循环神经网络

CS2916 大语言模型

飲水思源 愛國榮校

https://plms.ai/teaching/index.html

(该章节部分课件参照CS11-747, CS224n)

一个情感分类的例子

I hate this movie ?

very good
good
neutral
bad
very bad

I love this movie ?

very good
good
neutral
bad
very bad

一个情感分类的例子

I hate this movie

very good
good
neutral
bad
very bad

I love this movie

very good
good
neutral
bad
very bad

一个情感分类的例子

I hate this movie

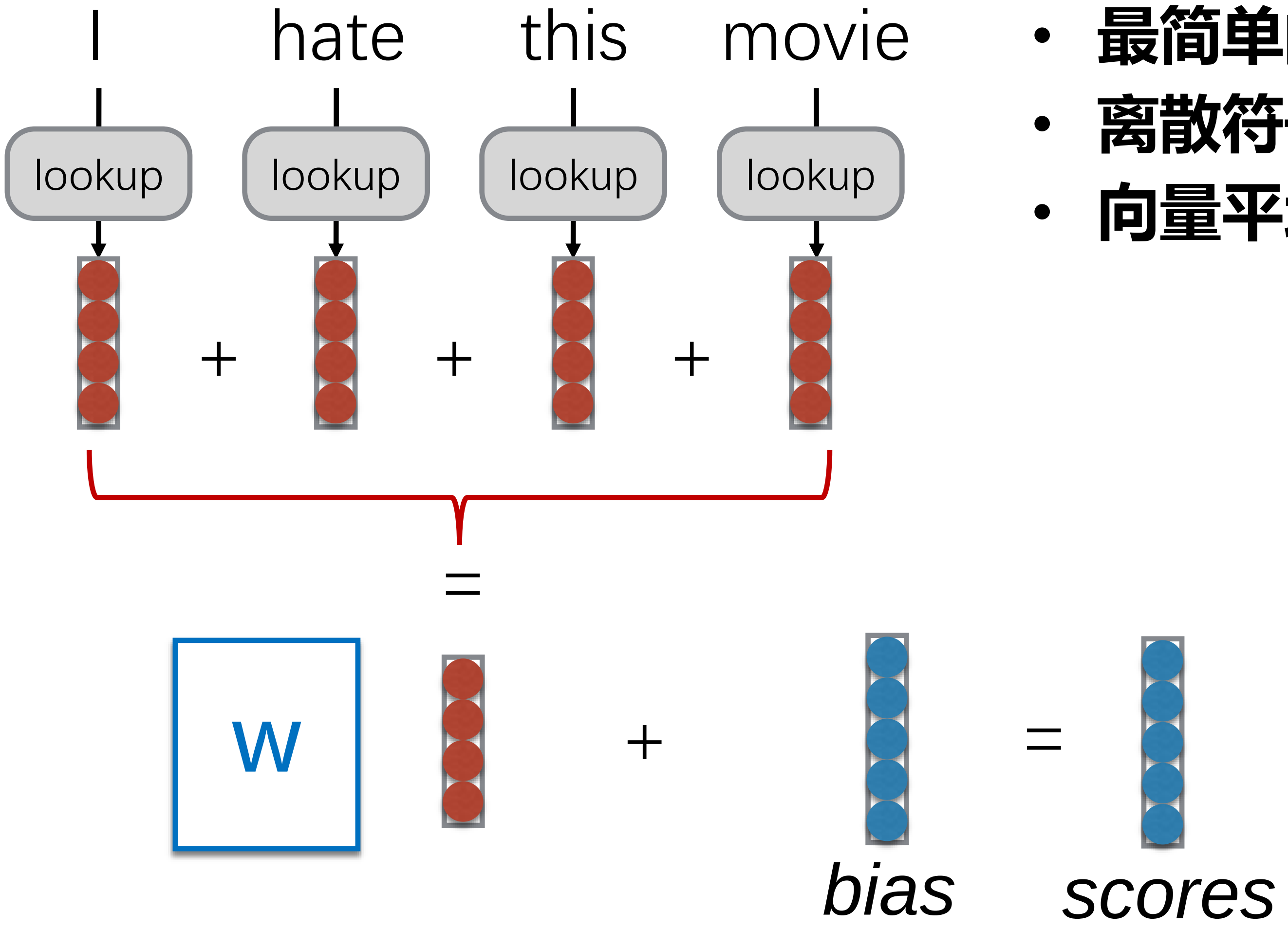
very good
good
neutral
bad
very bad

I love this movie

very good
good
neutral
bad
very bad

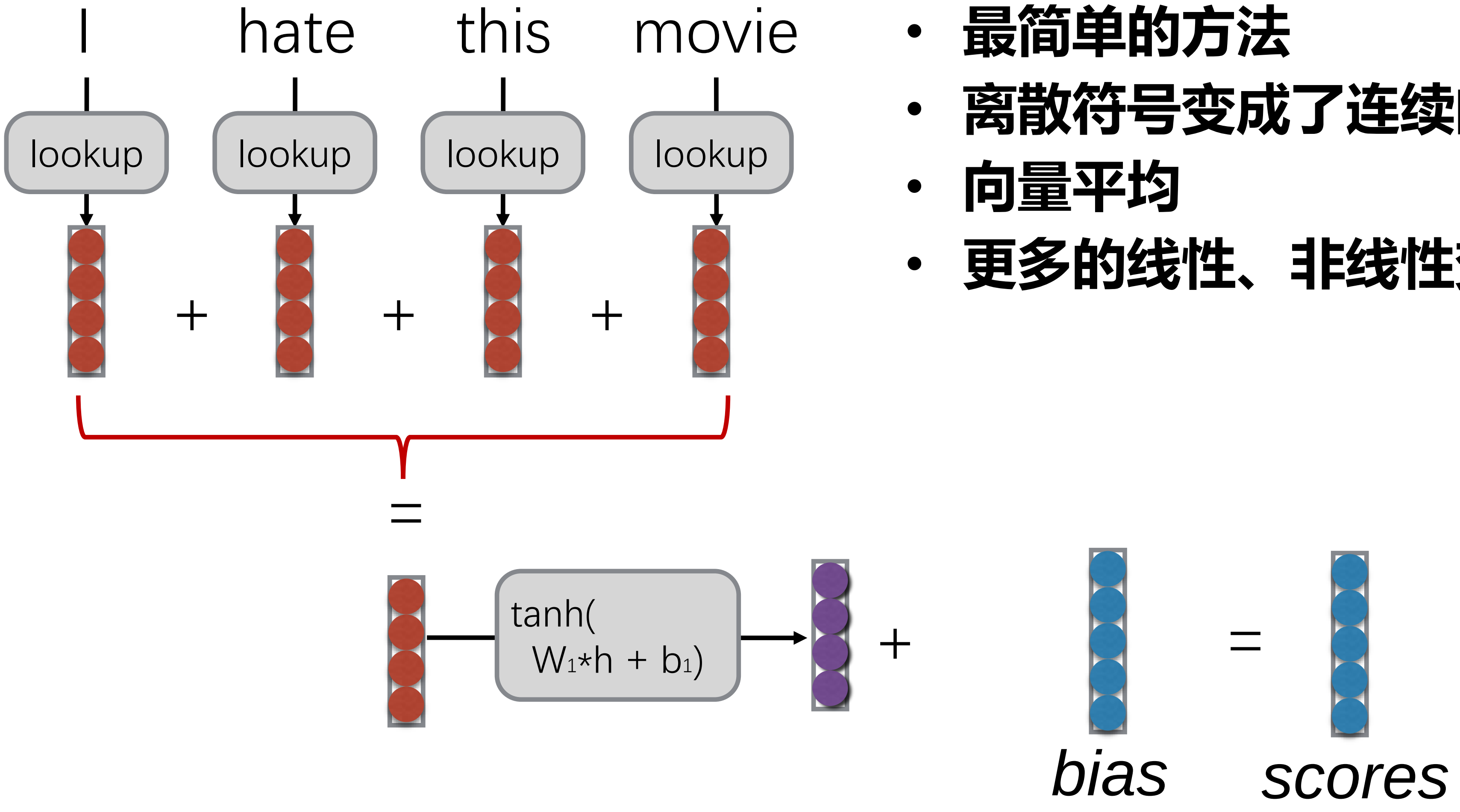
我们的机器如何完成这项任务?

Continuous Bag of Words (CBOW)



- 最简单的方法
- 离散符号变成了连续向量
- 向量平均

Continuous Bag of Words (CBOW)



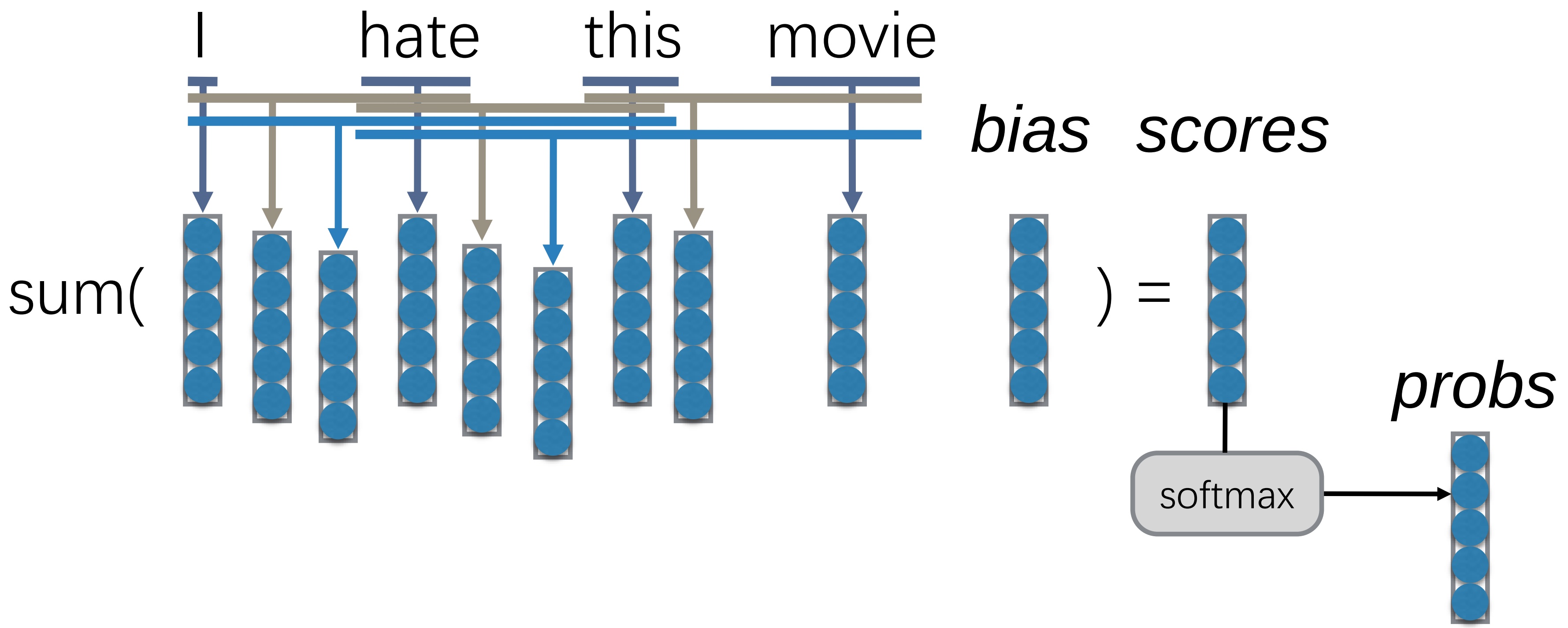
- 最简单的方法
- 离散符号变成了连续向量
- 向量平均
- 更多的线性、非线性变换

深度网络层的作用

- 多个MLP（多层感知器）层可以让我们轻松学习特征组合
- 例如，捕捉到像“not”和“hate”这样的组合
- 但是！无法处理“not hate”这样的情况

如何处理处理组合？

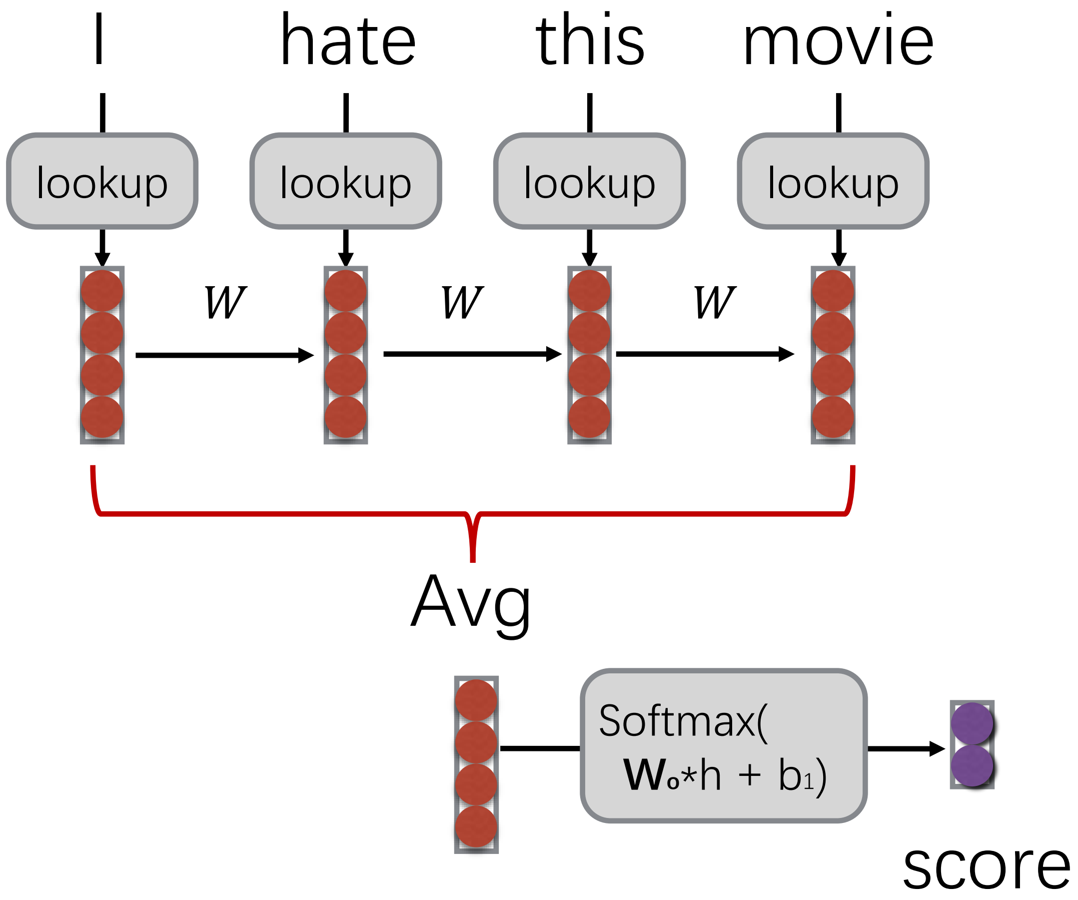
Bag of n-grams



Bag of n-grams的问题

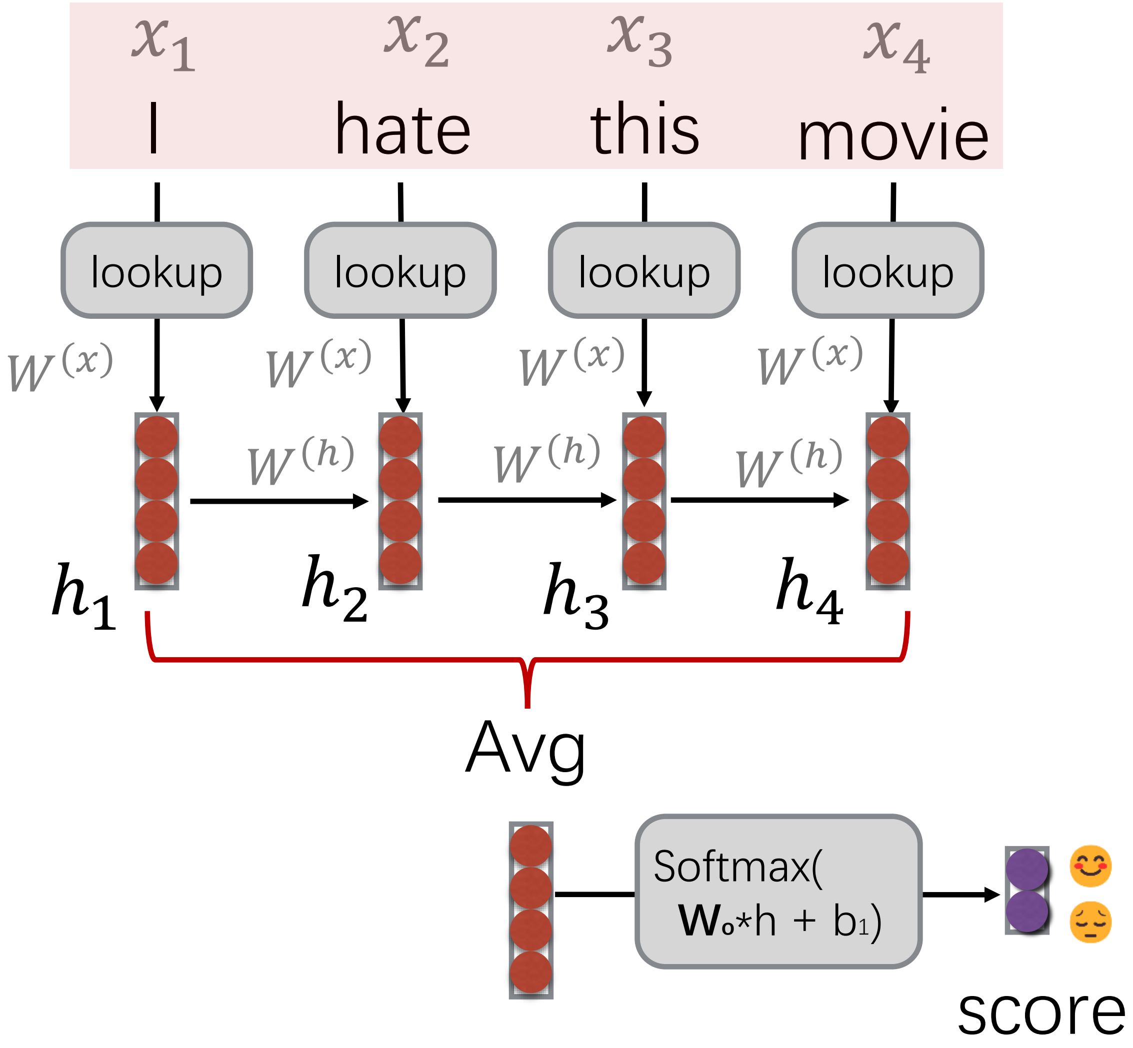
- 与之前相同：参数爆炸
- 相似词/词组之间没有共享
- 丢失了全局序列顺序

循环神经网络(Recurrent Neural Networks)



- 不断用相同的权重处理输入的单词
- 位置敏感
- 支持任意长度的句子

循环神经网络(Recurrent Neural Networks)



输入： x_1, x_2, x_3, x_4

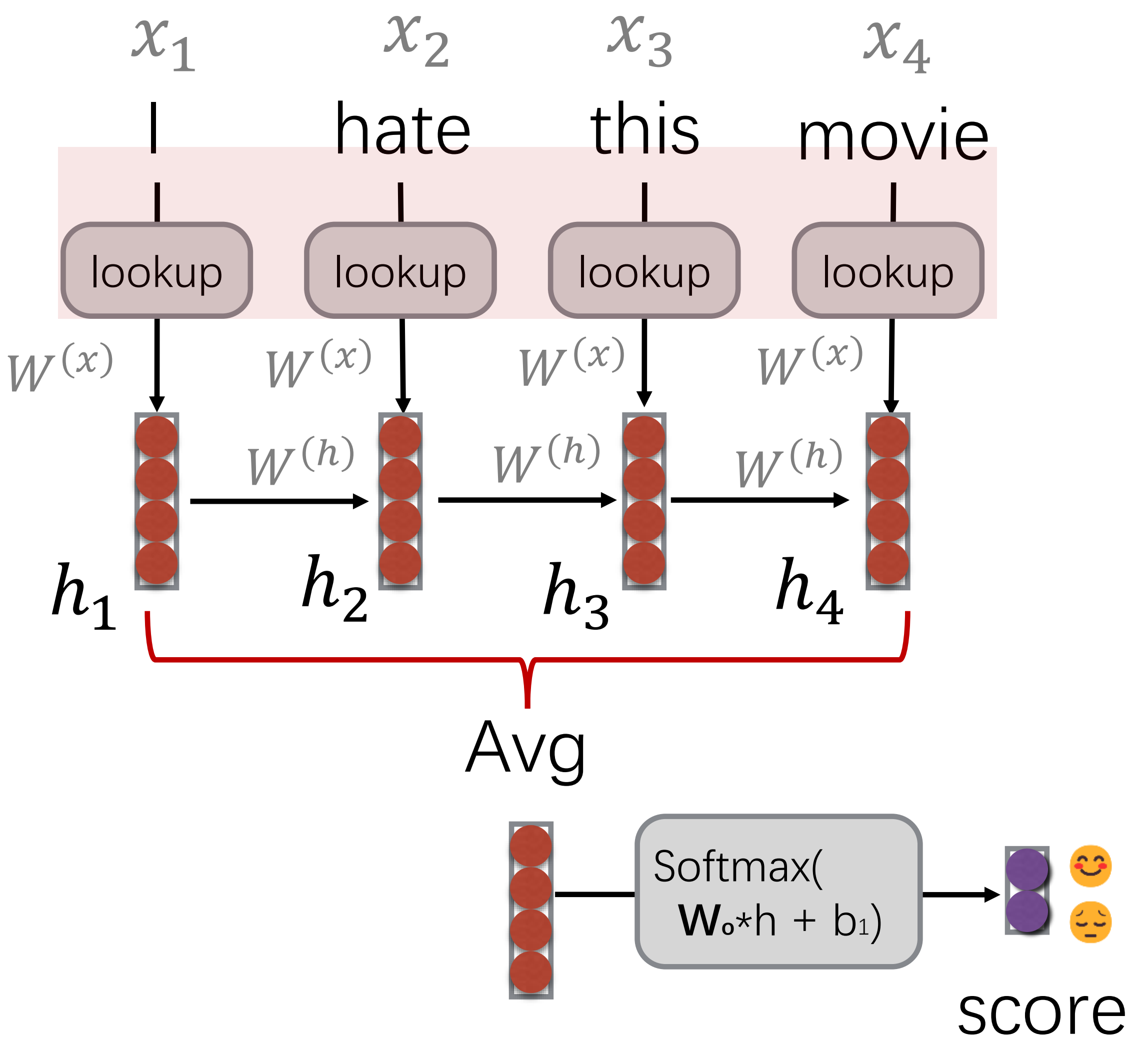
向量化： $\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \boldsymbol{x_4}$

循环计算： $\boldsymbol{h_t} = f\left(\boldsymbol{W^h h_{t-1}} + \boldsymbol{W^x x_t}\right)$
$\boldsymbol{h_0}$需要初始化

向量聚合： $\boldsymbol{h} = \frac{1}{N}\sum_t \boldsymbol{h_t}$

输出计算： $\hat{\boldsymbol{y}} = \mathrm{Soft\,max}\left(\boldsymbol{W^0 h} + b\right)$

循环神经网络(Recurrent Neural Networks)



输入：　x_1, x_2, x_3, x_4

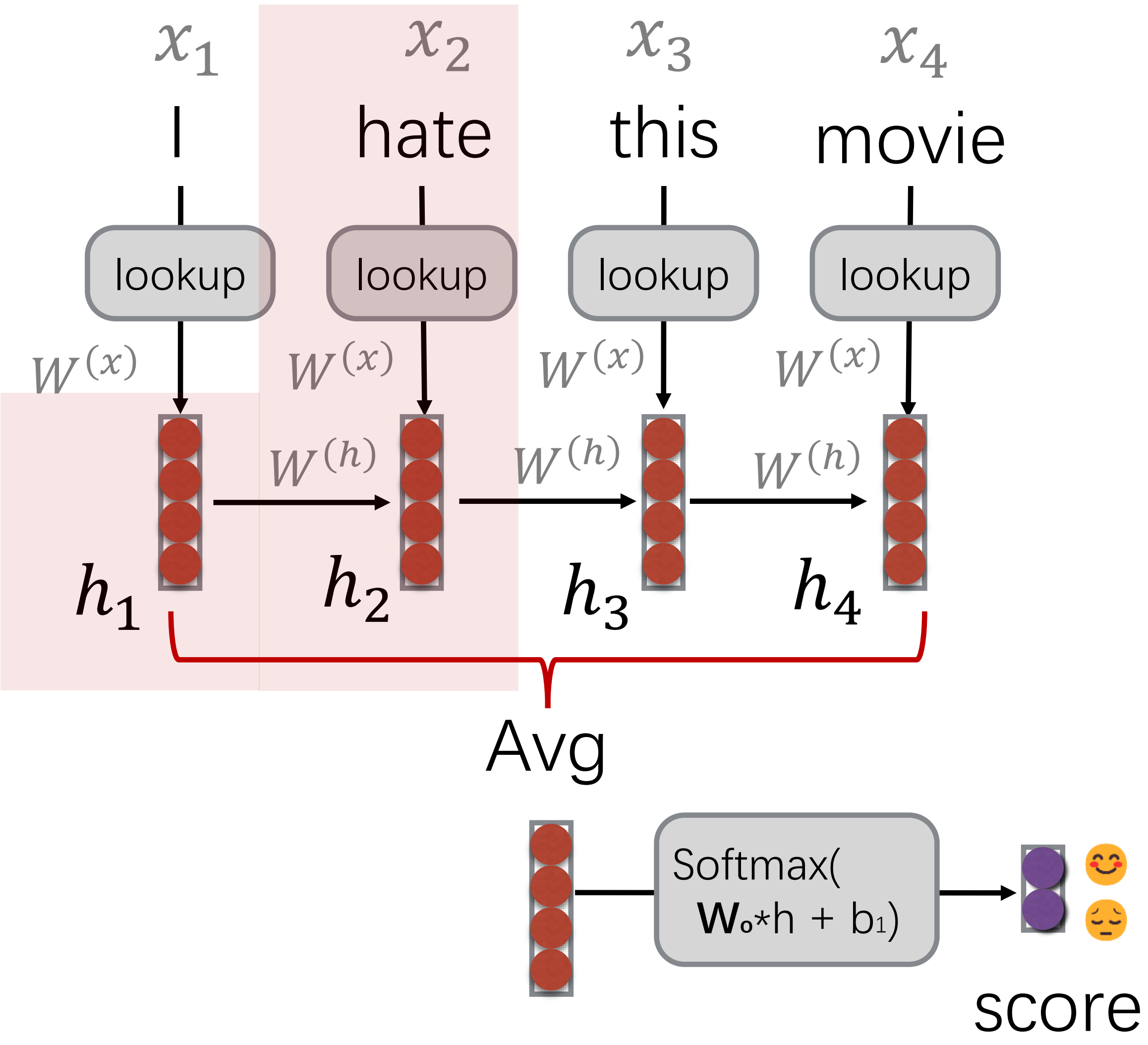
向量化：　$\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \boldsymbol{x_4}$

循环计算：$\boldsymbol{h_t} = f\left(\boldsymbol{W^h h_{t-1}} + \boldsymbol{W^x x_t}\right)$
$\boldsymbol{h}_0$需要初始化

向量聚合：$\boldsymbol{h} = \frac{1}{N}\sum_t \boldsymbol{h_t}$

输出计算：$\hat{\boldsymbol{y}} = \mathrm{Soft\,max}\left(\boldsymbol{W^0 h} + b\right)$

循环神经网络(Recurrent Neural Networks)



输入： x_1, x_2, x_3, x_4

向量化： $\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \boldsymbol{x_4}$

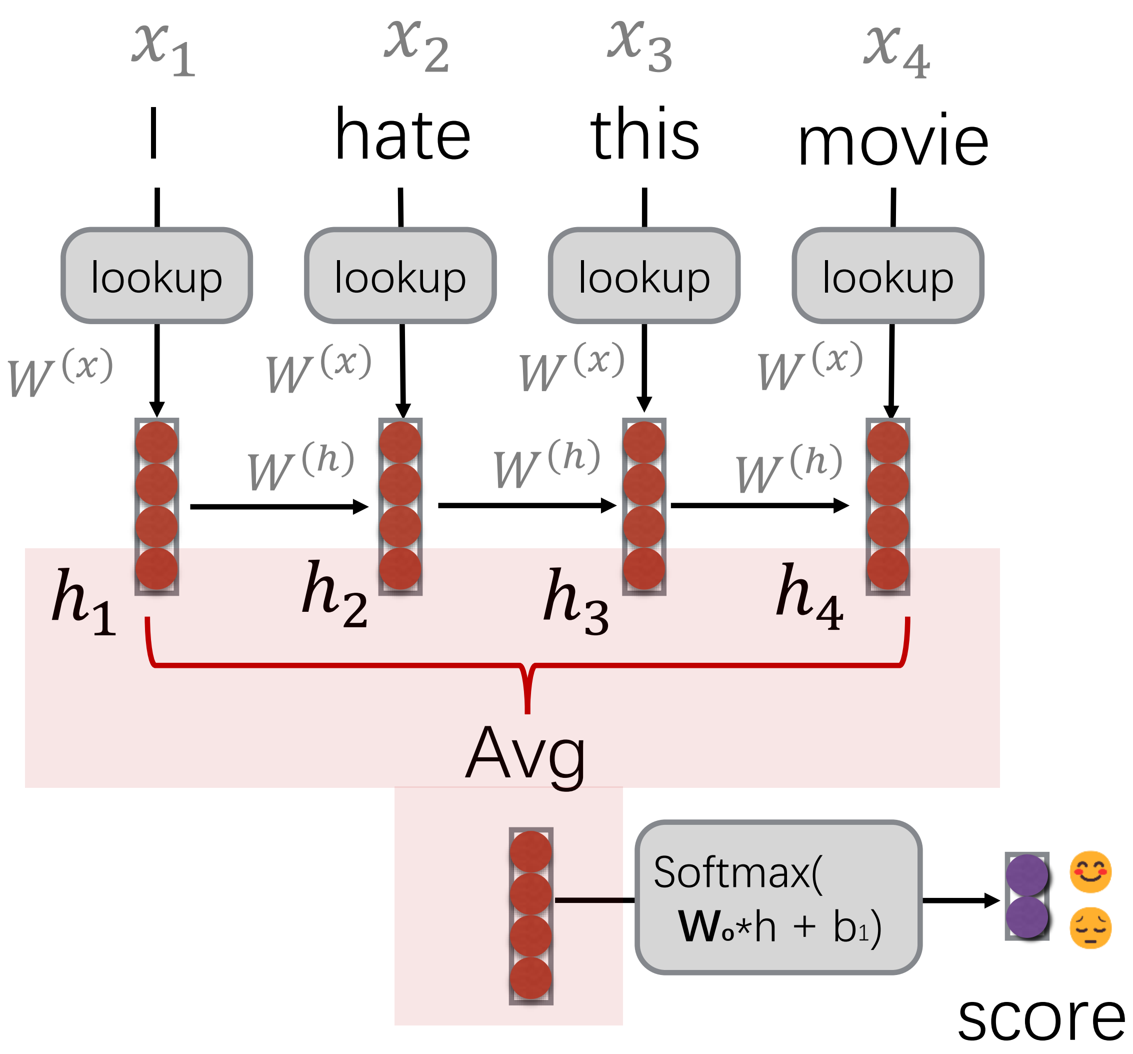
循环计算： $\boldsymbol{h_t} = f\left(\boldsymbol{W^h h_{t-1}} + \boldsymbol{W^x x_t}\right)$

$\boldsymbol{h_0}$需要初始化

向量聚合： $\boldsymbol{h} = \frac{1}{N}\sum_t \boldsymbol{h_t}$

输出计算： $\hat{\boldsymbol{y}} = \mathrm{Soft\,max}\left(\boldsymbol{W^0 h} + b\right)$

循环神经网络(Recurrent Neural Networks)



输入： x_1, x_2, x_3, x_4

向量化： $\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \boldsymbol{x_4}$

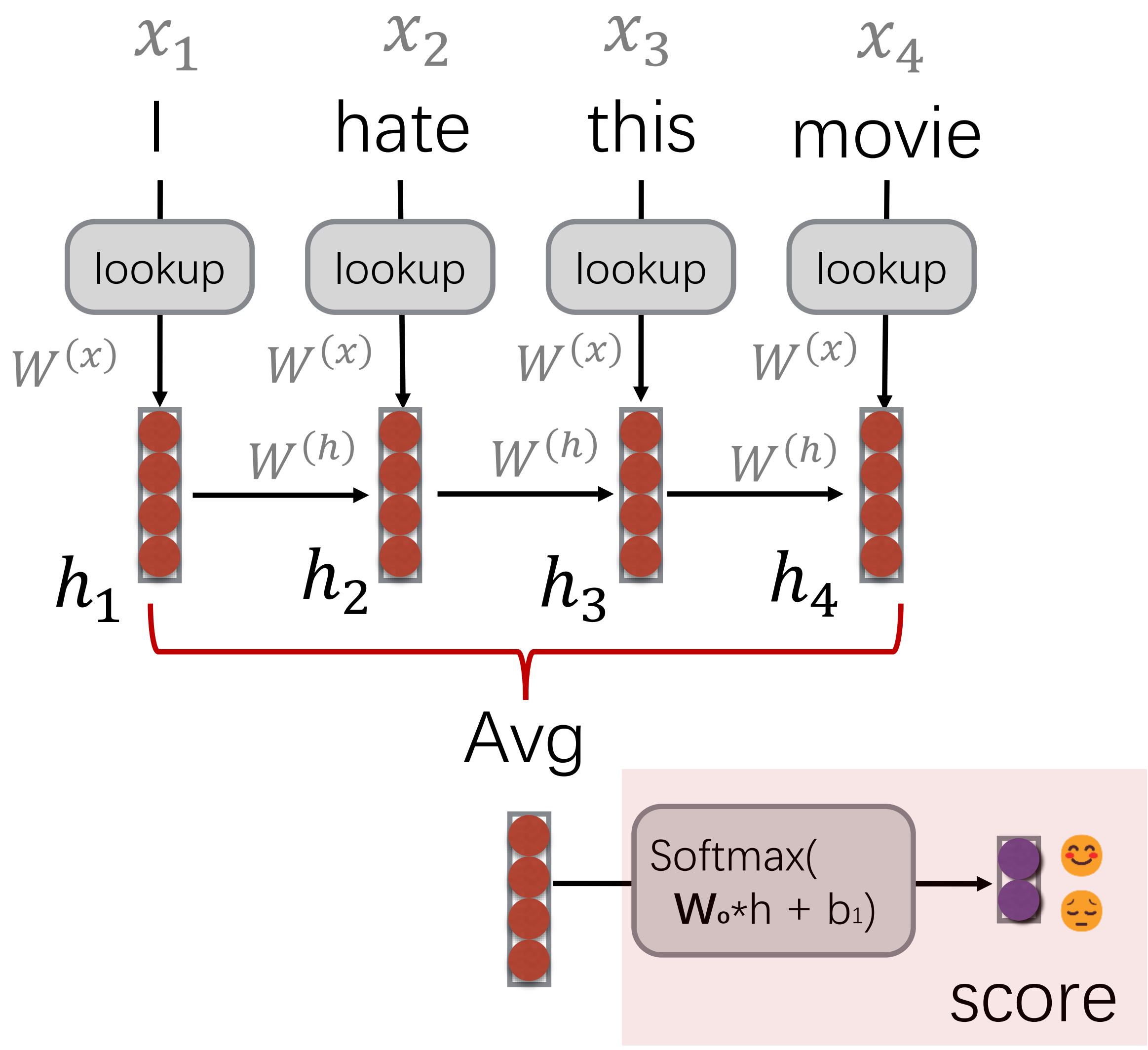
循环计算： $\boldsymbol{h_t} = f\left(\boldsymbol{W}^h \boldsymbol{h}_{t-1} + \boldsymbol{W}^x \boldsymbol{x}_t\right)$

$\boldsymbol{h}_0$需要初始化

向量聚合： $\boldsymbol{h} = \frac{1}{N} \sum_t \boldsymbol{h}_t$

输出计算： $\hat{\boldsymbol{y}} = \mathrm{Soft\max}\left(\boldsymbol{W}^{\boldsymbol{0}} \boldsymbol{h} + b\right)$

循环神经网络(Recurrent Neural Networks)



输入： x_1, x_2, x_3, x_4

向量化： $\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \boldsymbol{x_4}$

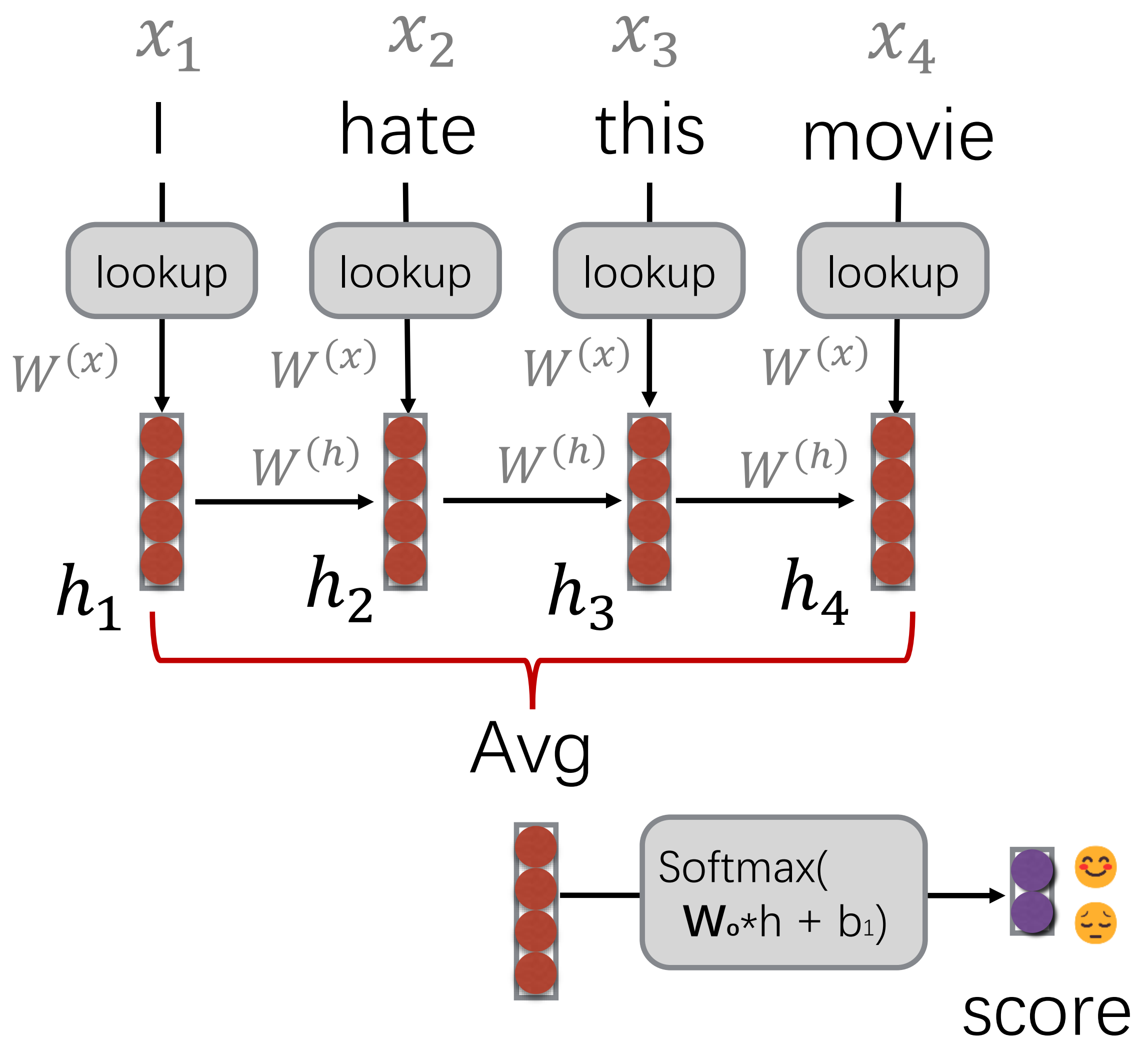
循环计算： $\boldsymbol{h_t} = f\left(\boldsymbol{W}^h \boldsymbol{h}_{t-1} + \boldsymbol{W}^x \boldsymbol{x}_t\right)$

$\boldsymbol{h}_0$需要初始化

向量聚合： $\boldsymbol{h} = \frac{1}{N} \sum_t \boldsymbol{h}_t$

输出计算： $\hat{\boldsymbol{y}} = \mathrm{Soft\,max}\left(\boldsymbol{W}^{\mathbf{0}} \boldsymbol{h} + b\right)$

循环神经网络(Recurrent Neural Networks)

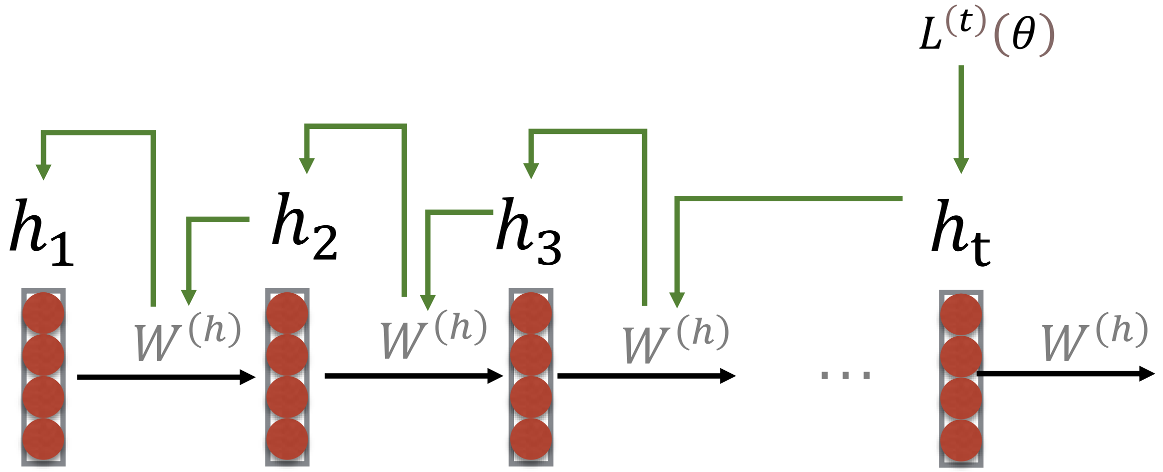


- 优点
  - 可以处理变长的序列
  - 可以把词序考虑进去
  - 可以考虑上下文信息
  - 参数数量和序列长度无关
- 缺点
  - 计算难以并行
  - 难以捕捉长距离依赖

循环神经网络的训练

如何计算 $\frac{\partial L^{(t)}}{\partial W^h}$

答案：$\frac{\partial L^{(t)}}{\partial w^h} = \sum_{i=1}^{t} \frac{\partial L^{(t)}}{\partial w^h}\bigg|_i$



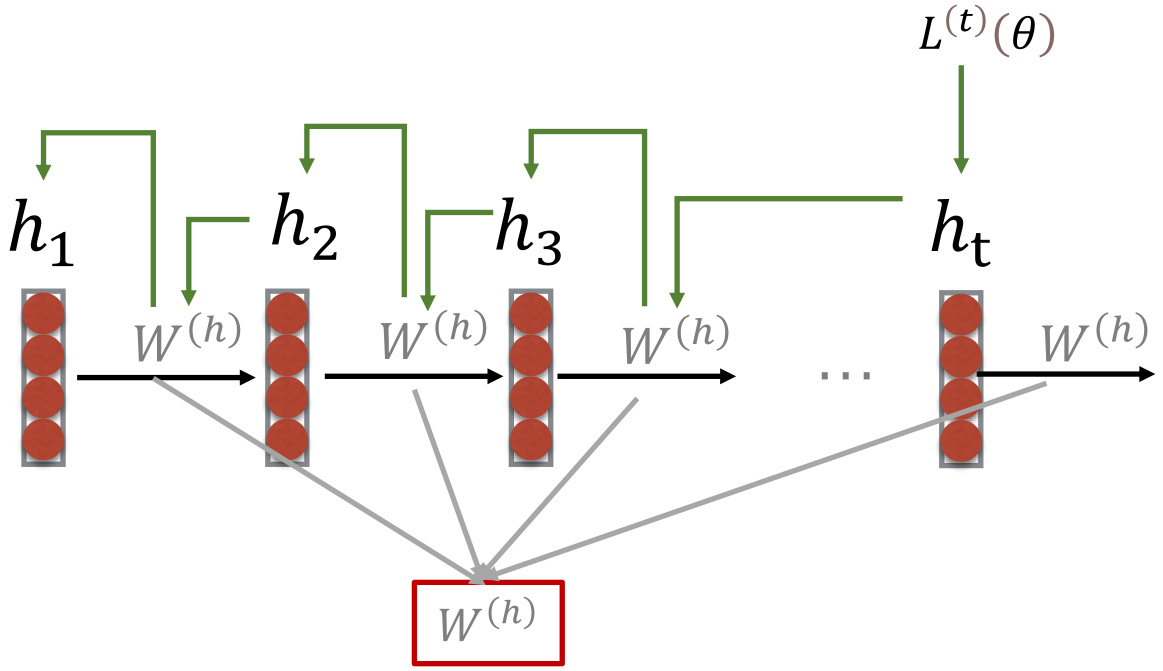
在神经网络里，参数参与过多少次运算，就应该获得多少次梯度的“奖励”

$$\underbrace{f\Big(W^h \underbrace{f\Big(W^h \underbrace{f\Big(W^h \underbrace{f\big(W^h h_0 + W^x x_1\big)}_{h_1} + W^x x_2\Big)}_{h_2} + W^x x_3\Big)}_{h_3} \cdots W^x x_t\Big)}_{h_t}$$

循环神经网络的训练

如何计算 $\frac{\partial L^{(t)}}{\partial W^h}$

答案： $\frac{\partial L^{(t)}}{\partial w^h} = \sum_{i=1}^{t} \left. \frac{\partial L^{(t)}}{\partial w^h} \right|_i$

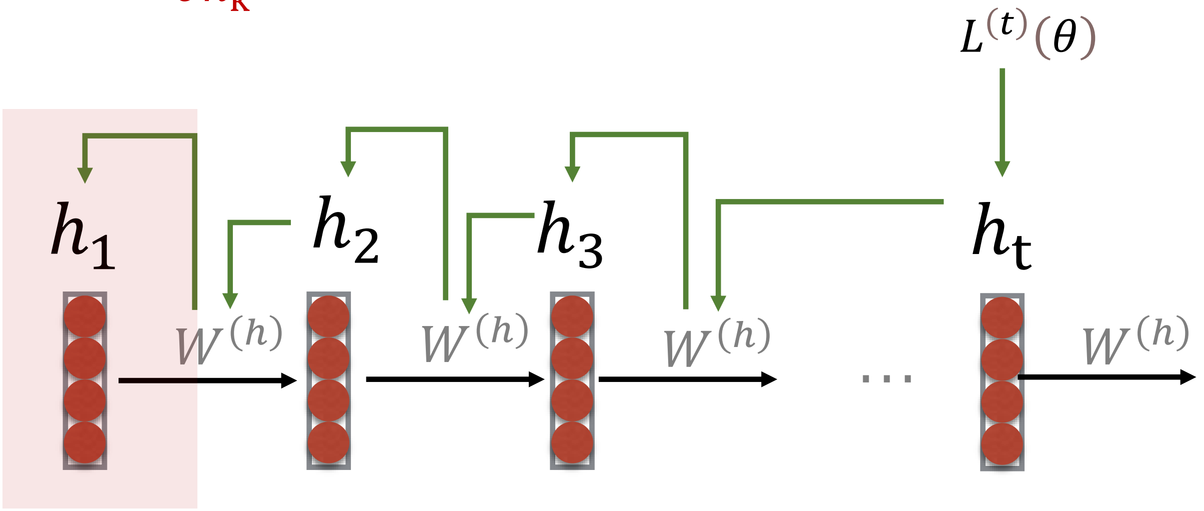


在神经网络里，参数参与过多少次运算，就应该获得多少次梯度的“奖励”

$$\underbrace{f(W^h \underbrace{f\left(W^h \underbrace{f\left(W^h f\left(\underbrace{W^h h_0 + W^x x_1}_{h_1}\right) + W^x x_2\right)}_{h_2} + W^x x_3\right)}_{h_3} \cdots W^x x_t)}_{h_t}$$

循环神经网络的训练

如何计算 $\frac{\partial L^{(t)}}{\partial h_k}$



在神经网络里，参数参与过多少次运算，就应该获得多少次梯度的“奖励”

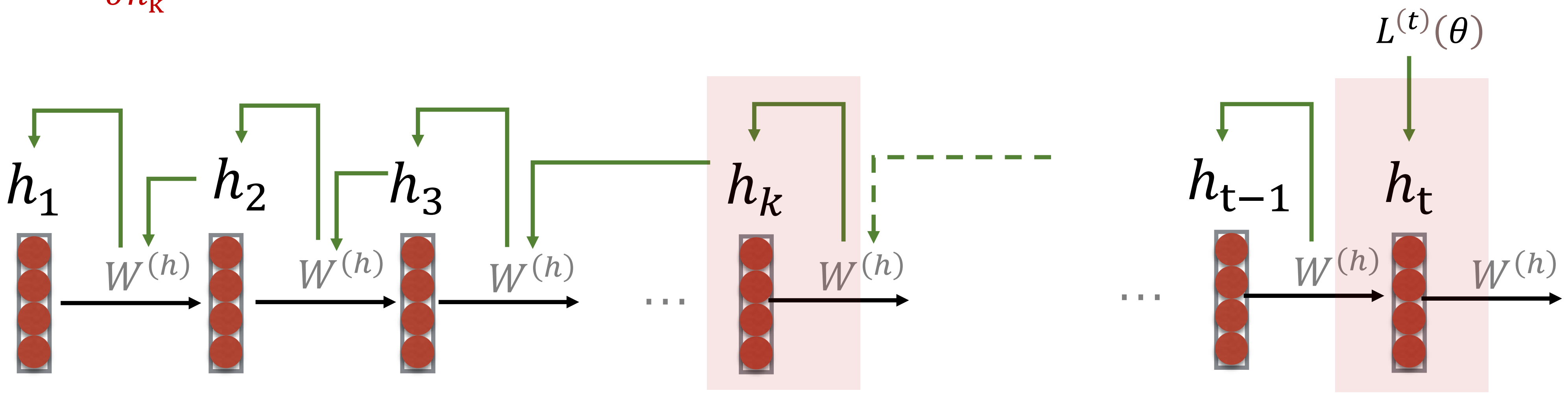
$$\frac{\partial L^{(t)}}{\partial h_1} = \frac{\partial h_2}{\partial h_1}\frac{\partial L^{(t)}}{\partial h_2}$$

$$\frac{\partial L^{(t)}}{\partial h_1} = \frac{\partial h_2}{\partial h_1}\frac{\partial h_3}{\partial h_2}\frac{\partial L^{(t)}}{\partial h_3}$$

$$\frac{\partial L^{(t)}}{\partial h_1} = \frac{\partial h_2}{\partial h_1}\frac{\partial h_3}{\partial h_2}\cdots\frac{\partial h_t}{\partial h_{t-1}}\frac{\partial L^{(t)}}{\partial h_t}$$

循环神经网络的训练

如何计算 $\frac{\partial L^{(t)}}{\partial h_{\mathrm{k}}}$



$$\frac{\partial L^{(t)}}{\partial h_{\mathrm{k}}} = \frac{\partial h_{\mathrm{k+1}}}{\partial h_{\mathrm{k}}} \cdots \boxed{\frac{\partial h_{\mathrm{t}}}{\partial h_{\mathrm{t-1}}}} \frac{\partial L^{(t)}}{\partial h_{\mathrm{t}}} \longrightarrow$$

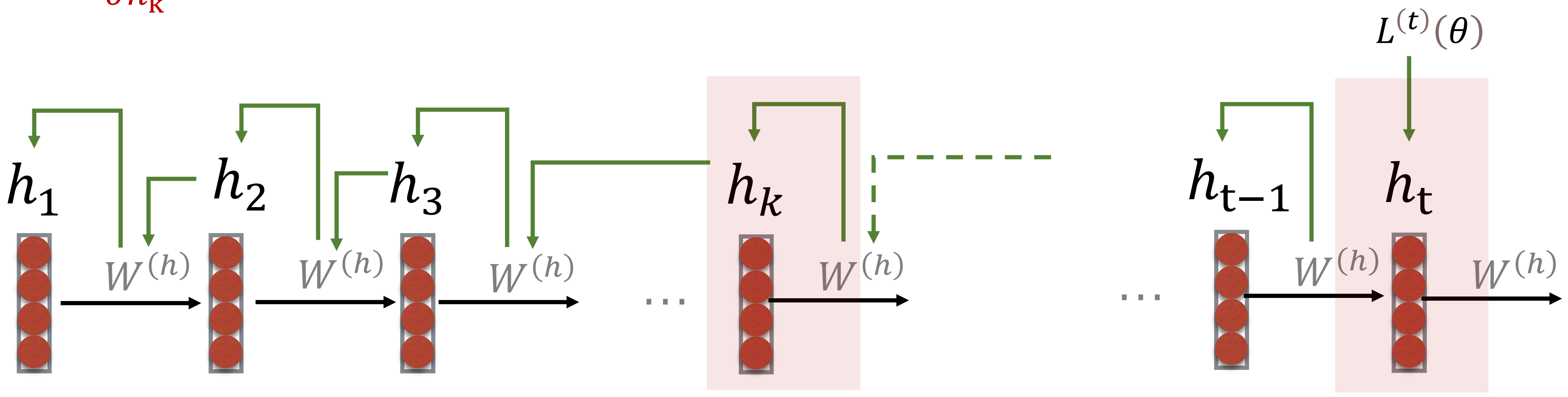
$$\boxed{\begin{aligned} \boldsymbol{h_t} &= f\left(\boldsymbol{W}^h \boldsymbol{h}_{t-1} + \boldsymbol{W}^x \boldsymbol{x}_t\right) \\ \frac{\partial h_t}{\partial h_{t-1}} &= \operatorname{diag}\left(f'\left(\boldsymbol{W}^h \boldsymbol{h}_{t-1} + \boldsymbol{W}^x \boldsymbol{x}_t\right)\right) \boldsymbol{W}_h \end{aligned}}$$

$$= \frac{\partial L^{(t)}}{\partial h_{\mathrm{t}}} \prod_{k<i\le t} \frac{\partial h_i}{\partial h_{i-1}}$$

$$= \frac{\partial L^{(t)}}{\partial h_{\mathrm{t}}} \prod_{k<i\le t} \operatorname{diag}\left(f'\left(\boldsymbol{W}^h \boldsymbol{h}_{\mathrm{i}-1} + \boldsymbol{W}^x \boldsymbol{x}_{\boldsymbol{i}}\right)\right) \boldsymbol{W}_h$$

循环神经网络的训练

如何计算 $\frac{\partial L^{(t)}}{\partial h_k}$



$$\frac{\partial L^{(t)}}{\partial h_k} = \frac{\partial h_{k+1}}{\partial h_k} \cdots \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial L^{(t)}}{\partial h_t}$$

$$= \frac{\partial L^{(t)}}{\partial h_t} \prod_{k<i\le t} \frac{\partial h_i}{\partial h_{i-1}}$$

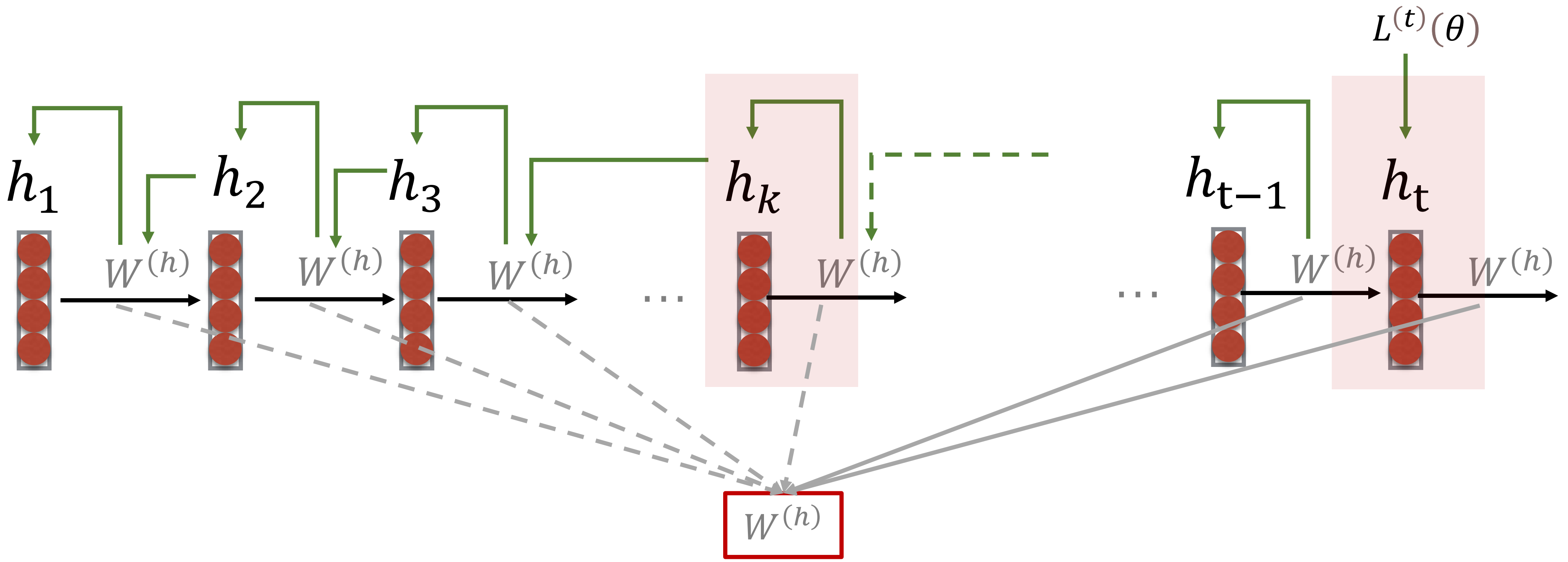
$$\boxed{= \frac{\partial L^{(t)}}{\partial h_t} \prod_{k<i\le t} \operatorname{diag}\left(f'\left(\boldsymbol{W}^h \boldsymbol{h}_{i-1} + \boldsymbol{W}^x \boldsymbol{x}_i\right)\right) \boldsymbol{W}_h}$$

如果$t-k$很大，且 $\operatorname{diag}\left(f'\left(\boldsymbol{W}^h \boldsymbol{h}_{i-1} + \boldsymbol{W}^x \boldsymbol{x}_i\right)\right) \boldsymbol{W}_h$很小 $\frac{\partial L^{(t)}}{\partial h_k}$趋近于零

梯度弥散

如果很大则会有梯度梯度爆炸

梯度弥散的影响



梯度弥散并不意味着$\boldsymbol{W}^{(\boldsymbol{h})}$无法**获得梯度**，而是无法获得**长距离单词传过来的梯度**，使得模型难以**捕捉到长距离的依赖关系**

长距离依赖

在一个晴朗的秋日，李明决定带着他的狗旺财去附近的公园散步。公园里人不多，让他感到非常放松。他注意到公园的一角正在举行一个小型的户外画展，展出了一些当地艺术家的作品。李明对艺术一直有所兴趣，尤其是绘画，所以他决定过去看看。在画展中，他被一幅描绘秋天景象的油画深深吸引，画中的色彩和细节处理让他联想到了他孩提时代的一些美好回忆。心动之下，李明决定购买这幅画作为他的生日礼物。然而，他突然意识到自己没有带足够的现金，也忘记带银行卡。这时，画展的组织者提出一个建议，如果李明能够解答一个关于展览主题的小谜题，他们愿意以优惠价提供这幅画。**李明的狗的名字是__**.

常见解决方案

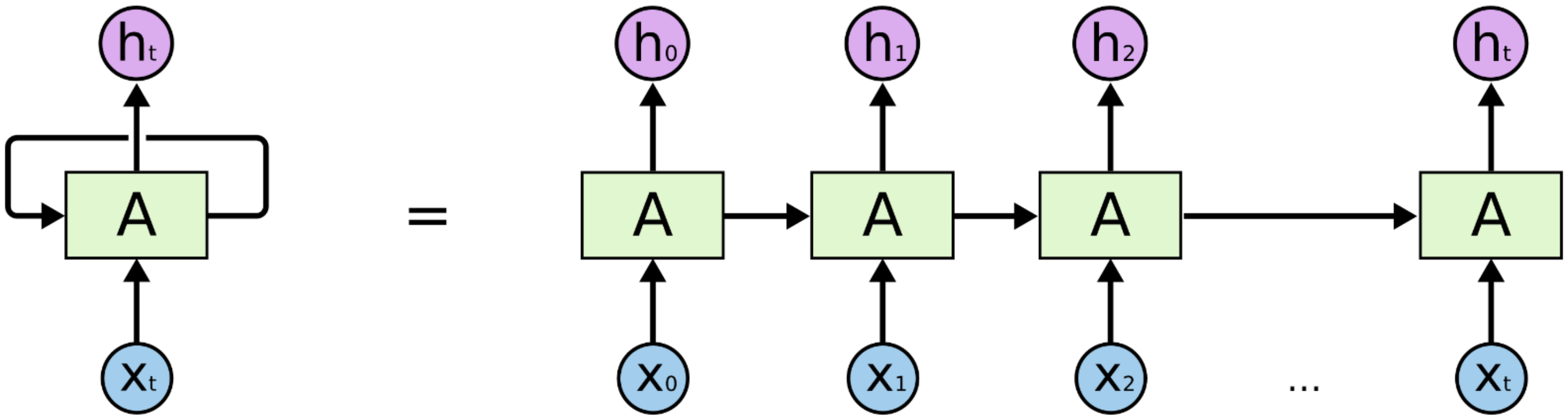
- 梯度爆炸
  - 梯度阶段 (Gradient Clipping)

Algorithm 1 Pseudo-code for norm clipping

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$
end if

- 梯度弥散
  - 是一个更难得问题，即如何让神经网络长时间的保留信息

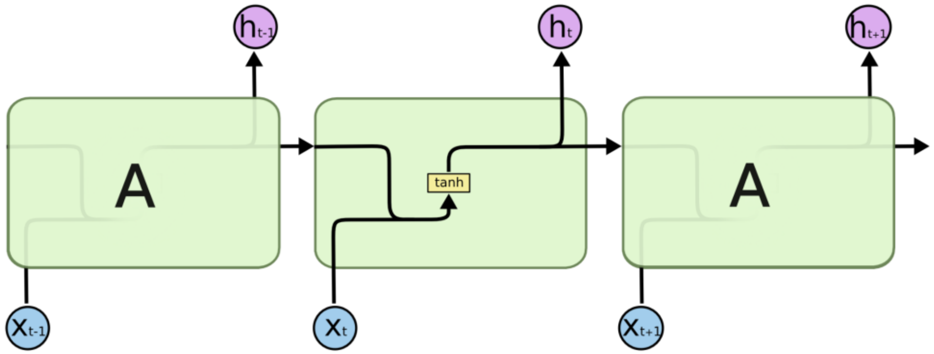
长短期记忆网络 (Long Short-Term Memory)



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

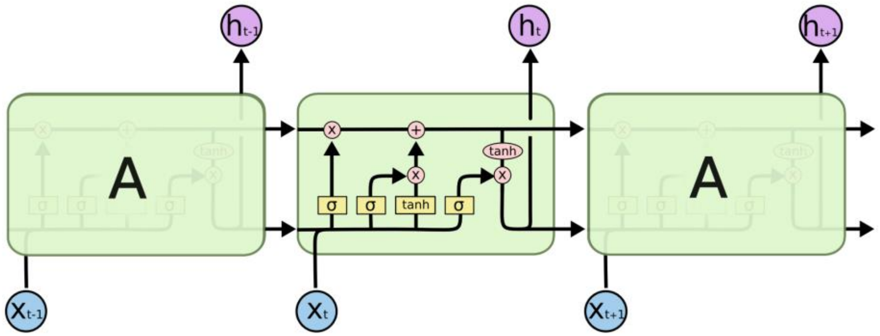
长短期记忆网络 (Long Short-Term Memory)

普通的RNN



$$\boxed{\boldsymbol{h_t} = \boldsymbol{tanh}\left(\boldsymbol{W}^h \boldsymbol{h}_{t-1} + \boldsymbol{W}^x \boldsymbol{x}_t\right)}$$

LSTM

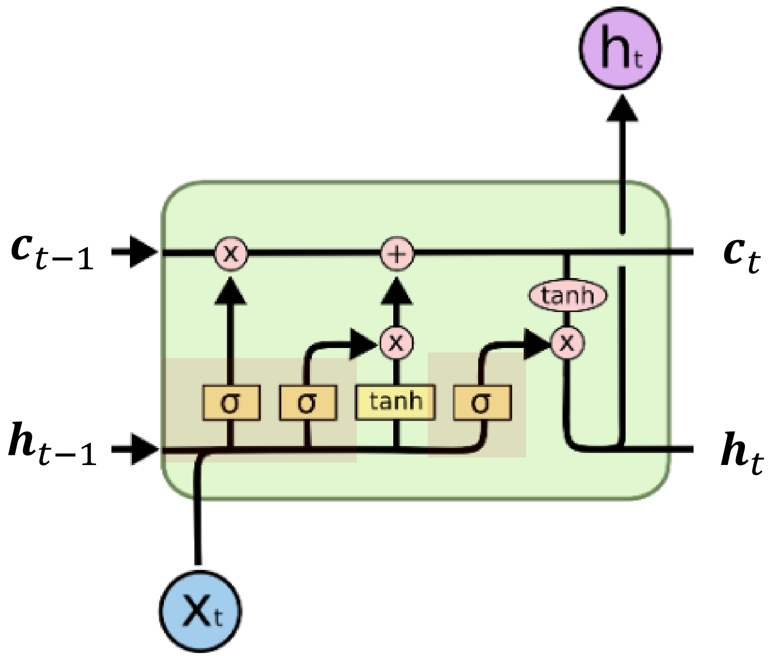


$$\boldsymbol{f}_t = \sigma\left(\boldsymbol{W}^f h_{t-1} + \boldsymbol{U}^f \boldsymbol{x_t} + b_f\right)$$
$$\boldsymbol{i}_t = \sigma\left(\boldsymbol{W}^i h_{t-1} + \boldsymbol{U}^i \boldsymbol{x_t} + b_i\right)$$
$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}^o h_{t-1} + \boldsymbol{U}^o \boldsymbol{x_t} + b_o)$$

$$\widetilde{\boldsymbol{c}}_{\boldsymbol{t}} = \sigma(\boldsymbol{W}^c h_{t-1} + \boldsymbol{U}^c \boldsymbol{x_t} + b_c)$$
$$\boldsymbol{c}_t = \boldsymbol{f}_t \cdot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \cdot \widetilde{\boldsymbol{c}}_{\boldsymbol{t}}$$
$$\boxed{\boldsymbol{h_t} = \boldsymbol{o}_t \cdot \boldsymbol{tanh}(\boldsymbol{c}_t)}$$

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

长短期记忆网络 (Long Short-Term Memory)



$$\boldsymbol{f}_t = \sigma\left(\boldsymbol{W}^f \boldsymbol{h}_{t-1} + \boldsymbol{U}^f \boldsymbol{x}_t + b_f\right)$$

遗忘门：控制上个状态信息是否保留和遗忘

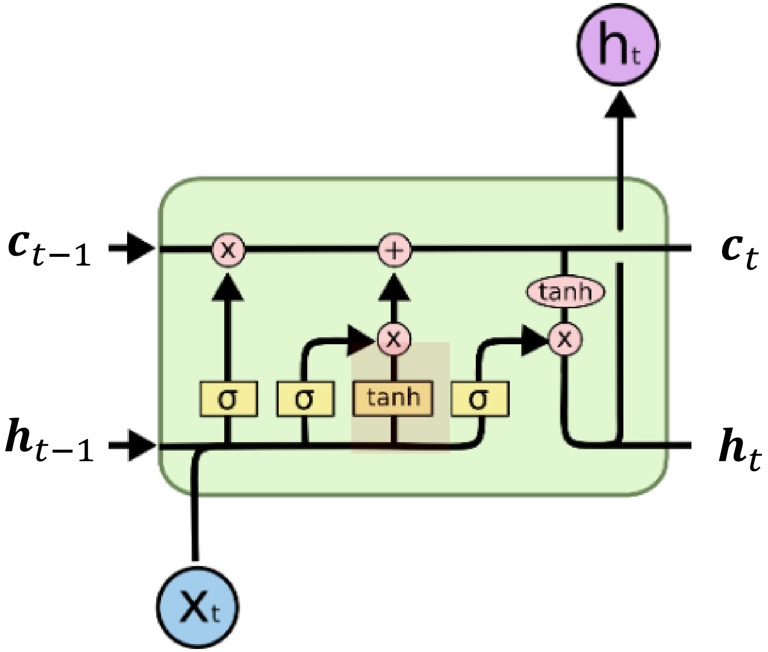
$$\boldsymbol{i}_t = \sigma\left(\boldsymbol{W}^i \boldsymbol{h}_{t-1} + \boldsymbol{U}^i \boldsymbol{x}_t + b_i\right)$$

输入门：控制哪些新的信息被写入

$$\boldsymbol{o}_t = \sigma\left(\boldsymbol{W}^o \boldsymbol{h}_{t-1} + \boldsymbol{U}^o \boldsymbol{x}_t + b_o\right)$$

输出门：控制那部分信息被输出到隐层

长短期记忆网络 (Long Short-Term Memory)



$$\boldsymbol{f}_t = \sigma\left(\boldsymbol{W}^f \boldsymbol{h}_{t-1} + \boldsymbol{U}^f \boldsymbol{x}_t + b_f\right)$$

遗忘门：控制上个状态信息是否保留和遗忘

$$\boldsymbol{i}_t = \sigma\left(\boldsymbol{W}^i \boldsymbol{h}_{t-1} + \boldsymbol{U}^i \boldsymbol{x}_t + b_i\right)$$

输入门：控制哪些新的信息被写入

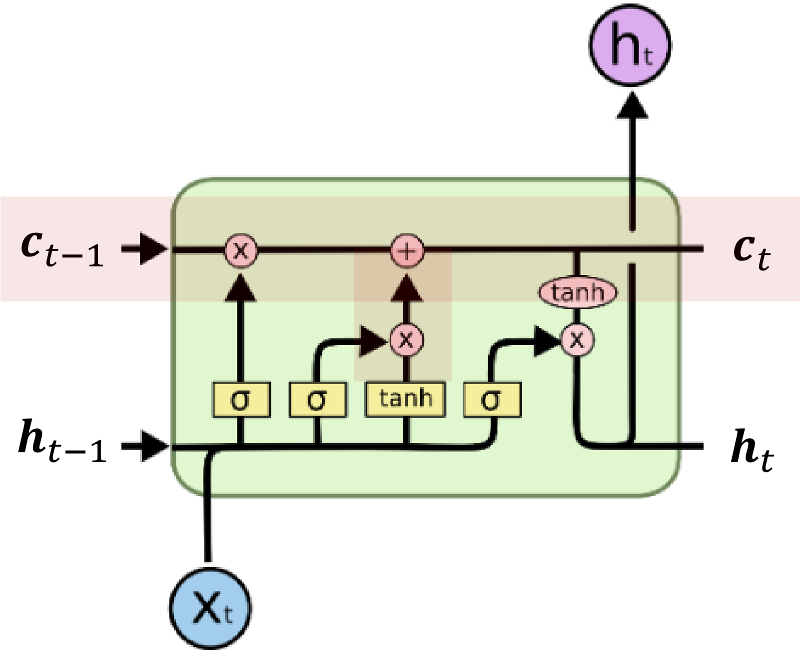
$$\boldsymbol{o}_t = \sigma\left(\boldsymbol{W}^o \boldsymbol{h}_{t-1} + \boldsymbol{U}^o \boldsymbol{x}_t + b_o\right)$$

输出门：控制那部分信息被输出到隐层

$$\tilde{\boldsymbol{c}}_t = \sigma\left(\boldsymbol{W}^c \boldsymbol{h}_{t-1} + \boldsymbol{U}^c \boldsymbol{x}_t + b_c\right)$$

新记忆单元：新生成的信息内容

长短期记忆网络 (Long Short-Term Memory)



$$f_t = \sigma\left(W^f h_{t-1} + U^f x_t + b_f\right)$$

遗忘门：控制上个状态信息是否保留和遗忘

$$i_t = \sigma\left(W^i h_{t-1} + U^i x_t + b_i\right)$$

输入门：控制哪些新的信息被写入

$$o_t = \sigma\left(W^o h_{t-1} + U^o x_t + b_o\right)$$

输出门：控制那部分信息被输出到隐层

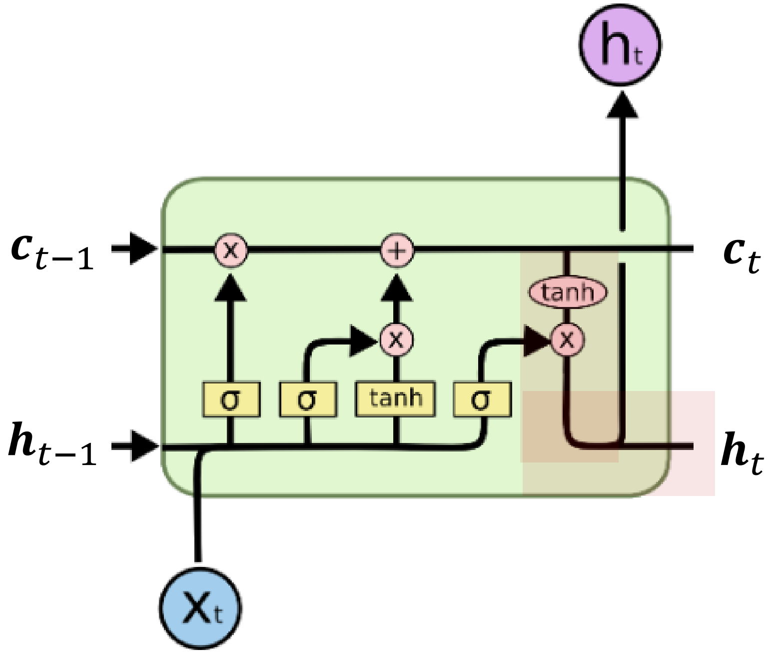
$$\tilde{c}_t = \sigma\left(W^c h_{t-1} + U^c x_t + b_c\right)$$

新记忆单元：新生成的信息内容

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

记忆单元：擦掉旧的并写入新的内容

长短期记忆网络 (Long Short-Term Memory)



$$\boldsymbol{f}_t = \sigma\left(\boldsymbol{W}^f \boldsymbol{h}_{t-1} + \boldsymbol{U}^f \boldsymbol{x}_t + b_f\right)$$

遗忘门：控制上个状态信息是否保留和遗忘

$$\boldsymbol{i}_t = \sigma\left(\boldsymbol{W}^i \boldsymbol{h}_{t-1} + \boldsymbol{U}^i \boldsymbol{x}_t + b_i\right)$$

输入门：控制哪些新的信息被写入

$$\boldsymbol{o}_t = \sigma\left(\boldsymbol{W}^o \boldsymbol{h}_{t-1} + \boldsymbol{U}^o \boldsymbol{x}_t + b_o\right)$$

输出门：控制那部分信息被输出到隐层

$$\tilde{\boldsymbol{c}}_t = \sigma\left(\boldsymbol{W}^c \boldsymbol{h}_{t-1} + \boldsymbol{U}^c \boldsymbol{x}_t + b_c\right)$$

新记忆单元：新生成的信息内容

$$\boldsymbol{c}_t = \boldsymbol{f}_t \cdot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \cdot \tilde{\boldsymbol{c}}_t$$

记忆单元：擦掉旧的并写入新的内容

$$\boldsymbol{h}_t = \boldsymbol{o}_t \cdot \boldsymbol{tanh}(\boldsymbol{c}_t)$$

隐层状态：读出一部分记忆内容作为隐层状态

LSTM与梯度弥散理解

- LSTM不能解决梯度弥散/爆炸，只是为学习长距离依赖**提供了架构基础**
- 如果$\boldsymbol{f}_t = 1, \boldsymbol{i}_t$=0, 记忆单元信息将会被永远保存（对于一般的RNN，不具备这个能力）

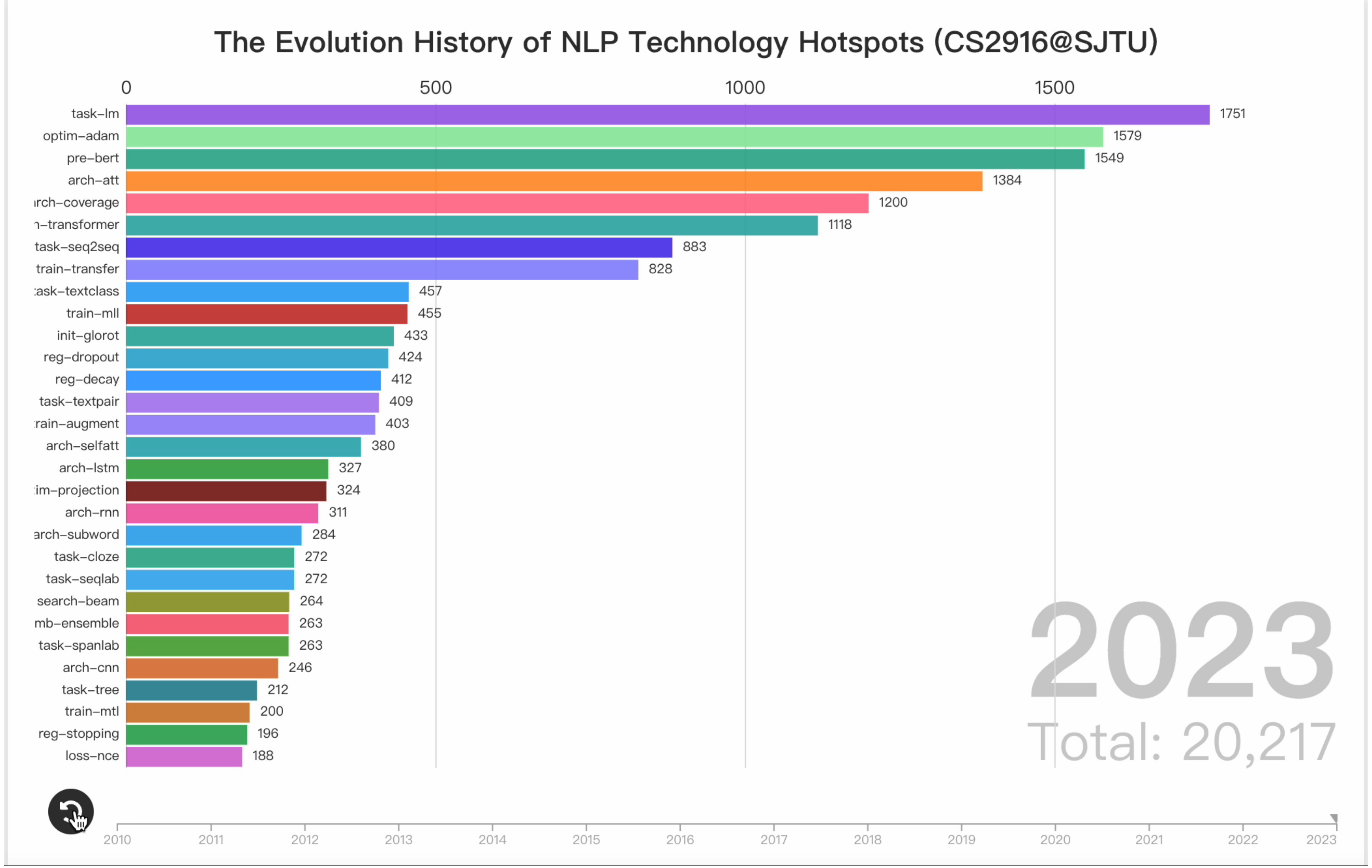
$$\boldsymbol{f}_t = \sigma\left(\boldsymbol{W}^f h_{t-1} + \boldsymbol{U}^f \boldsymbol{x}_t + b_f\right)$$
$$\boldsymbol{i}_t = \sigma\left(\boldsymbol{W}^i h_{t-1} + \boldsymbol{U}^i \boldsymbol{x}_t + b_i\right)$$
$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}^o h_{t-1} + \boldsymbol{U}^o \boldsymbol{x}_t + b_o)$$
$$\tilde{\boldsymbol{c}}_t = \sigma(\boldsymbol{W}^c h_{t-1} + \boldsymbol{U}^c \boldsymbol{x}_t + b_c)$$
$$\boldsymbol{c}_t = \boldsymbol{f}_t \cdot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \cdot \tilde{\boldsymbol{c}}_t$$
$$\boldsymbol{h}_t = \boldsymbol{o}_t \cdot \boldsymbol{tanh}(\boldsymbol{c}_t)$$

LSTM跌宕起伏、颠沛流离的一生

- 1997：Long Short-Term Memory提出
- 2000：引入“遗忘门”
- 2014: 首次成功应用真实任务（并开始流行起来）
- 2019：逐渐退出舞台

Long Short-Term Memory Hochreiter et al. 1997
Learning to forget continual prediction with LSTM Gers et al.2000
Sequence to Sequence Learning with Neural Networks, Sutskever et al 2014

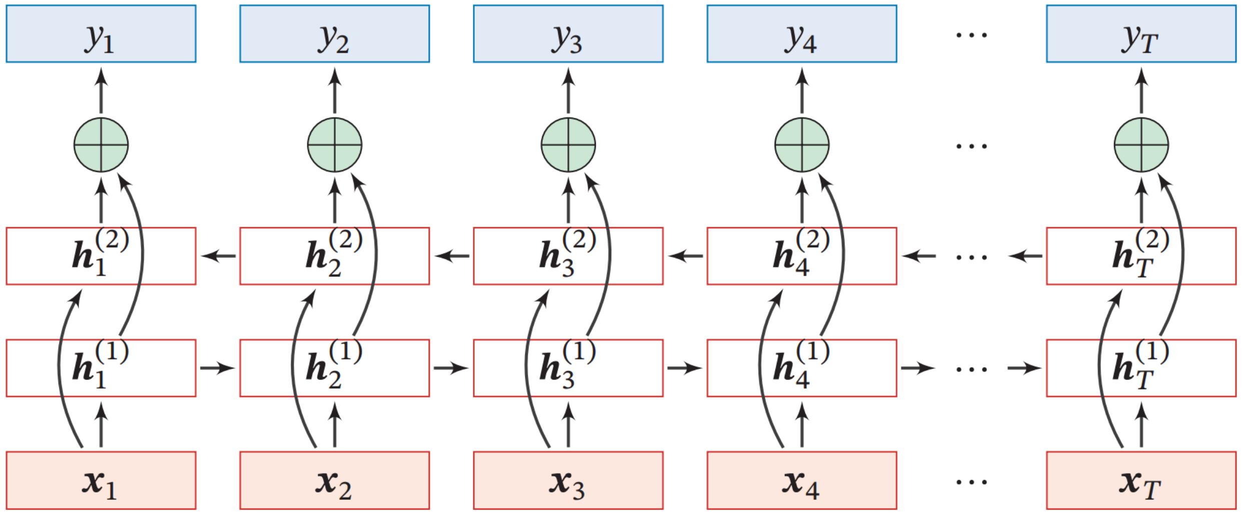
LSTM历史



The Evolution History of NLP Technology Hotspots (CS2916@SJTU)
0
500
1000
1500
task–lm 1751
optim–adam 1579
pre–bert 1549
arch–att 1384
ırch–coverage 1200
ı–transformer 1118
task–seq2seq 883
train–transfer 828
:ask–textclass 457
train–mll 455
init–glorot 433
reg–dropout 424
reg–decay 412
task–textpair 409
rain–augment 403
arch–selfatt 380
arch–lstm 327
im–projection 324
arch–rnn 311
arch–subword 284
task–cloze 272
task–seqlab 272
search–beam 264
mb–ensemble 263
task–spanlab 263
arch–cnn 246
task–tree 212
train–mtl 200
reg–stopping 196
loss–nce 188
2023
Total: 20,217
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023

LSTM的变体

- 双向循环神经网络
  - 适用条件：可以获得全部数据序列
  - 往往非常有效

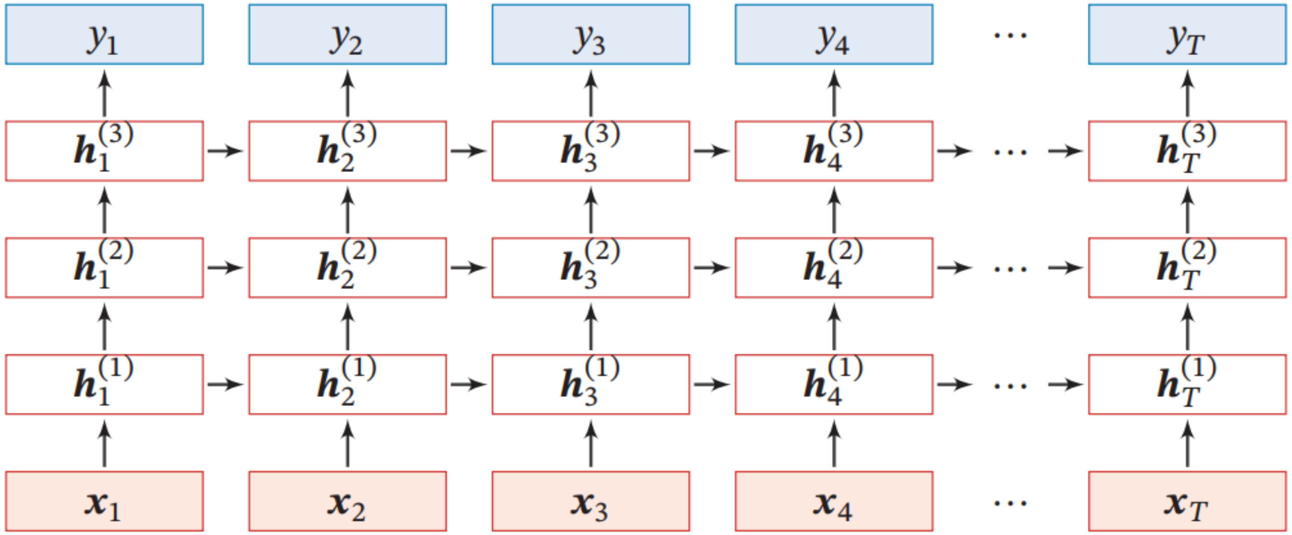


按时间展开的双向循环神经网络

图来源：https://nndl.github.io/nndl-book.pdf

LSTM的变体

- 堆叠的LSTM
  - 时间维度上的“deep”，但是参数共享
  - 网络结构上的“deep”参数不共享



按时间展开的堆叠循环神经网络

图来源：https://nndl.github.io/nndl-book.pdf

梯度弥散在非RNN中的探索

- 深层的网络都会遇到梯度弥散的问题，
  - 链式法则、非线性函数的求导
  - 远离输出层的梯度会越来越小
- 解决方法：
  - 添加直连边
  - 添加Gate边

$$y = F(x, w) + x$$

Residual Network
(He et al.2016)

$$y = \alpha F(x, w) + (1 - \alpha)x$$
$$\alpha = \sigma(Wx + b)$$

Highway Network
(Kumar et al.2015)