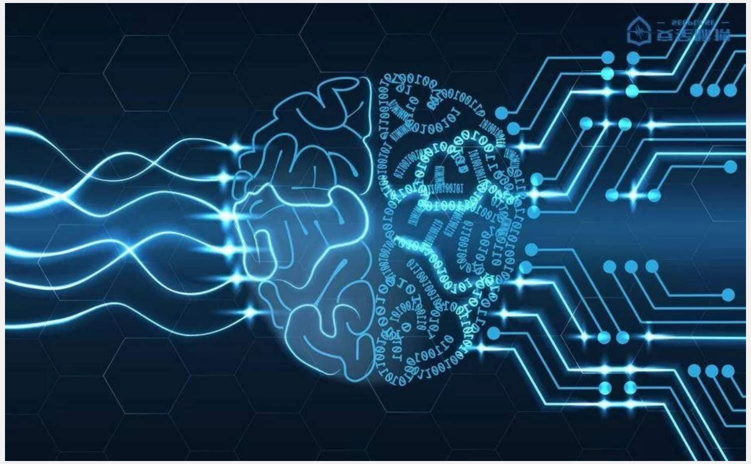


# 机器学习

## Machine learning

# 第八章：循环神经网络

目录

CONTENTS



为人师表

# 数值特征与分类特征

| Age | Gender | Nationality | Nationality |
|---|---|---|---|
| 35 | 1 | US | $[1, 0, 0, 0, \cdots, 0]$ |
| 31 | 1 | China | $[0, 1, 0, 0, \cdots, 0]$ |
| 29 | 0 | India | $[0, 0, 1, 0, \cdots, 0]$ |
| 27 | 1 | US | $[1, 0, 0, 0, \cdots, 0]$ |

数值特征：有顺序的　　二元特征：0或1　　分类特征

➢ 分类特征数值化

1、建立字典来标记国家：US→1，China → 2，India →3，Germany →4，…

2、进行one-hot encoding： US→1 →[1,0,0,…0]；China →2 →[0,1,0,…0]

✓ 从1开始编码，一般用0表示该特征未知的情况

✓ 思考：为什么不采用标量来编码？

# 词性标注（Parts of Speech）

**Input1：** My **work** is easy

**Output1：** pronouns nouns verb adjective

- 词表征：one-hot vector（独热编码）

把词典中的词展开为一向量

维度可能为10000

如my的输入为：

$$\begin{bmatrix} 0 \\ 0 \\ \dots \\ 1 \\ \dots \\ 0 \\ \dots \end{bmatrix} \begin{matrix} a \\ aaron \\ \dots \\ my \\ \dots \\ work \\ \dots \end{matrix}$$

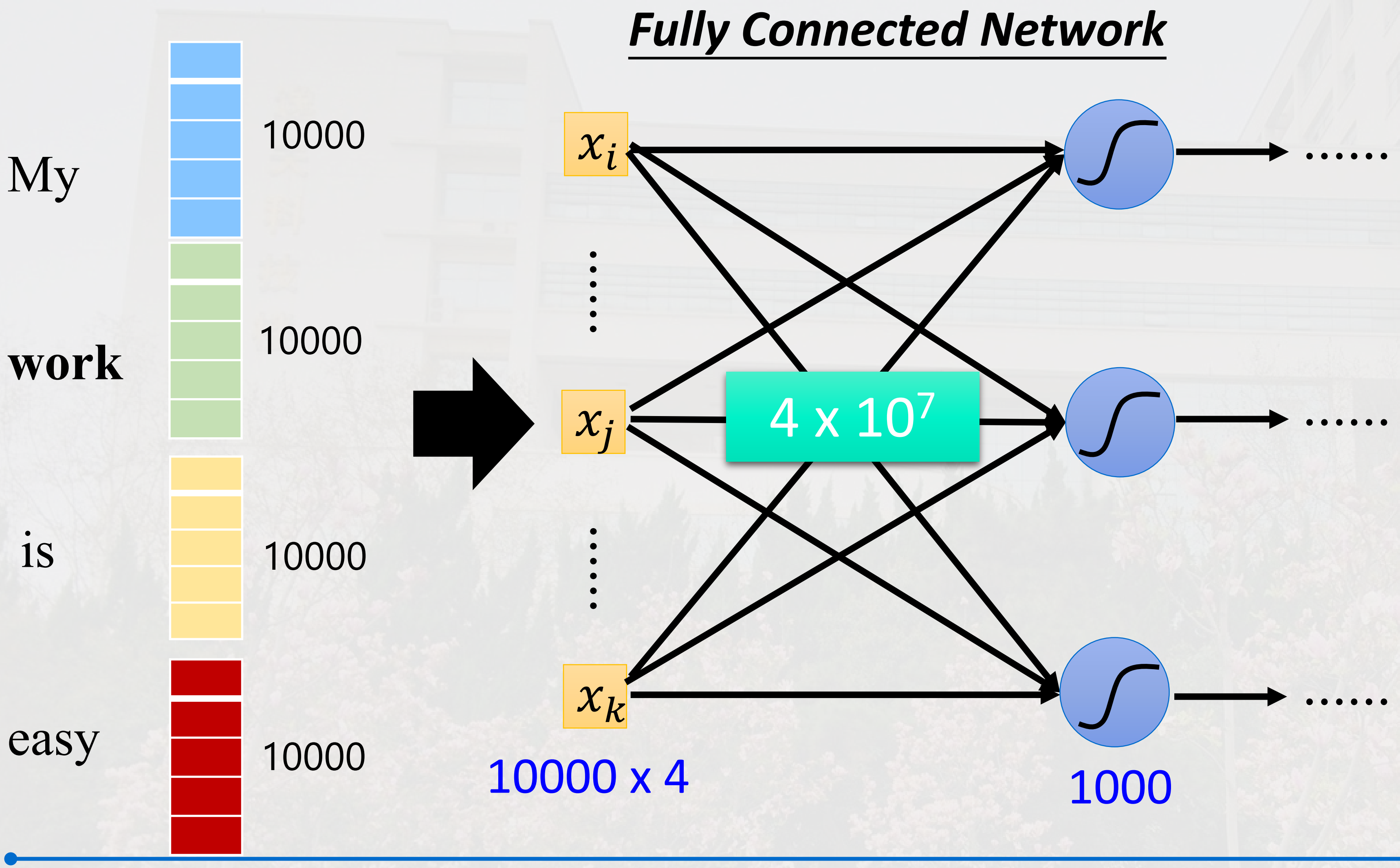
- 输出：9维向量，如my的输出为

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- *Nouns（名词）*
- *Pronouns（代名词）*
- *Verbs（动词）*
- *Adjectives（形容词）*
- *Adverbs（副词）*
- *Prepositions（介词）*
- *Conjunctions（连词）*
- *Articles/determiners（冠词）*
- *Interjections（感叹词）*

# 词性标注（Parts of Speech）

- 用全连接神经网络来建模




- 参数过多
- 句子长度可发生变化
- 某些词的词性与前后文有关系

# 词性标注（Parts of Speech）

| | | | | |
|---|---|---|---|---|
| **Input1：** | My | **work** | is | easy |
| **Output1：** | pronouns | nouns | verb | adjective |

| | | | |
|---|---|---|---|
| **Input2：** | I | **work** | hard |
| **Output2：** | Pronouns | verb | adverb |

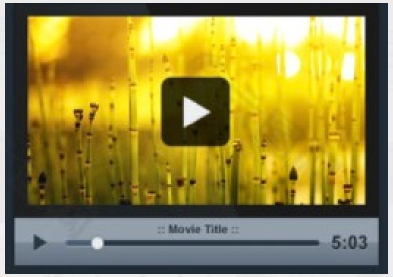
# 1. 为什么需要RNN?

## 前馈网络的一些不足：

- ➢ 连接存在层与层之间，每层的节点之间是无连接的。（无循环）
- ➢ 输入和输出的维数都是固定的，不能任意改变。无法处理变长的序列数据。
- ➢ 假设每次输入都是独立的，也就是说每次网络的输出只依赖于当前的输入。

**时间序列数据**是指在不同时间点上收集到的数据，这类数据反映了某一事物、现象等随时间的变化状态或程度。

视频

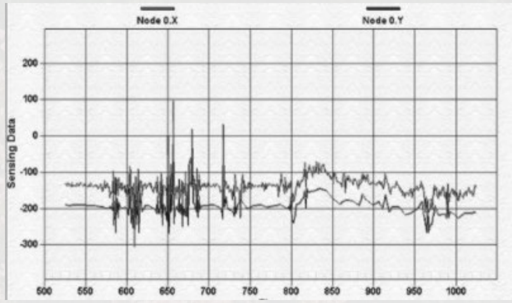



音乐

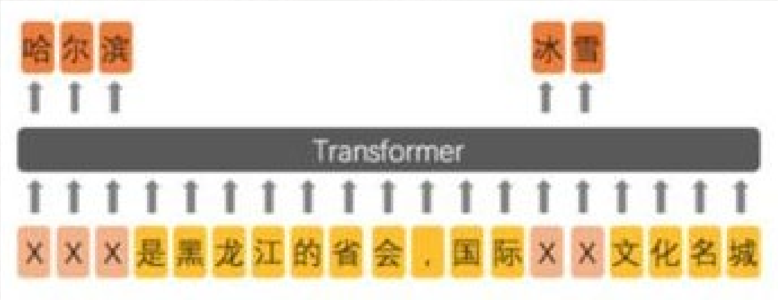
生物序列(DNA)

实际应用中，某些任务需要能够更好的处理**序列信息**，即前面的输入和后面的输入是有关系的

工业传感数据



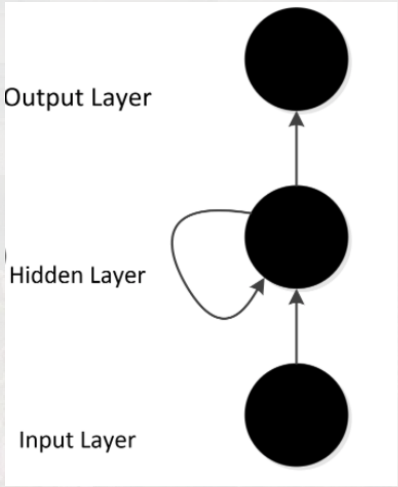

语义识别NLP



# 2. 循环神经网络

- 循环神经网络通过使用带自反馈的神经元，能够处理任意长度的序列。
- 循环神经网络比前馈神经网络更加符合生物神经网络的结构。
- 循环神经网络已经被广泛应用在语音识别、语言模型以及自然语言生成等任务上。

典型的RNN网络



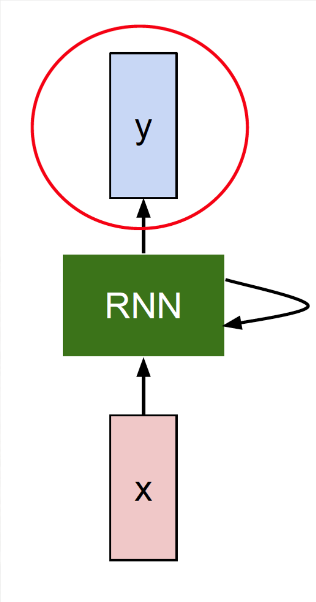
# 2.循环神经网络-一些定义

在进一步了解RNN之前，我们首先给出一些基本定义：

- $x_t$: $t$时刻的输入
- $y_t$: $t$时刻的真实输出
- $\hat{y}_t$: $t$时刻的预测输出
- $h_t$: $t$时刻的隐状态
- $W_{hh}$:隐状态单元之间的共享权重
- $W_{hy}$:隐含层与输出层之间的权重
- $W_{xh}$:输入层与隐含层之间的权重
- $\theta$: 泛指一般权重
- $g_\theta$: 非线性激活函数
- $L_t$: $t$时刻预测值与真实值之间的损失函数
- $E_t$: $t$时刻的交叉熵损失函数

# 2.循环神经网络-基本单元结构

任务：预测在某一时间节点的输出向量



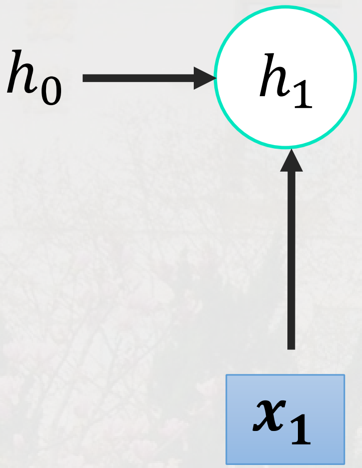

可以通过一个递归函数来对一个时序信息$x$进行建模：

$$\boxed{h_t} = \boxed{g_\theta}(\boxed{h_{t-1}}, \boxed{x_t})$$

新的隐状态　非线性函数　旧的状态　某一时刻的输入

# 2. 经典循环神经网络结构-1

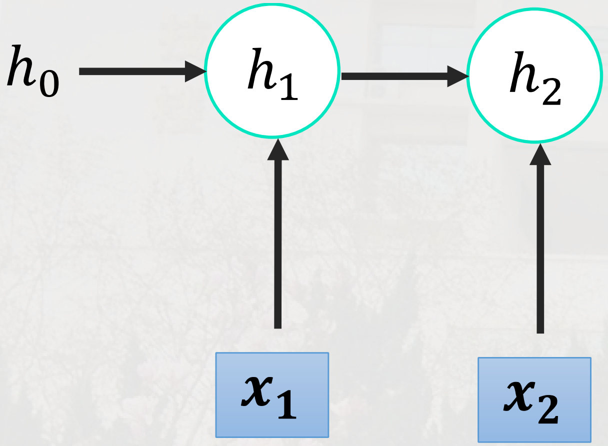
RNN引入了隐状态$h$（hidden state），$h$可对序列数据提取特征，接着再转换为输出。首先从一个时间节点出发，先计算$h_1$：




$$h_1 = g_\theta(W_{xh}x_1 + W_{hh}h_0 + b_x)$$

# 2. 经典循环神经网络结构-2

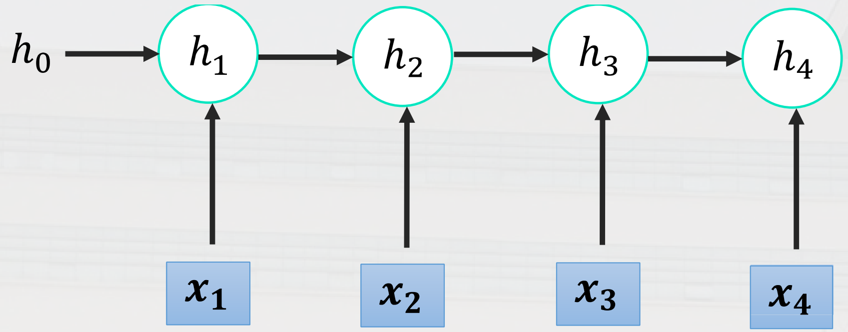
在第一步的基础上，上一步的隐状态$h_1$可以随时间传播到下一步用来计算$h_2$，在RNN中每个步骤使用用的参数$W_\theta$都相同且共享，其计算结果如下：




$$h_2 = g_\theta(W_{xh}x_2 + W_{hh}h_1 + b_x)$$

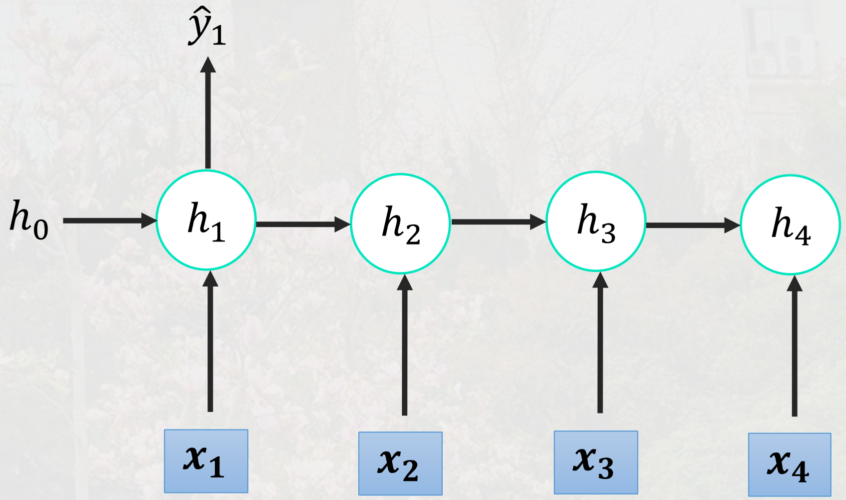
# 2. 经典循环神经网络结构-3

计算$h_3$,$h_4$，网络结构如下图所示：



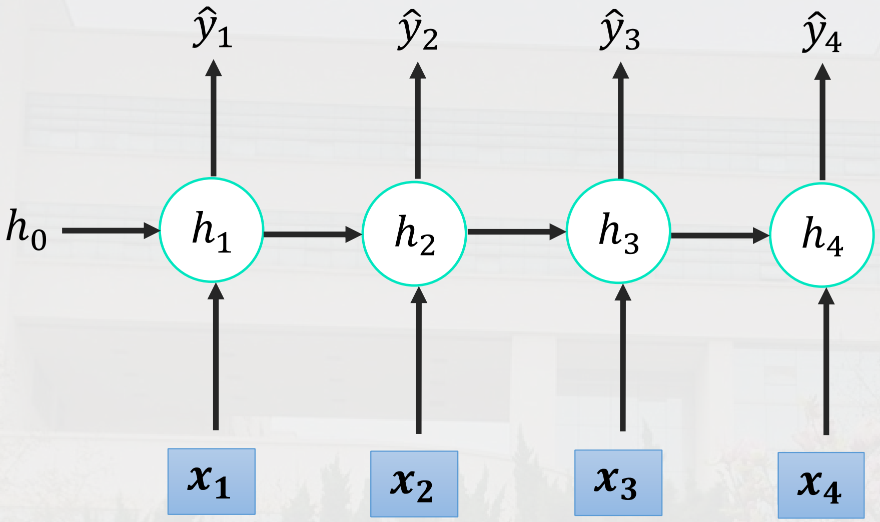

基于此，计算RNN的预测输出$\hat{y}_1$：

$$\hat{y}_1 = f_\theta(W_{hy}h_1 + b_y)$$



# 2. 经典循环神经网络结构-4

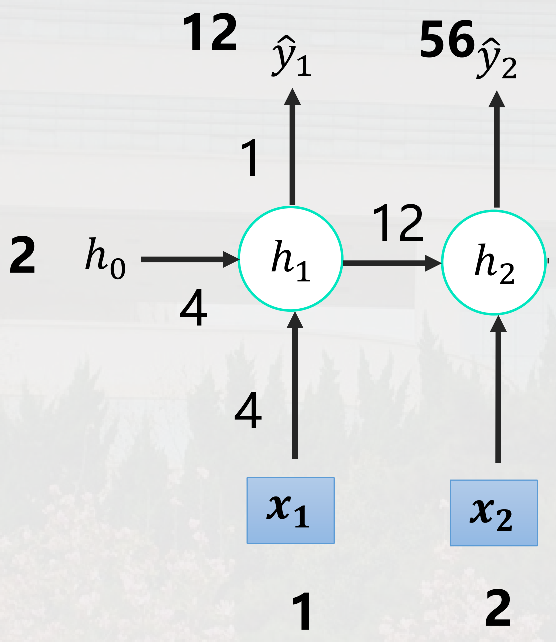
使用和$\hat{y}_1$相同的参数，得到$\hat{y}_1$, $\hat{y}_2$, $\hat{y}_3$, $\hat{y}_4$的输出结构：




以上即为最经典的RNN结构(Vanilla RNN)，其输入为$x_1, x_2, x_3, x_4$，输出为$\hat{y}_1$, $\hat{y}_2$, $\hat{y}_3$, $\hat{y}_4$。实际应用中，序列数据长度往往较长，最大值为$\hat{y}_n$，这里简化后只计算4个输入和输出。以上结构是经典的RNN结构，可以看出输入和输出等长。

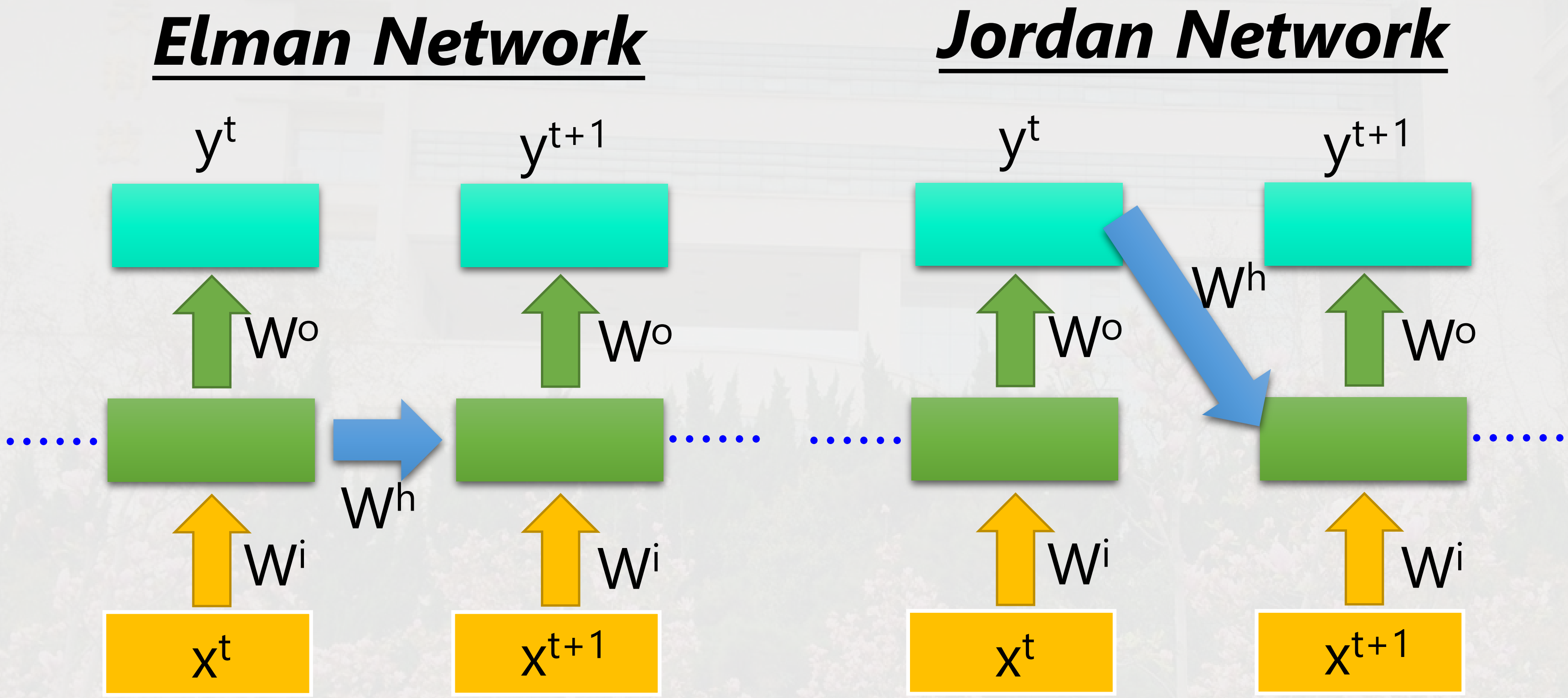
# 2. 经典循环神经网络结构

举例：$x_1=1; x_2=2; h_0=2; W_{xh}=4; W_{hh}=4; W_{hy}=1; y_1=?; y_2=?$（假设没有激活函数）。



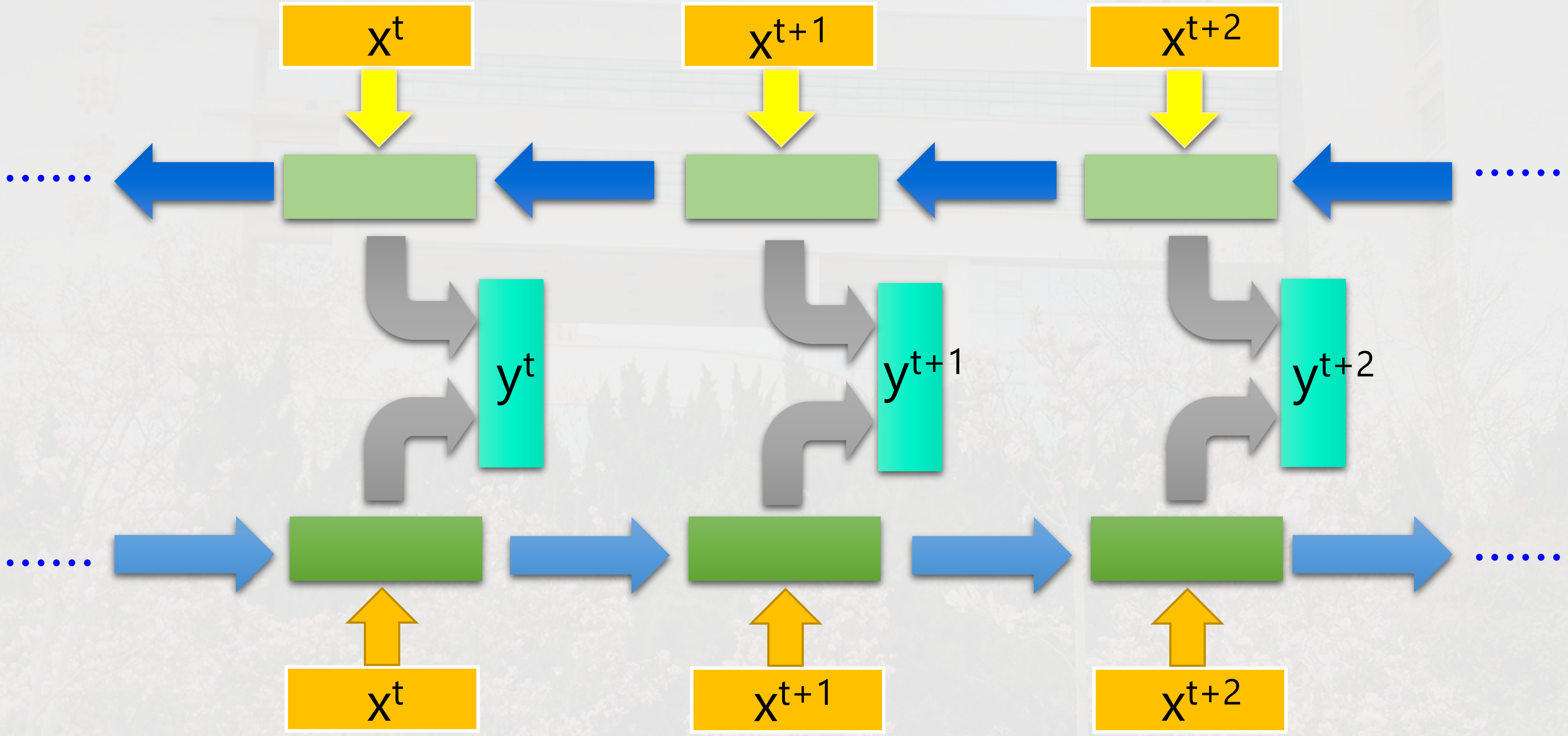
# 循环神经网络的结构

前面我们介绍的RNN也叫Elman network，RNN还有其他几种不同的结构类型，如Jordan network：



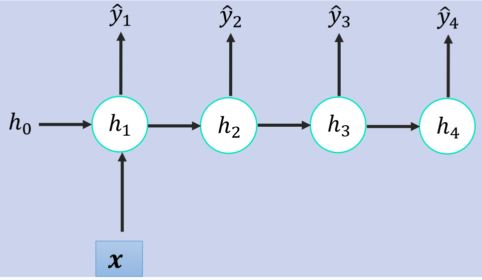
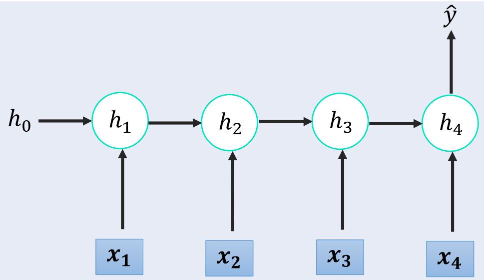
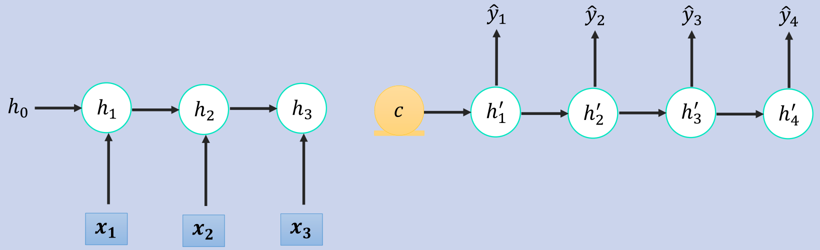
# 循环神经网络的结构

前面我们介绍的RNN也叫Elman network，RNN还有其他几种不同的结构类型，如Bidirectional RNN：

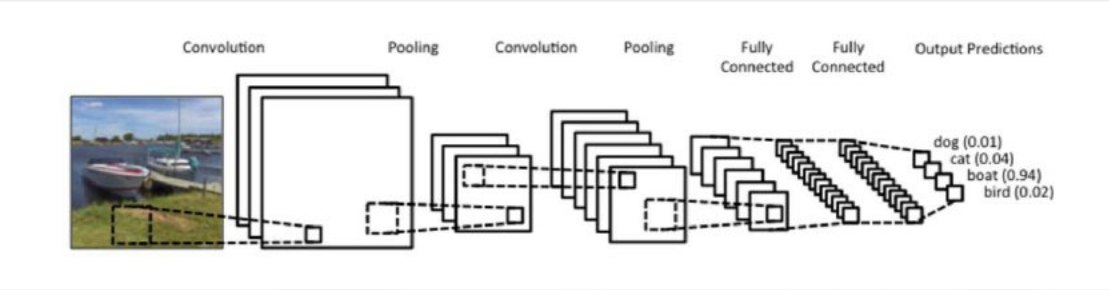


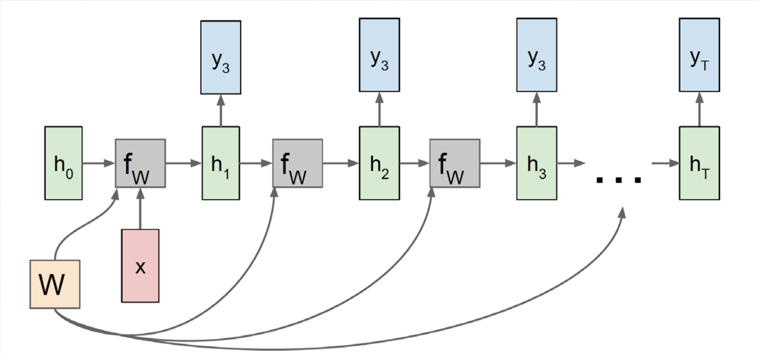
# 循环神经网络的其他结构

除了上面的经典结构（又称为Many-to-many结构），RNN还有其他几种不同的结构类型用来解决不同需求下的序列问题。下表中列出了常见的几种结构类型：

| 网络结构 | 结构视图 | 应用场景举例 |
|---|---|---|
| 1 to Many |  | 1、从图像生成文字，输入为图像的特征，输出为一段句子<br>2、根据图像生成语音或音乐，输入为图像特征，输出为一段语音或音乐 |
| Many to 1 |  | 1、输入一段文字，判断其所属类别<br>2、输入一个句子，判断其情感倾向<br>3、输入一段视频，判断其所属类别 |
| Sequence to Sequence |  | 1、机器翻译，输入一种语言文本序列，输出另外一种语言的文本序列<br>2、文本摘要，输入文本序列，输出这段文本序列摘要<br>3、阅读理解，输入文章，输出问题答案<br>4、语音识别，输入语音序列信息，输出文字序列 |

# 2. 循环神经网络与CNN比较

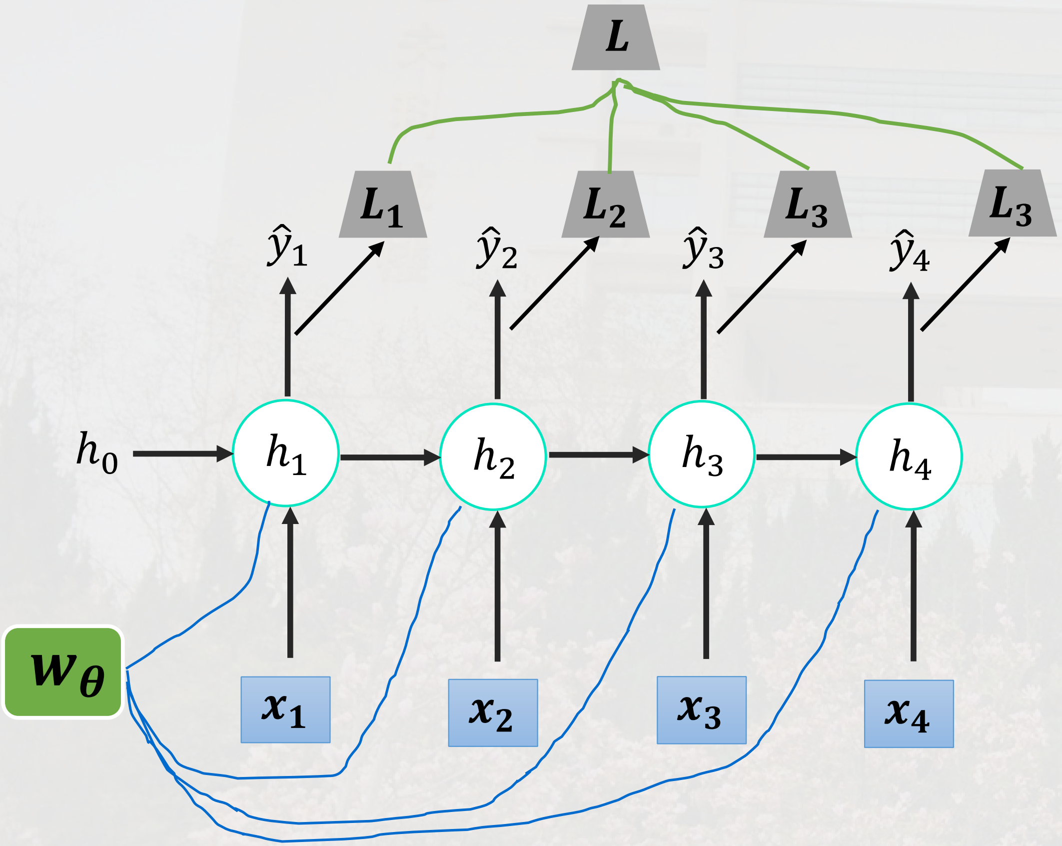


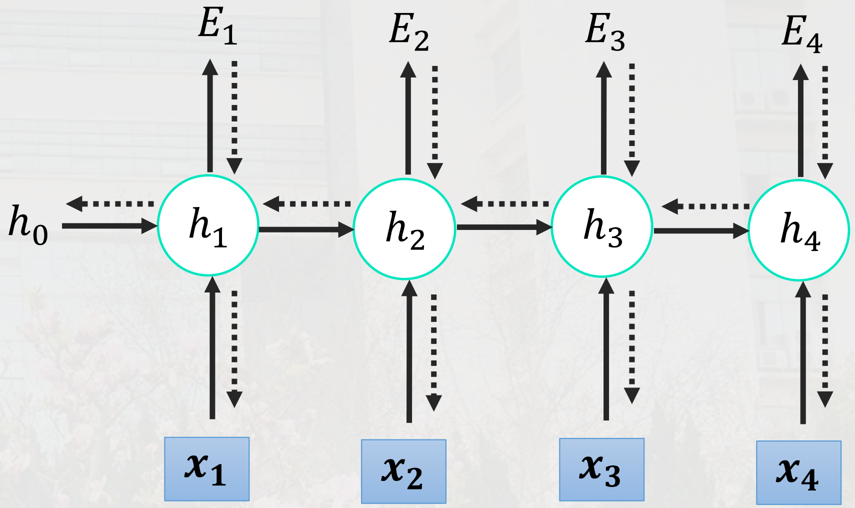




| 类别 | 特点描述 |
|---|---|
| 相同点 | 1、传统神经网络的扩展。<br>2、前向计算产生结果，反向计算模型更新。<br>3、每层神经网络横向可以多个神经元共存，纵向可以有多层神经网络连接。 |
| 不同点 | 1、CNN空间扩展，神经元与特征卷积；RNN时间扩展，神经元与多个时间输出计算。<br>2、RNN可以用于描述时间上连续状态的输出，有记忆功能，CNN用于静态输出。 |

# 3. BPTT（back-propagation through time）算法

BPTT（back-propagation through time）算法是常用的训练RNN的方法，其本质还是BP算法，只不过RNN处理时间序列数据，所以要基于时间反向传播，故叫**随时间反向传播。**




- $E_t$是在时间步$t$上的损失函数，即RNN的输入与输出之间的误差，是$y_t$的函数，可以用$E_t = \mathcal{L}(y_t)$ 表示。
- 最终的损失函数是每一个时刻损失函数的加和

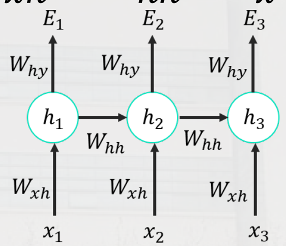
# 3. BPTT（back-propagation through time）算法

➢ $E_t$是在时间步$t$上的损失函数，即RNN的输入与输出之间的误差，是$\hat{y}_t$的函数，可以用$E_t=\mathcal{L}(y_t,\hat{y}_t)$ 表示

$$\hat{y}_t = f_\theta\left(W_{hy}h_t + b_y\right) \qquad h_t = g_\theta(W_{xh}x_t + W_{hh}h_{t-1} + b_x)$$

➢ 整个模型的损失函数$E=\sum_t E_t$, 因此，可得$E$对于网络权重$W_\theta$（表示 $W_{xh}$， $W_{hh}$或$b_x$ ）的梯度为

$$\frac{\partial E}{\partial W_\theta} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W_\theta} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_t}\boxed{\frac{\partial h_t}{\partial W_\theta}}$$




$$\frac{\partial h_t}{\partial W_\theta} = \frac{\partial g_\theta(W_\theta, h_{t-1})}{\partial W_\theta} + \frac{\partial g_\theta(W_\theta, h_{t-1})}{\partial h_{t-1}}\boxed{\frac{\partial h_{t-1}}{\partial W_\theta}}$$

$$\frac{\partial h_{t-1}}{\partial W_\theta} = \frac{\partial g_\theta(W_\theta, h_{t-2})}{\partial W_\theta} + \frac{\partial g_\theta(W_\theta, h_{t-2})}{\partial h_{t-2}}\frac{\partial h_{t-2}}{\partial W_\theta}$$

…… ……

$$\frac{\partial h_t}{\partial W_\theta} = \frac{\partial g_\theta(W_\theta, h_{t-1})}{\partial W_\theta} + \frac{\partial g_\theta(W_\theta, h_{t-1})}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial W_\theta} + \cdots + \left(\prod_{j=k+1}^{t}\frac{\partial g_\theta(W_\theta, h_{j-1})}{\partial h_{j-1}}\right)\frac{\partial h_k}{\partial W_\theta} + \cdots$$

$$\frac{\partial E}{\partial W_\theta} = \sum_{t=1}^{T}\sum_{k=1}^{t}\frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_t}\left(\prod_{j=k+1}^{t}\boxed{\frac{\partial g_\theta(W_\theta, h_{j-1})}{\partial h_{j-1}}}\right)\frac{\partial h_k}{\partial W_\theta}$$

$$\frac{\partial h_j}{\partial h_{j-1}}$$

# 3. BPTT（back-propagation through time）算法

$$\frac{\partial E}{\partial W_\theta}=\sum_{t=1}^{T}\frac{\partial E_t}{\partial W_\theta}=\sum_{t=1}^{T}\sum_{k=1}^{t}\frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_t}\left(\prod_{j=k+1}^{t}\frac{\partial h_j}{\partial h_{j-1}}\right)\frac{\partial h_k}{\partial W_\theta}$$

$\frac{\partial h_k}{\partial W_\theta}$仅表示回传一步的偏导数，可以观察到，求解某一时刻的梯度，需要追溯这个时刻之前所有时刻的信息

进一步求解的关键在于求解$\frac{\partial h_t}{\partial h_{t-1}}$，假设$h_t$和$h_{t-1}$都是$p$维列向量，$h_t[i]$为$h_i$第$i$维元素，可进一步写为

$$h_t=\left[h_t[1],h_t[2],\dots,h_t[p]\right]^T=\begin{bmatrix} g_\theta(x_t,h_{t-1})[1]\\ g_\theta(x_t,h_{t-1})[2]\\ \vdots\\ g_\theta(x_t,h_{t-1})[p]\end{bmatrix}=\begin{bmatrix} g_\theta(W_{xh}[1]x_t+W_{hh}[1]h_{t-1}+b_x[1])\\ g_\theta(W_{xh}[2]x_t+W_{hh}[2]h_{t-1}+b_x[2])\\ \vdots\\ g_\theta(W_{xh}[p]x_t+W_{hh}[p]h_{t-1}+b_x[3])\end{bmatrix}$$

其中$g_\theta(W_{xh}[i]x_t+W_{hh}[i]h_{t-1}+b_x[i])$为$h_t$的第$i$维值，且为标量。

# 3. BPTT（back-propagation through time）算法

$$\frac{\partial E}{\partial W_\theta}=\sum_{t=1}^{T}\frac{\partial E_t}{\partial W_\theta}=\sum_{t=1}^{T}\sum_{k=1}^{t}\frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_t}\left(\prod_{j=k+1}^{t}\frac{\partial h_j}{\partial h_{j-1}}\right)\frac{\partial h_k}{\partial W_\theta}$$

进一步求导可得**（向量对向量求偏导，会得到什么？）**：

$$\frac{\partial h_t}{\partial h_{t-1}}=\left[\frac{\partial g_\theta(x_t,h_{t-1})[1]}{\partial h_{t-1}},\frac{\partial g_\theta(x_t,h_{t-1})[2]}{\partial h_{t-1}},\dots,\frac{\partial g_\theta(x_t,h_{t-1})[p]}{\partial h_{t-1}}\right]^T$$

$$=\begin{bmatrix}W_{hh}[1]g'_\theta(x_t,h_{t-1})[1]\\ W_{hh}[2]g'_\theta(x_t,h_{t-1})[2]\\ \vdots\\ W_{hh}[p]g'_\theta(x_t,h_{t-1})[p]\end{bmatrix}$$

$$=diag\left[g'_\theta(x_t,h_{t-1})[i]\right]W_{hh}$$

其中$diag[g'_\theta(x_t,h_{t-1})[i]]$为对角阵，其对角元素为$g'_\theta(x_t,h_{t-1})[i]$。

# 3. BPTT（back-propagation through time）算法

$$\frac{\partial E}{\partial W_\theta}=\sum_{t=1}^{T}\frac{\partial E_t}{\partial W_\theta}=\sum_{t=1}^{T}\sum_{k=1}^{t}\frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_t}\left(\prod_{j=k+1}^{t}\frac{\partial h_j}{\partial h_{j-1}}\right)\frac{\partial h_k}{\partial W_\theta}$$

将上式代入到网络权重$W_\theta$的梯度公式中，可得

$$\frac{\partial E}{\partial W_\theta}=\sum_{t=1}^{T}\sum_{k=1}^{t}\frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_\theta}\left(\prod_{j=k+1}^{t} diag\left[g'_\theta\left(x_j,h_{j-1}\right)\right]W_{hh}\right)\frac{\partial h_k}{\partial W_\theta}$$

RNN通过上式实现沿时间的反向传播。

# 3. RNN中的梯度消失和梯度爆炸

**时间的短期依赖问题：**

- ❑ 预测短时间序列信息

举例：从语境中预测下一个词汇

*“有朵云彩在？” →　“天空”*

基于云彩这个词汇很容易实现准确预测（短期时间依赖问题）。  RNN Good so far

**时间的长期依赖问题：**

- ❑ 预测较长时间序列的信息

举例：考虑一个很长的语境

*“我是一名华科的学生，。。。。。，我不能哭，因为我是华科人啊”*

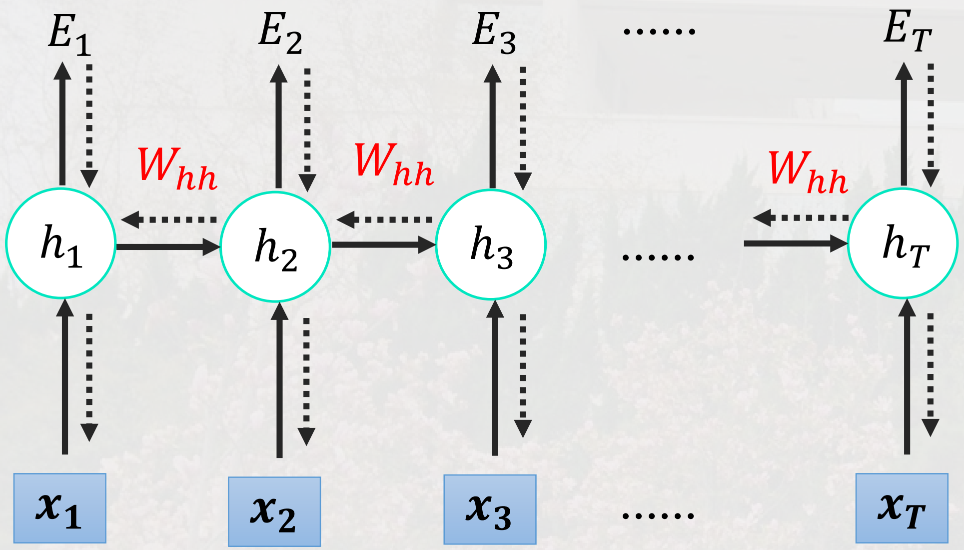
预测的信息需要追溯到很久之前的语境。  Practically difficult for RNN

# 3. RNN中的梯度消失和梯度爆炸

RNN梯度更新公式：

$$\frac{\partial E}{\partial W_\theta}=\sum_{t=1}^{T}\sum_{k=1}^{t}\frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_\theta}\left(\prod_{j=k+1}^{t} diag\left[g'_\theta\left(x_j,h_{j-1}\right)\right]W_{hh}\right)\frac{\partial h_k}{\partial W_\theta}$$

**激活函数的导数**　**权重矩阵**




**误差随着时间从第t步传递到k步**
**(重复矩阵相乘导致梯度的消失和爆炸)**

# 3. RNN中的梯度消失和梯度爆炸

$w = 1 \Rightarrow y^{1000} = 1$

$w = 1.01 \Rightarrow y^{1000} \approx 20000$

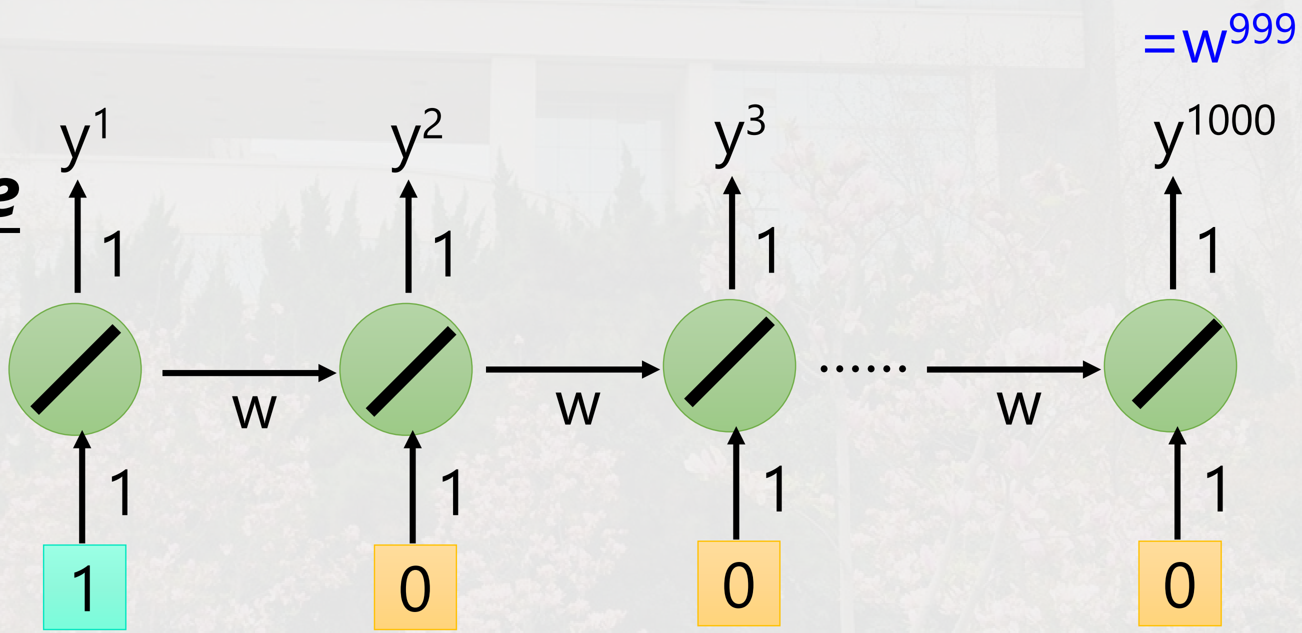
Large $\partial L / \partial w$ → Small Learning rate?

$w = 0.99 \Rightarrow y^{1000} \approx 0$

$w = 0.01 \Rightarrow y^{1000} \approx 0$

small $\partial L / \partial w$ → Large Learning rate?

***Toy Example***



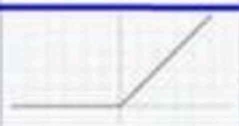
# 3. RNN中的梯度消失和梯度爆炸

## WHY

$$\frac{\partial E}{\partial W_\theta}=\sum_{t=1}^{T}\sum_{k=1}^{t}\frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_\theta}\left(\prod_{j=k+1}^{t} diag\left[g_\theta'(x_j,h_{j-1})\right]W_{hh}\right)\frac{\partial h_k}{\partial W_\theta}$$

**激活函数的导数**

首先考虑激活函数：

| Name | Plot | Equation | Derivative (with respect to x) | Range |
|---|---|---|---|---|
| Logistic |  | $f(x)=\frac{1}{1+e^{-x}}$ | $f'(x)=f(x)(1-f(x))\rightarrow(0,\ 0.25]$ | $(0,1)$ |
| TanH |  | $f(x)=\tanh(x)=\frac{2}{1+e^{-2x}}-1$ | $f'(x)=1-f(x)^2\rightarrow(0,\ 1]$ | $(-1,1)$ |
| Rectified linear unit (ReLU) |  | $f(x)=\begin{cases}0 & \text{for } x<0\\ x & \text{for } x\ge 0\end{cases}$ | $f'(x)=\begin{cases}0 & \text{for } x<0\\ 1 & \text{for } x\ge 0\end{cases}$ 点击查看源网页 | $[0,\infty)$ |
| Leaky rectified linear unit (Leaky ReLU) |  | $f(x)=\begin{cases}0.01x & \text{for } x<0\\ x & \text{for } x\ge 0\end{cases}$ | $f'(x)=\begin{cases}0.01 & \text{for } x<0\\ 1 & \text{for } x\ge 0\end{cases}$ | $(-\infty,\infty)$ |

# 3. RNN中的梯度消失和梯度爆炸

**WHY**

$$\frac{\partial E}{\partial W_\theta}=\sum_{t=1}^{T}\sum_{k=1}^{t}\frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_\theta}\left(\prod_{j=k+1}^{t} diag\left[g_\theta'(x_j,h_{j-1})\right]W_{hh}\right)\frac{\partial h_k}{\partial W_\theta}$$

权重矩阵

考虑权重矩阵：

已知 $g_\theta'(x_j,h_{j-1})$存在最大值，因此$\|diag[g_\theta'(x_j,h_{j-1})]\|\le\gamma$。

假设$\lambda_1$是$W_{hh}$矩阵的奇异值分解后的最大值，如果$\lambda_1<\frac{1}{\gamma}$，可知：

$$\forall t,\left\|\frac{\partial h_t}{\partial h_{t-1}}\right\|\le\|diag[g_\theta'(x_j,h_{j-1})]\|\|W_{hh}\|=\eta<\gamma\frac{1}{\gamma}<1$$

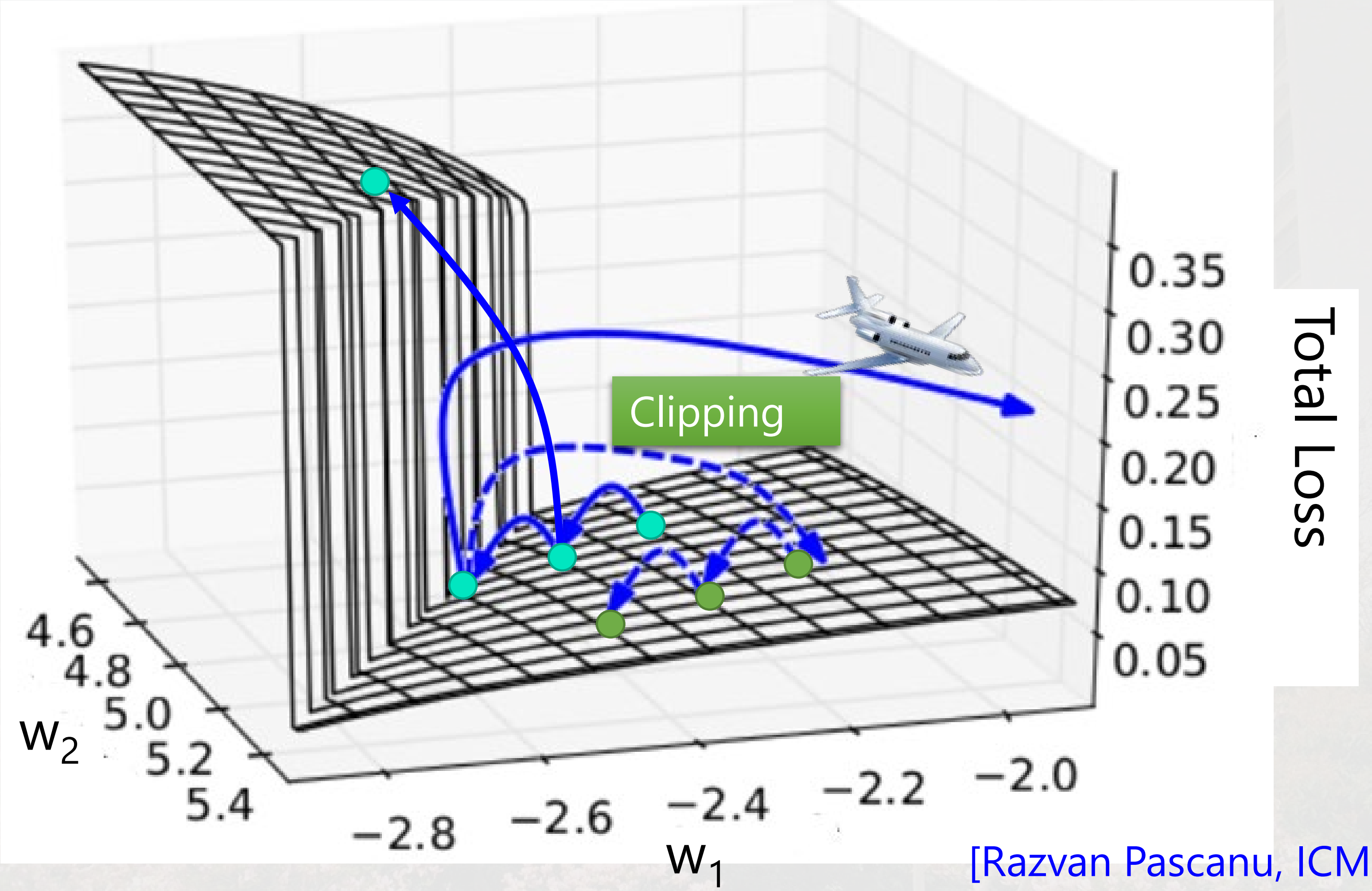
通过多次连乘，得到

$$\left\|\prod_{j=k+1}^{t} diag\left[g_\theta'(x_j,h_{j-1})\right]W_{hh}\right\|\le\eta^{t-k}$$

**因为$\eta<1$，当沿着时间方向回传时 $\eta^{t-k}$接近于0。**
**相反，如果$\lambda_1>\frac{1}{\gamma}$，可证明会出现梯度爆炸。**

# 3. RNN中的梯度消失和梯度爆炸

➢ 误差曲面要么很平坦要么很陡峭




[Razvan Pascanu, ICML'13]

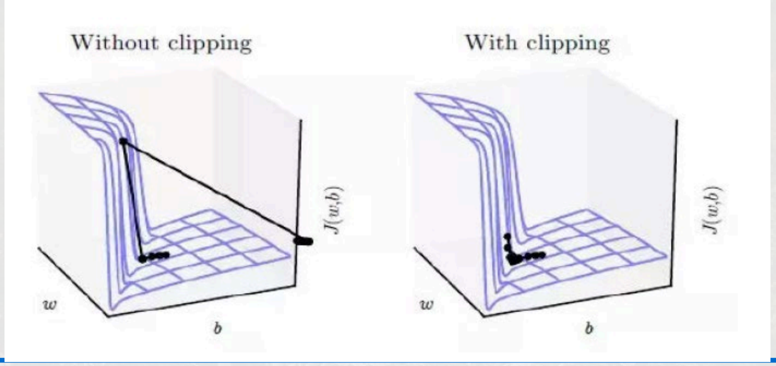
# 3. 解决梯度消失和梯度爆炸的方法

导致梯度消失和梯度爆炸的原因都是RNN无法很好的处理序列间长期依赖问题，回传的步数越多，这种现象就越严重。目前有如下方法来缓解这个问题：

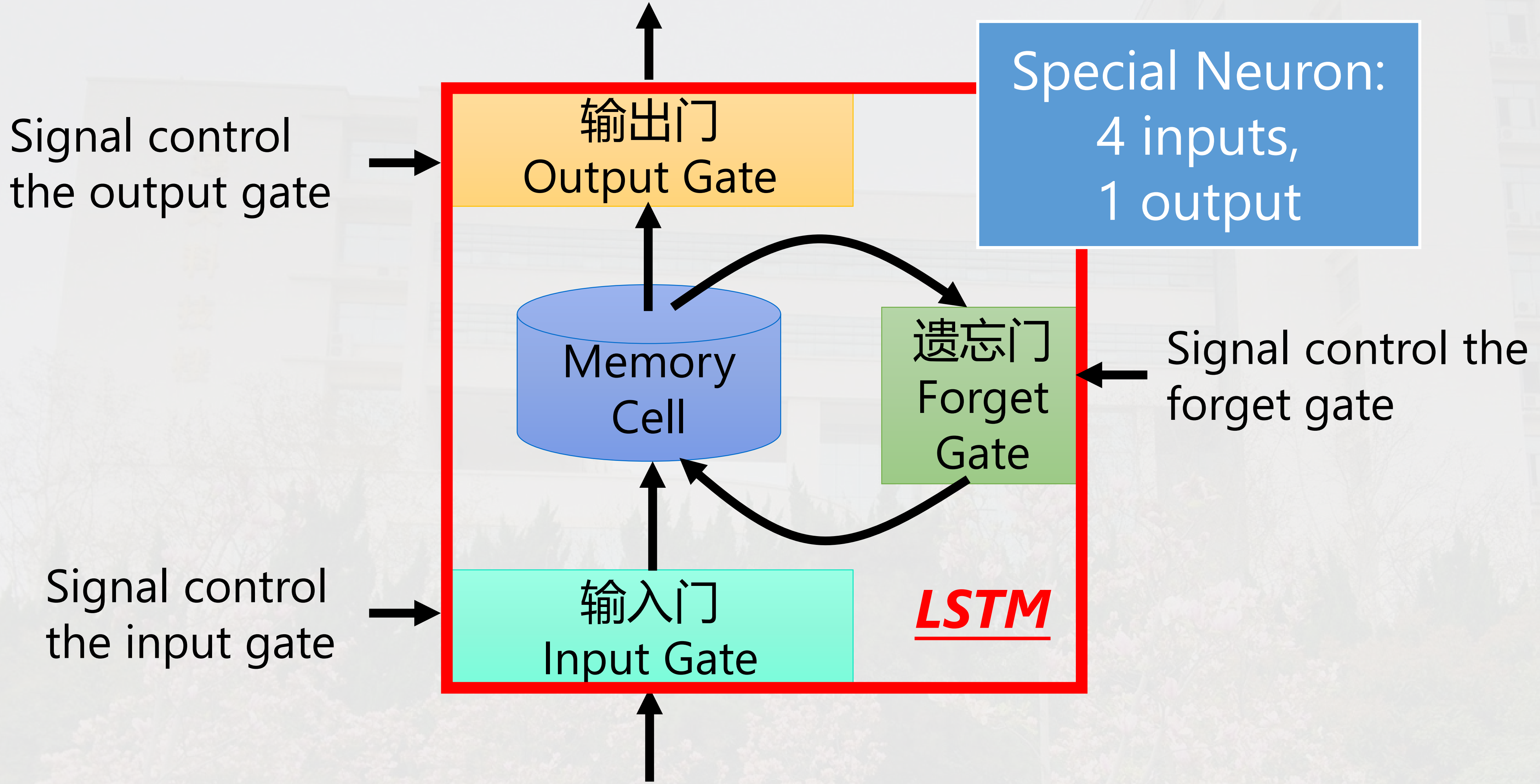
1）使用梯度截断的方法来避免梯度爆炸，即设置一个梯度截断的阈值，如果梯度的范数（Norm）超过这个阈值则对其进行强制截断，并设置梯度为该阈值；

2）引入正则项来惩罚网络权重大小，从而避免出现梯度过大的问题；

3）为了缓解梯度消失，研究者提出了多种基于RNN的变种，如长短期记忆神经网络、门控循环单元等。



# 3. 长短期记忆神经网络（LSTM）



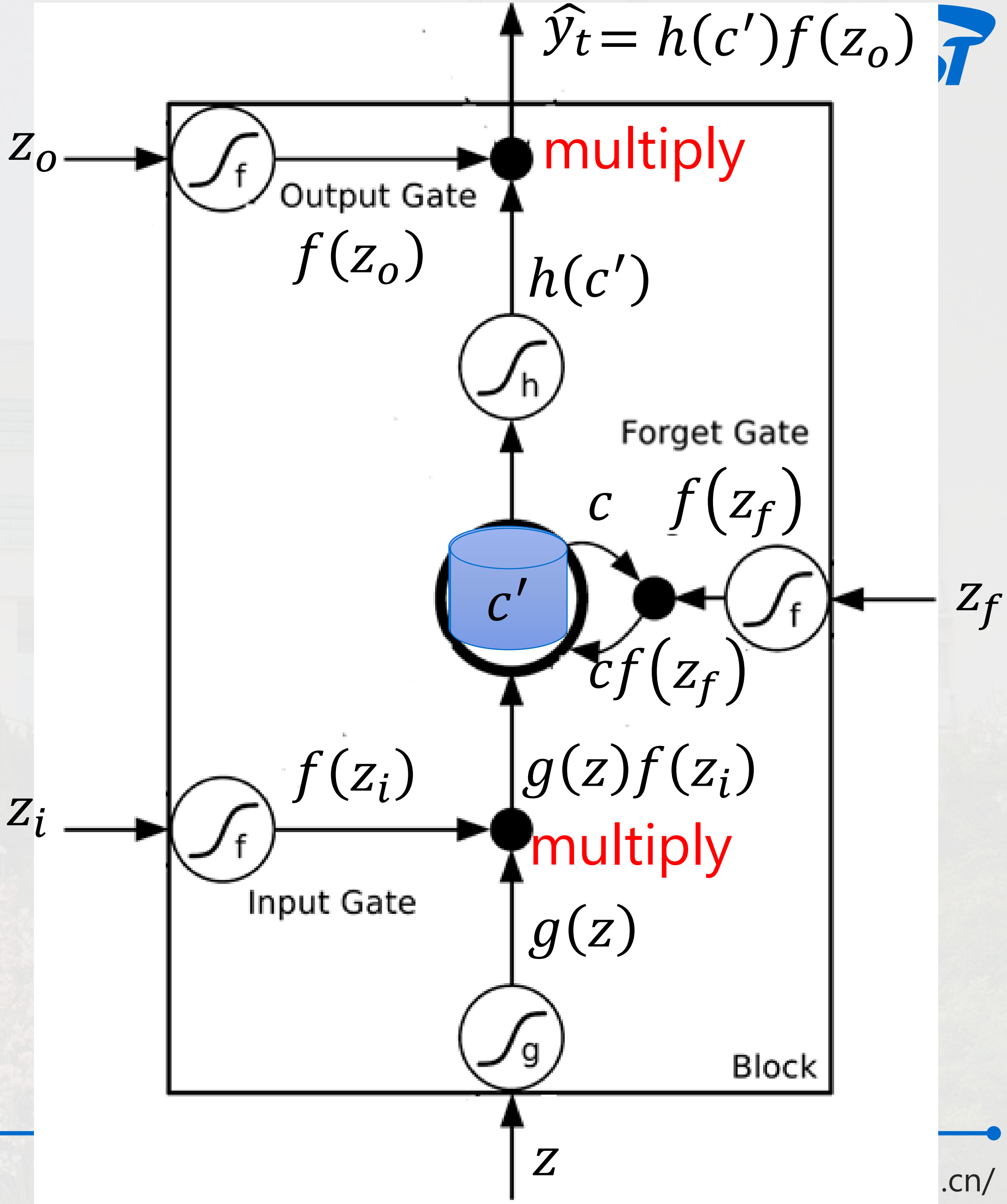
# 3. 长短期记忆神经网络（LSTM）

- 门控单元激活函数 f 一般是 sigmoid 函数

  值介于 0 和1之间

  模拟门的打开和关闭

$$c' = g(z)f(z_i) + cf(z_f)$$



# 3. 长短期记忆神经网络（LSTM）

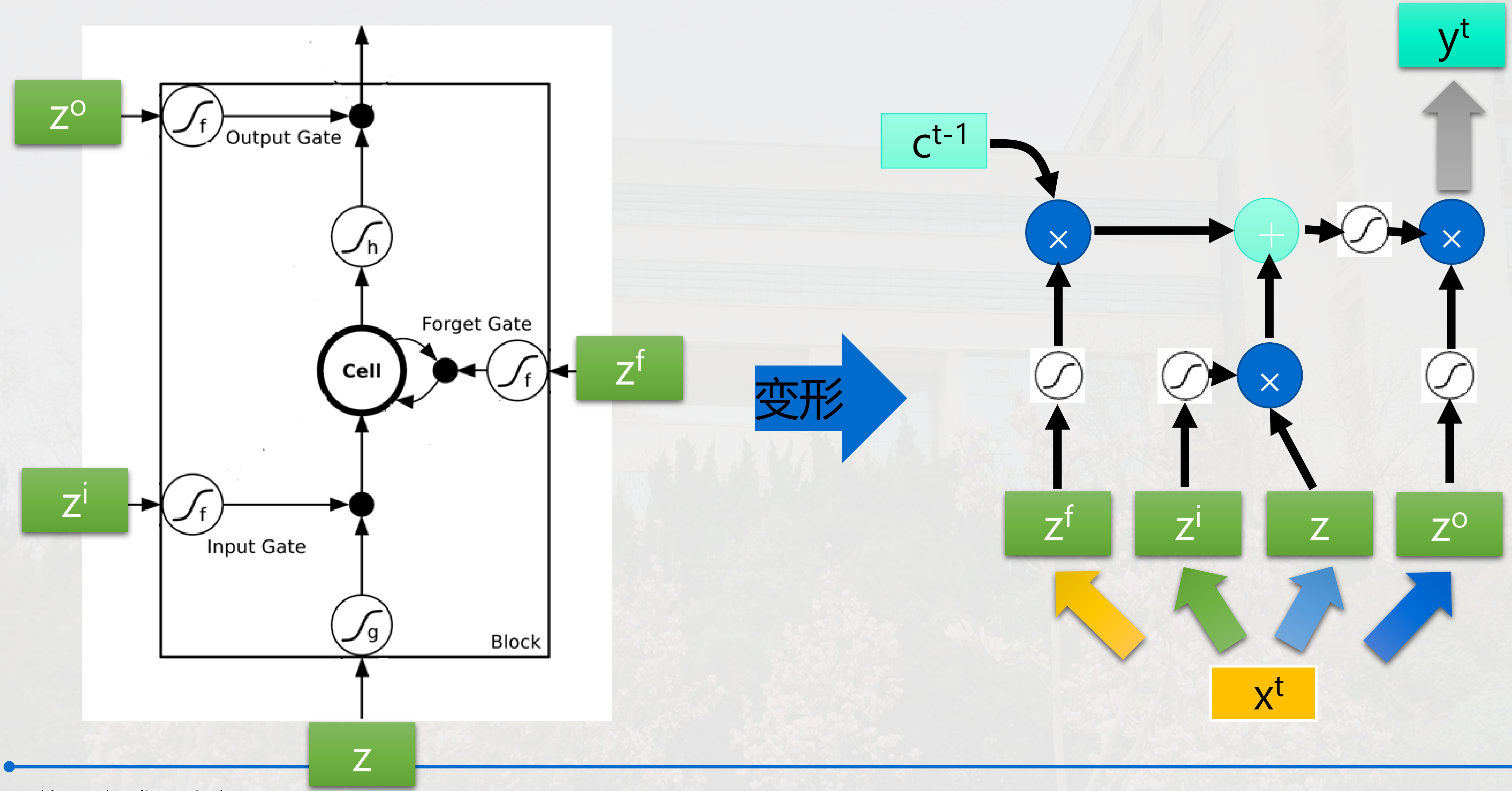
| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 3 | 3 | 7 | 7 | 7 | 0 | 6 |
| $x_1$ | 1 | 3 | 2 | 4 | 2 | 1 | 3 | 6 | 1 |
| $x_2$ | 0 | 1 | 0 | 1 | 0 | 0 | -1 | 1 | 0 |
| $x_3$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| $y$ | 0 | 0 | 0 | 0 | 0 | 7 | 0 | 0 | 6 |

When $x_2$ = 1, add the numbers of $x_1$ into the memory

When $x_2$ = -1, reset the memory

When $x_3$ = 1, output the number in the memory.

# 3. 长短期记忆神经网络（LSTM）

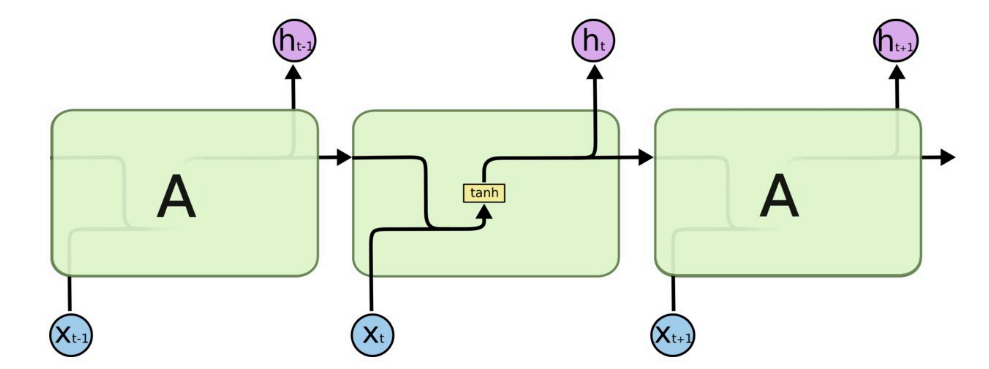


# 3. 长短期记忆神经网络（LSTM）

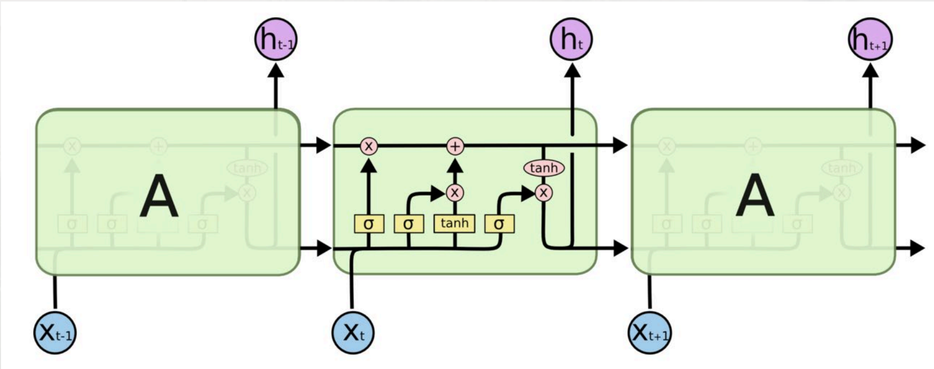
## LSTM与RNN的区别

长短期记忆神经网络（Long Short-term Memory, LSTM）是门限类RNN中最著名的一种，它改善了RNN的记忆能力同时减轻了梯度爆炸和梯度消失问题。基于RNN的基础结构，LSTM主要将原有的隐含层循环函数从简单的全连接改为使用**三个控制门的记忆单元**(Memory Cell)，假设隐含层的激活函数为*tanh*，通过下图对比看到RNN与LSTM单层结构的区别。

RNN

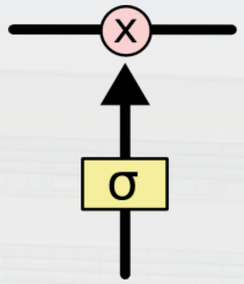



LSTM



# 3. 长短期记忆神经网络（LSTM）

**可以看到，最主要的区别为加入了三个下面的门控单元：**



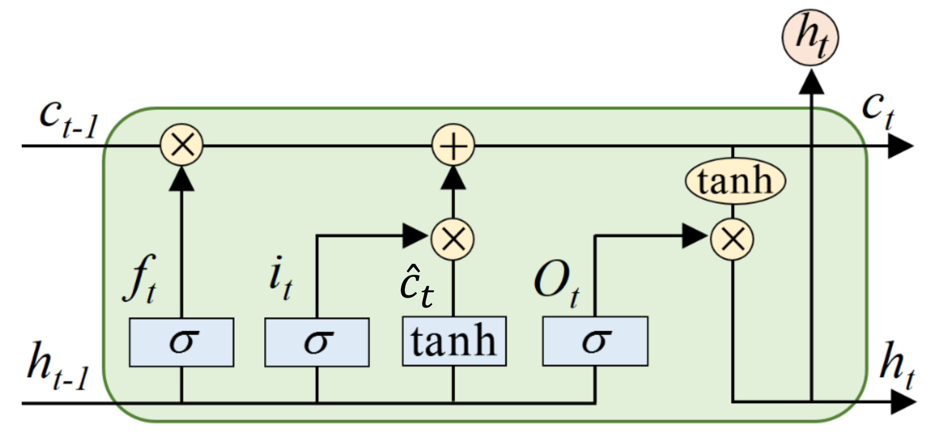

门控单元是控制信息通过量多少的一种函数，即对于向量$y$，通过向量$x$来控制$y$通过的信息量，具体通过下式表示：

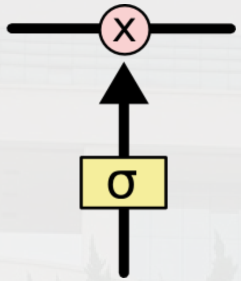
$$\boldsymbol{o} = \boldsymbol{\sigma}(\boldsymbol{x}) \otimes \boldsymbol{y}$$

其中⊗表示元素逐个相乘，$\sigma(x)$为sigmoid函数，其输出的每个元素取值范围在0到1之间。$\sigma(x)$中的元素约接近1，$y$对应位置的保留信息就越多；反之，$\sigma(x)$中的元素越接近0，$y$对应位置保留的信息就越少。$o$为门控单元的输出，即输入向量$y$通过该单元的输出信息。

# 3. 长短期记忆神经网络（LSTM）-循环函数

在$t$时刻，LSTM的循环函数可写为




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

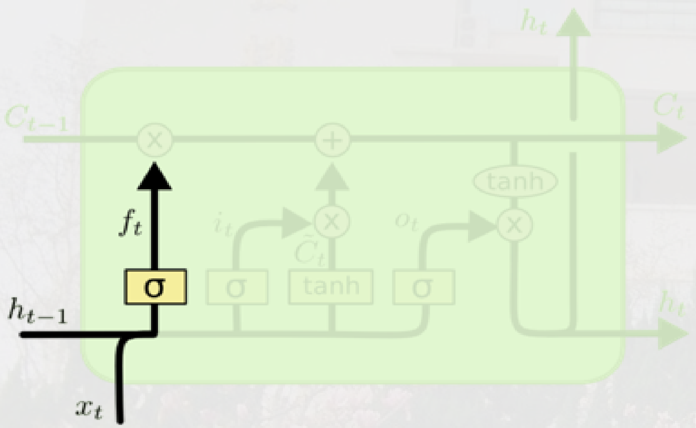
$$c_t = f_t \otimes c_{t-1} + i_t \otimes \hat{c}_t$$

$$\hat{c}_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$

其中$c_t$为$t$时刻的单元状态，存储序列的历史信息。三个门控单元分别为$i_t$，$o_t$，$f_t$，分别称之为输入门、输出门、遗忘门，下面分别解释三种门的工作模式 。

# 3. 长短期记忆神经网络（LSTM）-循环函数之遗忘门

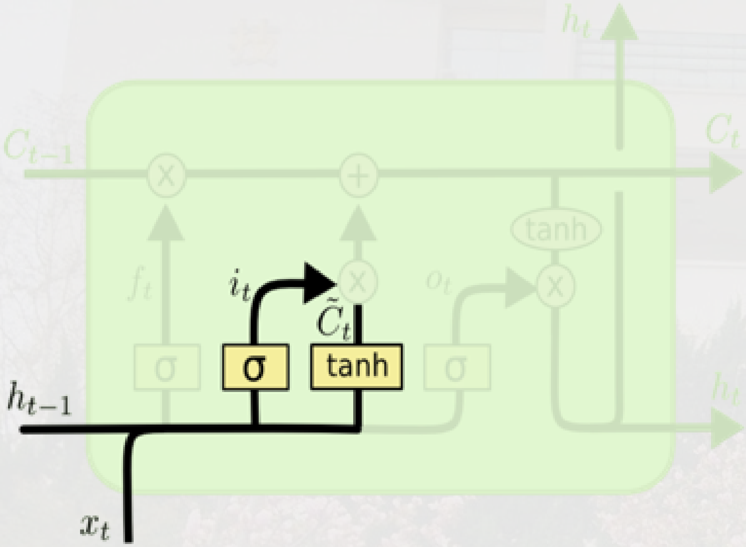
遗忘门$f_t$，该门的作用是对上一个单元状态信息选择性的遗忘。它读取的是上一个单元的隐变量$h_{t-1}$与此刻的$x_t$，并输出一个0到1之间的数值，具体计算公式如下：




$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + b_f\right)$$

# 3. 长短期记忆神经网络（LSTM）-循环函数之输入门

输入门$i_t$，该门的作用是决定当前时刻单元的隐变量需要更新的信息量，它将与遗忘门共同作用决定什么信息需要丢弃，什么新的信息需要保留，从而决定当前单元状态$c_t$的信息量，具体计算公式如下：

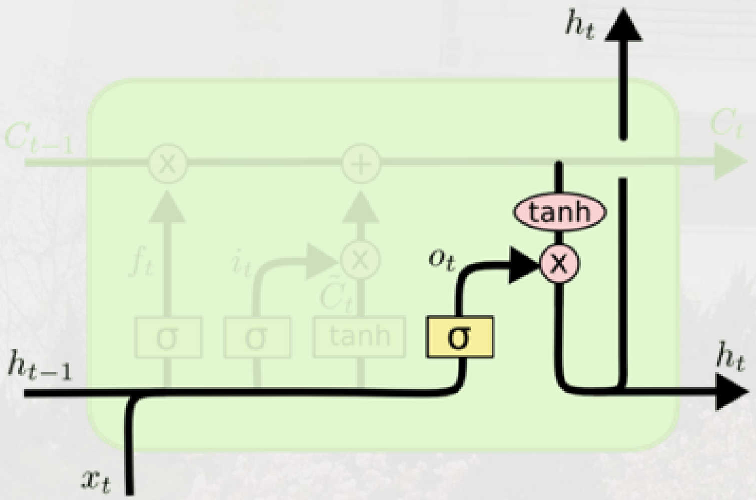



$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$

$$\tilde{C}_t = tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$

# 3. 长短期记忆神经网络（LSTM）-循环函数之输出门

输出门$o_t$， 其目的是从记忆单元$c_t$产生隐层单元$h_t$，具体公式如下图所示:




$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

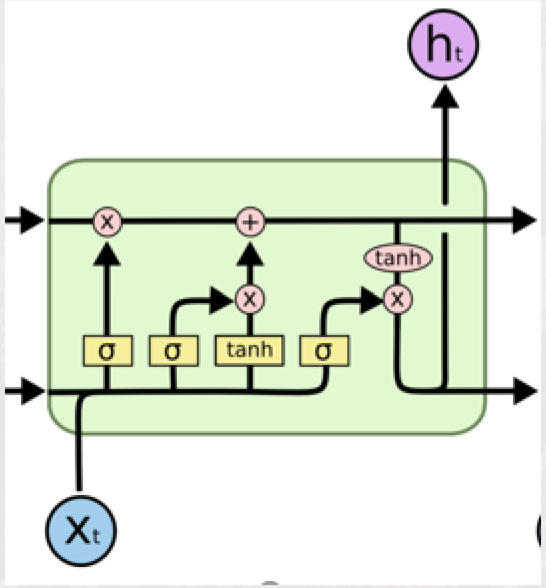
# 3. 长短期记忆神经网络（LSTM）-总结特点

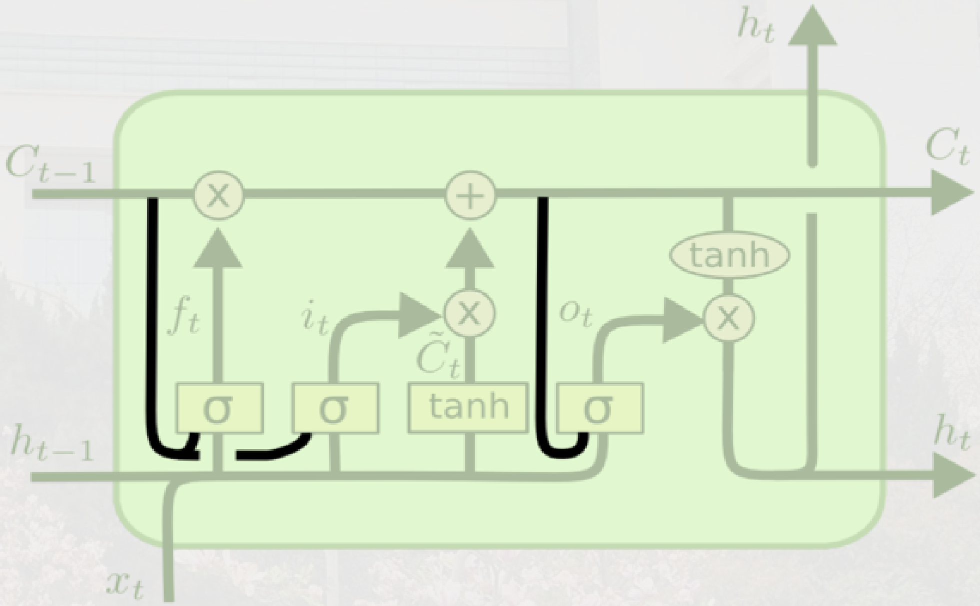
**LSTM通过这种复杂的循环函数在每个时间步上对当前的输入和记忆的历史信息进行重新的组合，很大程度上解决了梯度爆炸和梯度消失的问题，LSTM的特点可总结为如下几点:**

- 通过增加循环函数的复杂程度，引入门控单元，使得网络在梯度回传的过程中经历了更多的较小的激活函数，从而降低了梯度爆炸发生的可能性。
- 通过遗忘门的使用减小梯度消失的可能性。遗忘门中的偏置项$b_f$在初始化时设定为一个较大的值，从而使得$f_t$接近于1， 即这一时刻的单元状态$c_t$与上一时刻尽可能接近$c_{t-1}$。因此在训练时，即使其他的路径依然面临梯度消失的风险，$c_t$上的梯度能够通过遗忘门的引入一直回传不易消失。
- 解决梯度消失和梯度爆炸并不是设计LSTM的初衷，通过引入门控单元更好的改善RNN的模型结构，能够使得模型自由的选择信息的传递，是LSTM能够广泛应用和受到喜爱的重要原因。
- 研究者通过在RNN和LSTM的网络架构上进行改进，开发了多种RNN和LSTM的流行变体结构，下面我们将介绍几种经典的流行架构。

# 4. RNN和LSTM的其他架构变体 –Peephole连接

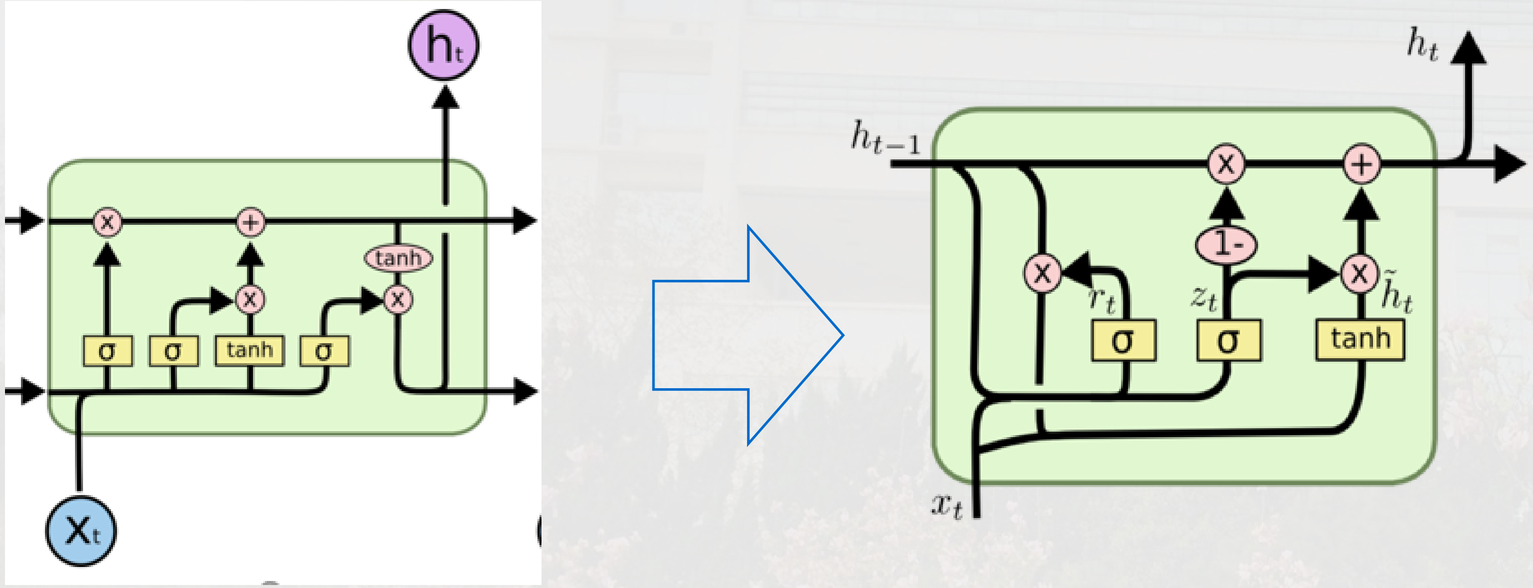
Gers & Schmidhuber (2000) 提出了增加 “peephole connections”在LSTM中，具体来讲，通过peephole connections使得隐状态不但受到$h_{t-1}$的影响，也会受上个单元状态$c_{t-1}$的影响。在实际应用中，可以根据需求选择加入适当位置的connections.



# 4. RNN和LSTM的其他架构变体 –门控循环单元（Gated Recurrent Unit）

门控循环单元（Gated Recurrent Unit，GRU）通过简化LSTM神经网络循环函数达到了类似的效果并节省了计算成本。在GRU中，遗忘门和输入门合并成了一个新的重置门$z_t$，且加入了一个更新门$r_t$，具体更新方式如下:




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

$$\hat{h}_t = \tanh(W_{x\hat{h}} x_t + W_{h\hat{h}}(r_t \otimes h_{t-1}))$$

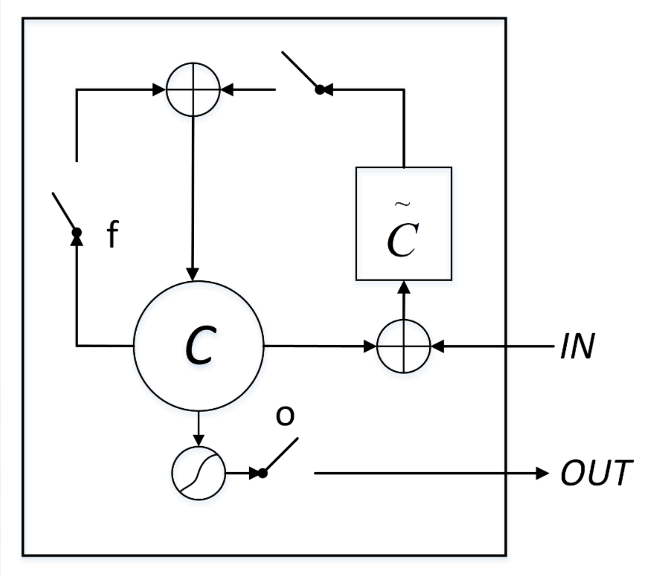
$$r_t = \sigma(W_{xr} x_t + W_{hr} h_{t\_1})$$

$$z_t = \sigma(W_{xz} x_t + W_{hz} h_{t-1})$$

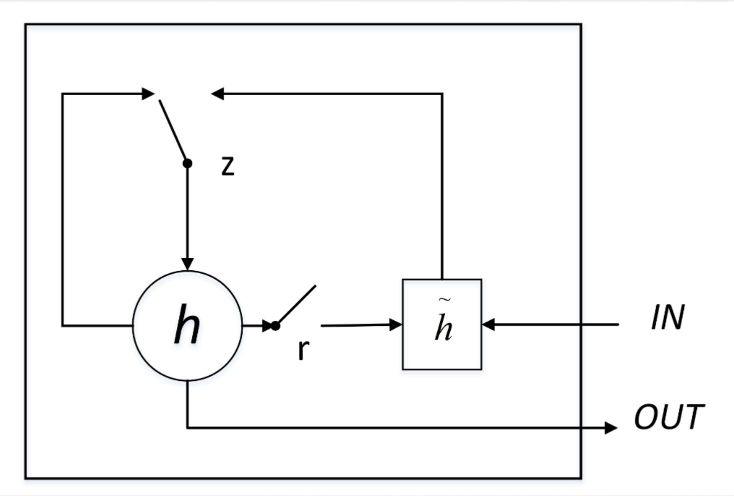
通过以上的简化，GRU可以达到和LSTM类似的训练效果，同时提高了网络运算的效率，降低了计算成本。

# 4. RNN和LSTM的其他架构变体 –门控循环单元（Gated Recurrent Unit)

门控循环单元（Gated Recurrent Unit，GRU）通过简化LSTM神经网络循环函数达到了类似的效果并节省了计算成本。在GRU中，遗忘门和输入门合并成了一个新的重置门$z_t$，且加入了一个更新门$r_t$，具体更新方式如下:




LSTM结构图

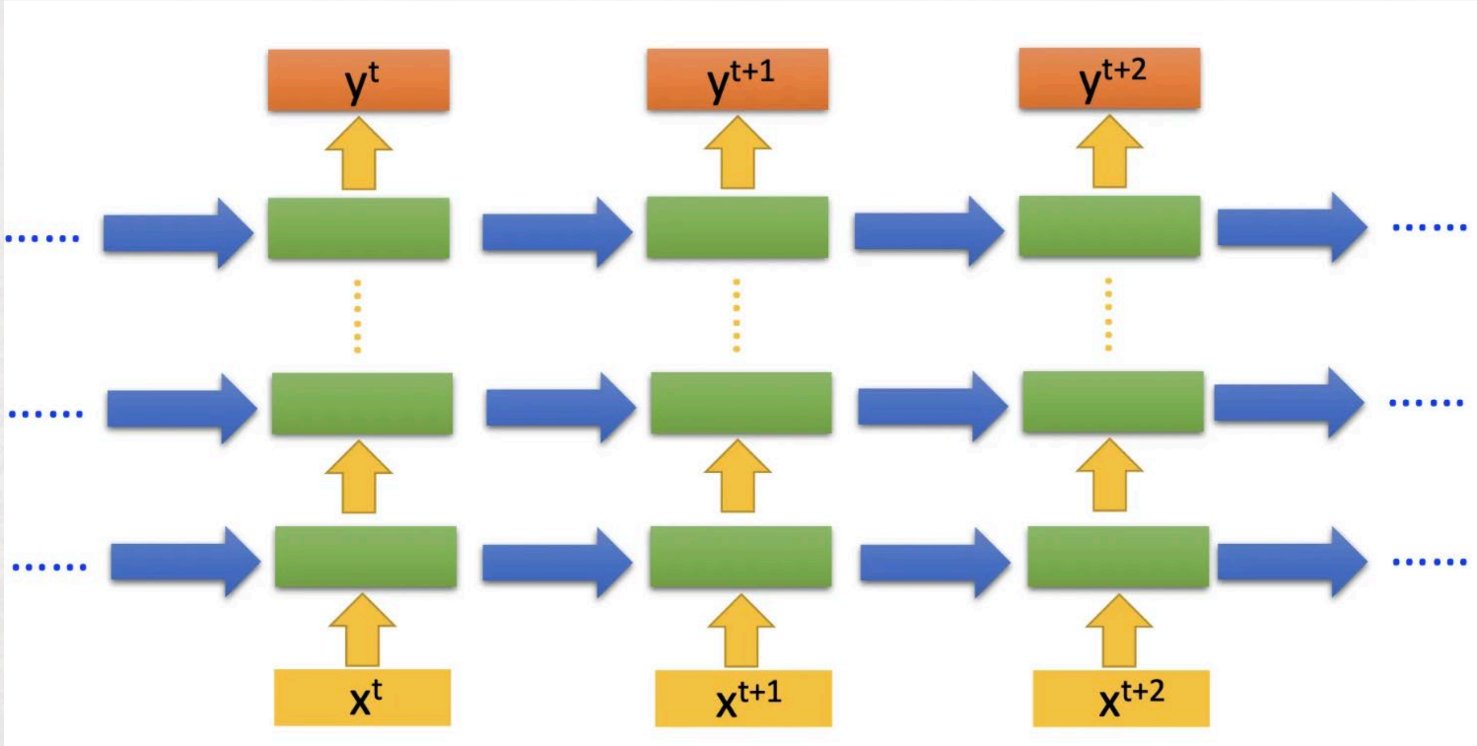



GRU结构图

通过以上的简化，GRU可以达到和LSTM类似的训练效果，同时提高了网络运算的效率，降低了计算成本。

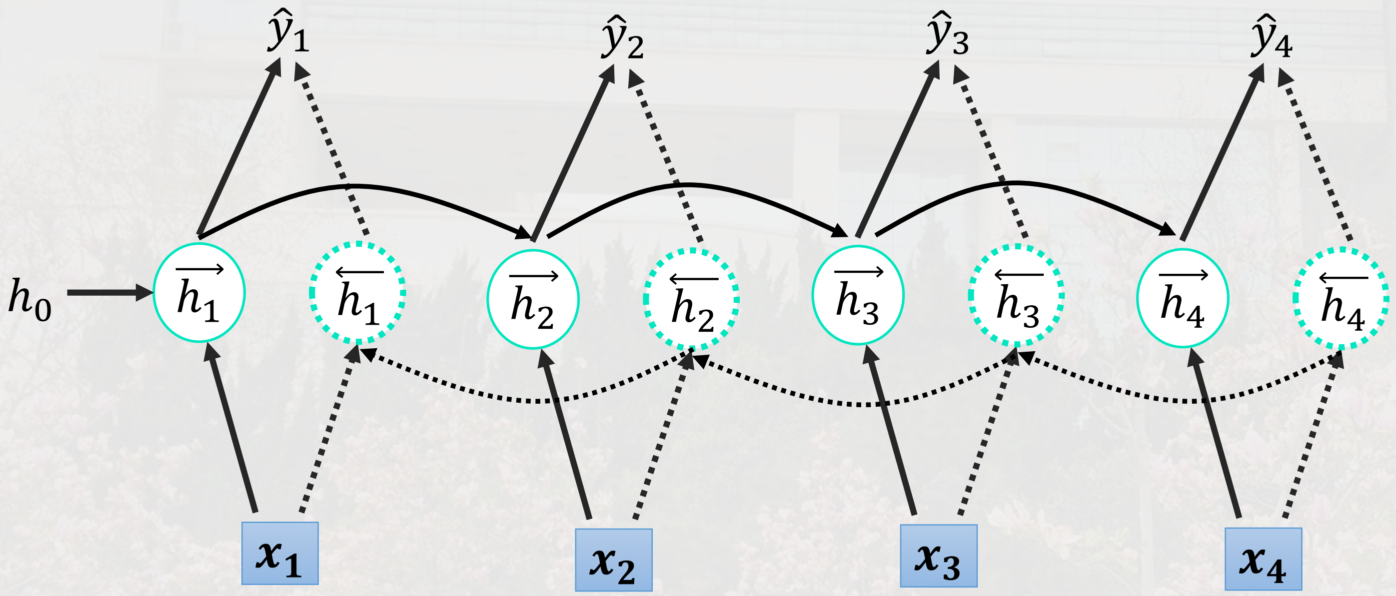
# 4. RNN和LSTM的其他架构变体 –多层RNN

如果将RNN 隐含层中的循环函数用多层全连接神经网络表示，便可得到多层RNN。与标准的RNN网络相同，第一层隐含层的输入是上一时刻该层隐变量和当前时刻的输入的信息融合，而其他隐含层的输入则是上一时刻的隐状态和上一层隐含层在当前时刻的隐状态。相比于简单的RNN，该网具有更强大的表达与学习能力，但是复杂性也随之提高，同时需要更多的训练数据。Deep RNNs的结构如下图所示：



# 4. RNN和LSTM的其他架构变体 –双向RNN/LSTM

单向RNN对于输入的时间序列按照时间顺序从左至右以此进行编码，每个时刻的隐状态的影响主要来自于当前时刻的输入与之前时刻的隐状态。但在一些问题中，当前时刻和之后时刻的信息均会对当前时刻的输出产生作用。因此研究者设计了将两层RNN叠加在一起，构成了双向RNN，双向RNN的隐状态$h_t$由两个方向的编码得到的隐状态组成。与双向RNN (Bidirectional RNN) 类似，Bidirectional LSTM有两层LSTM。



# 总结

- 了解为什么要使用循环神经网络
- 掌握RNN的基本结构
- RNN训练：沿时间的反向传播，梯度爆炸与梯度消失
- RNN的常见变体LSTM和GRU