法律声明

课程详情请咨询

- 微信公众号：深度之眼
- 客服微信号：deepshare0920

公众号

微 信

deepshare.net
深度之眼

Deep Learning
循环神经网络

导师：Johnson

deepshare.net
深度之眼

循环神经网络

Recurrent Neural Network

主要内容

contents

deepshare.net
深度之眼

- 循环神经网络
  - 简单循环网络
    - 表现方式
    - 反向传播
  - 循环网络常用结构
    - 双向循环网络
    - 深度循环网络
    - 递归神经网络
  - 长期依赖问题
    - 梯度消失
    - 梯度爆炸
  - 门控制循环网络
    - LSTM
    - GRU
  - 长期依赖优化

权值共享

Shared Weight

deepshare.net
深度之眼

循环神经网络就是为了学习卷积神经网络中权值共享等思路，来处理序列化数据，
这就造成了他们有很多类似的地方。

卷积神经网络 vs. 循环神经网络

CNN vs. RNN

deepshare.net
深度之眼

输入的区别：

循环神经网络是一类用于处理序列数据的神经网络

卷积神经网络是一类用于处理网格化数据（如一个图像）的神经网络

卷积神经网络 vs. 循环神经网络

CNN vs. RNN

deepshare.net
深度之眼

数据的输入对比：

循环网络可以扩展到更长的序列。大多数循环网络也能处理可变长度的序列

卷积网络可以很容易地扩展到具有很大宽度和高度的图像，以及处理大小可变的图像

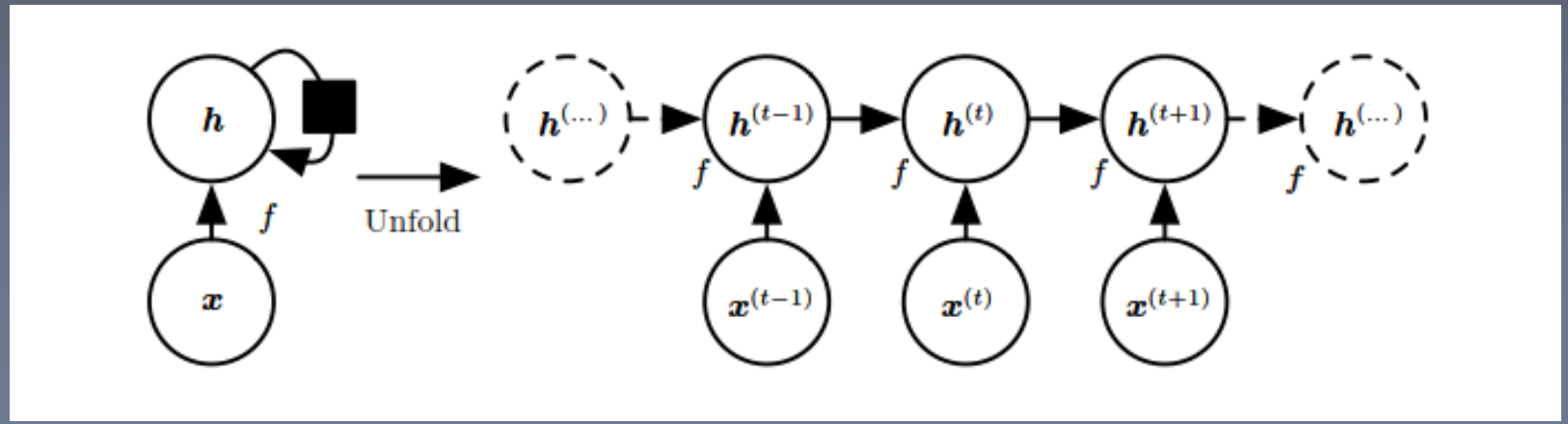
RNN基本表达方式-循环图

RNN presentation

循环图和展开图都有其用途
循环图简洁

展开图能够明确描述其中的计算流程

展开图还通过显式的信息流动路径帮助说明信息在时间上向前（计算输出和损失）和向后（计算梯度）的思想



(左) 循环图。黑色方块表示单个时间步的延迟。

(右) 展开计算图。其中每个节点现在与一个特定的时间实例相关联。

RNN基本训练方式

RNN Basic Training Mode

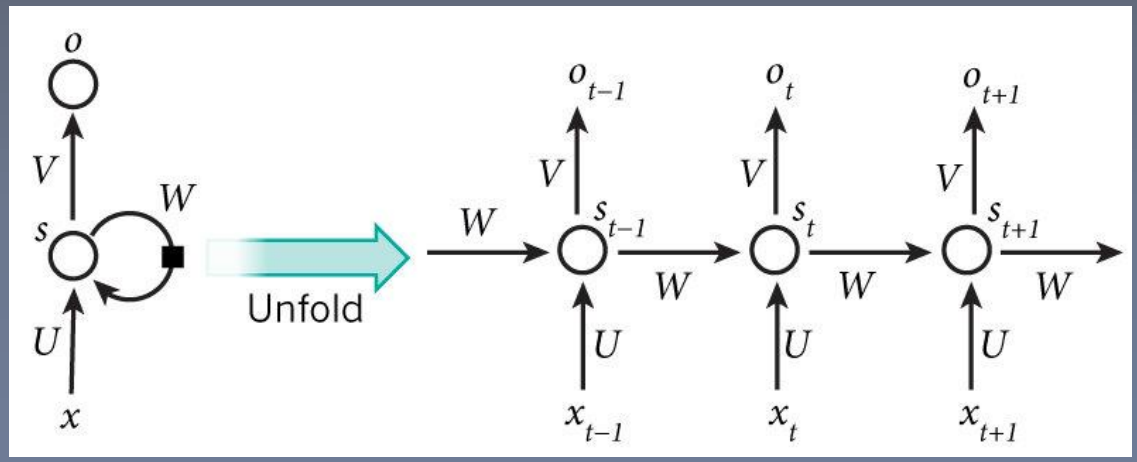
deepshare.net
深度之眼

1. 最简单的RNN
2. 前向传播
3. 反向延时传播(Bptt, Back Propagation Through Time)

最简单的RNN

Simple RNN

在左边循环图中，x是神经网络的输入，U是输入层到隐藏层之间的权重矩阵，W是记忆单元到隐藏层之间的权重矩阵，V是隐藏层到输出层之间的权重矩阵，S是隐藏层的输出，同时也是要保存到记忆单元中，并与下一时刻的一起作为输入，O是神经网络的输出。



从右边的展开计算图中可以更清楚的看到，RNN每个时刻隐藏层的输出都会传递给下一时刻，因此每个时刻的网络都会保留一定的来自之前时刻的历史信息，并结合当前时刻的网络状态一并再传给下一时刻。

最简单的RNN

Simple RNN

循环神经网络中一些重要的设计模式

1、每个时间步都有输出，并且隐藏单元之间有循环连接的循环网络

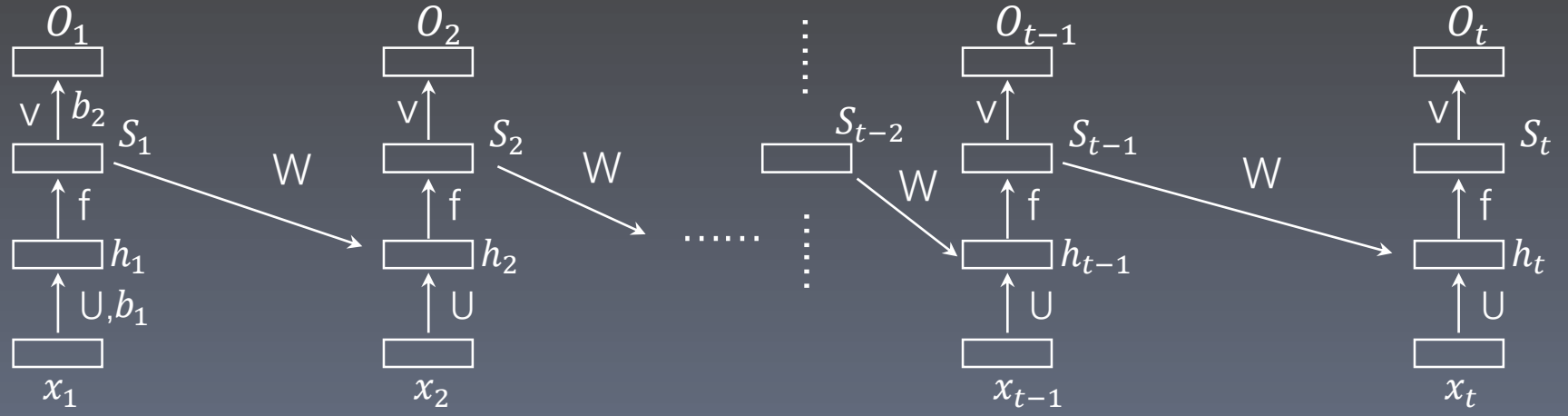
PS：这是最基础的循环神经网络，作为研究的基线

2、每个时间步都产生一个输出，只有当前时刻的输出到下个时刻的隐藏单元之间有循环连接的循环网络

PS：这是在基线的基础上改变了隐藏单元的链接方式

3、隐藏单元之间存在循环连接，但读取整个序列后产生单个输出的循环网络

PS：这是在基线的基础上改变输出方式



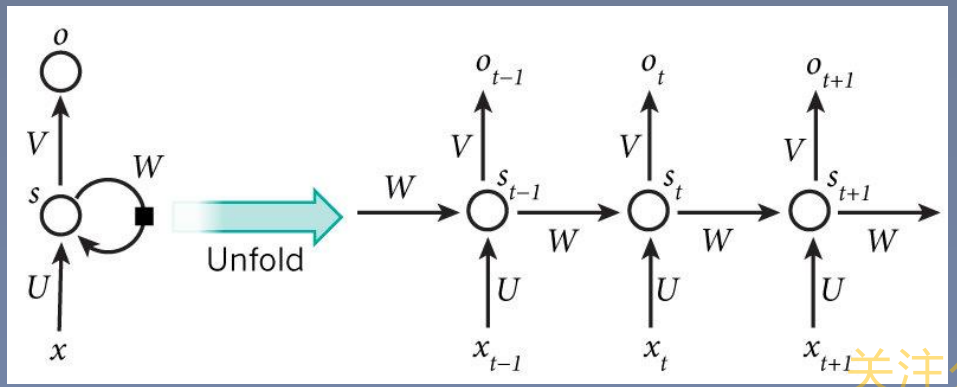
x_1, x_2,..... x_t可以想象成语音的输入特征：1000ms语音为”早上好”10ms取一个特征，共有100个特征(40维,t=100)输出(o1,o2,....ot)=(zao ,zao,...,shang,shang,.....,hao,hao)

$h_1 = x_1U + b_1$ | $h_2 = x_2U + S_1\mathrm{W} + b_1$ | $h_{t-1} = x_{t-1}U + S_{t-2}\mathrm{W} + b_1$ | $h_t = x_tU + S_{t-1}\mathrm{W} + b_1$

$S_1 = f(h_1)$ | $S_2 = f(h_2)$ | $S_{t-1} = f(h_{t-1})$ | $S_t = f(h_t)$

$O_1 = S_1\mathrm{V} + b_2$ | $O_2 = S_2\mathrm{V} + b_2$ | $O_{t-1} = S_{t-1}\mathrm{V} + b_2$ | $O_t = S_t\mathrm{V} + b_2$

其中，x_i ， S_i ， h_i ， O_i均为一维行向量



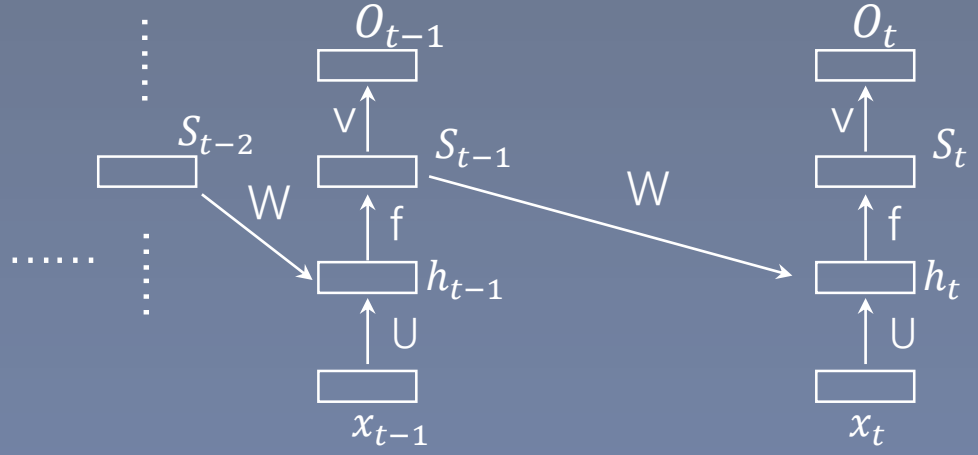
此时可以理解权值共享

$$J = \sum_{i=1}^{t} \|o_i - \tilde{o_i}\|^2 = J_1 + J_2 + \cdots + J_t \quad (J_i为MSE或CE损失)$$

$$\frac{\partial J}{\partial o_i} = \frac{\partial (J_1 + J_2 + \cdots + J_t)}{\partial o_i} = \frac{\partial J_i}{\partial o_i}$$

已知: $\frac{\partial J}{\partial o_1}, \frac{\partial J}{\partial o_2}, \cdots, \frac{\partial J}{\partial o_t}$

注意全连接的基本结构



$$\frac{\partial J}{\partial S_t} = \frac{\partial J}{\partial O_t} V^T$$

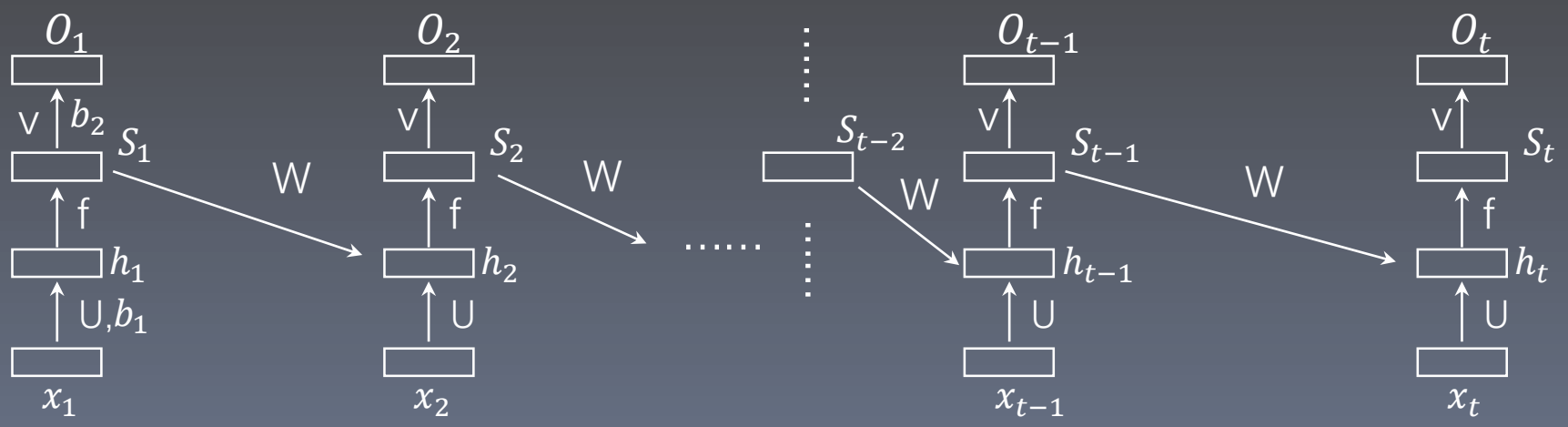
$$\frac{\partial J}{\partial h_t} = \frac{\partial J}{\partial S_t} \frac{dS_t}{dh_t}$$

$$\frac{\partial J}{\partial x_t} = \frac{\partial J}{\partial h_t} U^T$$

$$\frac{\partial J}{\partial S_{t-1}} = \frac{\partial J}{\partial O_{t-1}} V^T + \frac{\partial J}{\partial h_t} W^T$$

$$\frac{\partial J}{\partial h_{t-1}} = \frac{\partial J}{\partial S_{t-1}} \frac{dS_{t-1}}{dh_{t-1}}$$

$$\frac{\partial J}{\partial x_{t-1}} = \frac{\partial J}{\partial h_{t-1}} U^T$$



$$\frac{\partial J}{\partial S_{t-2}} = \frac{\partial J}{\partial O_{t-2}} V^T + \frac{\partial J}{\partial h_{t-1}} W^T$$

$$\frac{\partial J}{\partial h_{t-2}} = \frac{\partial J}{\partial S_{t-2}} \frac{dS_{t-2}}{dh_{t-2}}$$

$$\frac{\partial J}{\partial x_{t-2}} = \frac{\partial J}{\partial h_{t-2}} U^T$$

$$\frac{\partial J}{\partial S_2} = \frac{\partial J}{\partial O_2} V^T + \frac{\partial J}{\partial h_3} W^T$$

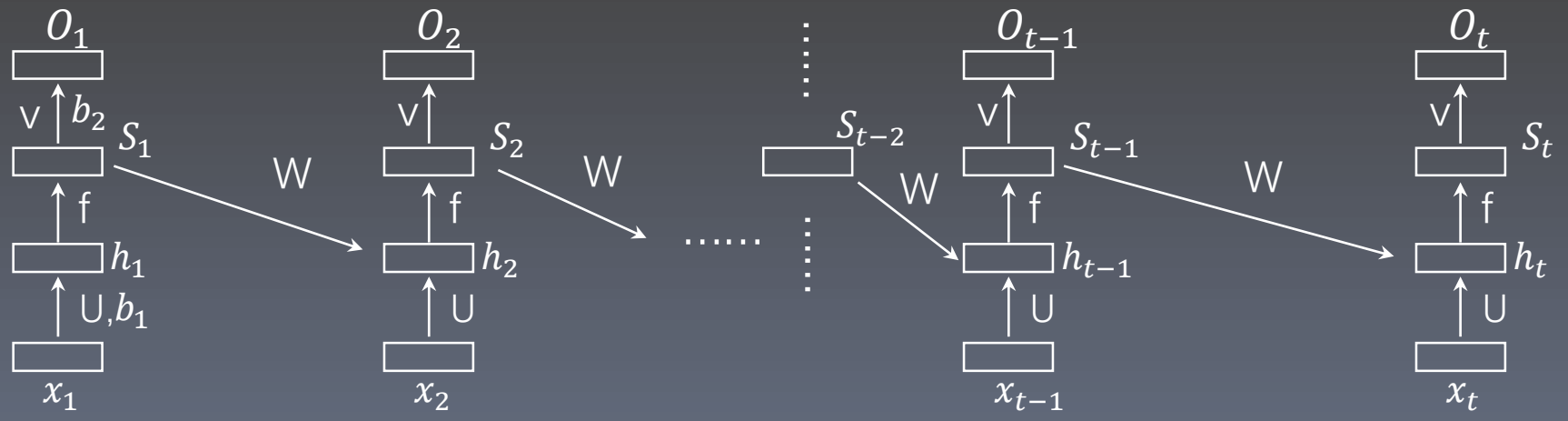
$$\frac{\partial J}{\partial h_2} = \frac{\partial J}{\partial S_2} \frac{dS_2}{dh_2}$$

$$\frac{\partial J}{\partial x_2} = \frac{\partial J}{\partial h_2} U^T$$

$$\frac{\partial J}{\partial S_1} = \frac{\partial J}{\partial O_1} V^T + \frac{\partial J}{\partial h_2} W^T$$

$$\frac{\partial J}{\partial h_1} = \frac{\partial J}{\partial S_1} \frac{dS_1}{dh_1}$$

$$\frac{\partial J}{\partial x_1} = \frac{\partial J}{\partial h_1} U^T$$



$$\frac{\partial J}{\partial V} = S_t^T \frac{\partial J}{\partial o_t}$$
$$\frac{\partial J}{\partial V} = S_{t-1}^T \frac{\partial J}{\partial o_{t-1}}$$
$$\vdots$$
$$\frac{\partial J}{\partial V} = S_1^T \frac{\partial J}{\partial o_1}$$
$$\Downarrow$$
$$\frac{\partial J}{\partial V} = \sum_{i=1}^{t} S_i^T \frac{\partial J}{\partial o_i}$$
$$= (S_1^T, S_2^T, \dots, S_t^T) \begin{pmatrix} \frac{\partial J}{\partial o_1} \\ \vdots \\ \frac{\partial J}{\partial o_t} \end{pmatrix}$$

$$\frac{\partial J}{\partial W} = S_{t-1}^T \frac{\partial J}{\partial h_t}$$
$$\frac{\partial J}{\partial W} = S_{t-2}^T \frac{\partial J}{\partial h_{t-1}}$$
$$\vdots$$
$$\frac{\partial J}{\partial W} = S_1^T \frac{\partial J}{\partial h_2}$$
$$\Downarrow$$
$$\frac{\partial J}{\partial W} = \sum_{i=1}^{t-1} S_i^T \frac{\partial J}{\partial h_{i+1}}$$
$$= (S_1^T, S_2^T, \dots, S_{t-1}^T) \begin{pmatrix} \frac{\partial J}{\partial h_2} \\ \vdots \\ \frac{\partial J}{\partial h_t} \end{pmatrix}$$

$$\frac{\partial J}{\partial U} = x_t^T \frac{\partial J}{\partial h_t}$$
$$\frac{\partial J}{\partial U} = x_{t-1}^T \frac{\partial J}{\partial h_{t-1}}$$
$$\vdots$$
$$\frac{\partial J}{\partial U} = x_1^T \frac{\partial J}{\partial h_1}$$
$$\Downarrow$$
$$\frac{\partial J}{\partial U} = \sum_{i=1}^{t} x_i^T \frac{\partial J}{\partial h_i}$$
$$= (x_1^T, x_2^T, \dots, x_{t-1}^T) \begin{pmatrix} \frac{\partial J}{\partial h_1} \\ \vdots \\ \frac{\partial J}{\partial h_t} \end{pmatrix}$$

都是向量和矩阵的乘法，没办法做到矩阵矩阵的相乘，没法做batch训练，无法发挥GPU的性能

思考：如何并行化训练，无法一句话并行，那就多句并行训练

设：

$$\begin{matrix} x_1^1 & x_2^1 & x_3^1 & \cdots & x_t^1 \\ x_1^2 & x_2^2 & x_3^2 & \cdots & x_t^2 \\ & & \vdots & & \\ x_1^N & x_2^N & x_3^N & \cdots & x_t^N \end{matrix}$$

为第一句话的t个样本
为第二句话的t个样本
⋮
为第N句话的t个样本

（以最长的序列为准，不够补O）

1:

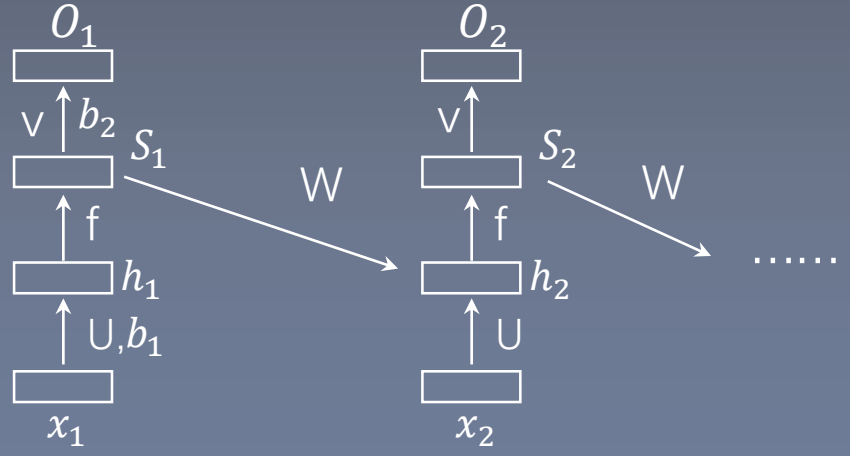
$$h_1^1 = x_1^1 U + b_1$$
$$S_1^1 = f(h_1^1)$$
$$O_1^1 = S_1^1 V + b_2$$

$$\begin{pmatrix} h_1^1 \\ \vdots \\ h_1^N \end{pmatrix} = \begin{pmatrix} x_1^1 \\ \vdots \\ x_1^N \end{pmatrix} U + \begin{pmatrix} b_1 \\ \vdots \\ b_1 \end{pmatrix}$$

2:

$$h_1^2 = x_1^2 U + b_1$$
$$s_1^2 = f(h_1^2)$$
$$O_1^2 = S_1^2 V + b_2$$

$$\begin{pmatrix} S_1^1 \\ \vdots \\ S_1^N \end{pmatrix} = f\begin{pmatrix} h_1^1 \\ \vdots \\ h_1^N \end{pmatrix}$$



…

N:

$$h_1^N = x_1^N U + b_1$$
$$s_1^N = f(h_1^N)$$
$$O_1^N = s_1^N V + b_2$$

$$\begin{pmatrix} O_1^1 \\ \vdots \\ O_1^N \end{pmatrix} = \begin{pmatrix} s_1^1 \\ \vdots \\ s_1^N \end{pmatrix} V + \begin{pmatrix} b_2 \\ \vdots \\ b_2 \end{pmatrix}$$

思考：如何并行化训练，无法一句话并行，那就多句并行训练

设：

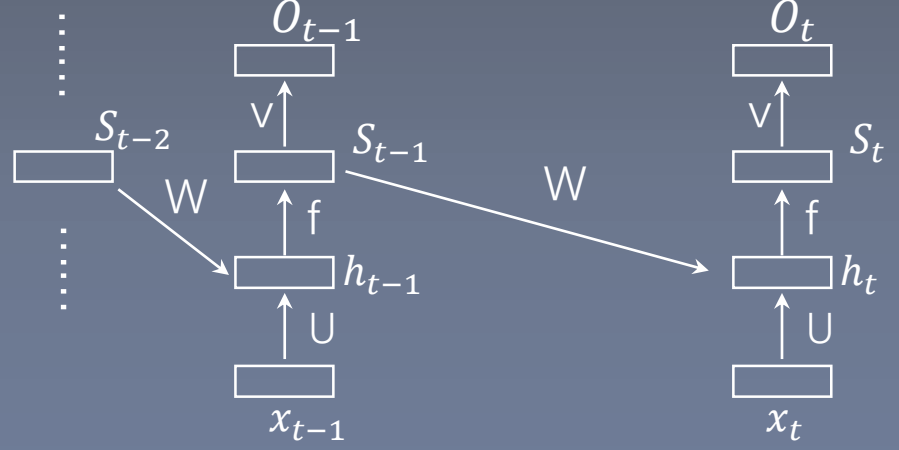
$$\begin{matrix} x_1^1 & x_2^1 & x_3^1 & \cdots & x_t^1 \\ x_1^2 & x_2^2 & x_3^2 & \cdots & x_t^2 \\ & & \vdots & & \\ x_1^N & x_2^N & x_3^N & \cdots & x_t^N \end{matrix}$$

为第一句话的t个样本
为第二句话的t个样本
⋮
为第N句话的t个样本

（以最长的序列为准，不够补O）

$$\mathrm{h}_{t-1}^1 = x_{t-1}^1 U + \mathrm{s}_{t-2}^1 w + b_1$$
$$\mathrm{s}_{t-1}^1 = f(\mathrm{h}_{t-1}^1)$$
$$\mathrm{o}_{t-1}^1 = \mathrm{s}_{t-1}^1 V + b_2$$

$$\begin{pmatrix} \mathrm{h}_{t-1}^1 \\ \vdots \\ \mathrm{h}_{t-1}^N \end{pmatrix} = \begin{pmatrix} x_{t-1}^1 \\ \vdots \\ x_{t-1}^N \end{pmatrix} U + \begin{pmatrix} \mathrm{s}_{t-2}^1 \\ \vdots \\ \mathrm{s}_{t-2}^N \end{pmatrix} w + \begin{pmatrix} b_1 \\ \vdots \\ b_1 \end{pmatrix}$$



$$\mathrm{h}_{t-1}^2 = x_{t-1}^2 U + \mathrm{s}_{t-2}^2 w + b_1$$
$$\mathrm{s}_{t-1}^2 = f(\mathrm{h}_{t-1}^2)$$
$$\mathrm{O}_{t-1}^2 = \mathrm{s}_{t-1}^2 V + b_2$$
$$\cdots$$

$$\begin{pmatrix} \mathrm{s}_{t-1}^1 \\ \vdots \\ \mathrm{s}_{t-1}^N \end{pmatrix} = f\begin{pmatrix} \mathrm{h}_{t-1}^1 \\ \vdots \\ \mathrm{h}_{t-1}^N \end{pmatrix}$$

$$\mathrm{h}_{t-1}^N = x_{t-1}^N U + \mathrm{s}_{t-2}^N w + b_1$$
$$\mathrm{s}_{t-1}^N = f(\mathrm{h}_{t-1}^N)$$
$$\mathrm{O}_{t-1}^N = \mathrm{s}_{t-1}^N V + b_2$$

$$\begin{pmatrix} \mathrm{O}_{t-1}^1 \\ \vdots \\ \mathrm{O}_{t-1}^N \end{pmatrix} = \begin{pmatrix} \mathrm{s}_{t-1}^1 \\ \vdots \\ \mathrm{s}_{t-1}^N \end{pmatrix} V + \begin{pmatrix} b_2 \\ \vdots \\ b_2 \end{pmatrix}$$

思考：如何并行化训练，无法一句话并行，那就多句并行训练

设：

$$\begin{matrix} x_1^1 & x_2^1 & x_3^1 & \cdots & x_t^1 \\ x_1^2 & x_2^2 & x_3^2 & \cdots & x_t^2 \\ & & \vdots & & \\ x_1^N & x_2^N & x_3^N & \cdots & x_t^N \end{matrix}$$

为第一句话的t个样本
为第二句话的t个样本
⋮
为第N句话的t个样本

（以最长的序列为准，不够补O）

1：

$$\frac{\partial J}{\partial S_{t-1}^1} = \frac{\partial J}{\partial O_{t-1}^1} V^T + \frac{\partial J}{\partial h_t^1} W^T$$

$$\frac{\partial J}{\partial h_{t-1}^1} = \frac{\partial J}{\partial S_{t-1}^1} \frac{\partial S_{t-1}^1}{\partial h_{t-1}^1}$$

$$\frac{\partial J}{\partial x_{t-1}^1} = \frac{\partial J}{\partial h_{t-1}^1} U^T$$

⋮

N：

$$\frac{\partial J}{\partial S_{t-1}^N} = \frac{\partial J}{\partial O_{t-1}^N} V^T + \frac{\partial J}{\partial h_t^N} W^T$$

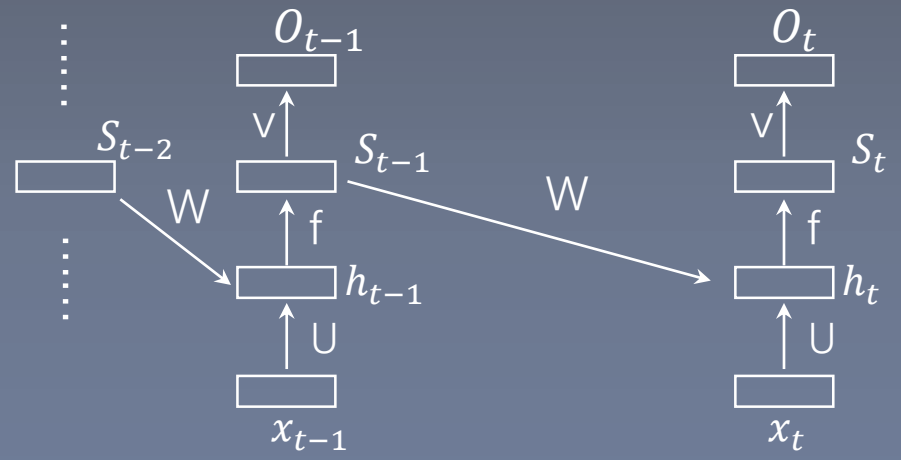
$$\frac{\partial J}{\partial h_{t-1}^N} = \frac{\partial J}{\partial S_{t-1}^N} \frac{\partial S_{t-1}^N}{\partial h_{t-1}^N}$$

$$\frac{\partial J}{\partial x_{t-1}^N} = \frac{\partial J}{\partial h_{t-1}^N} U^T$$

$$\begin{pmatrix} \frac{\partial J}{\partial S_{t-1}^1} \\ \vdots \\ \frac{\partial J}{\partial S_{t-1}^N} \end{pmatrix} = \begin{pmatrix} \frac{\partial J}{\partial o_{t-1}^1} \\ \vdots \\ \frac{\partial J}{\partial o_{t-1}^N} \end{pmatrix} V^T + \begin{pmatrix} \frac{\partial J}{\partial h_t^1} \\ \vdots \\ \frac{\partial J}{\partial h_t^N} \end{pmatrix} W^T$$

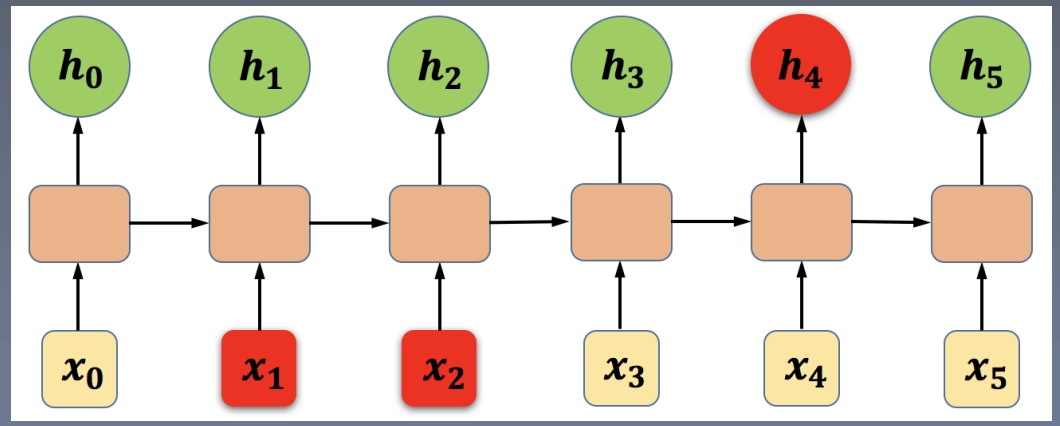
$$\begin{pmatrix} \frac{\partial J}{\partial h_{t-1}^1} \\ \vdots \\ \frac{\partial J}{\partial h_{t-1}^N} \end{pmatrix} = \begin{pmatrix} \frac{\partial J}{\partial S_{t-1}^1} \\ \vdots \\ \frac{\partial J}{\partial S_{t-1}^N} \end{pmatrix} \odot \begin{pmatrix} \frac{\partial S_{t-1}^1}{\partial h_{t-1}^1} \\ \vdots \\ \frac{\partial S_{t-1}^N}{\partial h_{t-1}^N} \end{pmatrix}$$

$$\begin{pmatrix} \frac{\partial J}{\partial x_{t-1}^1} \\ \vdots \\ \frac{\partial J}{\partial x_{t-1}^N} \end{pmatrix} = \begin{pmatrix} \frac{\partial J}{\partial h_{t-1}^1} \\ \vdots \\ \frac{\partial J}{\partial h_{t-1}^N} \end{pmatrix} U^T$$

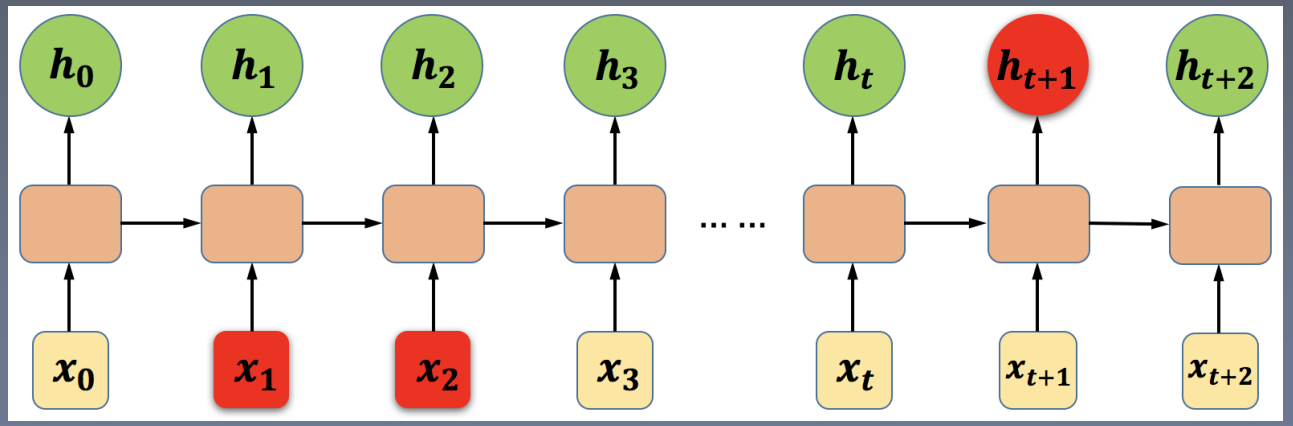


长期依赖问题

Long-term dependency problem



时间跨度较小的依赖关系示意图



时间跨度较大的依赖关系示意图

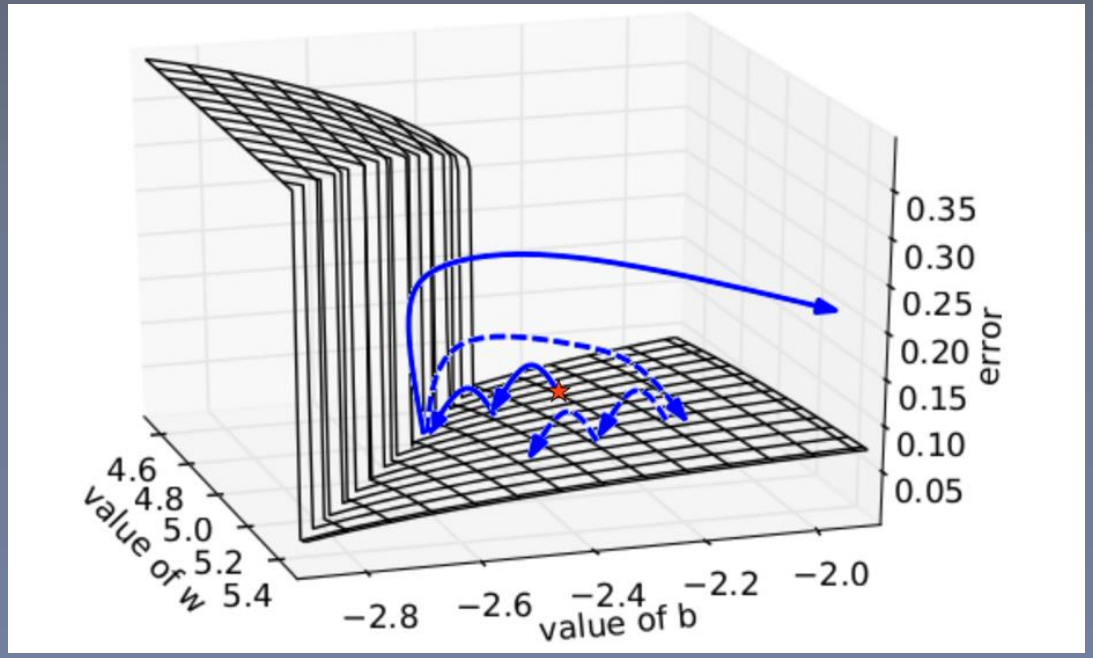
经过许多阶段传播后的梯度倾向于消失（大部分情况）或爆炸（很少，但对优化过程影响很大）。

长期依赖问题

Long-term dependency problem

根本问题：经过许多阶段传播后的梯度倾向于消失（大部分情况）或爆炸（很少，但对优化过程影响很大）

在深层的神经网络中，由于多个权重矩阵的相乘，会出现很多如图所示的陡峭区域，当然也有可能会出现很多非常平坦的区域。在这些陡峭的地方，Loss函数的倒数非常大，导致最终的梯度也很大，对参数进行更新后可能会导致参数的取值超出有效的取值范围，这种情况称之为梯度爆炸。



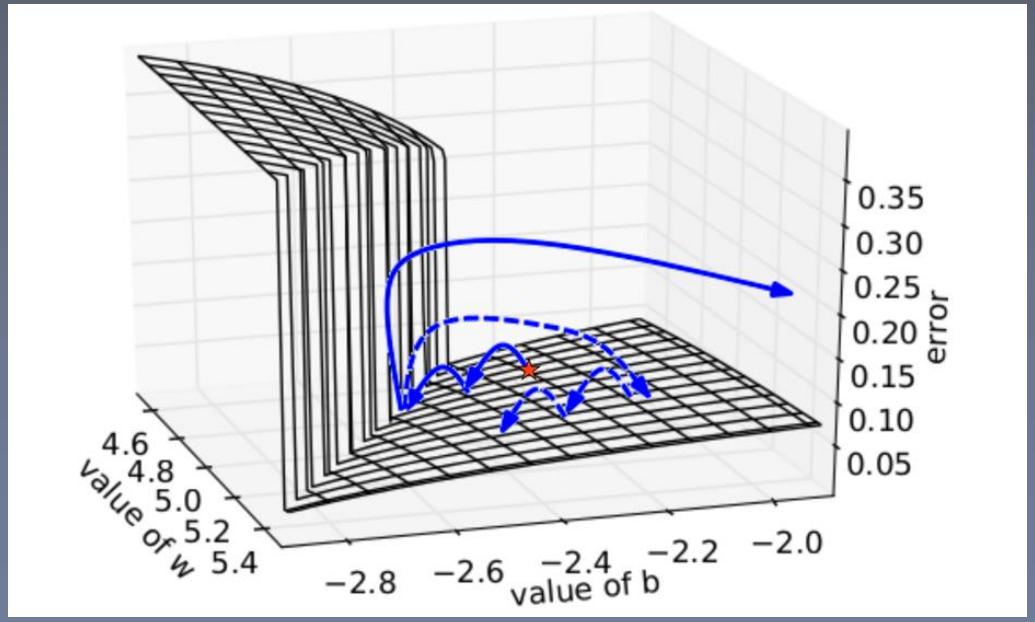
长期依赖问题

Long-term dependency problem

根本问题：经过许多阶段传播后的梯度倾向于消失（大部分情况）或爆炸（很少，但对优化过程影响很大）。

而在那些非常平坦的地方，Loss的变化很小，这个时候梯度的值也会很小（可能趋近于0），导致参数的更新非常缓慢，甚至更新的方向都不明确，这种情况称之为梯度消失。

长期依赖问题的存在会导致循环神经网络没有办法学习到时间跨度较长的依赖关系。



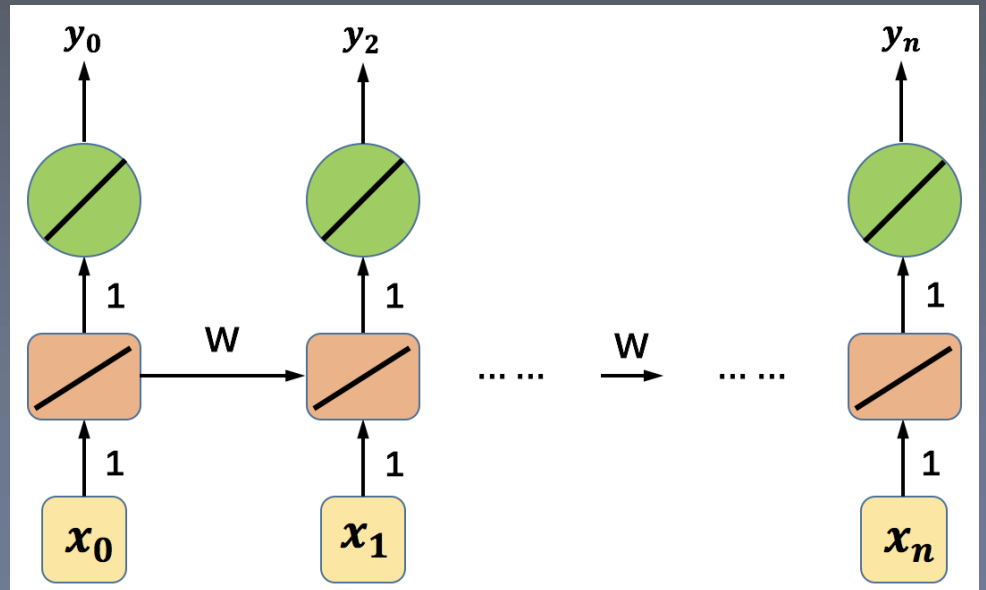
长期依赖问题

Long-term dependency problem

我们定义一个简化的循环神经网络，该网络中的所有激活函数均为线性的，除了在每个时间步上共享的参数W以外，其它的权重矩阵均设为1，偏置项均设为0。

假设输入的序列中除了的值为1以外，其它输入的值均为0。根据我们前面学的知识，我们最终可以得到，神经网络的输出是关于权重矩阵W的指数函数。

例子



$$h_0 = 1$$
$$h_1 = W$$
$$h_2 = W^2$$
$$\vdots$$
$$h_n = W^n$$

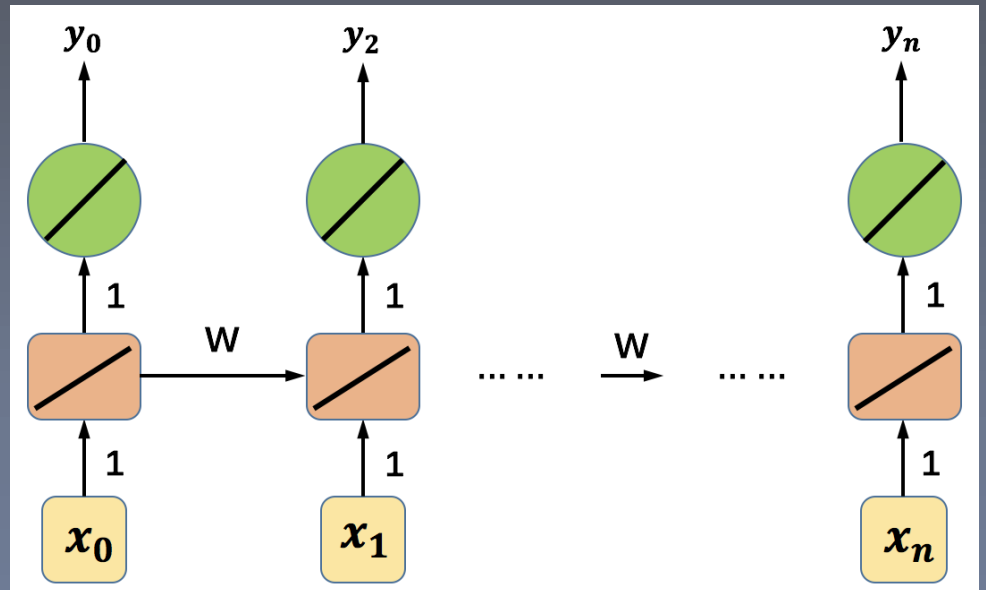
长期依赖问题

Long-term dependency problem

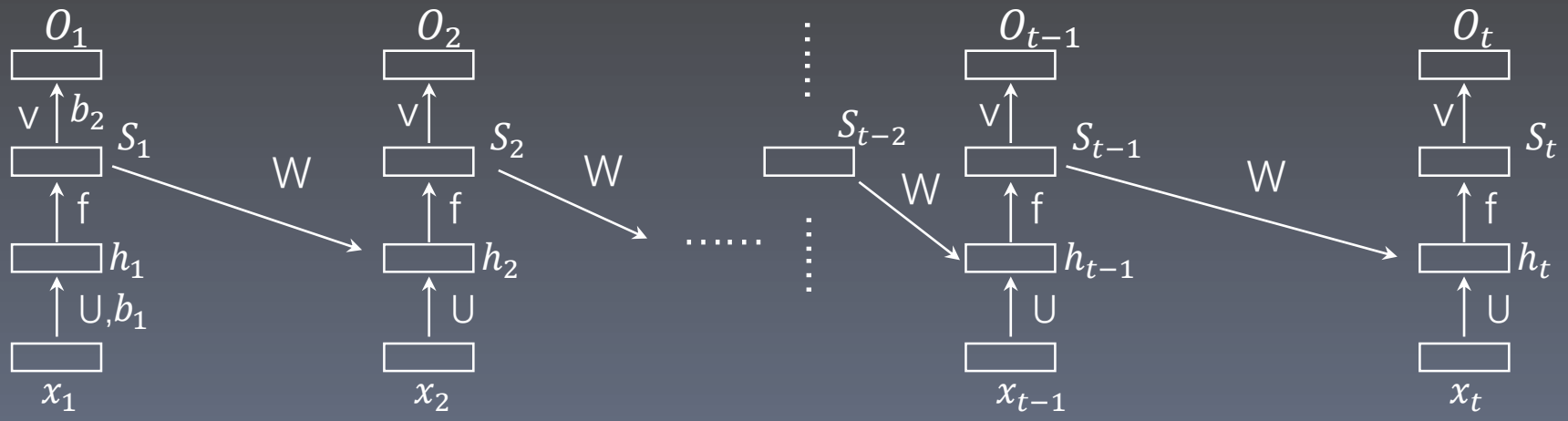
当W的值大于1时，随着n的增加，神经网络最终输出的值也成指数级增长，而当W的值小于1时，随着n的值增加，神经网络最终的输出则会非常小。这两种结果分别是导致梯度爆炸和梯度消失的根本原因。

从例子可以看到，循环神经网络中梯度消失和梯度爆炸问题产生的根本原因，是由于参数共享导致的。

例子



$$h_0 = 1$$
$$h_1 = W$$
$$h_2 = W^2$$
$$\vdots$$
$$h_n = W^n$$



$$\frac{\partial J}{\partial S_{t-1}} = \frac{\partial J}{\partial O_{t-1}} V^T + \frac{\partial J}{\partial h_t} W^T \quad \Rightarrow \quad \frac{\partial J}{\partial S_{t-1}} = \frac{\partial J}{\partial O_{t-1}} V^T + \frac{\partial J}{\partial S_t} \frac{dS_t}{dh_t} W^T \approx f(t) \frac{\partial J}{\partial S_t}$$

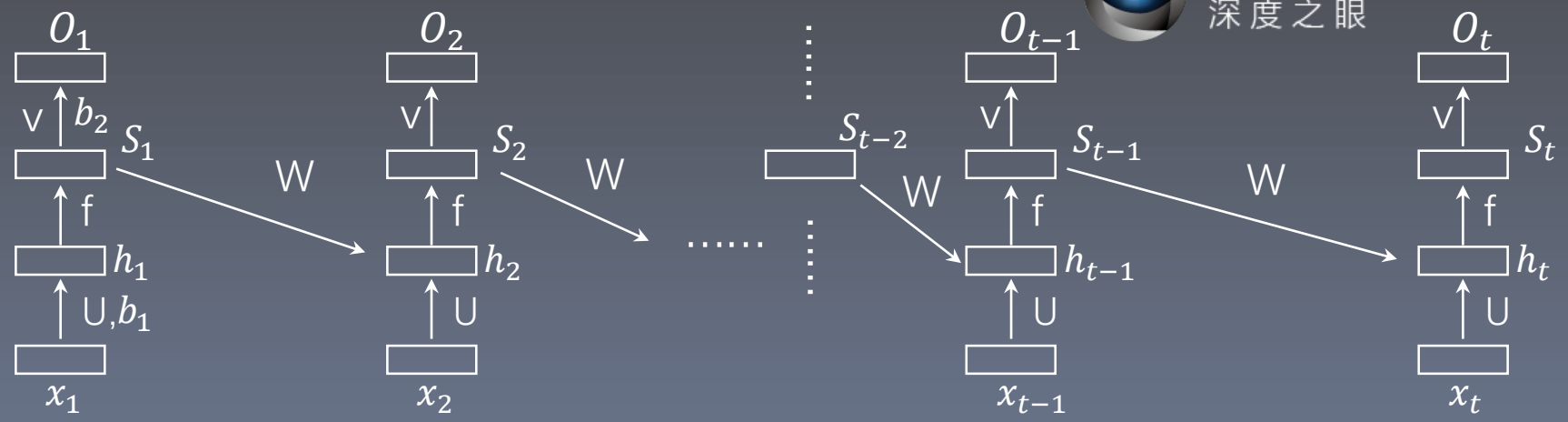
$$\frac{\partial J}{\partial h_t} = \frac{\partial J}{\partial S_t} \frac{dS_t}{dh_t}$$

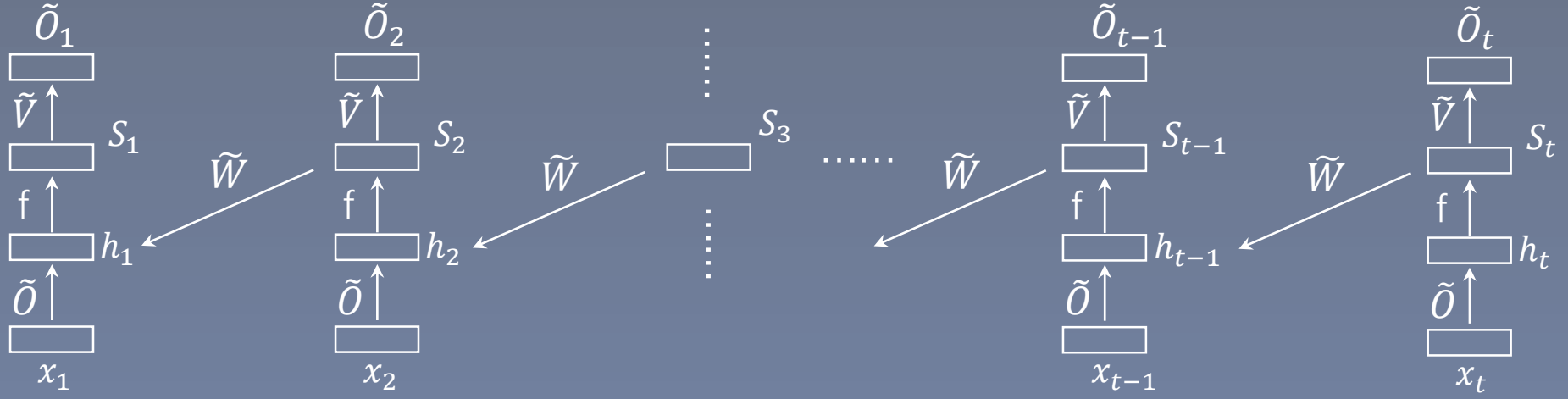
$$\frac{\partial J}{\partial S_{t-k}} = f_1(t) f_2(t) \cdots f_k(t) \frac{\partial J}{\partial S_t}$$

f1*f2……fk与W的k次方相关，因此会出现梯度消失或者梯度爆炸，一种简单的方式是bptt时候每隔T清除下一时刻传来的梯度

$$\frac{\partial J}{\partial S_{t-1}} = \frac{\partial J}{\partial O_{t-1}} V^T$$

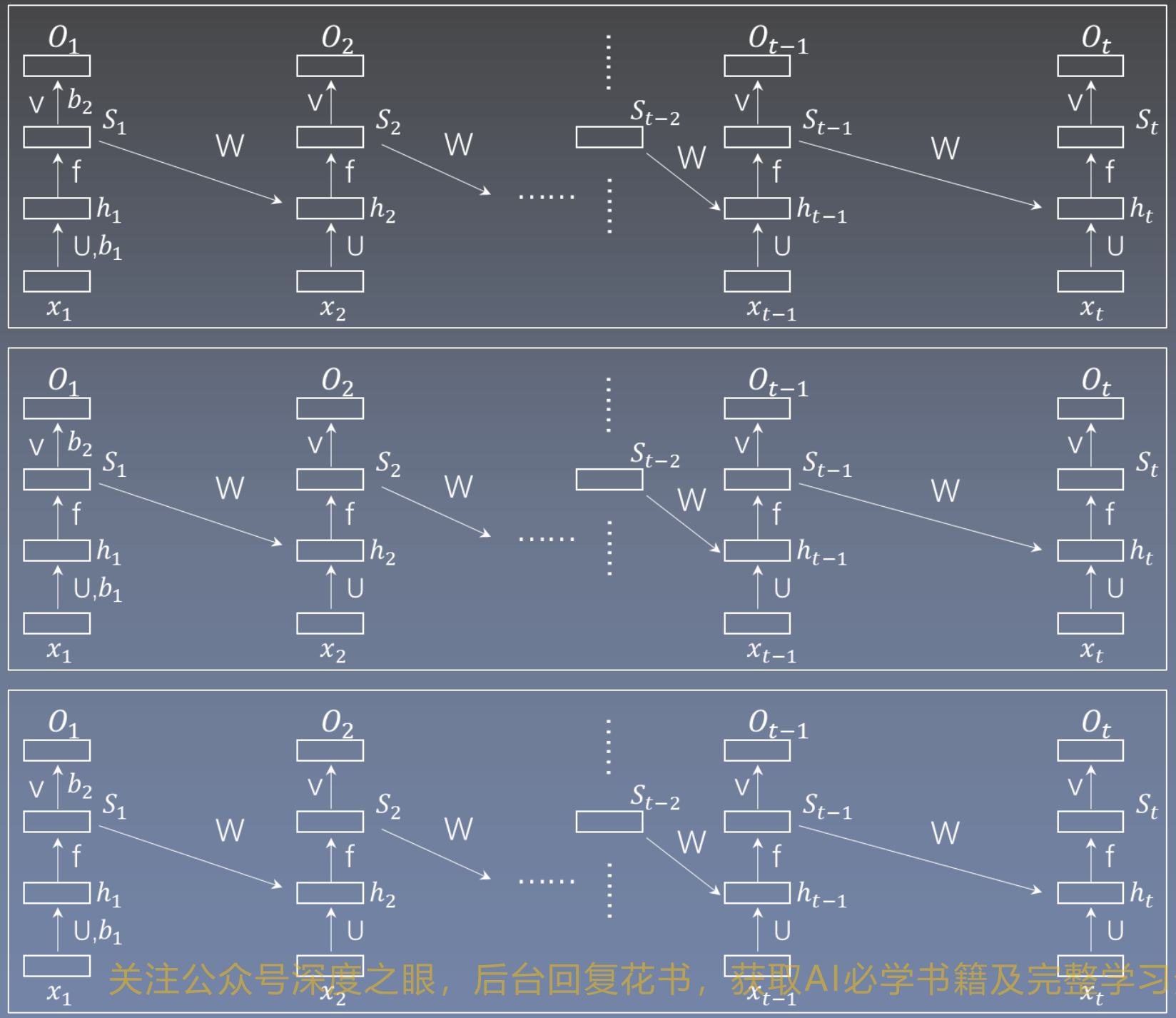
双向RNN





$$O_{\text{total}}^{t} = [O_t, \tilde{O}_t]$$

Deep RNN



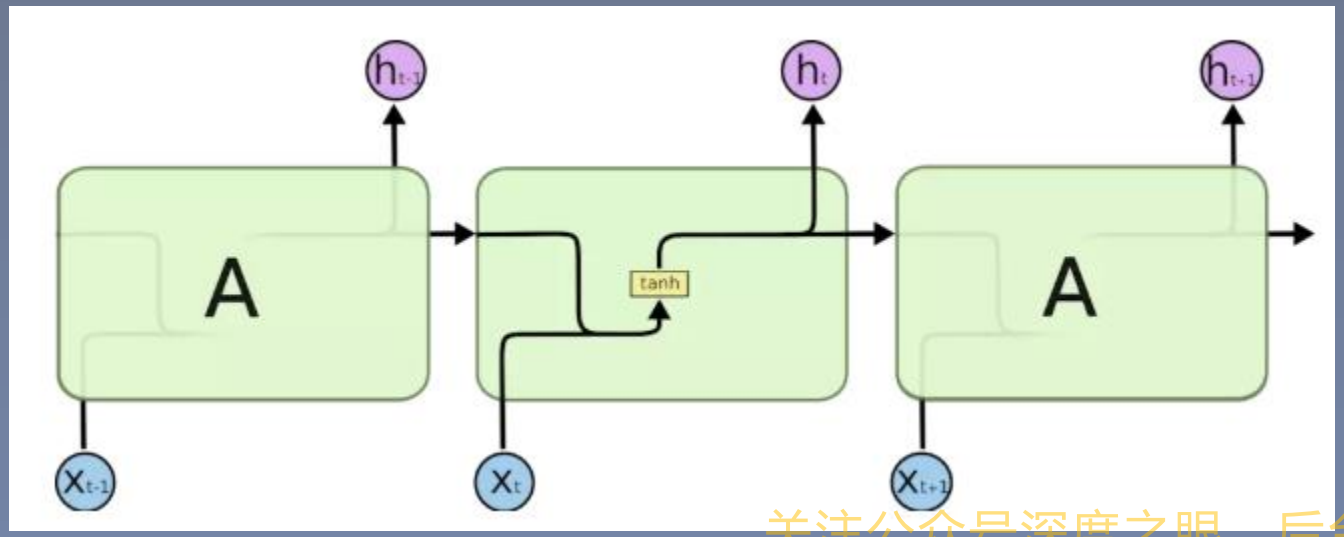
长短期记忆网络LSTM

Long short-term memory

LSTM 通过刻意的设计来避免长期依赖问题。

记住长期的信息在实践中是 LSTM 的默认行为，而非需要付出很大代价才能获得的能力！

所有 RNN 都具有一种重复神经网络模块的链式的形式。在标准的 RNN 中，这个重复的模块只有一个非常简单的结构，例如一个 tanh 层。

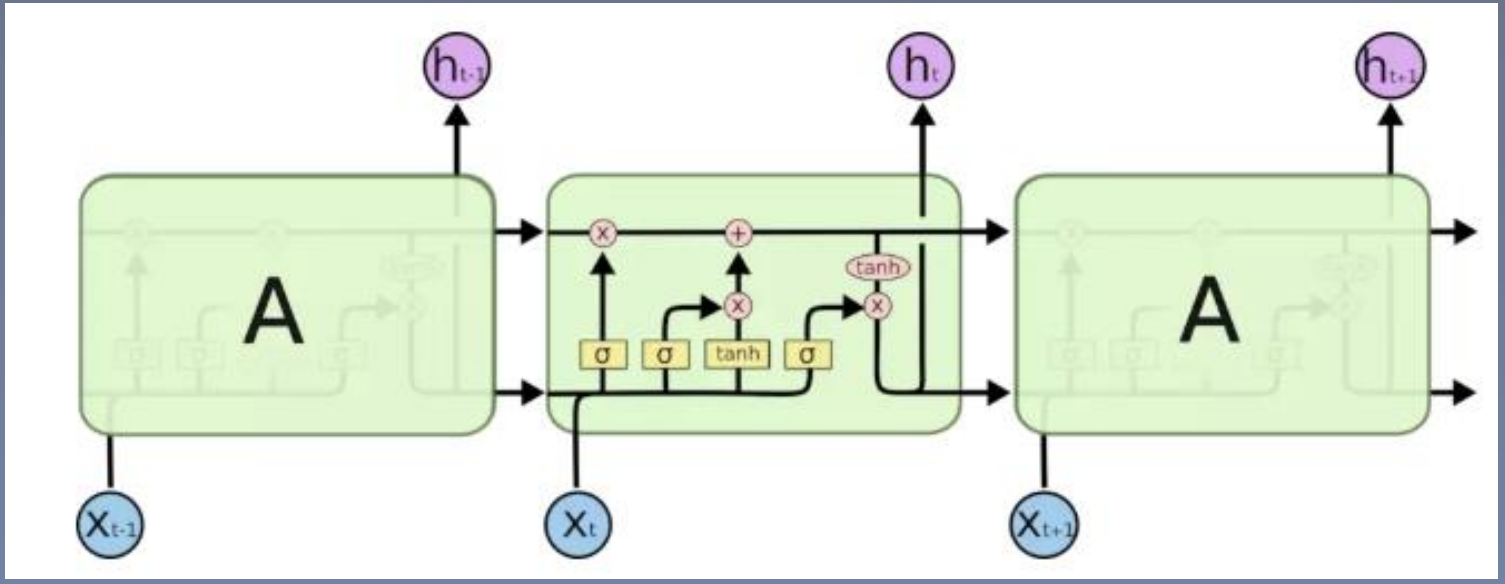


长短期记忆网络LSTM

Long short-term memory

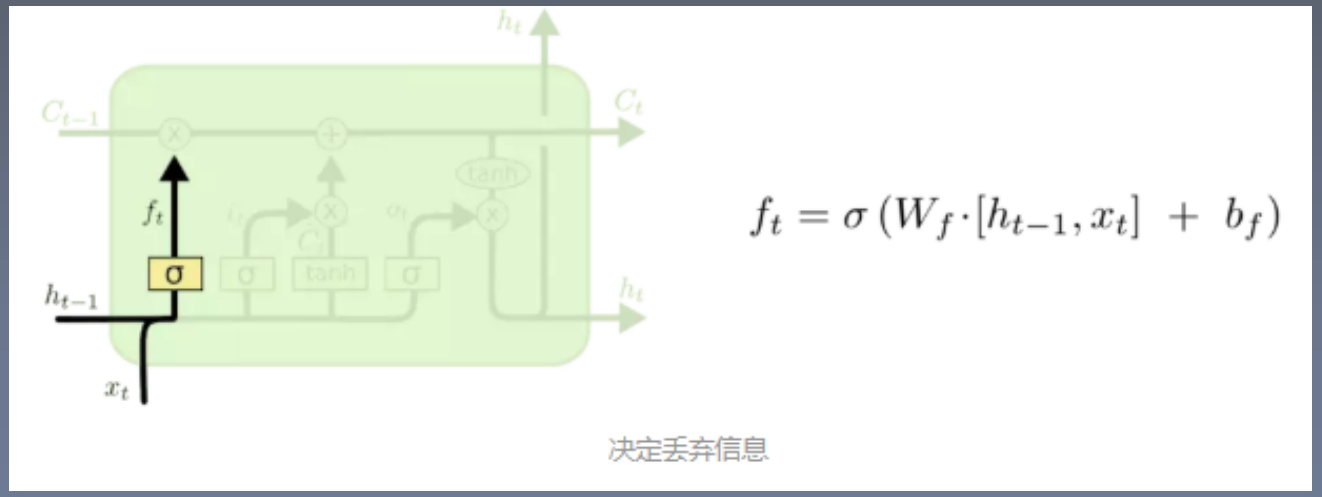
LSTM 同样是这样的结构，但是重复的模块拥有一个不同的结构。

不同于 单一神经网络层，这里是有四个，以一种非常特殊的方式进行交互。



长短期记忆网络LSTM

Long short-term memory



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

决定丢弃信息

在我们 LSTM 中的第一步是决定我们会从细胞状态中丢弃什么信息。

这个决定通过一个称为忘记门层完成。

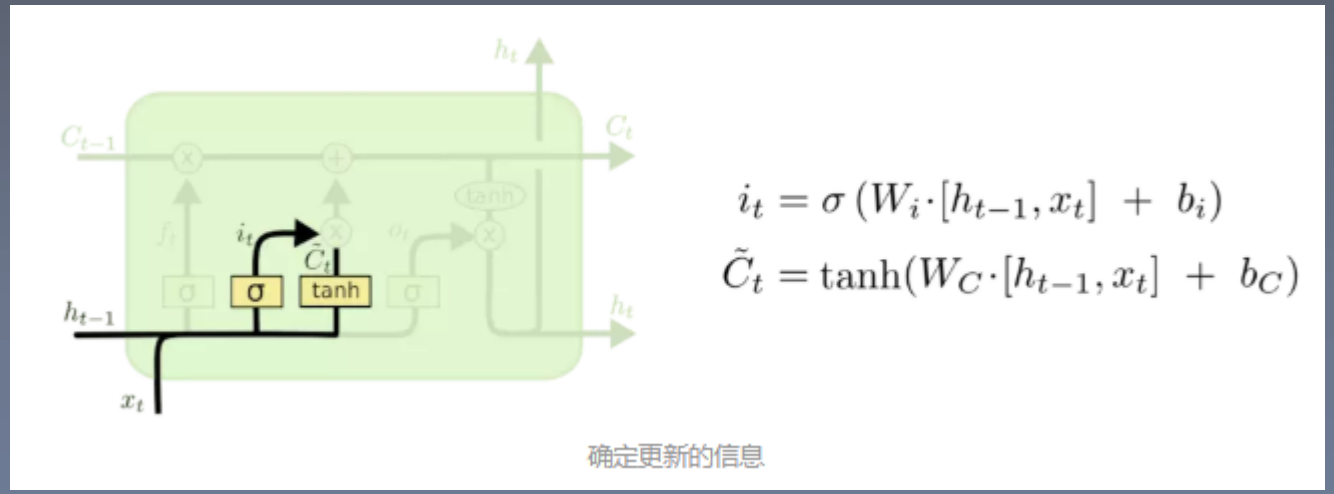
该门会读取 h_{t-1} 和 x_t，输出一个在 0 到 1 之间的数值给每个在细胞状态 C_{t-1} 中的数字。

1 表示“完全保留”，0 表示“完全舍弃”。

逐步理解 LSTM

长短期记忆网络LSTM

Long short-term memory



确定更新的信息

下一步是确定什么样的新信息被存放在细胞状态中。这里包含两个部分。

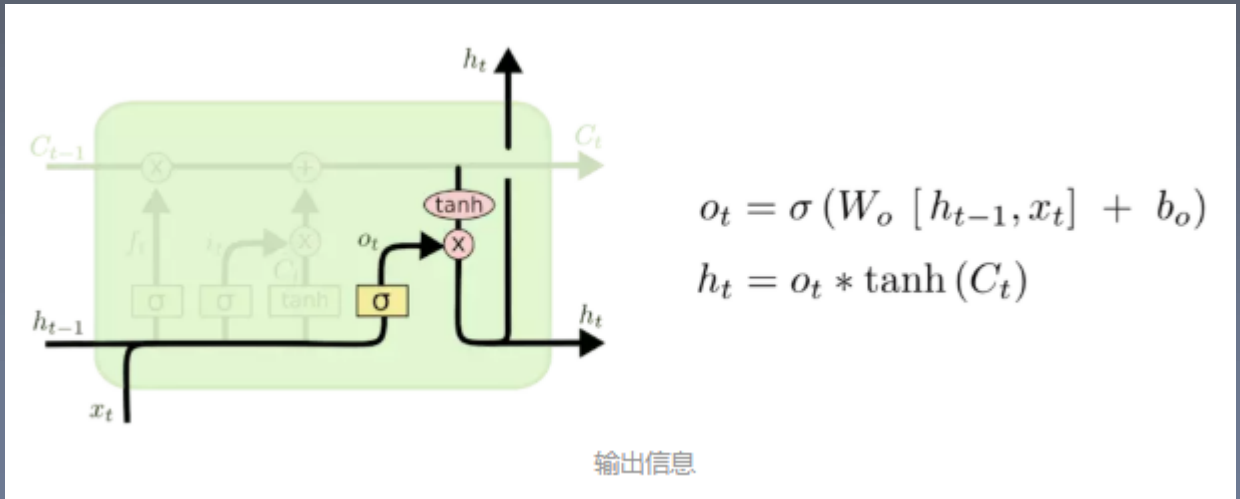
第一，sigmoid 层称 “输入门层” 决定什么值我们将要更新。然后，一个 tanh 层创建一个新的候选值向量，\tilde{C}_t，会被加入到状态中。

下一步，我们会讲这两个信息来产生对状态的更新。

逐步理解 LSTM

长短期记忆网络LSTM

Long short-term memory



逐步理解 LSTM

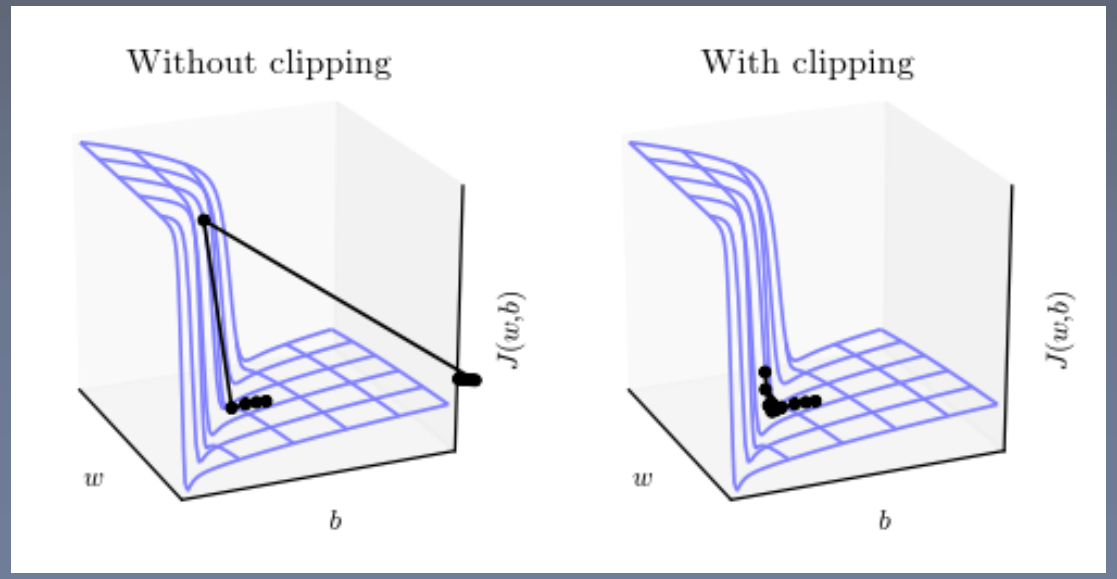
最终，我们需要确定输出什么值。

这个输出将会基于我们的细胞状态，但是也是一个过滤后的版本。

首先，我们运行一个 sigmoid 层来确定细胞状态的哪个部分将输出出去。

接着，我们把细胞状态通过 tanh 进行处理（得到一个在 到 之间的值）并将它和 sigmoid 门的输出相乘，最终我们仅仅会输出我们确定输出的那部分。

截断梯度



高度非线性函数导致梯度非常大或非常小，这样导致参数更新时步伐较大，可能进入目标函数较大区域。

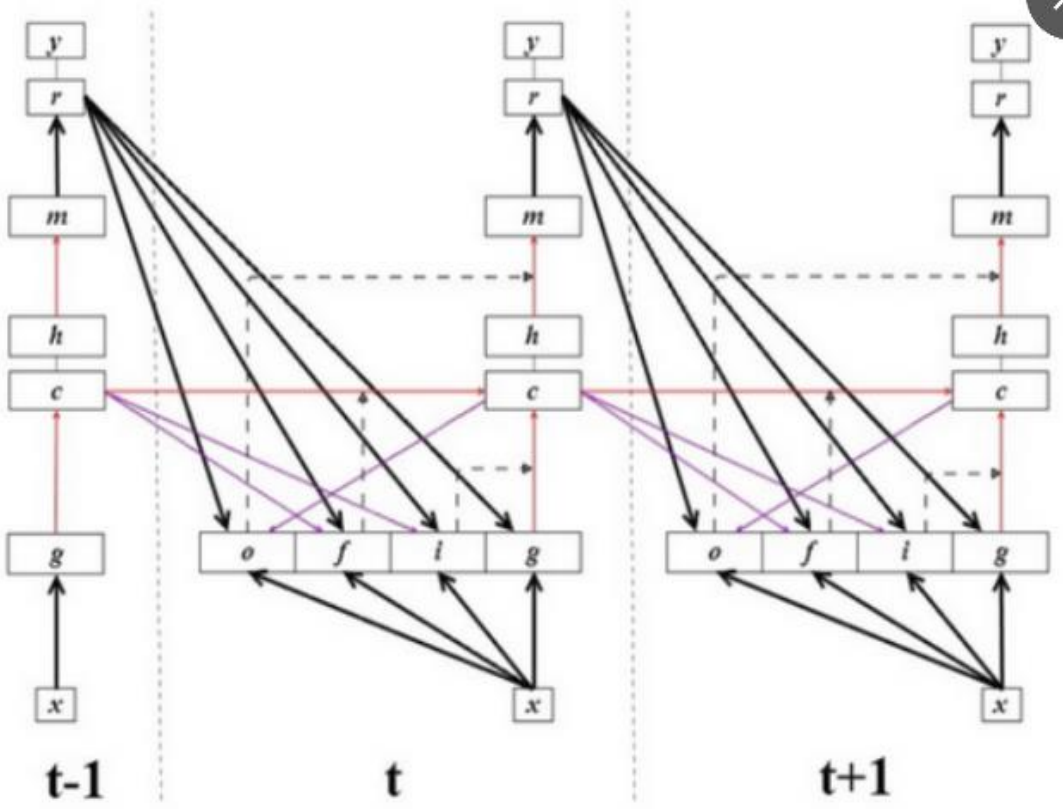
1）如左图，没有截断时，参数直接被推离到小峡谷之外。

2）当使用梯度截断后，更新步长受到限制，对于这种大梯度的反应较为温和。

梯度截断的方法：

一种选择是在参数更新之前，逐元素地截断小批量产生的参数)。

另一种是在参数更新之前截断梯度 g 的范数 $\|g\|$



For traditional LSTM:

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1} + W_{ic}c_{t-1} + b_i) \quad (1)$$
$$f_t = \sigma(W_{fx}x_t + W_{fm}m_{t-1} + W_{fc}c_{t-1} + b_f) \quad (2)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g(W_{cx}x_t + W_{cm}m_{t-1} + b_c) \quad (3)$$
$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1} + W_{oc}c_t + b_o) \quad (4)$$
$$m_t = o_t \odot h(c_t) \quad (5)$$
$$y_t = \phi(W_{ym}m_t + b_y) \quad (6)$$

For recurrent projected layer, substitute (6) with (7) and (8):

$$r_t = W_{rm}m_t \quad (7)$$
$$y_t = \phi(W_{yr}r_t + b_y) \quad (8)$$

$$\frac{\partial J}{\partial m_t} = \frac{\partial J}{\partial y_t}W_{ym}^T + \frac{\partial J}{\partial g_{t+1}}W_{cm}^T + \frac{\partial J}{\partial i_{t+1}}W_{im}^T + \frac{\partial J}{\partial f_{t+1}}W_{fm}^T + \frac{\partial J}{\partial O_{t+1}}W_{om}^T$$

$$\frac{\partial J}{\partial h_t} = \frac{\partial J}{\partial m_t} \odot O_t$$

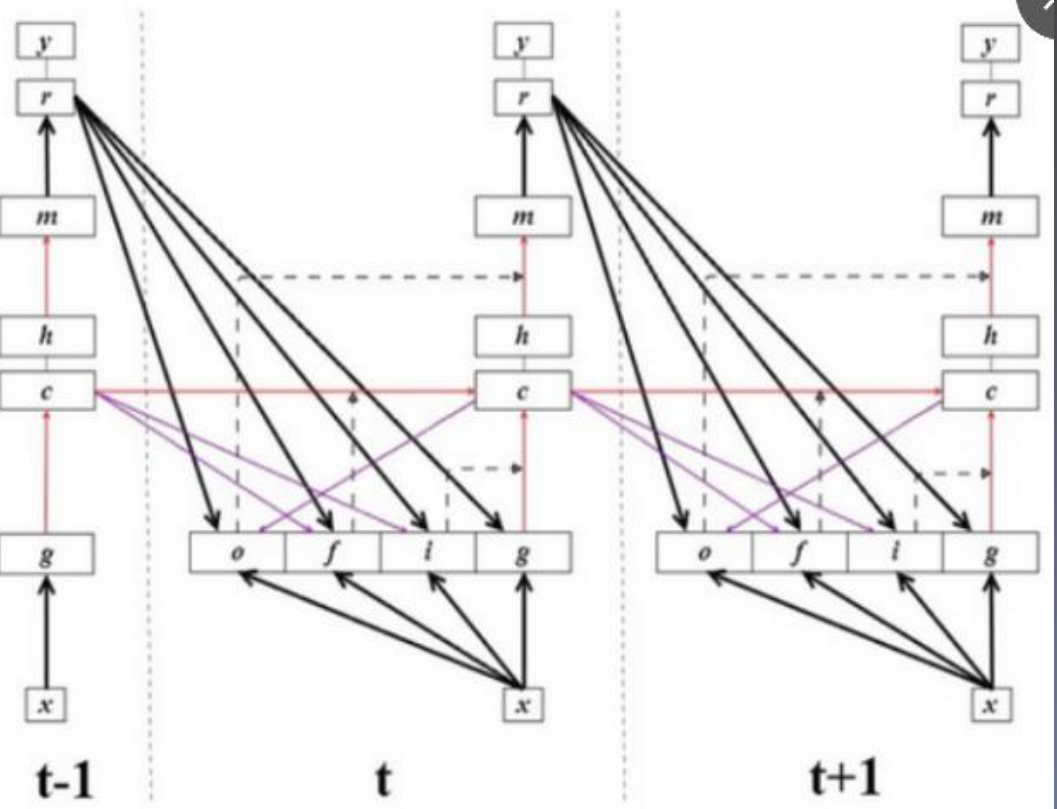
$$\frac{\partial J}{\partial C_t} = \frac{\partial J}{\partial h_t}\frac{dh_t}{dc_t} + \frac{\partial J}{\partial c_{t+1}} \odot f_{t+1} + \frac{\partial J}{\partial i_{t+1}}W_{ic}^T + \frac{\partial J}{\partial f_{t+1}}W_{fc}^T + + \frac{\partial J}{\partial O_t}W_{oc}^T$$

$$\left.\begin{aligned}
\frac{\partial J}{\partial g_t} &= \frac{\partial J}{\partial c_t} \odot i_t \\
\frac{\partial J}{\partial i_t} &= \frac{\partial J}{\partial c_t} \odot g_t \\
\frac{\partial J}{\partial f_t} &= \frac{\partial J}{\partial c_t} \odot c_{t-1} \\
\frac{\partial J}{\partial O_t} &= \frac{\partial J}{\partial m_t} \odot h_t
\end{aligned}\right\}$$

还有sigmoid或 tanh 非 线性变换，如

$$\begin{cases}
\frac{\partial J}{\partial \tilde{i}_t} = \frac{\partial J}{\partial i_t}\frac{di_t}{d\tilde{i}_t} = \frac{\partial J}{\partial i_t}i_t(1-i_t), \tilde{i}_t \text{为sigmoid之前} \\
\frac{\partial J}{\partial \tilde{g}_t} = \frac{\partial J}{\partial g_t}\frac{dg_t}{d\tilde{g}_t} = \frac{\partial J}{\partial g_t}(1-{g_t}^2), \tilde{g}_t \text{为tanh之前}
\end{cases}$$

$$\frac{\partial J}{\partial x_t} = \frac{\partial J}{\partial g_t}W_{cx}^T + \frac{\partial J}{\partial i_t}W_{ix}^T + \frac{\partial J}{\partial f_t}W_{fx}^T + \frac{\partial J}{\partial O_t}W_{ox}^T$$



For traditional LSTM:

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1} + W_{ic}c_{t-1} + b_i) \quad (1)$$
$$f_t = \sigma(W_{fx}x_t + W_{fm}m_{t-1} + W_{fc}c_{t-1} + b_f) \quad (2)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g(W_{cx}x_t + W_{cm}m_{t-1} + b_c) \quad (3)$$
$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1} + W_{oc}c_t + b_o) \quad (4)$$
$$m_t = o_t \odot h(c_t) \quad (5)$$
$$y_t = \phi(W_{ym}m_t + b_y) \quad (6)$$

For recurrent projected layer, substitute (6) with (7) and (8):

$$r_t = W_{rm}m_t \quad (7)$$
$$y_t = \phi(W_{yr}r_t + b_y) \quad (8)$$

$$\frac{\partial J}{\partial W_{ym}} = m_t^T \frac{\partial J}{\partial y_t}$$
$$\frac{\partial J}{\partial W_{cm}} = m_t^T \frac{\partial J}{\partial g_{t+1}}$$
$$\frac{\partial J}{\partial W_{im}} = m_t^T \frac{\partial J}{\partial i_{t+1}}$$
$$\frac{\partial J}{\partial W_{fm}} = m_t^T \frac{\partial J}{\partial f_{t+1}}$$
$$\frac{\partial J}{\partial W_{om}} = m_t^T \frac{\partial J}{\partial o_{t+1}}$$

$$\frac{\partial J}{\partial W_{cx}} = x_t^T \frac{\partial J}{\partial g_t}$$
$$\frac{\partial J}{\partial W_{ix}} = x_t^T \frac{\partial J}{\partial i_t}$$
$$\frac{\partial J}{\partial W_{fx}} = x_t^T \frac{\partial J}{\partial f_t}$$
$$\frac{\partial J}{\partial W_{ox}} = x_t^T \frac{\partial J}{\partial o_t}$$

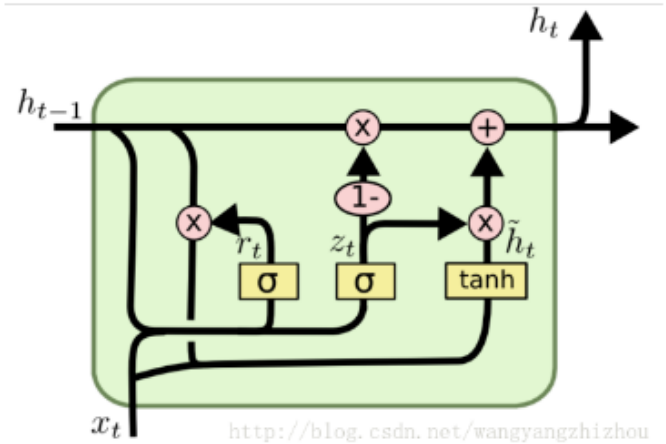
$$\frac{\partial J}{\partial W_{ic}} = c_t^T \frac{\partial J}{\partial i_{t+1}}$$
$$\frac{\partial J}{\partial W_{fc}} = c_t^T \frac{\partial J}{\partial f_{t+1}}$$
$$\frac{\partial J}{\partial W_{oc}} = c_t^T \frac{\partial J}{\partial o_t}$$

GRU

1、GRU概述

GRU是LSTM网络的一种效果很好的变体，它较LSTM网络的结构更加简单，而且效果也很好，因此也是当前非常流形的一种网络。GRU既然是LSTM的变体，因此也是可以解决RNN网络中的长依赖问题。

在LSTM中引入了三个门函数：输入门、遗忘门和输出门来控制输入值、记忆值和输出值。而在GRU模型中只有两个门：分别是更新门和重置门。具体结构如下图所示：



图中的z_t和r_t分别表示更新门和重置门。更新门用于控制前一时刻的状态信息被带入到当前状态中的程度，更新门的值越大说明前一时刻的状态信息带入越多。重置门控制前一状态有多少信息被写入到当前的候选集 $\tilde{h}_t$ 上，重置门越小，前一状态的信息被写入的越少。

2、GRU前向传播

根据上面的GRU的模型图，我们来看看网络的前向传播公式：

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

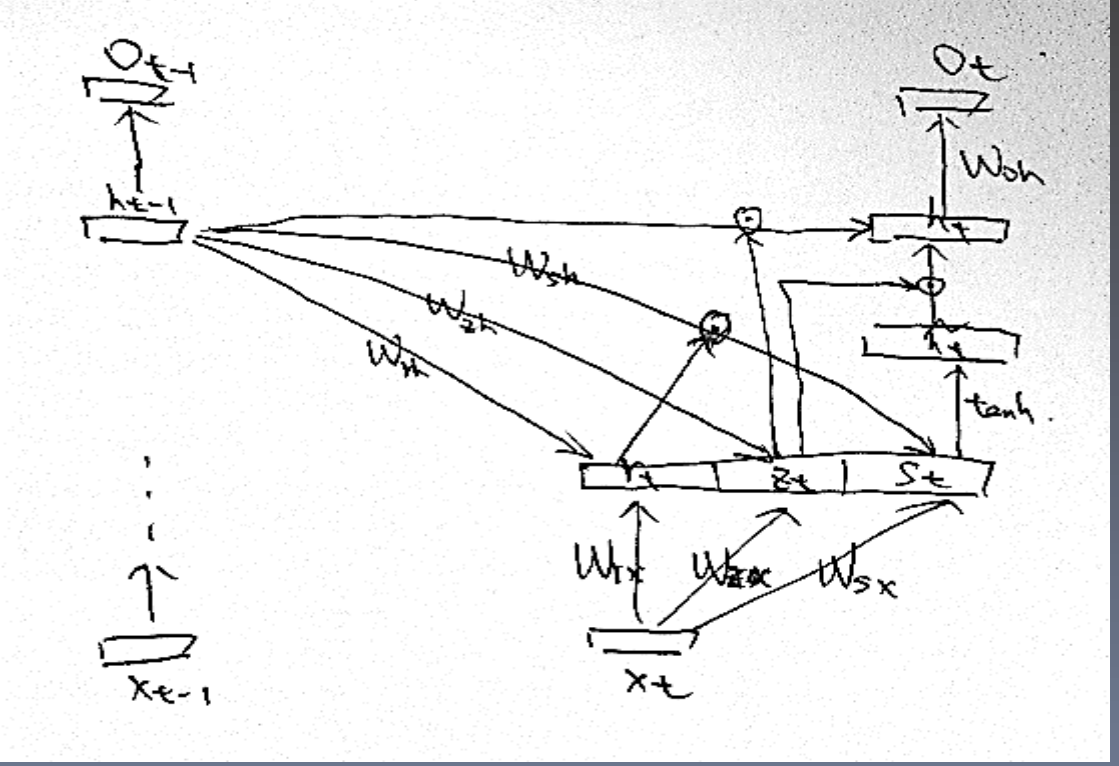
$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh(W_{\tilde{h}} \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

$$y_t = \sigma(W_o \cdot h_t)$$

其中[]表示两个向量相连，*表示矩阵的乘积。



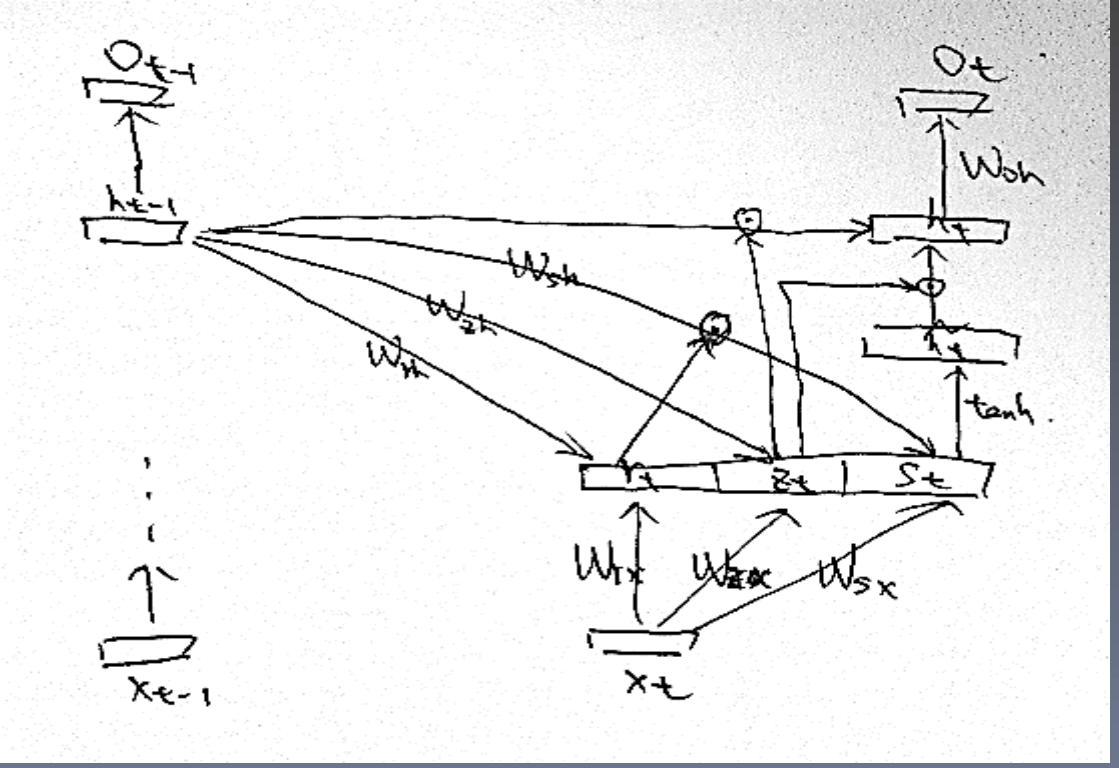
$$r_t = x_t W_{rx} + h_{t-1} W_{rh} + b_r$$
$$z_t = x_t W_{zx} + h_{t-1} W_{zh} + b_z$$
$$r_t = sigmoid(r_t), z_t = sigmoid(z_t)$$
$$s_t = x_t W_{sx} + (r_t \odot h_{t-1}) W_{sh} + b_s$$
$$\tilde{h}_t = \tanh(s_t)$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$
$$o_t = h_t W_{oh} + b_o$$

$$\frac{\partial J}{\partial h_t} = \frac{\partial J}{\partial O_t} W_{oh}^T + \frac{\partial J}{\partial r_{t+1}} W_{rh}^T + \frac{\partial J}{\partial z_{t+1}} W_{zh}^T + (\frac{\partial J}{\partial s_{t+1}} \odot r_{t+1}) W_{sh}^T + \frac{\partial J}{\partial h_{t+1}} \odot (1 - z_{t+1})$$

$$\frac{\partial J}{\partial \tilde{h}_t} = \frac{\partial J}{\partial h_t} \odot z_t$$

$$\frac{\partial J}{\partial s_t} = \frac{\partial J}{\partial \tilde{h}_t} \frac{d\tilde{h}}{ds_t} = \frac{\partial J}{\partial \tilde{h}_t} (1 - \tilde{h}_t^{\,2}), \frac{\partial J}{\partial x_t} = \frac{\partial J}{\partial r_t} W_{rx}^T + \frac{\partial J}{\partial z_t} W_{zx}^T + \frac{\partial J}{\partial s_t} W_{sx}^T$$

$$\left.\begin{aligned} \frac{\partial J}{\partial z_t} &= \frac{\partial J}{\partial h_t} \odot \tilde{h}_t + \frac{\partial J}{\partial h_t} \odot (-h_{t-1}) \\ \frac{\partial J}{\partial r_t} &= (\frac{\partial J}{\partial s_t} \odot h_{t-1}) W_{sh}^T \end{aligned}\right\}$$ 同样要多算一次$sigmoid$导数



$$r_t = x_t W_{rx} + h_{t-1} W_{rh} + b_r$$

$$z_t = x_t W_{zx} + h_{t-1} W_{zh} + b_z$$

$$r_t = sigmoid(r_t), z_t = sigmoid(z_t)$$

$$s_t = x_t W_{sx} + (r_t \odot h_{t-1}) W_{sh} + b_s$$

$$\tilde{h}_t = \tanh(s_t)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

$$o_t = h_t W_{oh} + b_o$$

$$\frac{\partial J}{\partial W_{oh}} = h_t^T \frac{\partial J}{\partial o_t}, \qquad \frac{\partial J}{\partial W_{rx}} = x_t^T \frac{\partial J}{\partial r_t}$$

$$\frac{\partial J}{\partial W_{rh}} = h_{t-1}^T \frac{\partial J}{\partial r_t}, \qquad \frac{\partial J}{\partial W_{zx}} = x_t^T \frac{\partial J}{\partial z_t}$$

$$\frac{\partial J}{\partial W_{zh}} = h_{t-1}^T \frac{\partial J}{\partial z_t}, \qquad \frac{\partial J}{\partial W_{sx}} = x_t^T \frac{\partial J}{\partial s_t}$$

$$\frac{\partial J}{\partial W_{sh}} = (r_t \odot h_{t-1})^T \frac{\partial J}{\partial s_t}$$

deepshare.net

深度之眼

联系我们:

电话: 18001992849

邮箱: service@deepshare.net

Q Q: 2677693114

公众号

客服微信