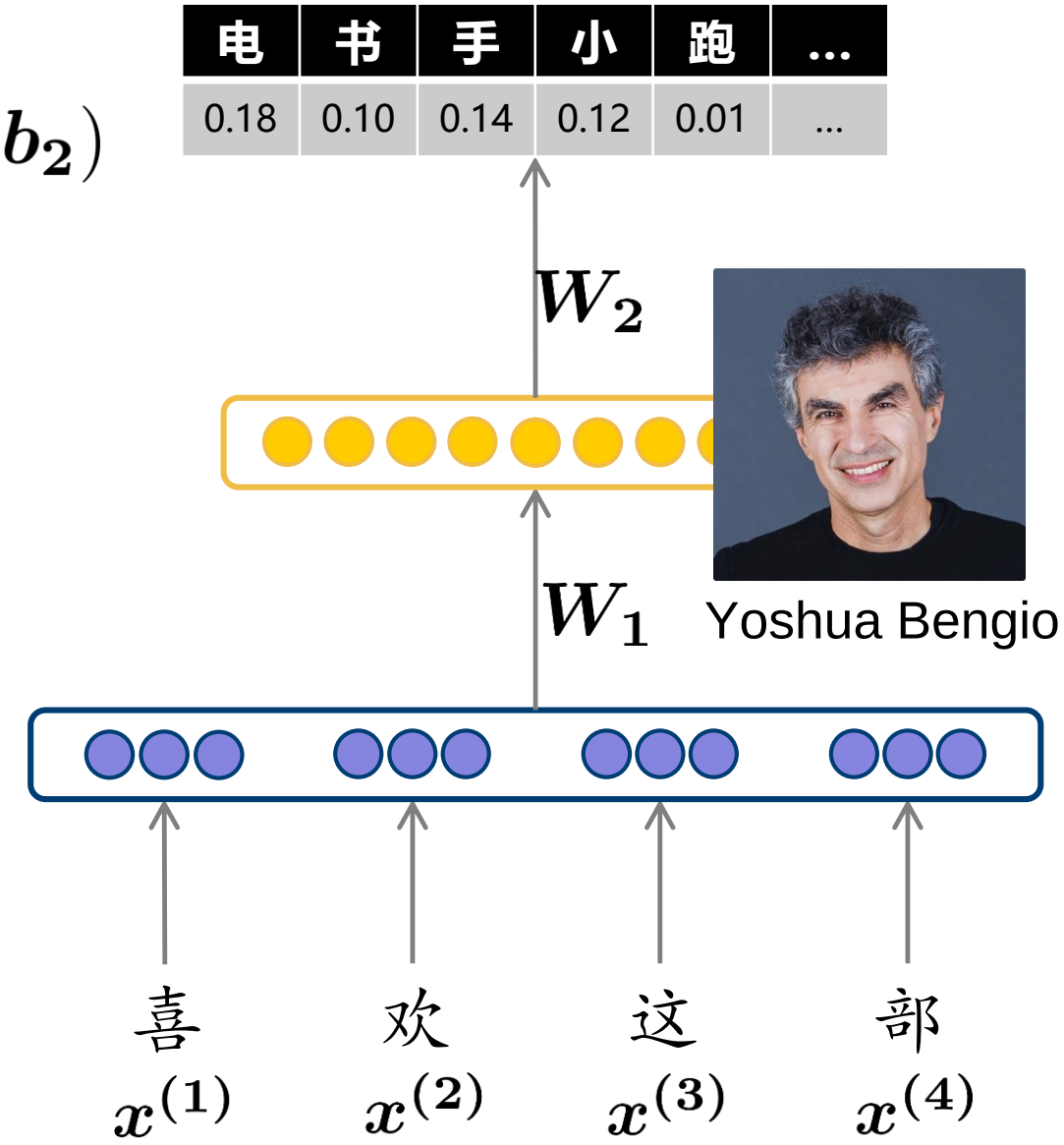
$$\boldsymbol{h} = f(\boldsymbol{W_1 e} + \boldsymbol{b_1})$$

查询拼接相应词向量

$$\boldsymbol{e} = [\boldsymbol{e}^{(1)}, \boldsymbol{e}^{(2)}, \boldsymbol{e}^{(3)}, \boldsymbol{e}^{(4)}]$$

各个字的单热向量

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$$



前馈神经网络语言模型的优点

- 通过学习单词分布式表示，有效缓解数据稀疏问题
- 避免维度灾难，相比n-gram模型占用更小的存储空间
- 可以并行计算不同时刻单词

前馈神经网络语言模型的问题

- 前馈神经网络需要保存的参数
  - 词向量矩阵 $\boldsymbol{E} \in \mathbb{R}^{|\mathcal{V}| \times d}$
  - 权重矩阵 $\boldsymbol{W_1} \in \mathbb{R}^{d_h \times (n \times d)}$
  - 输出矩阵 $\boldsymbol{W_2} \in \mathbb{R}^{|\mathcal{V}| \times d_h}$

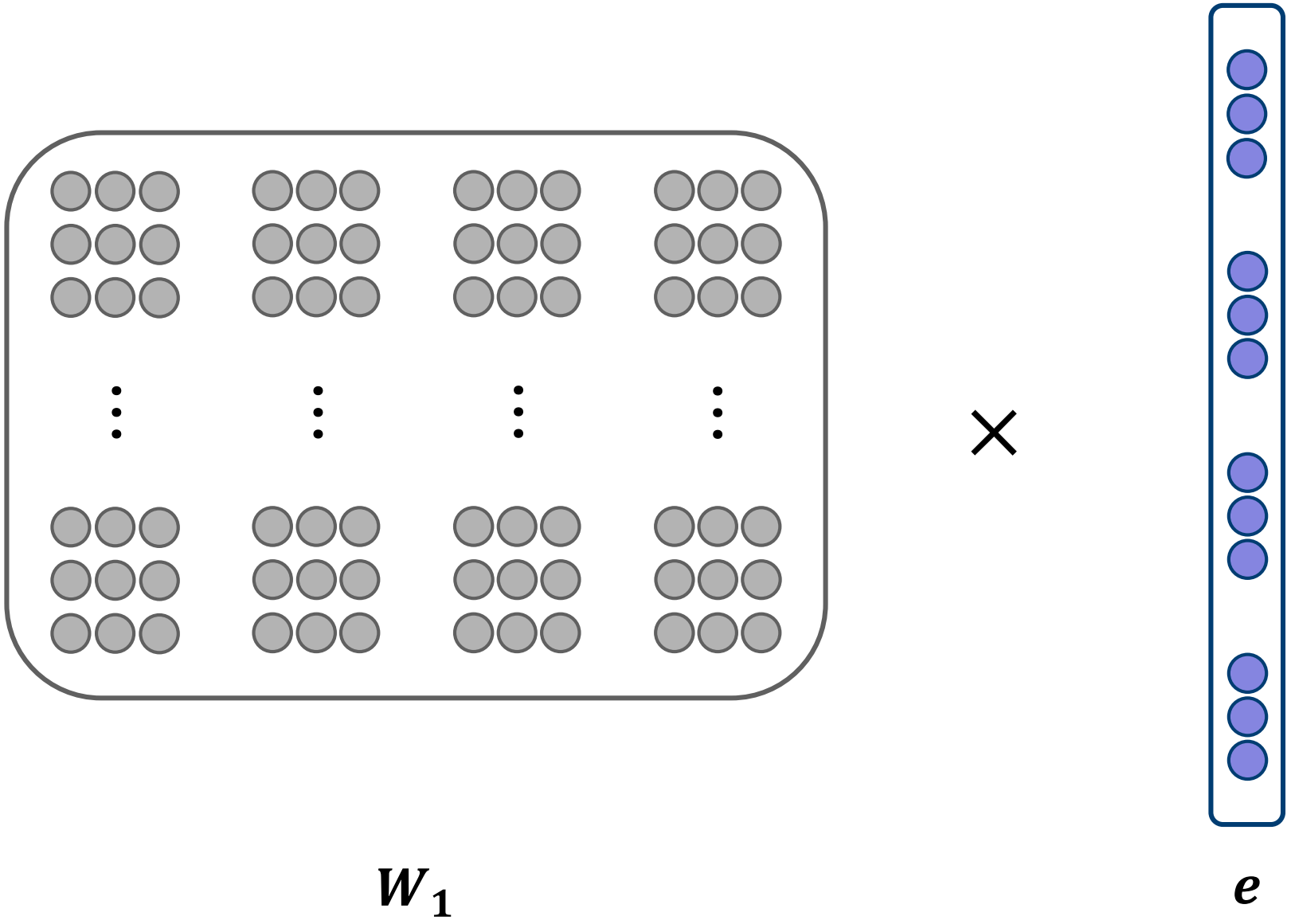
前馈神经网络语言模型的问题

- 前馈神经网络需要保存的参数
  - 词向量矩阵$\boldsymbol{E} \in \mathbb{R}^{|\mathcal{V}| \times d}$
  - 权重矩阵$\boldsymbol{W_1} \in \mathbb{R}^{d_h \times (n \times d)}$
  - 输出矩阵$\boldsymbol{W_2} \in \mathbb{R}^{|\mathcal{V}| \times d_h}$

这种增大是线性的

- 权重矩阵的大小仍会随着n的增大而增大
  - 不能完全解决Storage Problem
  - 仍不能很好的建模长时依赖

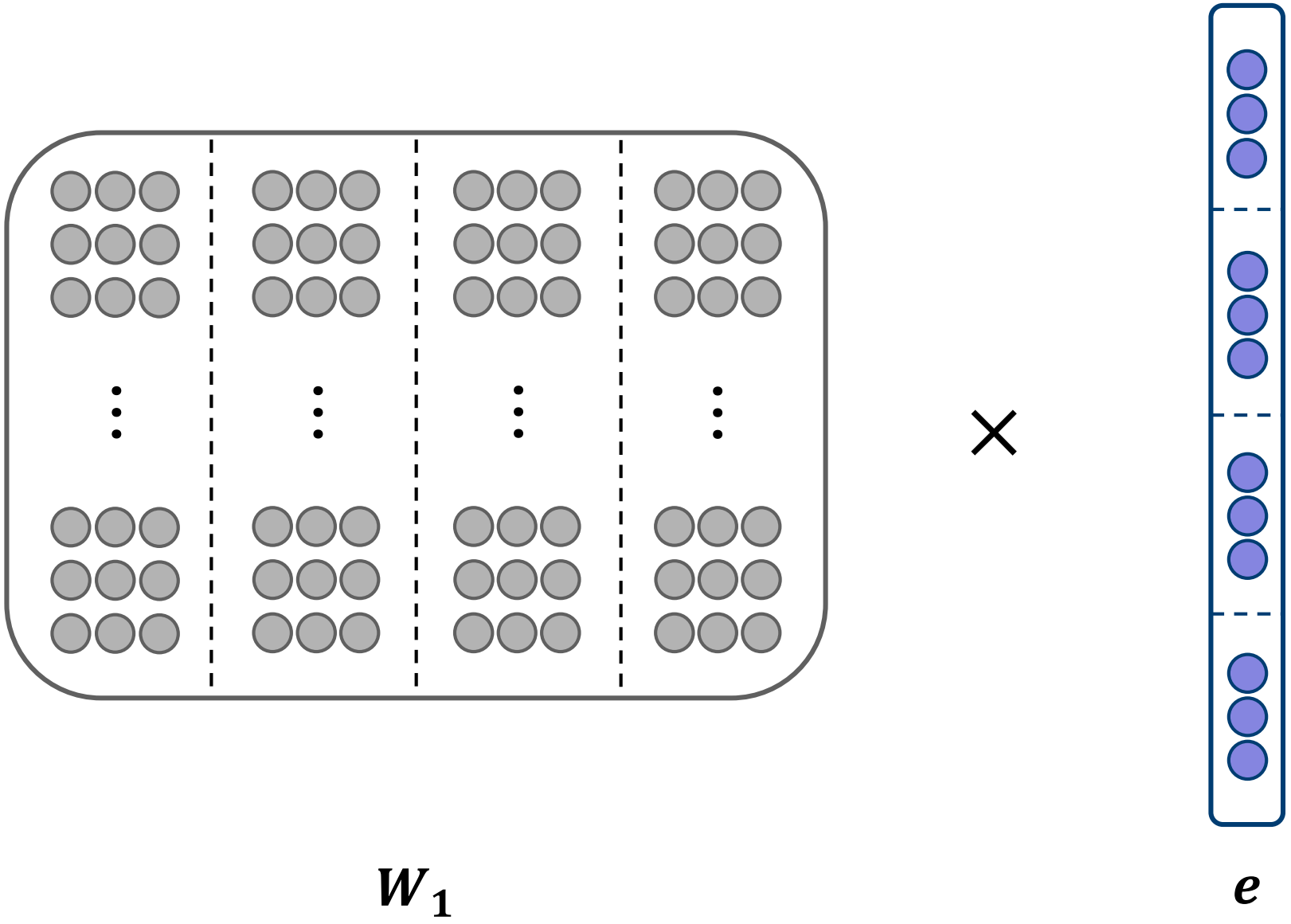
观察$W_1 \times e$的计算



W_1

e

观察$W_1 \times e$的计算



W_1

e

观察$W_1 \times e$的计算

$$\begin{bmatrix} W_1^{(1)}; & W_1^{(2)}; & W_1^{(3)}; & W_1^{(4)} \end{bmatrix} \times \begin{bmatrix} e^{(1)} \\ e^{(2)} \\ e^{(3)} \\ e^{(4)} \end{bmatrix} = \begin{matrix} W_1^{(1)} e^{(1)} \\ + \\ W_1^{(2)} e^{(2)} \\ + \\ W_1^{(3)} e^{(3)} \\ + \\ W_1^{(4)} e^{(4)} \end{matrix}$$

重构权重矩阵——循环神经网络的引出

- 权重矩阵的问题
  - 需要单独对每一个历史时刻的词学习参数
  - 不同时刻间共通的处理要不断重复学习

重构权重矩阵——循环神经网络的引出

- 权重矩阵的问题
  - 需要单独对每一个历史时刻的词学习参数
  - 不同时刻间共通的处理要不断重复学习

- 分解权重矩阵 $\boldsymbol{W_1} \in \mathbb{R}^{d_h \times (n \times d)}$
  - 不同时刻间相同的处理
    - $\boldsymbol{W_e} \in \mathbb{R}^{d_h \times d}$
  - 不同时刻间不同的处理
    - 假定已有信息向下一时刻传递时的处理是相同的
    - $\boldsymbol{W_h} \in \mathbb{R}^{d_h \times d_h}$

重构权重矩阵——循环神经网络的引出

- 权重矩阵的问题
  - 需要单独对每一个历史时刻的词学习参数
  - 不同时刻间共通的处理要不断重复学习

- 分解权重矩阵$\boldsymbol{W} \Rightarrow \boldsymbol{W_e} \in \mathbb{R}^{d_h \times d}, \boldsymbol{W_h} \in \mathbb{R}^{d_h \times d_h}$
  - 不再与历史长度n相关

- 从另一个角度理解
  - 在时间维度上大量使用共享参数的思想

循环神经网络语言模型的计算流程

各个字的单热向量$\boldsymbol{x}^{(t)}$

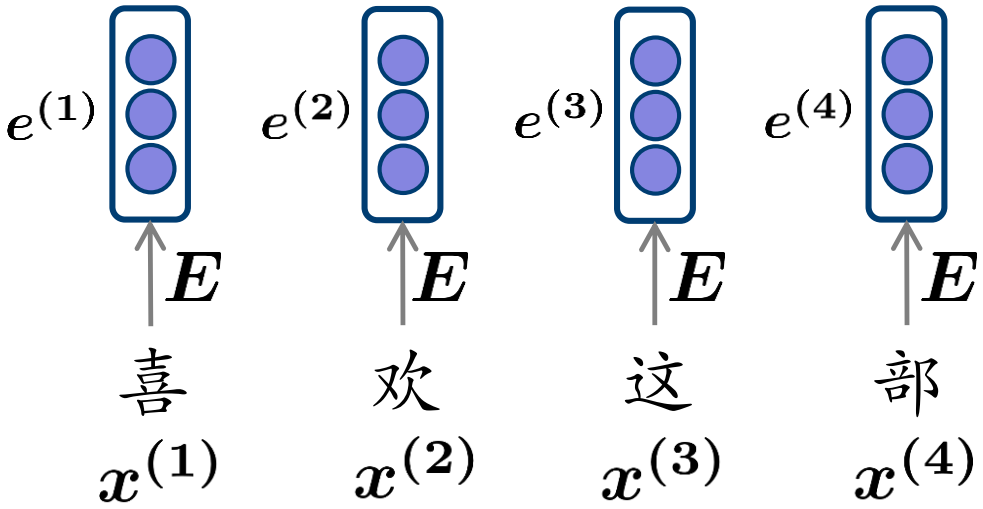
喜	欢	这	部
$\boldsymbol{x}^{(1)}$	$\boldsymbol{x}^{(2)}$	$\boldsymbol{x}^{(3)}$	$\boldsymbol{x}^{(4)}$

循环神经网络语言模型的计算流程

t时刻词向量

$$e^{(t)} = Ex^{(t)}$$

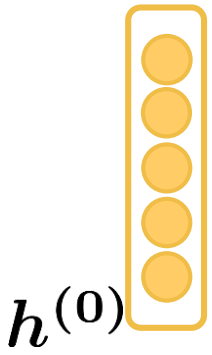
各个字的单热向量[t时刻词向量]



循环神经网络语言模型的计算流程

隐藏状态

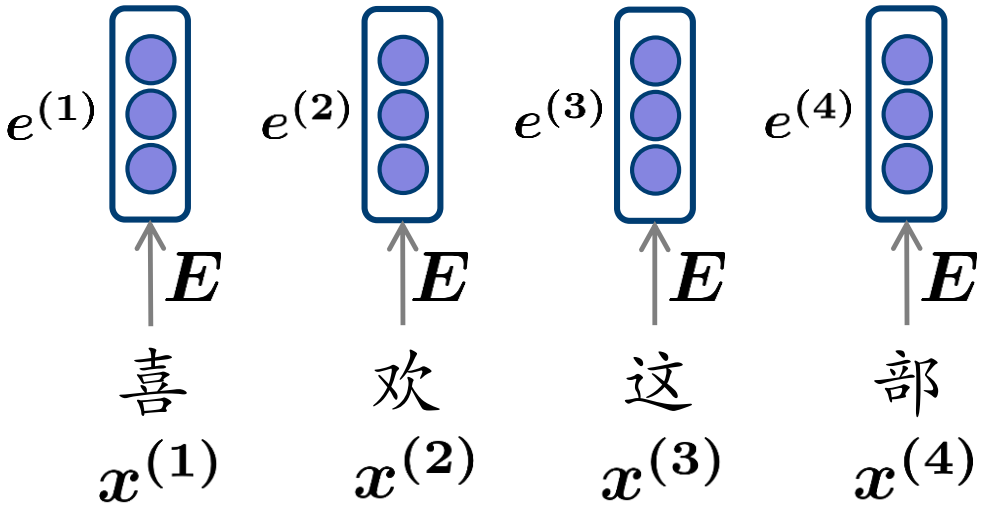
$\boldsymbol{h}^{(0)}$是初始隐藏状态



t时刻词向量

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

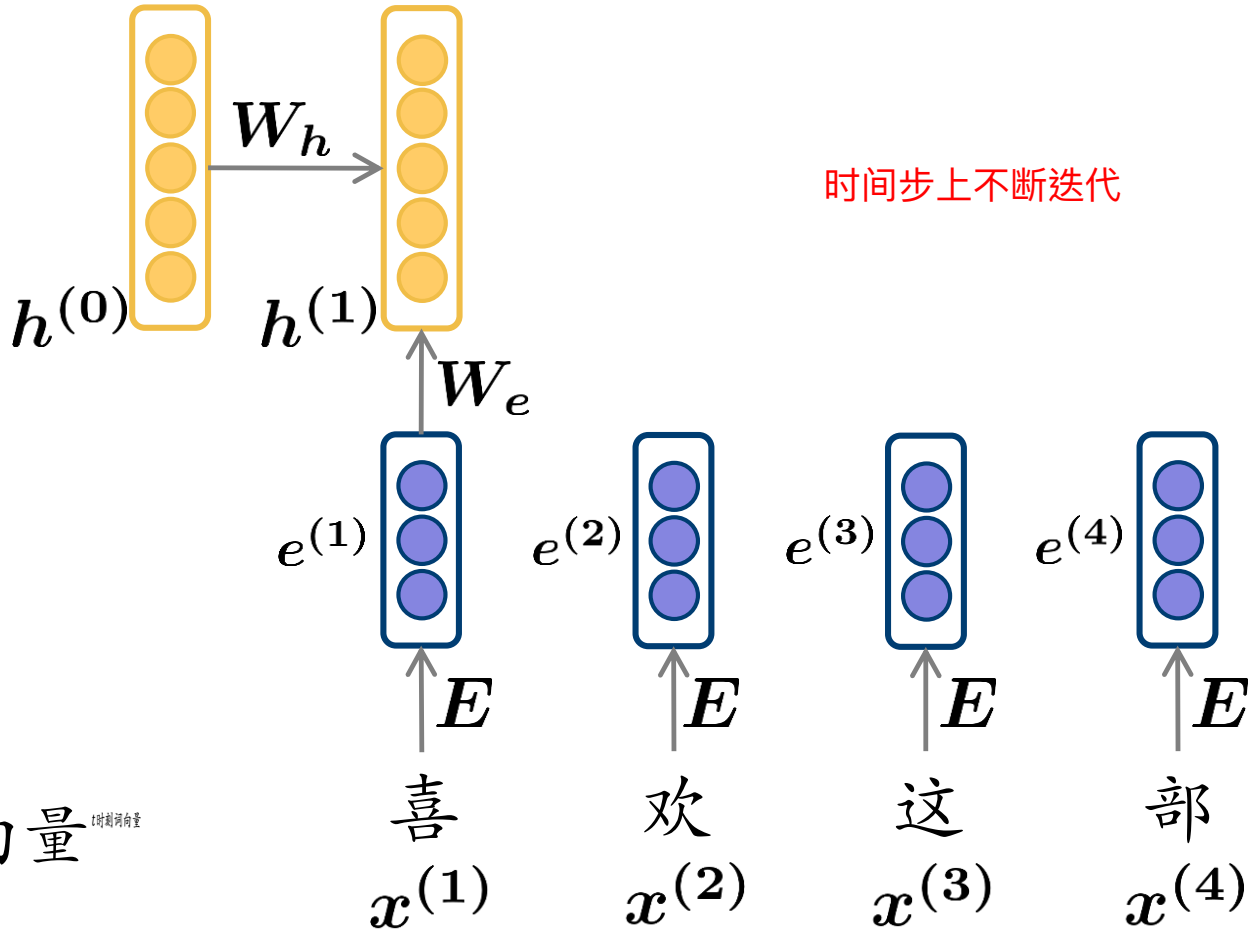
各个字的单热向量[t时刻词向量]



循环神经网络语言模型的计算流程

隐藏状态

$$h^{(t)} = \sigma(W_h h^{(t-1)} + W_e e^{(t)} + b_1)$$



时间步上不断迭代

t时刻词向量

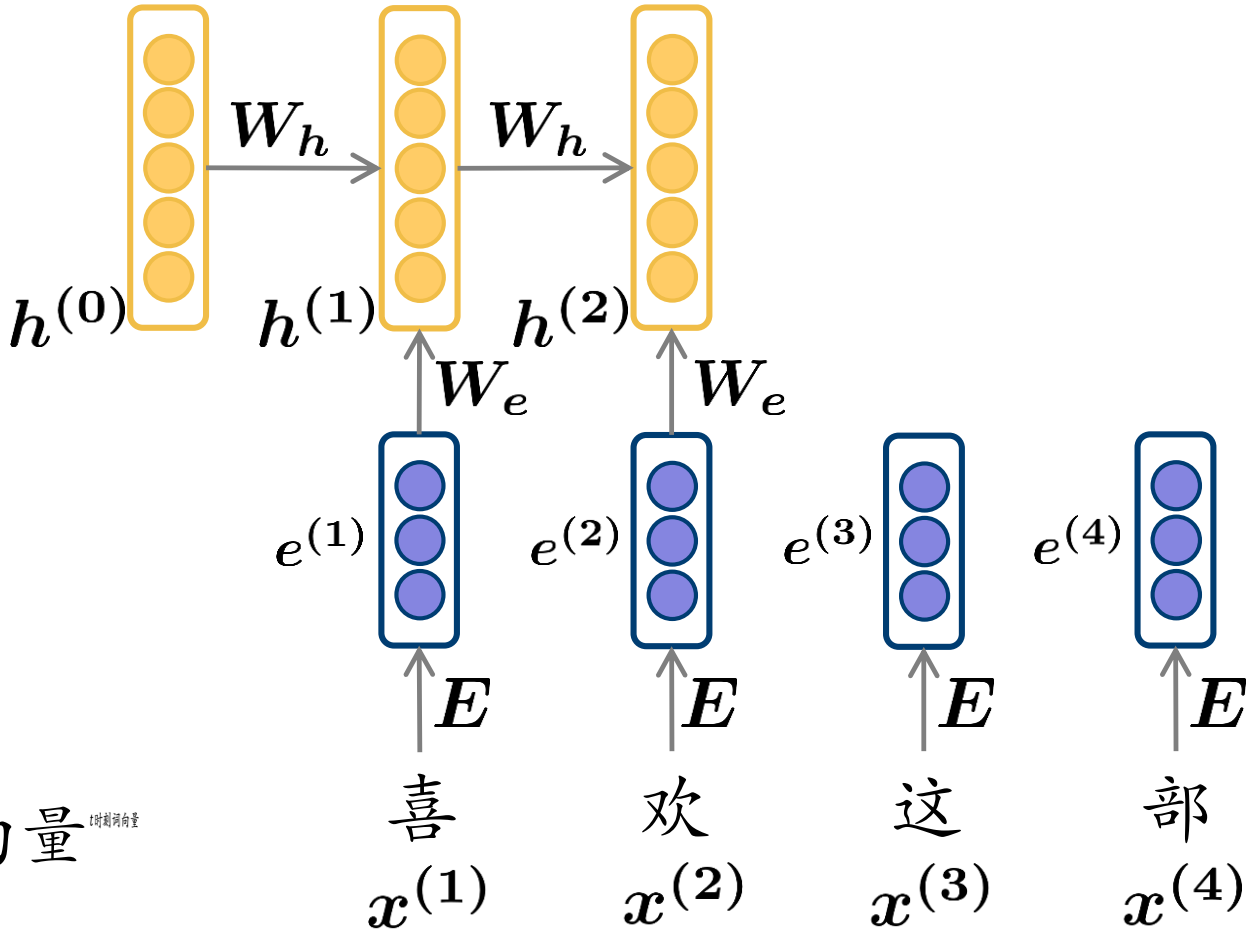
$$e^{(t)} = Ex^{(t)}$$

各个字的单热向量[t时刻词向量]

循环神经网络语言模型的计算流程

隐藏状态

$$\boldsymbol{h}^{(t)} = \sigma(\boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e \boldsymbol{e}^{(t)} + \boldsymbol{b}_1)$$



t时刻词向量

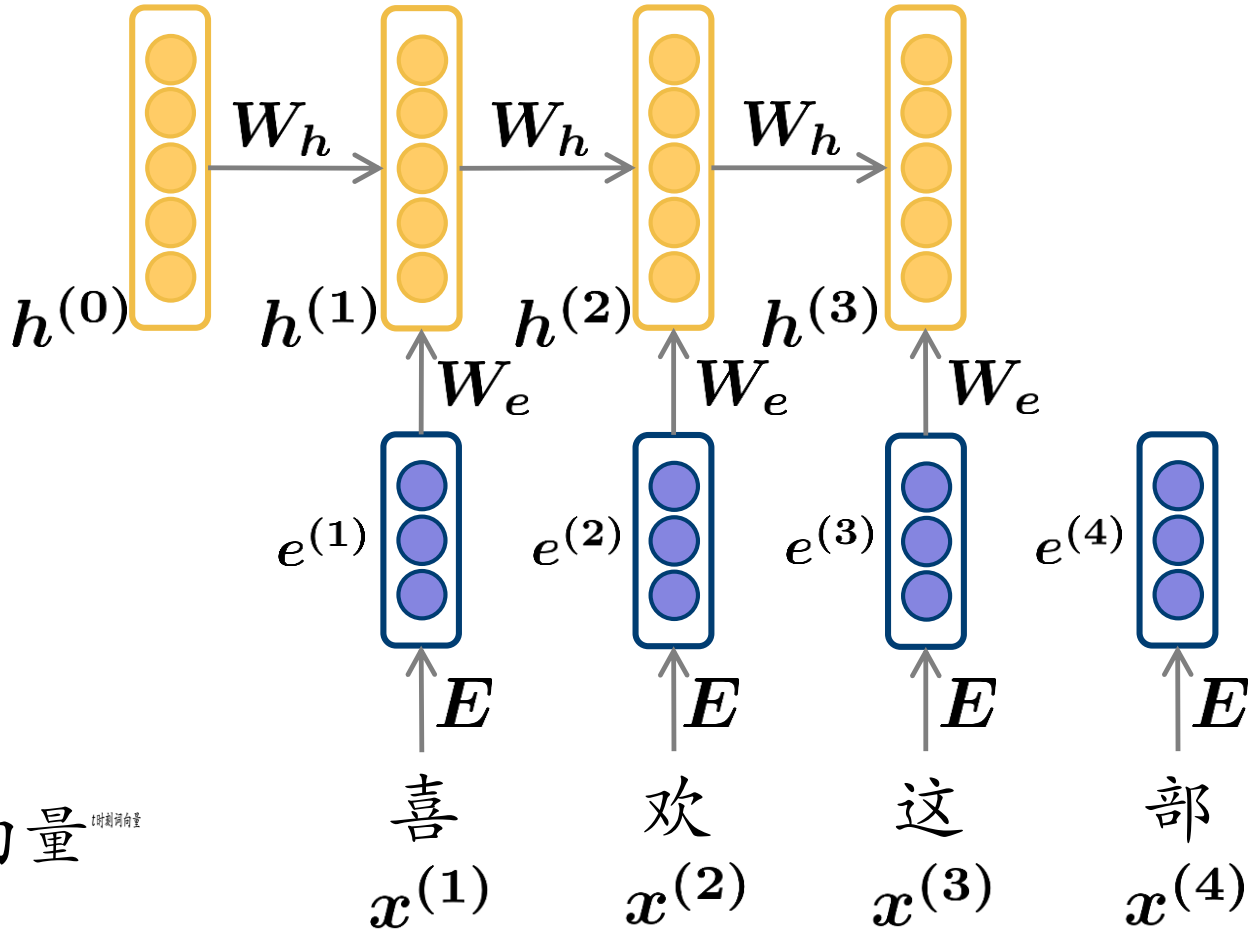
$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

各个字的单热向量[t时刻词向量]

循环神经网络语言模型的计算流程

隐藏状态

$$h^{(t)} = \sigma(W_h h^{(t-1)} + W_e e^{(t)} + b_1)$$



t时刻词向量

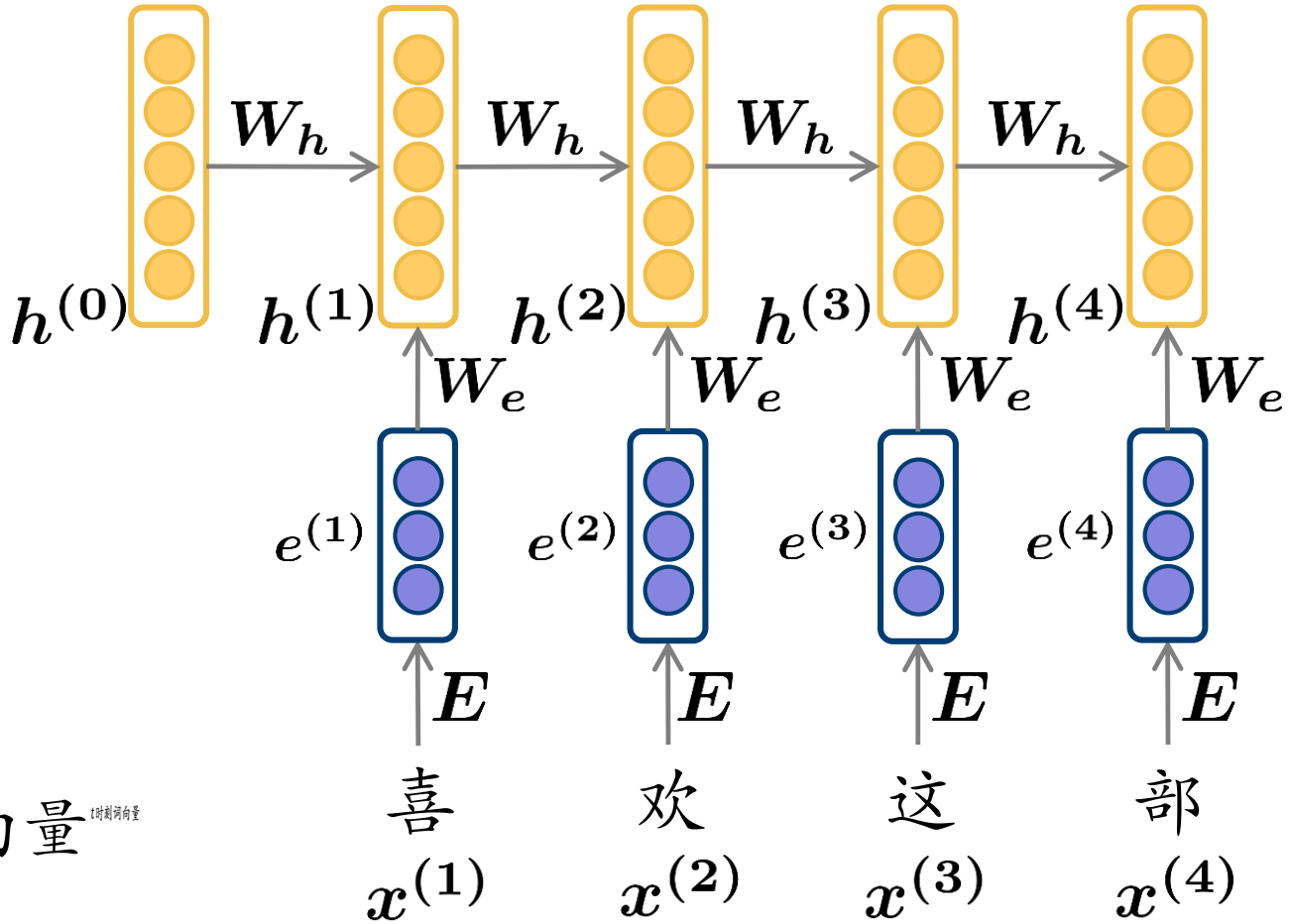
$$e^{(t)} = Ex^{(t)}$$

各个字的单热向量[t时刻词向量]

循环神经网络语言模型的计算流程

隐藏状态

$$\boldsymbol{h}^{(t)} = \sigma(\boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e \boldsymbol{e}^{(t)} + \boldsymbol{b}_1)$$



t时刻词向量

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

各个字的单热向量[t时刻词向量]

循环神经网络语言模型的计算流程

输出概率分布

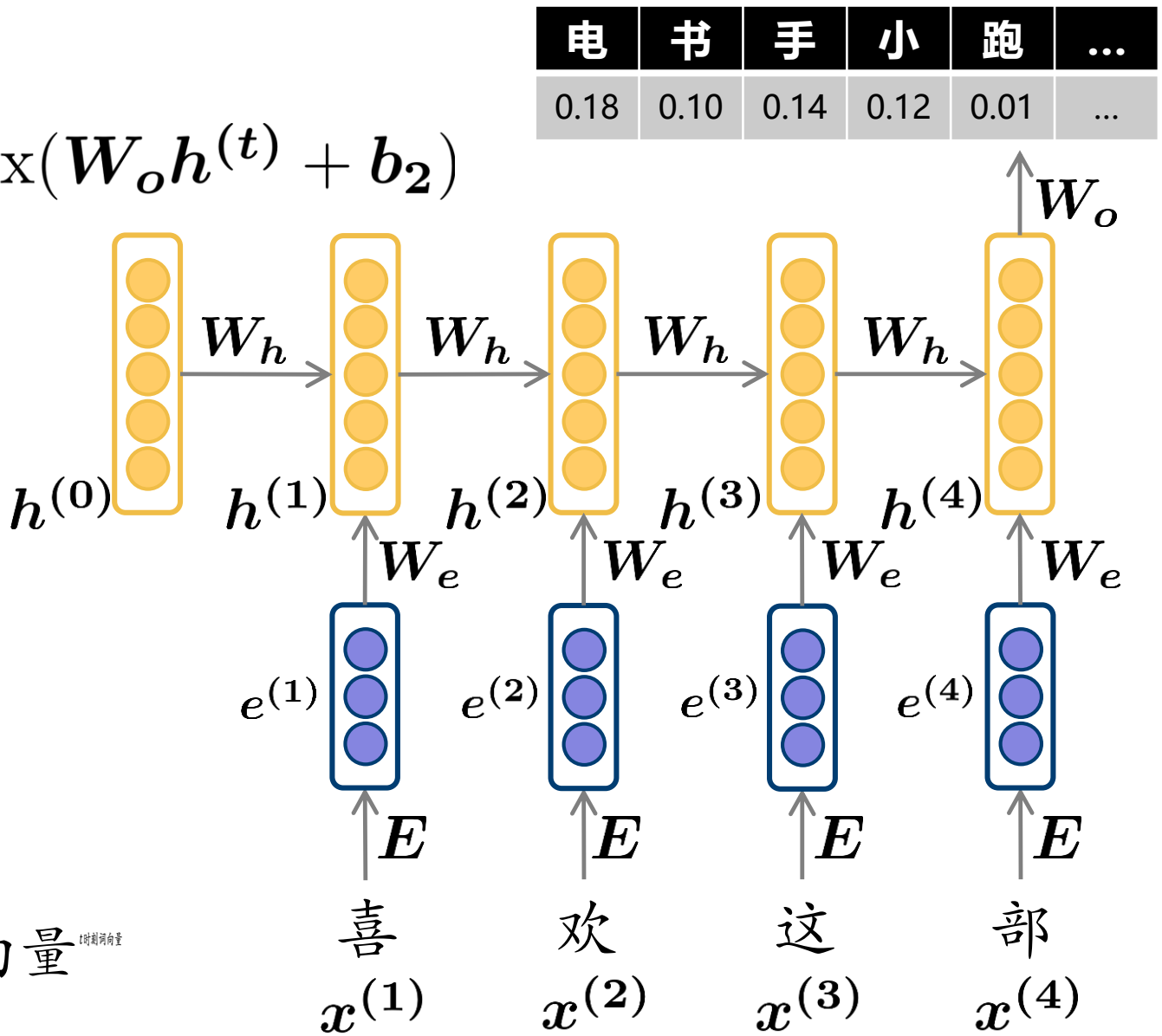
$$\hat{\boldsymbol{y}}^{(t)} = \text{Softmax}(\boldsymbol{W_o h}^{(t)} + \boldsymbol{b_2})$$

隐藏状态 $\boldsymbol{h}^{(t)}$

t时刻词向量

$$\boldsymbol{e}^{(t)} = \boldsymbol{E x}^{(t)}$$

各个字的单热向量[t时刻词向量]



循环神经网络的训练

- 采用交叉熵损失函数

$$\mathcal{L}^{(t)}(\theta) = -\sum_{w \in \mathcal{V}} p_w^{(t)} \log \hat{p}_w^{(t)} = -\log \hat{p}_{w_{t+1}}^{(t)}$$

循环神经网络语言模型的计算流程

输出概率分布

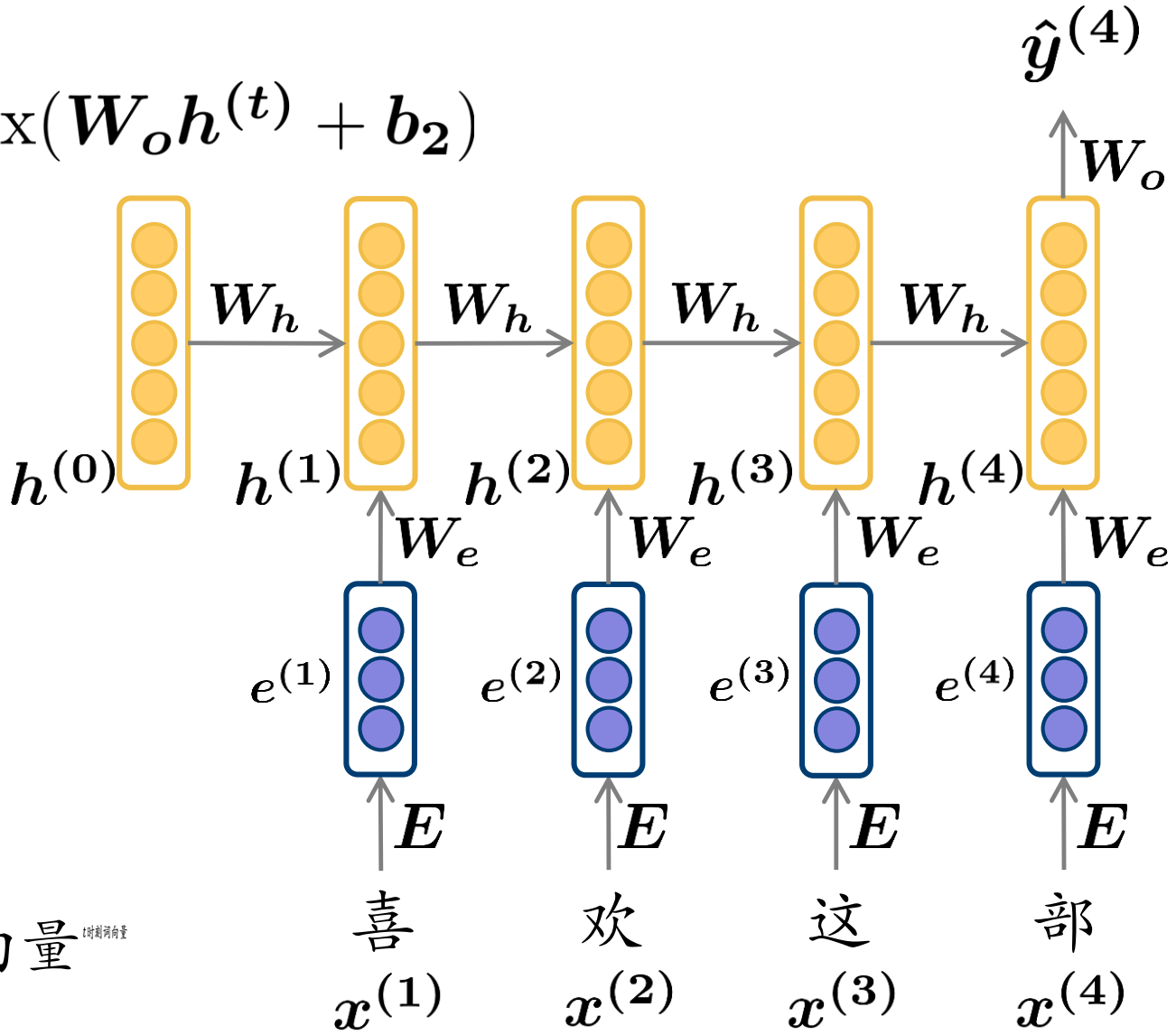
$$\hat{\boldsymbol{y}}^{(t)} = \text{Softmax}(\boldsymbol{W_o}\boldsymbol{h}^{(t)} + \boldsymbol{b_2})$$

隐藏状态 $\boldsymbol{h}^{(t)}$

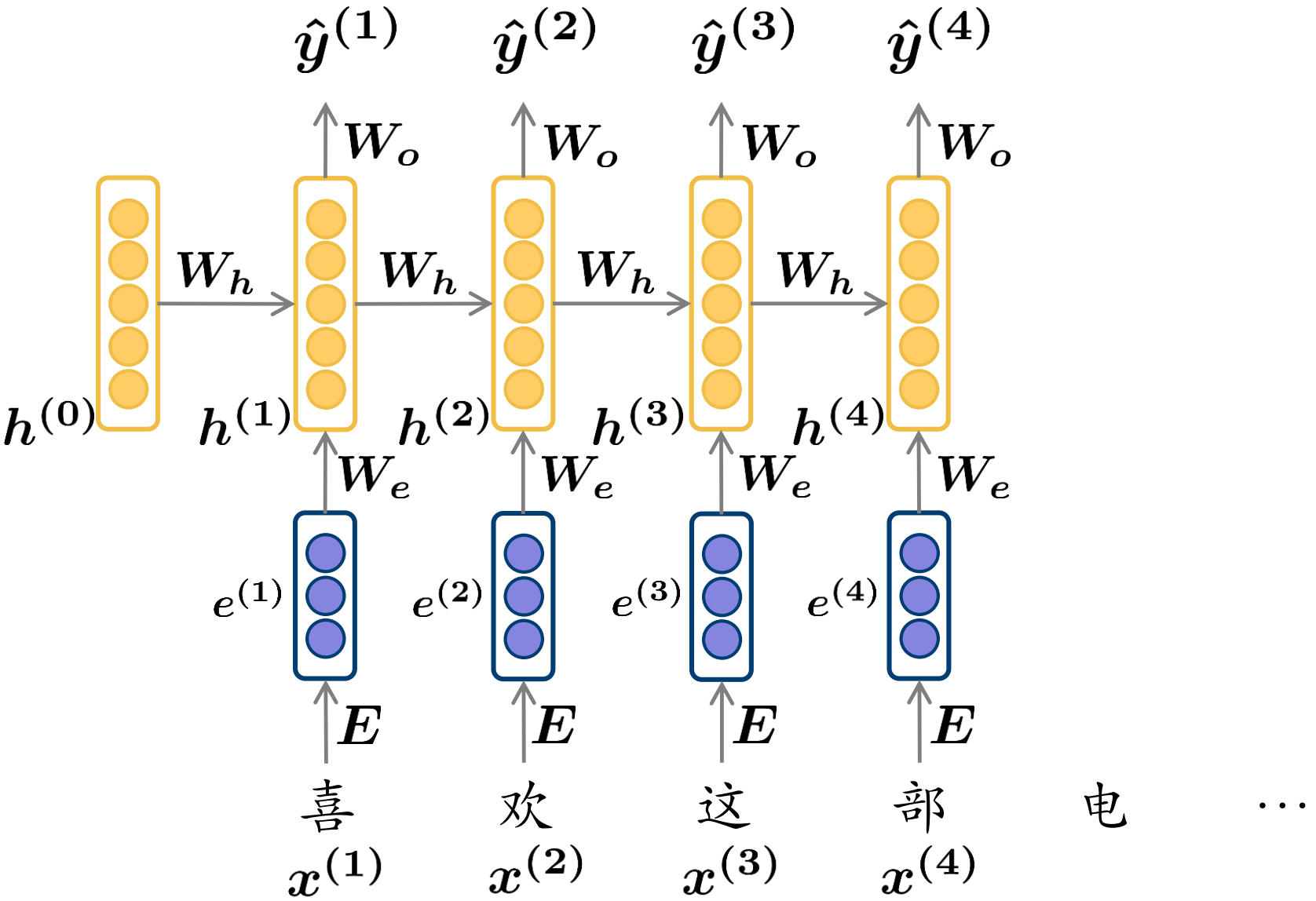
t时刻词向量

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

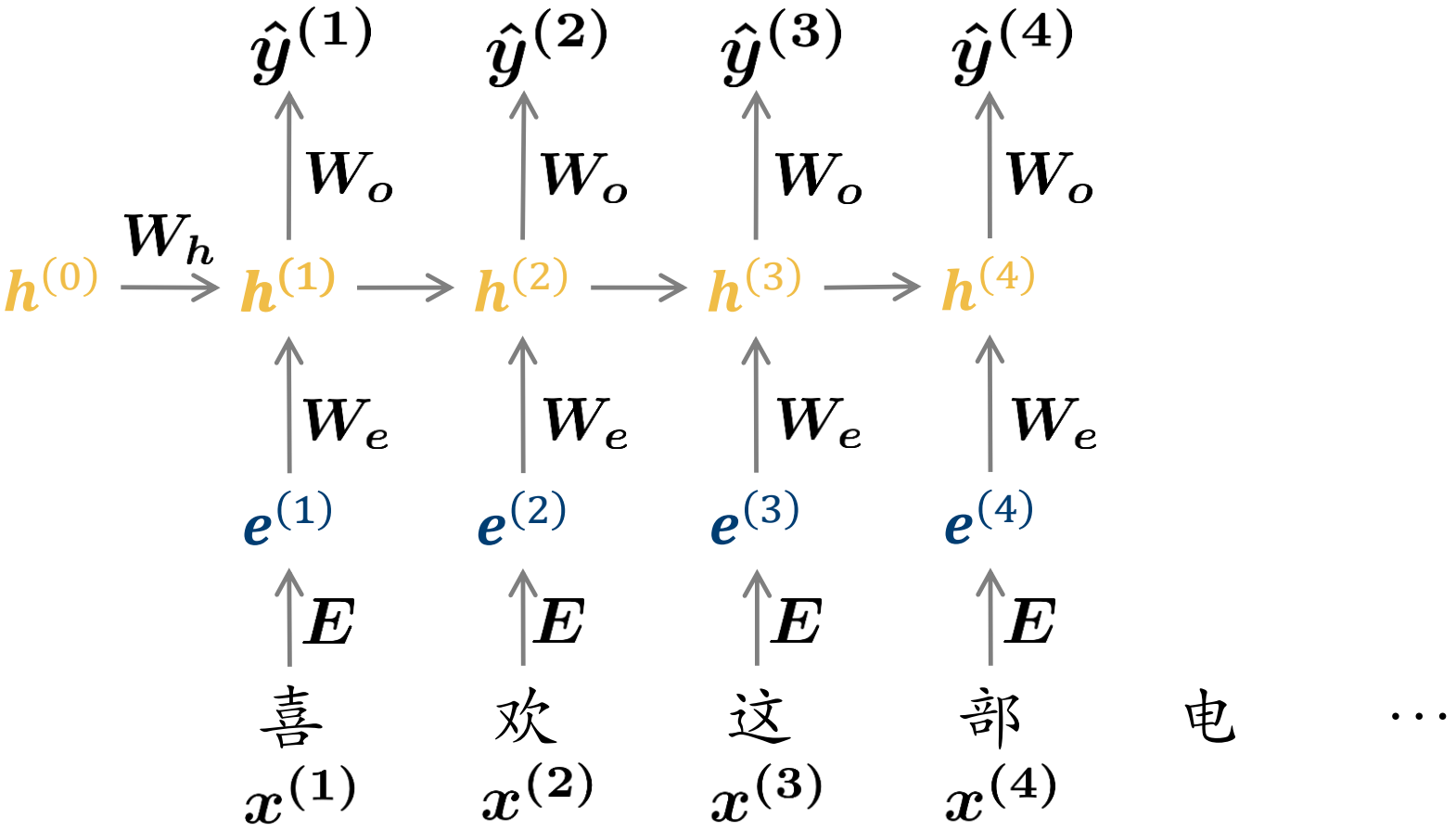
各个字的单热向量[t时刻词向量]



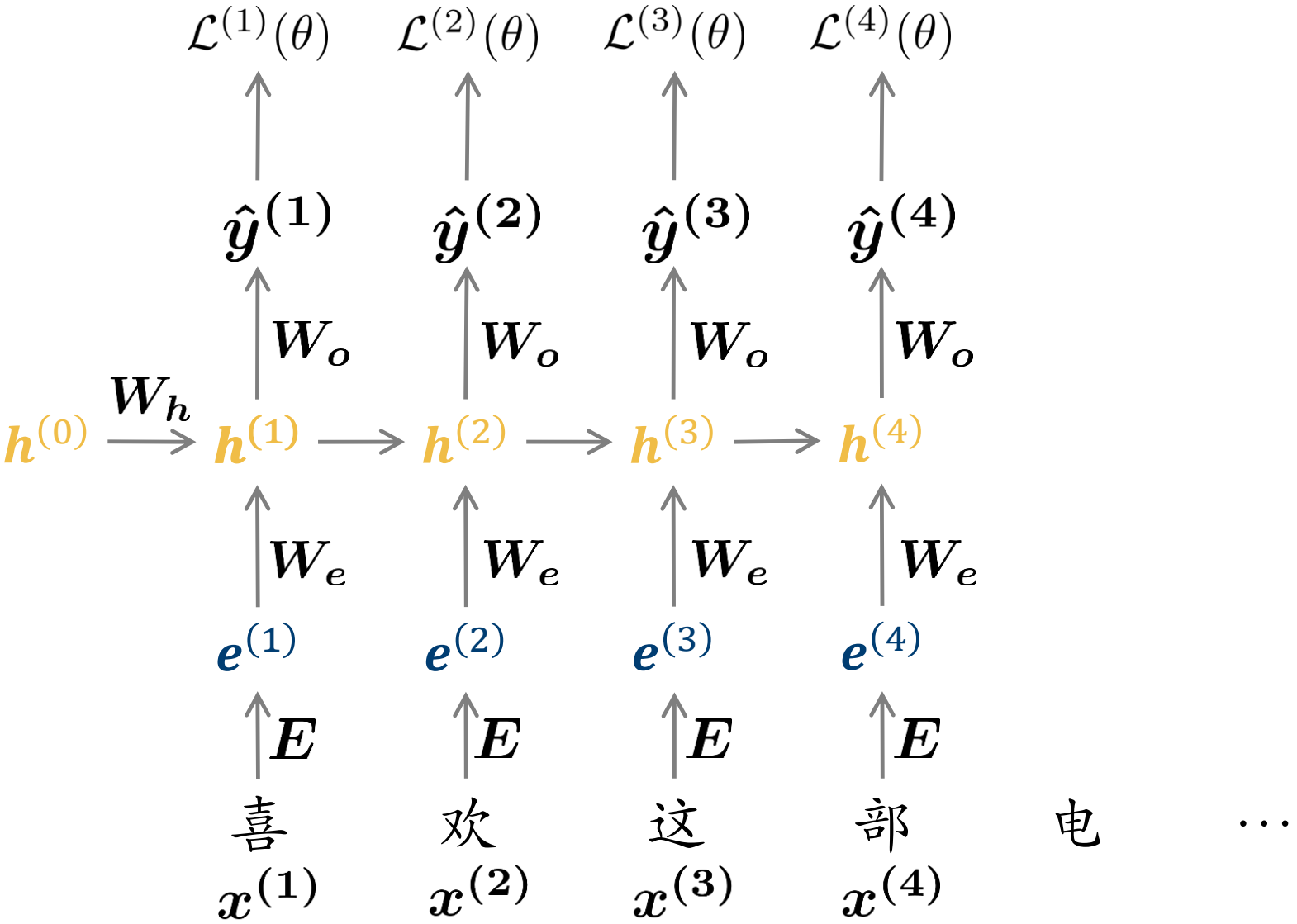
循环神经网络的训练


$\hat{y}^{(1)}$
$\hat{y}^{(2)}$
$\hat{y}^{(3)}$
$\hat{y}^{(4)}$
W_o
W_h
$h^{(0)}$
$h^{(1)}$
$h^{(2)}$
$h^{(3)}$
$h^{(4)}$
W_e
$e^{(1)}$
$e^{(2)}$
$e^{(3)}$
$e^{(4)}$
E
喜
欢
这
部
电
…
$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$

循环神经网络的训练


$\hat{y}^{(1)}$ $\hat{y}^{(2)}$ $\hat{y}^{(3)}$ $\hat{y}^{(4)}$
W_o W_o W_o W_o
$h^{(0)}$ W_h $h^{(1)}$ $h^{(2)}$ $h^{(3)}$ $h^{(4)}$
W_e W_e W_e W_e
$e^{(1)}$ $e^{(2)}$ $e^{(3)}$ $e^{(4)}$
E E E E
喜 欢 这 部 电 …
$x^{(1)}$ $x^{(2)}$ $x^{(3)}$ $x^{(4)}$

循环神经网络的训练


$\mathcal{L}^{(1)}(\theta)$
$\mathcal{L}^{(2)}(\theta)$
$\mathcal{L}^{(3)}(\theta)$
$\mathcal{L}^{(4)}(\theta)$
$\hat{y}^{(1)}$
$\hat{y}^{(2)}$
$\hat{y}^{(3)}$
$\hat{y}^{(4)}$
W_o
W_h
$h^{(0)}$
$h^{(1)}$
$h^{(2)}$
$h^{(3)}$
$h^{(4)}$
W_e
$e^{(1)}$
$e^{(2)}$
$e^{(3)}$
$e^{(4)}$
E
喜
欢
这
部
电
…
$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$

循环神经网络的训练

- 采用交叉熵损失函数

$$\mathcal{L}^{(t)}(\theta) = -\sum_{w \in \mathcal{V}} p_w^{(t)} \log \hat{p}_w^{(t)} = -\log \hat{p}_{w_{t+1}}^{(t)}$$

- 总的损失函数为：

$$\mathcal{L}(\theta) = \frac{1}{T} \sum_{t=1}^{T} \mathcal{L}^{(t)}(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \log \hat{p}_{w_{t+1}}^{(t)}$$

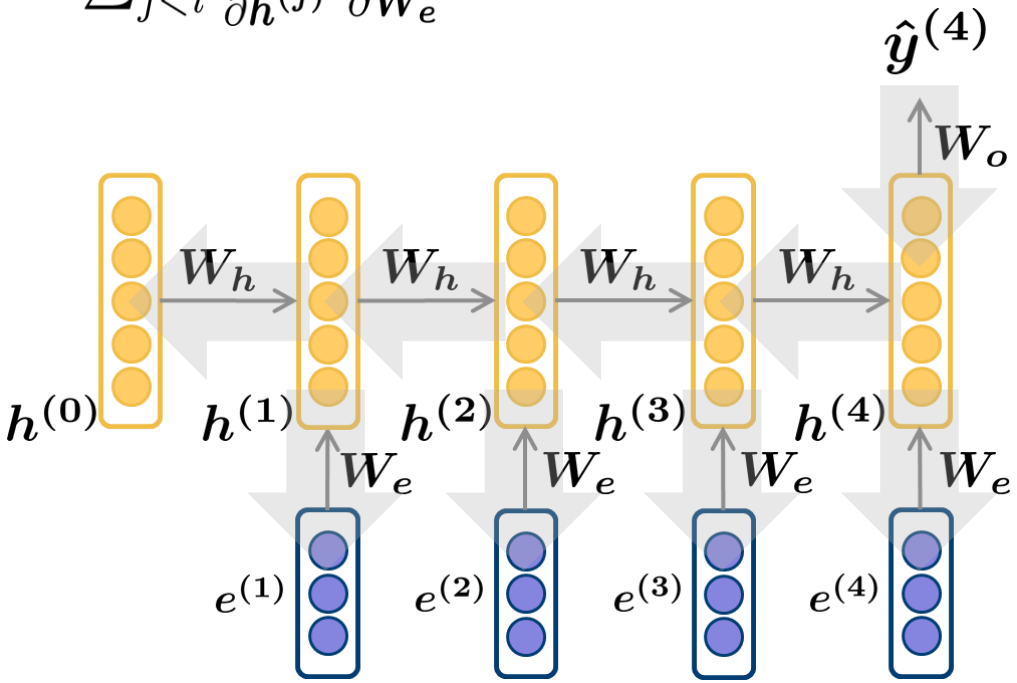
循环神经网络的梯度计算

循环神经网络的梯度计算

- 链式法则

循环神经网络的梯度计算

- 链式法则
  - $\frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{W}_h} = \sum_{j<i} \frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(j)}} \frac{\partial \boldsymbol{h}^{(j)}}{\partial \boldsymbol{W}_h}$
  - $\frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{W}_e} = \sum_{j<i} \frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(j)}} \frac{\partial \boldsymbol{h}^{(j)}}{\partial \boldsymbol{W}_e}$



循环神经网络的梯度计算

- 链式法则
  - $\frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{W}_h} = \sum_{j<i} \frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(j)}} \frac{\partial \boldsymbol{h}^{(j)}}{\partial \boldsymbol{W}_h}$
  - $\frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{W}_e} = \sum_{j<i} \frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(j)}} \frac{\partial \boldsymbol{h}^{(j)}}{\partial \boldsymbol{W}_e}$

- 考虑公共项：

$$\frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(j)}} = \frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(i)}} \prod_{j \le t < i} \frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}}$$

循环神经网络的梯度计算

- 考虑：$\frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}}$

$$\boldsymbol{h}^{(t+1)} = \sigma(\boldsymbol{W}_h \boldsymbol{h}^{(t)} + \boldsymbol{W}_e \boldsymbol{e}^{(t+1)} + \boldsymbol{b})$$

- 其中σ为激活函数

循环神经网络的梯度计算

- 考虑：$\dfrac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}}$

$$\boldsymbol{h}^{(t+1)} = \sigma(\tilde{\boldsymbol{h}}^{(t+1)}) = \sigma(\boldsymbol{W}_{\boldsymbol{h}}\boldsymbol{h}^{(t)} + \boldsymbol{W}_{\boldsymbol{e}}\boldsymbol{e}^{(t+1)} + \boldsymbol{b})$$

- 其中σ为激活函数

- 考虑简化情况：激活函数为线性 $\sigma(x) = x$
- 则

$$\begin{aligned}\frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}} &= \frac{\partial \sigma(\tilde{\boldsymbol{h}}^{(t+1)})}{\partial \tilde{\boldsymbol{h}}^{(t+1)}} \frac{\partial(\boldsymbol{W}_{\boldsymbol{h}}\boldsymbol{h}^{(t)} + \boldsymbol{W}_{\boldsymbol{e}}\boldsymbol{e}^{(t+1)} + \boldsymbol{b})}{\partial \boldsymbol{h}^{(t)}} \\ &= 1 \times \boldsymbol{W}_{\boldsymbol{h}}\end{aligned}$$

循环神经网络的梯度计算

- 于是

$$\frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(j)}} = \frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(i)}} \prod_{j \le t < i} \frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}}$$

$$= \frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(i)}} \prod_{j \le t < i} \boldsymbol{W}_{\boldsymbol{h}}$$

循环神经网络的梯度计算

- 于是

$$\frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(j)}} = \frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(i)}} \prod_{j \le t < i} \frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}}$$

$$= \frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(i)}} \prod_{j \le t < i} \boldsymbol{W_h}$$

求导的时候loss经过多步回传
梯度消失

- 继续简化：考虑标量情况

$$\frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(j)}} = \frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(i)}} \prod_{j \le t < i} W_h = \frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(i)}} W_h^{i-j}$$

循环神经网络的梯度计算

$$\frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(j)}} = \frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(i)}} \prod_{j \le t < i} W_h = \frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(i)}} W_h^{i-j}$$

- 当$W_h \neq 1$时，会出现什么情况？

循环神经网络的梯度计算

$$\frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(j)}} = \frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(i)}} \prod_{j \le t < i} W_h = \frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(i)}} W_h^{i-j}$$

- 当$W_h \neq 1$时，会出现什么情况？

- 类似的，将标量还原回向量、并引入激活函数，仍会存在这些问题。

循环神经网络存在的问题

- 梯度消失问题
  - 并不能有效的建立远距依赖关系
  - 例如：This kind of books is/are great.

循环神经网络存在的问题

- 梯度消失问题
  - 并不能有效的建立远距依赖关系
  - 例如：This kind of books is/are great.

- 梯度爆炸问题
  - 与神经网络中学习率过大时的情形类似

循环神经网络存在的问题

- 梯度消失问题
  - 并不能有效的建立远距依赖关系
  - 例如：This kind of books is/are great.

- 梯度爆炸问题
  - 与神经网络中学习率过大时的情形类似
  - 解决方法：clipping
    - 如果梯度的绝对值大于阈值，则将其绝对值设置为阈值，符号不变

解决梯度消失问题

- 为什么会出现梯度消失问题?

解决梯度消失问题

- 为什么会出现梯度消失问题?
  - 时间维度上的深度神经网络
  - RNN对于历史处理的共享假设加剧了这个现象

解决梯度消失问题

- 为什么会出现梯度消失问题?
  - 时间维度上的深度神经网络
  - RNN对于历史处理的共享假设加剧了这个现象

- 解决思路:
  - 向下一时刻传递信息时，考虑当前时刻的上下文情况

解决梯度消失问题

- 为什么会出现梯度消失问题?
  - 时间维度上的深度神经网络
  - RNN对于历史处理的共享假设加剧了这个现象
- 解决思路:
  - 向下一时刻传递信息时，考虑当前时刻的上下文情况
- 进一步理解
  - RNN中$\boldsymbol{h}^{(t-1)}$的意义：记录历史信息，指导后续表达
  - 当前时刻的输入w_t的意义：当前时刻进行的表达
  - w_t对于$\boldsymbol{h}^{(t-1)}$的影响:
    - 完成了部分已有表达任务
    - 为后续的表达添加了新的条件

目录

长短时记忆神经网络（LSTM）

核心思想：

- 将长期记忆和短期记忆分离
  - 长期记忆用来存储和传递整个表达过程的状态，每一时刻根据上下文情况更新，且不能被外界直接观察
  - 短期记忆是根据当前上下文情况，从长期记忆中获取的，用以指导当前时刻的输出以及下一时刻长期记忆变化的部分，相当于当前时刻被“激活”的长期记忆

长短时记忆神经网络（LSTM）

核心思想：

- 将长期记忆和短期记忆分离
  - 长期记忆：cell state $\boldsymbol{c}^{(t)} \in \mathbb{R}^n$
  - 短期记忆：hidden state $\boldsymbol{h}^{(t)} \in \mathbb{R}^n$

长短时记忆神经网络（LSTM）

核心思想：

- 将长期记忆和短期记忆分离
  - 长期记忆：单元状态 $\boldsymbol{c}^{(t)} \in \mathbb{R}^n$
  - 短期记忆：隐藏状态 $\boldsymbol{h}^{(t)} \in \mathbb{R}^n$

- 使用门（gate）控制记忆的消除、传递与读写
  - 门是一个与记忆向量相同维度的向量
  - 每个位置是一个在0（关闭）和1（开启）间的实数
  - 在使用时，将门与其作用的向量做逐元素的乘法
  - 门是在每个时刻根据当前上下文动态生成的

LSTM的公式

- 在时间点t，输入的词向量为$\boldsymbol{x}^{(t)}$，计算其隐藏状态$\boldsymbol{h}^{(t)}$和单元状态$\boldsymbol{c}^{(t)}$：
  - 遗忘门$\boldsymbol{f}^{(t)}$：控制原单元状态中内容的保留与遗忘
  - 输入门$\boldsymbol{i}^{(t)}$：控制当前时刻什么内容被写入单元状态
  - 输出门$\boldsymbol{o}^{(t)}$：控制当前时刻什么内容从单元状态中被输出至隐藏状态

LSTM的公式

- 在时间点t，输入的词向量为$\boldsymbol{x}^{(t)}$，计算其隐藏状态$\boldsymbol{h}^{(t)}$和单元状态$\boldsymbol{c}^{(t)}$：
  - 遗忘门：$\boldsymbol{f}^{(t)} = \sigma(\boldsymbol{W}_{\boldsymbol{f}}[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_{\boldsymbol{f}})$
  - 输入门：$\boldsymbol{i}^{(t)} = \sigma(\boldsymbol{W}_{\boldsymbol{i}}[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_{\boldsymbol{i}})$
  - 输出门：$\boldsymbol{o}^{(t)} = \sigma(\boldsymbol{W}_{\boldsymbol{o}}[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_{\boldsymbol{o}})$

LSTM的公式

- 在时间点t，输入的词向量为$\boldsymbol{x}^{(t)}$，计算其隐藏状态$\boldsymbol{h}^{(t)}$和单元状态$\boldsymbol{c}^{(t)}$：

cell state

concatenate

  - 遗忘门：$\boldsymbol{f}^{(t)} = \sigma(\boldsymbol{W}_f[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_f)$
  - 输入门：$\boldsymbol{i}^{(t)} = \sigma(\boldsymbol{W}_i[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_i)$
  - 输出门：$\boldsymbol{o}^{(t)} = \sigma(\boldsymbol{W}_o[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_o)$

0 遗忘，1记着

  - 候选单元状态：$\tilde{\boldsymbol{c}}^{(t)} = \tanh(\boldsymbol{W}_c[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_c)$

element-wise的乘

新的东西也不是全都要

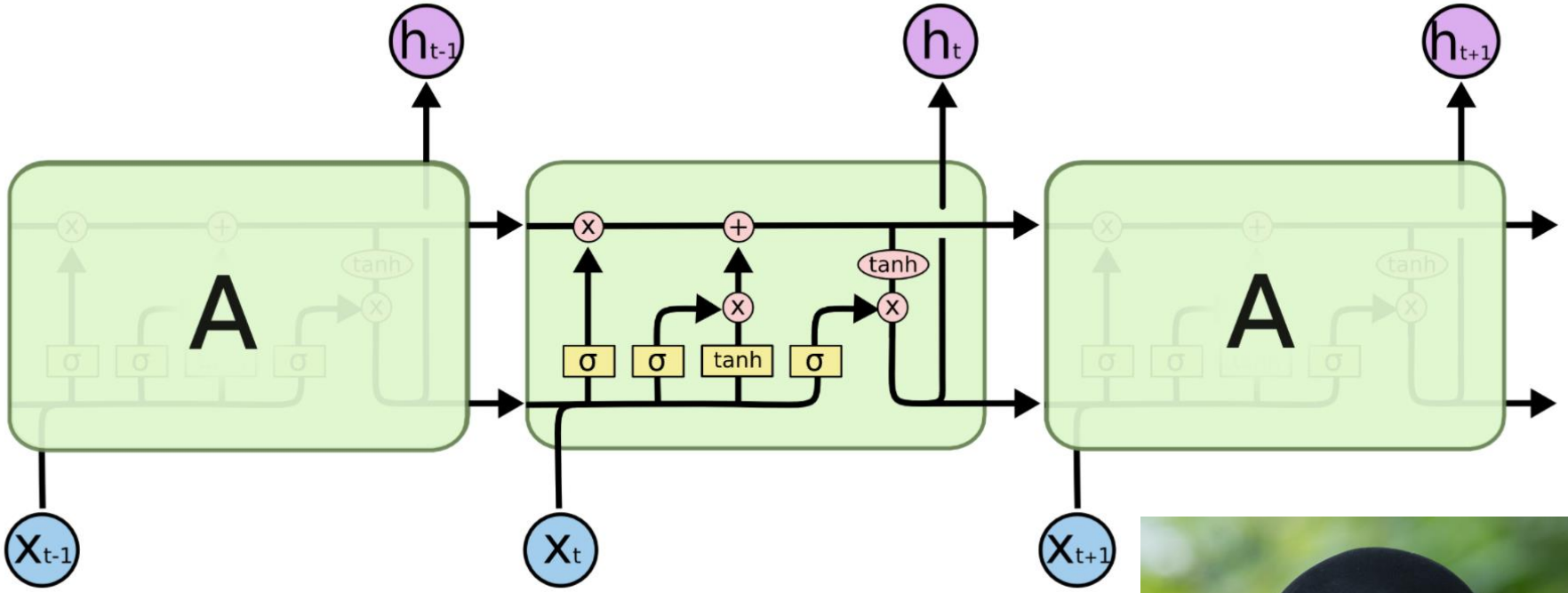
  - 单元状态：$\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \circ \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \circ \tilde{\boldsymbol{c}}^{(t)}$

关键

长期记忆的值，不需要的，就被遗忘掉

  - 隐藏状态：$\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)} \circ \tanh(\boldsymbol{c}^{(t)})$

LSTM的流程图示



$$f^{(t)} = \sigma(W_f[h^{(t-1)}; x^{(t)}] + b_f)$$
$$i^{(t)} = \sigma(W_i[h^{(t-1)}; x^{(t)}] + b_i)$$
$$o^{(t)} = \sigma(W_o[h^{(t-1)}; x^{(t)}] + b_o)$$
$$\tilde{c}^{(t)} = \tanh(W_c[h^{(t-1)}; x^{(t)}] + b_c)$$
$$c^{(t)} = f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)}$$
$$h^{(t)} = o^{(t)} \circ \tanh(c^{(t)})$$

乘的东西都跟当前时间步相关，并非向之前一样乘同一个Wh，从而比较好地解决梯度消失。

Juergen Schmidhuber

LSTM与梯度消失问题

- 思考：LSTM真的解决了梯度消失问题了吗?

LSTM与梯度消失问题

- 思考：LSTM真的解决了梯度消失问题了吗?

- 确切的讲，LSTM让网络能够更容易的保留远距的信息，从而极大的缓解了梯度消失的问题
  - 朴素RNN需要W_h为单位矩阵以完全保留历史信息
  - LSTM只需要遗忘门为全1便可完全保留历史信息

简化LSTM：GRU

Gated Recurrent Union

LSTM 参数量太大了

- 在时间点t，输入的词向量为$\boldsymbol{x}^{(t)}$，计算其隐藏状态$\boldsymbol{h}^{(t)}$（无单元状态）
  - 更新门$\boldsymbol{z}^{(t)}$：控制原隐藏状态中内容的更新与保留
  - 重置门$\boldsymbol{r}^{(t)}$：控制原隐藏状态中什么部分被用来和当前时刻的输入一起计算新的隐藏状态内容

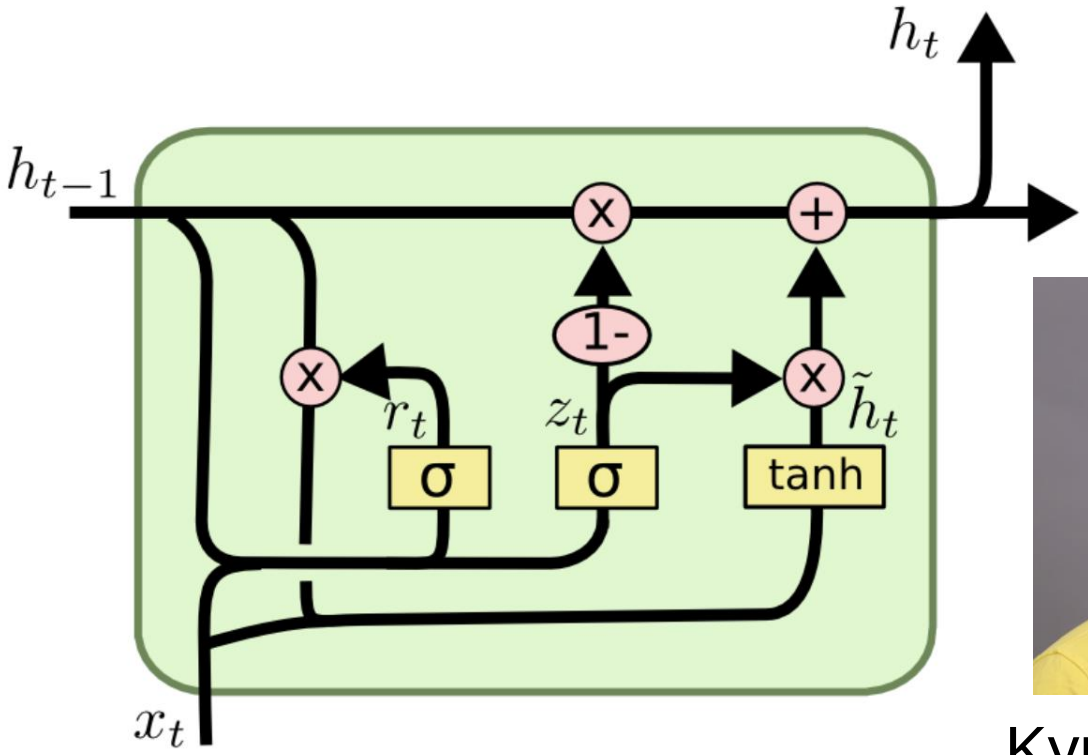
简化LSTM：GRU

- 在时间点t，输入的词向量为$\boldsymbol{x}^{(t)}$，计算其隐藏状态$\boldsymbol{h}^{(t)}$（无单元状态）
  - 更新门：$\boldsymbol{z}^{(t)} = \sigma(\boldsymbol{W}_{\boldsymbol{z}}[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_{\boldsymbol{z}})$
  - 重置门：$\boldsymbol{r}^{(t)} = \sigma(\boldsymbol{W}_{\boldsymbol{r}}[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_{\boldsymbol{r}})$

简化LSTM：GRU

- 在时间点t，输入的词向量为$\boldsymbol{x}^{(t)}$，计算其隐藏状态$\boldsymbol{h}^{(t)}$（无单元状态）
  - 更新门：$\boldsymbol{z}^{(t)} = \sigma(\boldsymbol{W}_{\boldsymbol{z}}[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_{\boldsymbol{z}})$
  - 重置门：$\boldsymbol{r}^{(t)} = \sigma(\boldsymbol{W}_{\boldsymbol{r}}[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_{\boldsymbol{r}})$
  - 候选状态：$\tilde{\boldsymbol{h}}^{(t)} = \tanh(\boldsymbol{W}_{\boldsymbol{h}}[\boldsymbol{r}^{(t)} \circ \boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_{\boldsymbol{h}})$
  - 隐藏状态：$\boldsymbol{h}^{(t)} = (1 - \boldsymbol{z}^{(t)}) \circ \boldsymbol{h}^{(t-1)} + \boldsymbol{z}^{(t)} \circ \tilde{\boldsymbol{h}}^{(t)}$ 核心

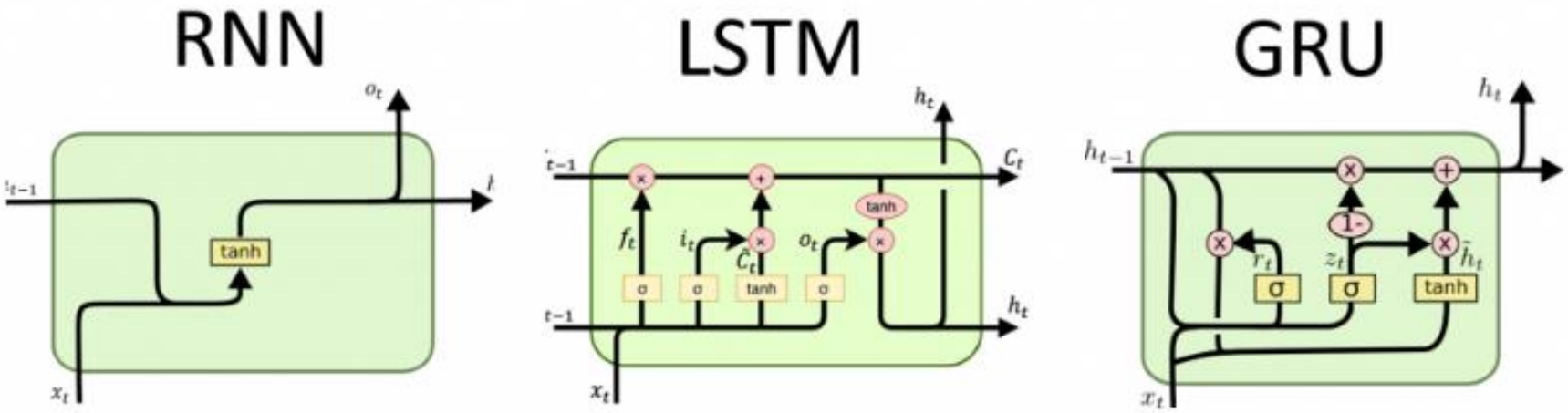
简化LSTM：GRU



Kyunghyun Cho

$$z^{(t)} = \sigma(W_z[h^{(t-1)}; x^{(t)}] + b_z)$$
$$r^{(t)} = \sigma(W_r[h^{(t-1)}; x^{(t)}] + b_r)$$
$$\tilde{h}^{(t)} = \tanh(W_h[r^{(t)} \circ h^{(t-1)}; x^{(t)}] + b_h)$$
$$h^{(t)} = (1 - z^{(t)}) \circ h^{(t-1)} + z^{(t)} \circ \tilde{h}^{(t)}$$

RNN， LSTM和GRU的对比



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

目录

神经网络语言模型效果对比

模型	PPL
3-gram	206.42
4-gram	160.68
5-gram	172.63
RNN	147.57
LSTM	78.63
GRU	78.82

PPL越小，生成sentence的真实度越高