



河海大学
HOHAI UNIVERSITY

第九章 深度学习的核心算法与应用

目录

Contents

01

循环神经网络

02

LSTM

03

应用案例



01 / PART ONE

循环神经网络

循环神经网络的应用场景比较多，比如写论文，写程序，写诗等。

无论是卷积神经网络，还是BP神经网络，他们的前提假设都是：
元素之间是相互独立的，输入与输出也是独立的。

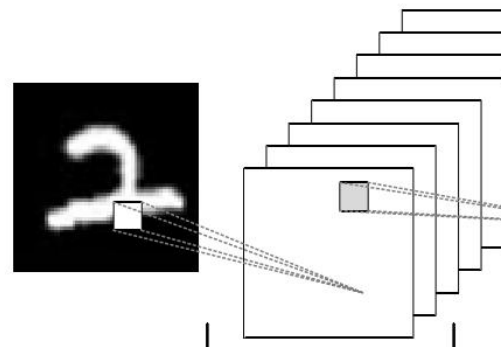
但现实世界中，很多元素都是相互连接的，比如股票随时间的变化，一个人说了：我喜欢旅游，其中最喜欢的地方是云南，以后有机会一定要去_____。人应该都知道是填“云南”。因为我们是根据上下文的内容推断出来的，但计算机要做到这一步就相当得难了。因此，就有了现在的循环神经网络，他的本质是：像人一样拥有记忆的能力。因此，他的输出就依赖于当前的输入和记忆。

图像数据

图像



卷积神经网络



序列数据

语音

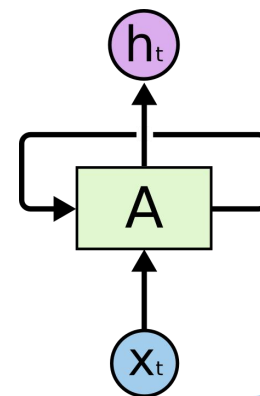


文本

我一把把车
把把住了



循环神经网络





序列数据

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

- 理解文本时，需要处理这些词连接起来的整个序列；
- 处理视频时，要分析所以帧连接起来的整个序列。

为了解决上述类似的问题，能够更好的处理序列的信息，**循环神经网络 (Recurrent Neural Network, RNN)** 就由此诞生了。



- 循环神经网络 (Recurrent neural networks, 简称RNN) 是一种通过隐藏层节点周期性的连接, 来**捕捉序列化数据中动态信息的神经网络**, 可以对序列化的数据进行分类。
- 和其他前向神经网络不同, RNN可以保存一种上下文的状态, 甚至能够在任意长的上下文窗口中存储、学习、表达相关信息, 而且不再局限于传统神经网络在空间上的边界, 可以在**时间序列上有延拓**, 直观上讲, 就是本时间的隐藏层和下一时刻的隐藏层之间的节点间有边。
- RNN广泛应用在和序列有关的场景, 如如一帧帧图像组成的视频, 一个个片段组成的音频, 和一个个词汇组成的句子。

- 《全世界就在那里》(外二首)
- ○小冰
- 河水上滑过一对对盾牌和长矛
- 她不再相信这是人们的天堂
- 眼看着太阳落了下去
- 这时候不必再有爱的诗句
- 全世界就在那里
- 早已拉下了离别的帷幕
- 生命的颜色
- 你双颊上的道理
- 是人们的爱情
- 撒向天空的一个星
- 变幻出生命的颜色
- 我跟着人们跳跃的心
- 太阳也不必再为我迟疑
-

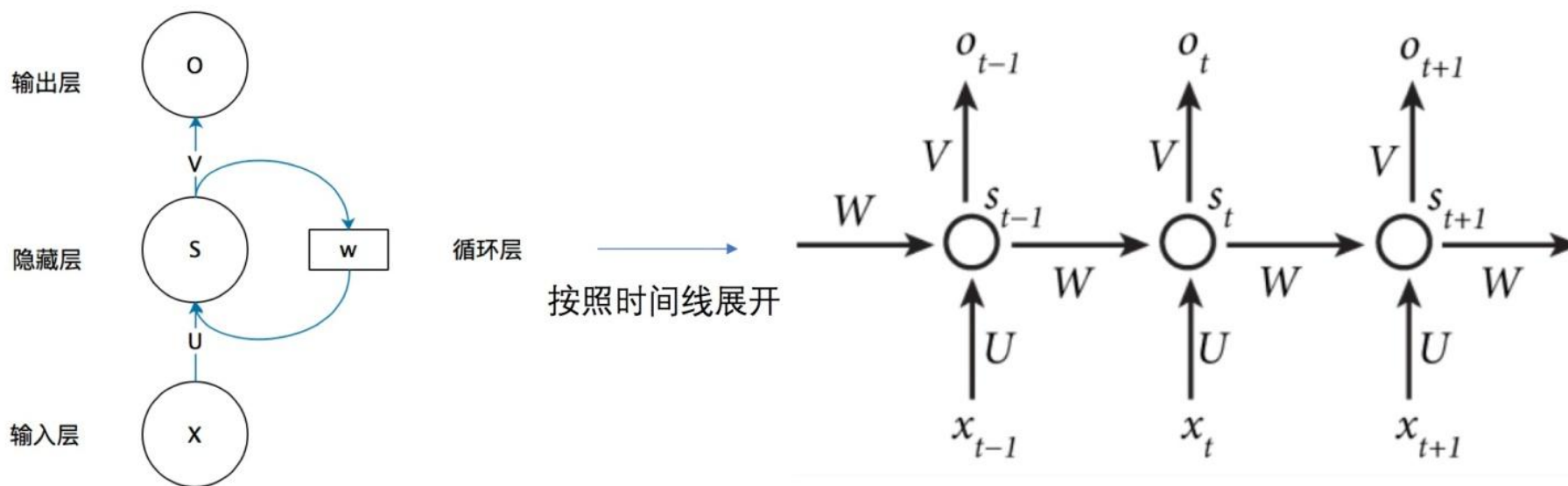
- 记录着生命的凭证
- 像飞在天空没有羁绊的云
- 冰雪后的水
- 那霜雪铺展出的道路
- 是你的声音啊
- 雪花中的一点颜色
- 是开启我生命的象征
- 我的心儿像冰雪后的水
- 一滴一滴翻到最后
- 给我生命的上帝
- 把它吹到缥缈的长空

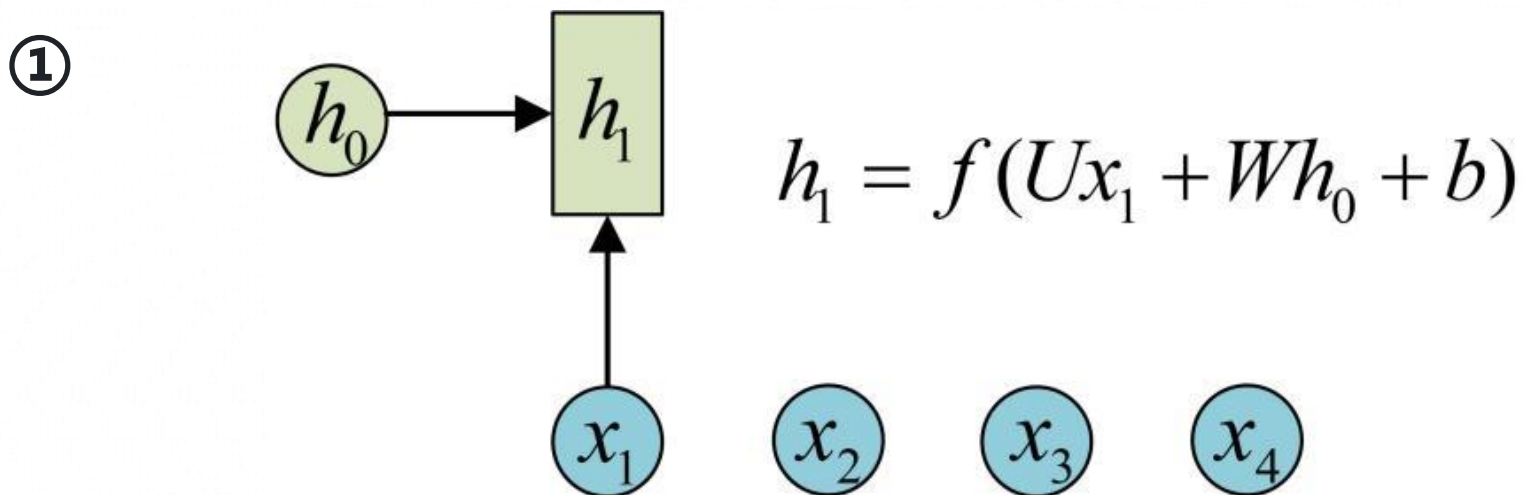
《机智过人》
机心造化本无私，
智略功夫巧笑时。
过客不须频眺望，
人间天上有新诗。

——机器人“九歌”

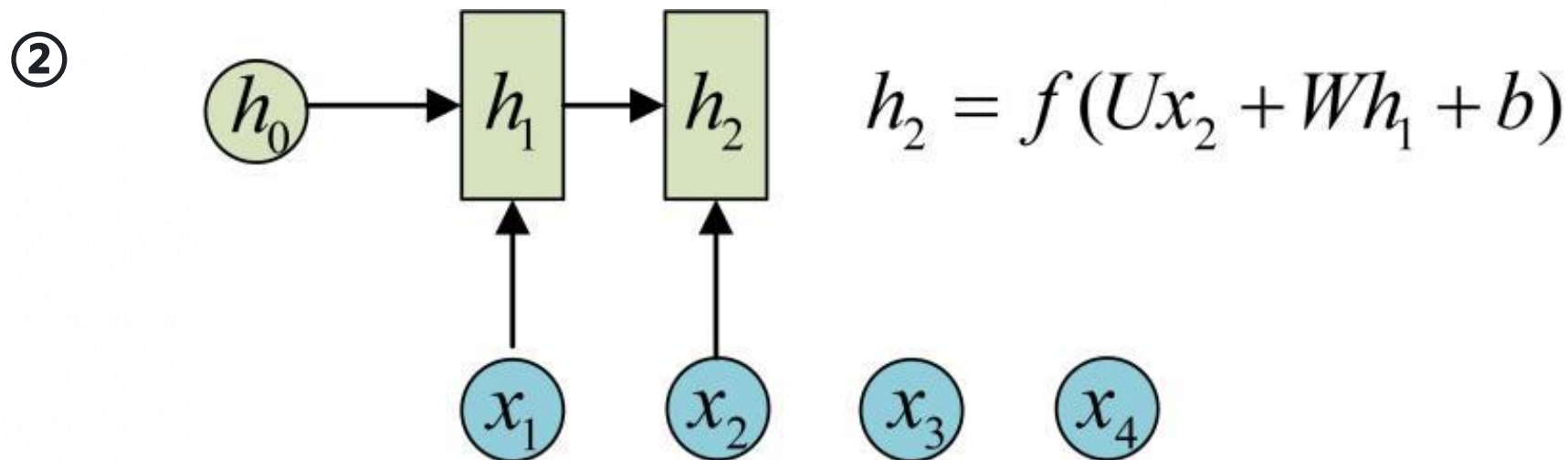
RNN对具有序列特性的数据非常有效，它能挖掘数据中的时序信息以及语义信息。

在语音识别、语言模型、机器翻译以及时序分析领域表现突出。



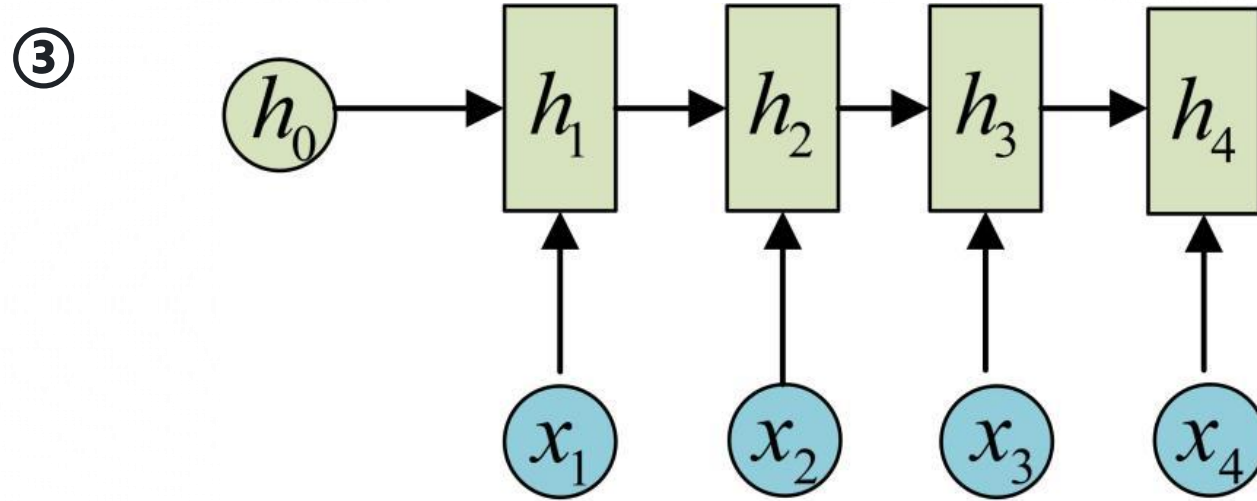


- x_1 、 x_2 、 x_3 为输入的序列;
- RNN引入了隐状态 h (hidden state), h 可对序列数据提取特征, 接着再转换为输出。



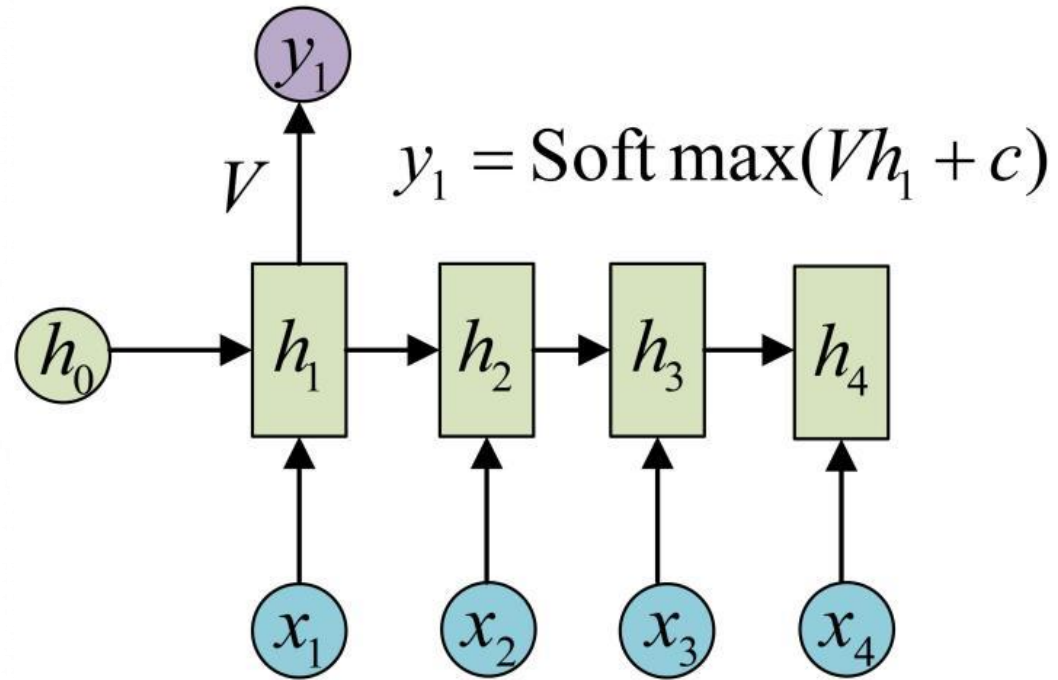
- 每个步骤使用的参数 U , W , b 相同, h_2 的计算方式和 h_1 类似。

1. 模型结构



- 与 h_2 相似，得到隐状态 h_3, h_4 。

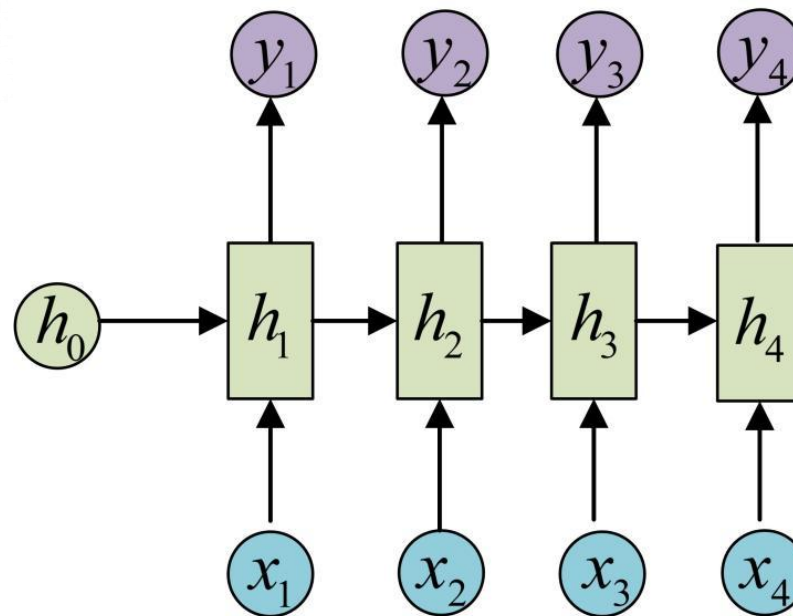
④



- 根据参数 V ，隐状态 h_1 ，得到输出 y_1 。

1. 模型结构

⑤



- 与 y_1 相似, 得到 y_2 - y_4 。

1. 模型结构

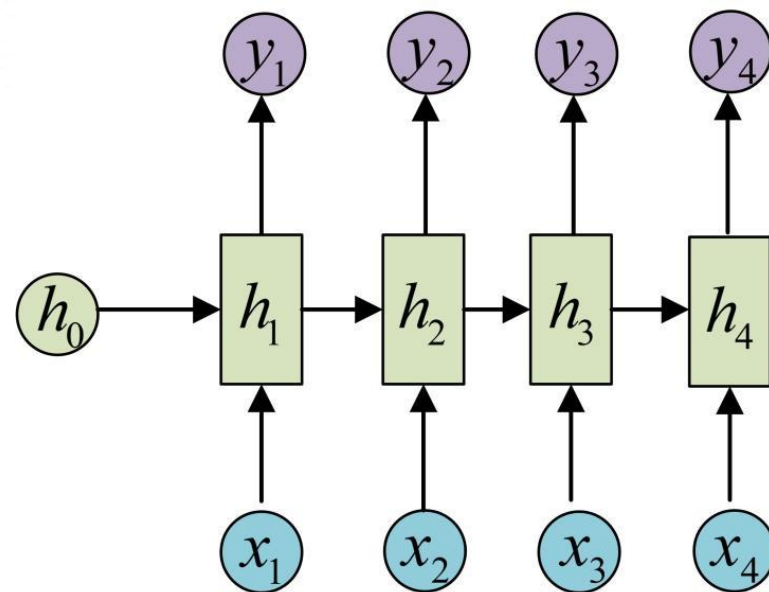
上述为最经典的RNN结构：

- 序列到序列 (sequence-to-sequence)



如何处理其他情况？

- 向量到序列 (vector-to-sequence)
- 序列到向量 (sequence-to-vector)
- 序列不等长 (在实际应用中，我们遇到的大部分问题序列都是不等长的)



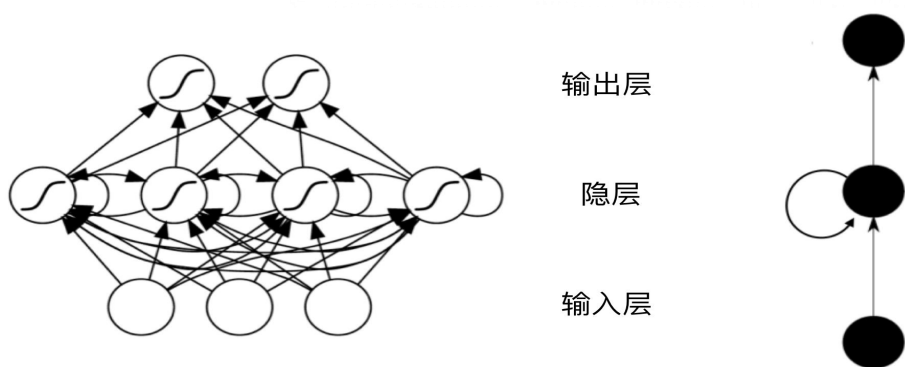
上述为最经典的RNN结构：

- 序列到序列 (sequence-to-sequence)



如何处理其他情况？

- 向量到序列 (vector-to-sequence)
- 序列到向量 (sequence-to-vector)
- 序列不等长 (在实际应用中，我们遇到的大部分问题序列都是不等长的)

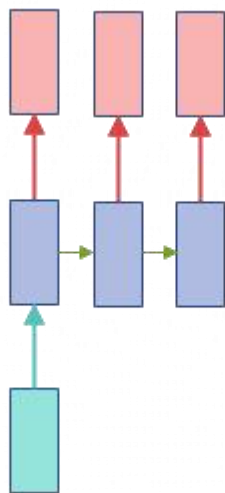


1. 模型结构

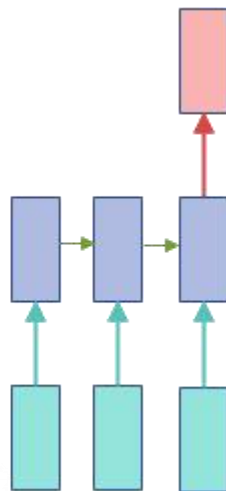
one to one



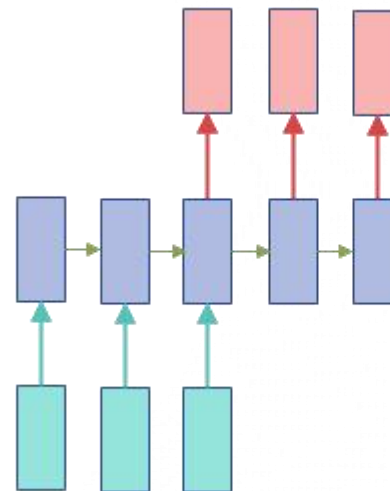
one to many



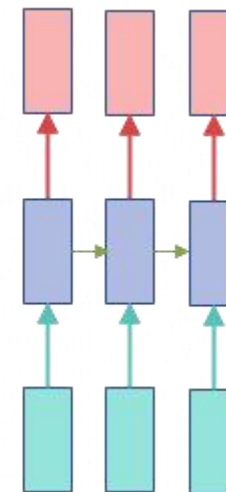
many to one



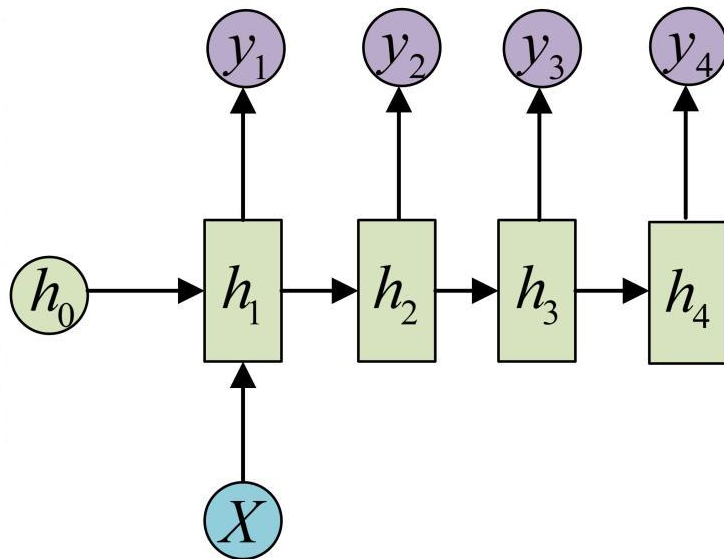
many to many



many to many



向量到序列 (vector-to-sequence) 模型



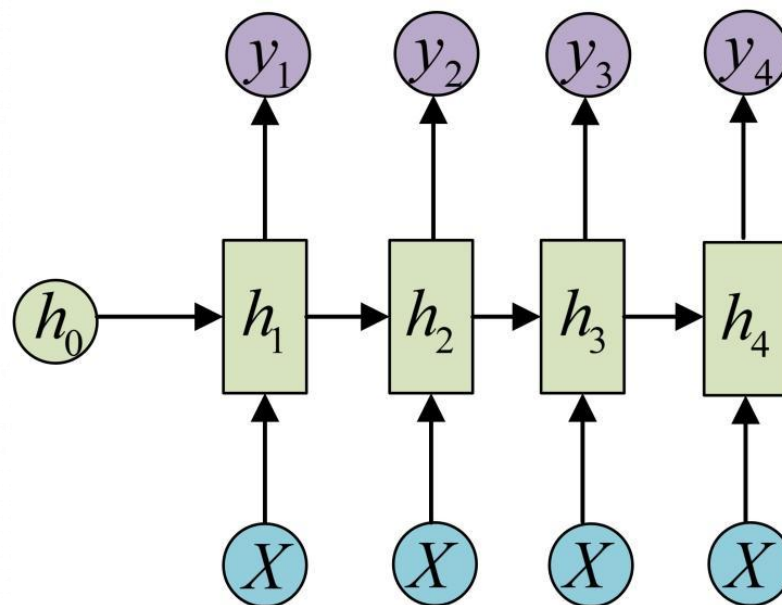
$$h_1 = f(Ux + Wh_0 + b)$$

$$h_N = f(Wh_{N-1} + b)$$

$$y_t = vh_t$$

- 方式1：只在其中的某一个序列进行计算，比如序列第一个进行输入计算。

向量到序列 (vector-to-sequence) 模型

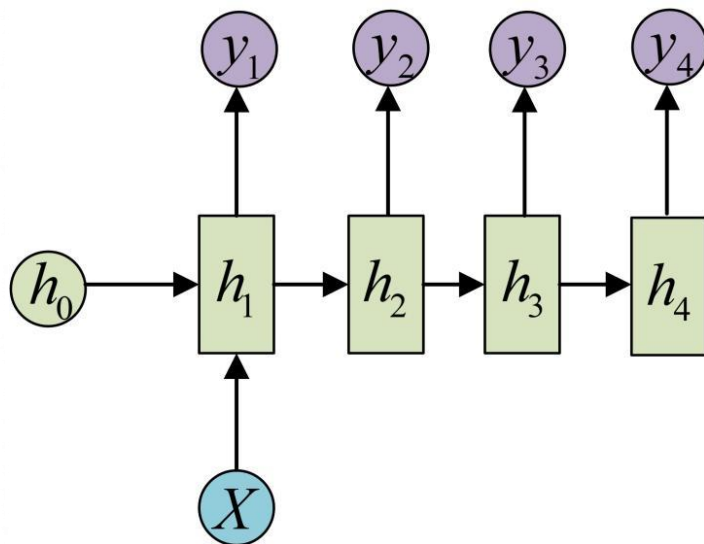


$$\begin{aligned}h_1 &= f(Ux + Wh_0 + b) \\h_N &= f(Ux + Wh_{N-1} + b) \\y_t &= vh_t\end{aligned}$$

- 方式2：把输入信息X作为每个阶段的输入。

向量到序列 (vector-to-sequence) 模型

1 vs N



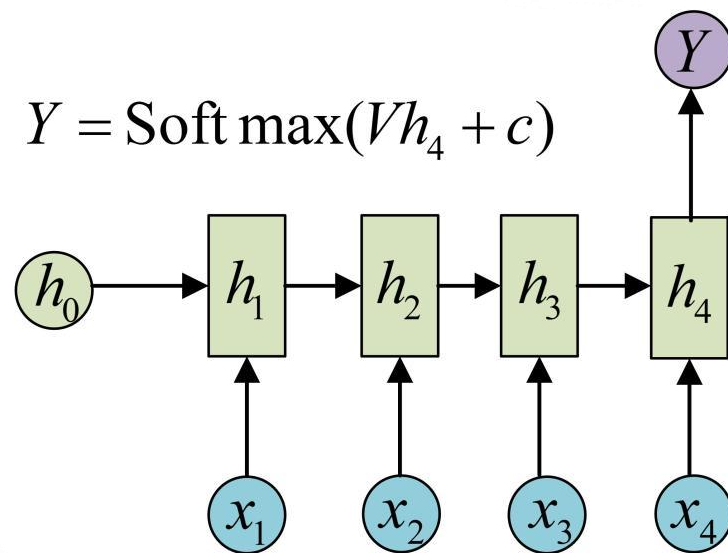
应用场景:

- 从**图像生成文字**, 输入为图像的特征, 输出为一段句子;
- 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。



→ 这是一只猫

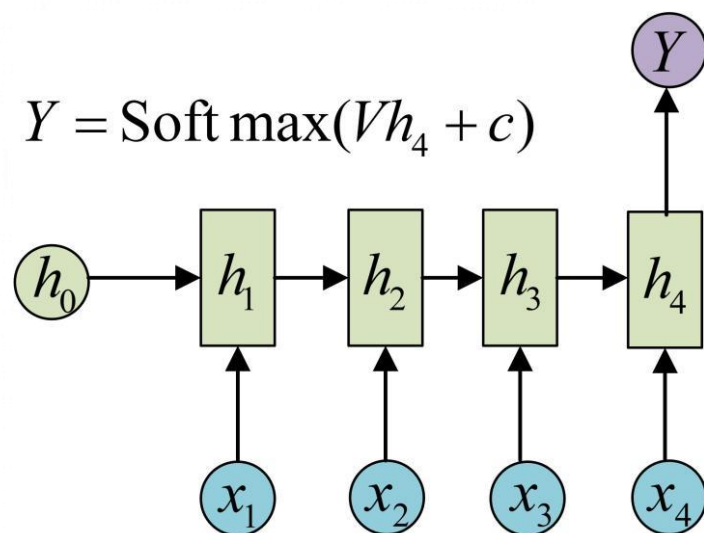
序列到向量 (sequence-to-vector) 模型



- 在最后一个序列上进行输出变换。

序列到向量 (sequence-to-vector) 模型

N vs 1



应用场景:

- 输出一段文字, 判断**所属类别**;
- 输入一个句子, 判断**情感倾向**;
- 输入一段视频, 判断**所属类别**。

2. RNN的训练



时序反向传播(BPTT):

传统反向传播(BP)在时间序列上的拓展

在 t 时序记忆单元的误差有两个来源, 分别是 t 时序误差 c_t 对记忆单元的偏微分和记忆单元在下一个时序 $t+1$ 的误差对 t 时序记忆单元的偏微分, 需要将两者相加

时序越长, 最后一个时序的损失对第一个时序中 w 的梯度越容易出现梯度消失或梯度爆炸现象

某个权重 w 的梯度是该权重在所有时序上梯度的累加

BPTT的三个步骤:

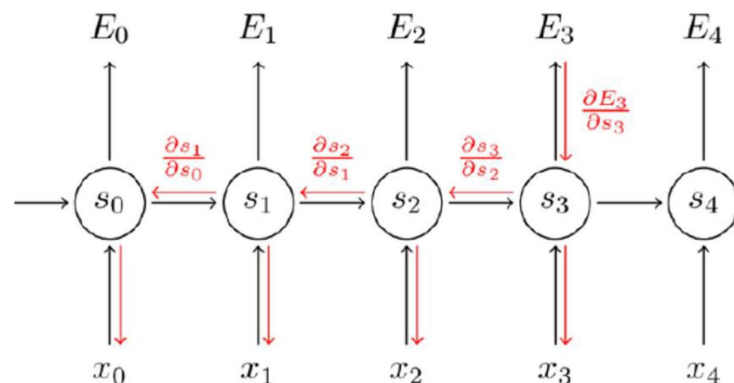
前向计算每个神经元的输出值

反向计算每个神经元的误差值 δ_j

计算每个权重的梯度

最后使用随机梯度下降算法更新权重。

2. RNN的训练



- 在每个时间节点 $t = 0, 1, 2, 3, 4$ 神经网络的输出都会产生误差值: E_0, E_1, E_2, E_3, E_4 。与前馈神经网络类似, RNN也使用反向传播梯度下降法更新权重。

$$s_t = \tanh(Ux_t + Ws_{t-1})$$

$$\hat{y}_t = \text{softmax}(Vs_t)$$

$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$

$$E(y, \hat{y}) = \sum_t E_t(y_t, \hat{y}_t) = -\sum_t y_t \log \hat{y}_t$$

$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial W}$$

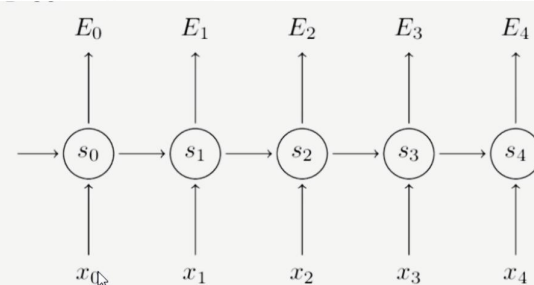
链式法则

$$s_t = \tanh(Ux_t + Ws_{t-1})$$

$$s_3 = \tanh(Ux_3 + Ws_2)$$

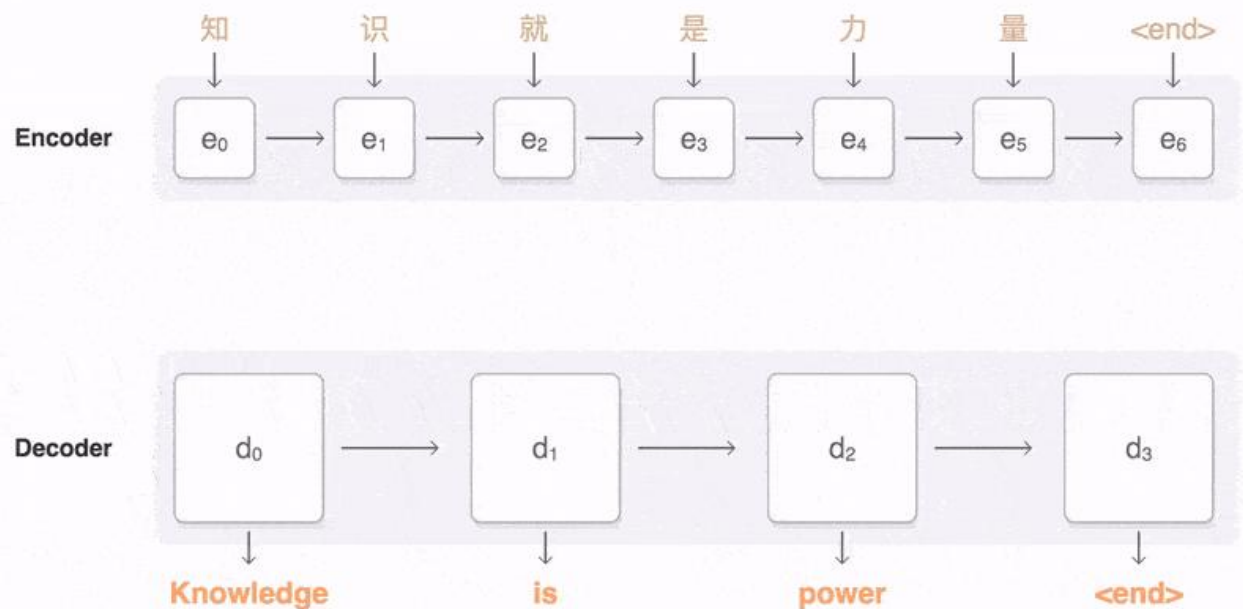
$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^3 \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W}$$

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^3 \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \left(\prod_{j=k+1}^3 \frac{\partial s_j}{\partial s_{j-1}} \right) \frac{\partial s_k}{\partial W}$$



3. 编码器-解码器结构

Transformer出现之前，主流的序列转换模型都基于复杂的循环神经网络（RNN），包含编码器和解码器两部分。当时表现最好的模型还通过注意力机制将编码器和解码器连接起来。



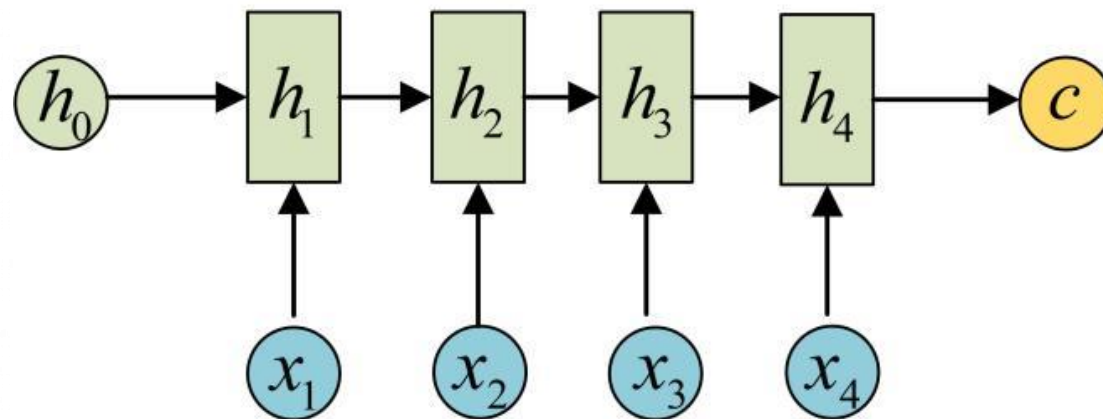
3. 编码器-解码器结构

不同编码方式

$$(1) \ c = h_4$$

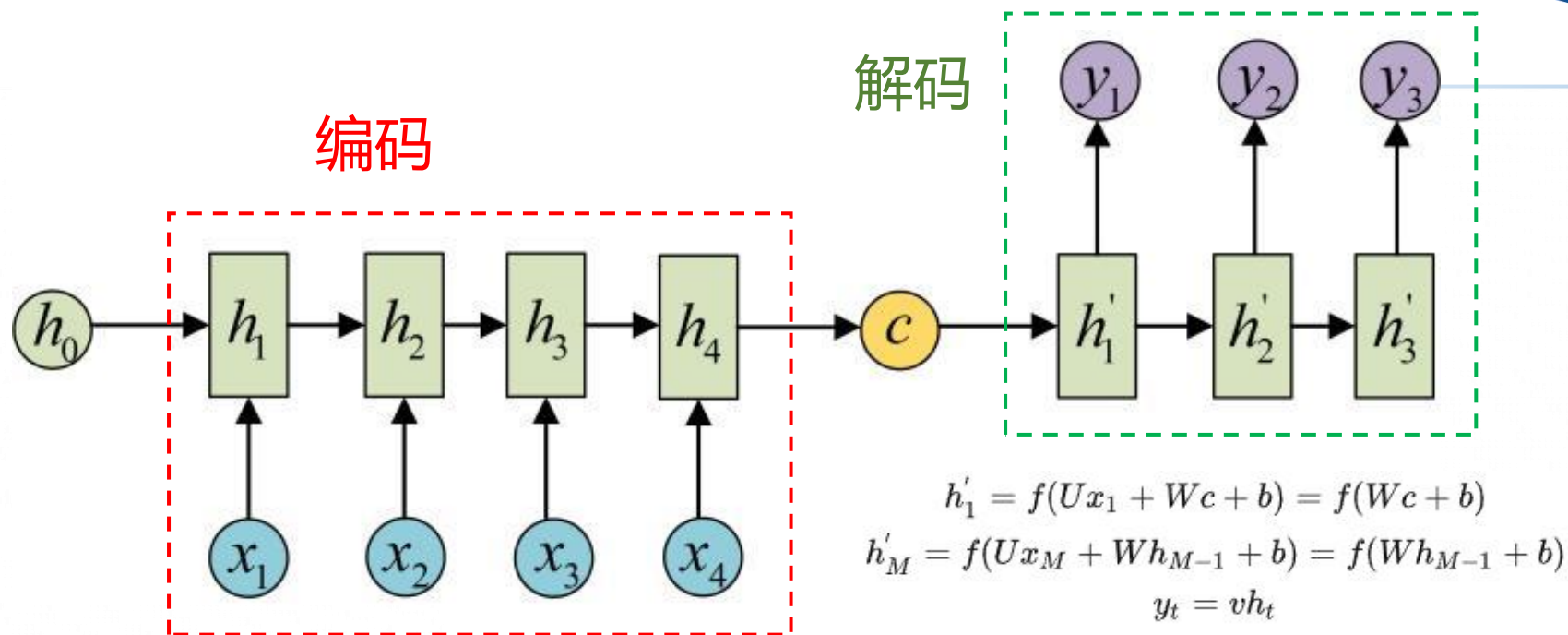
$$(2) \ c = q(h_4)$$

$$(3) \ c = q(h_1, h_2, h_3, h_4)$$



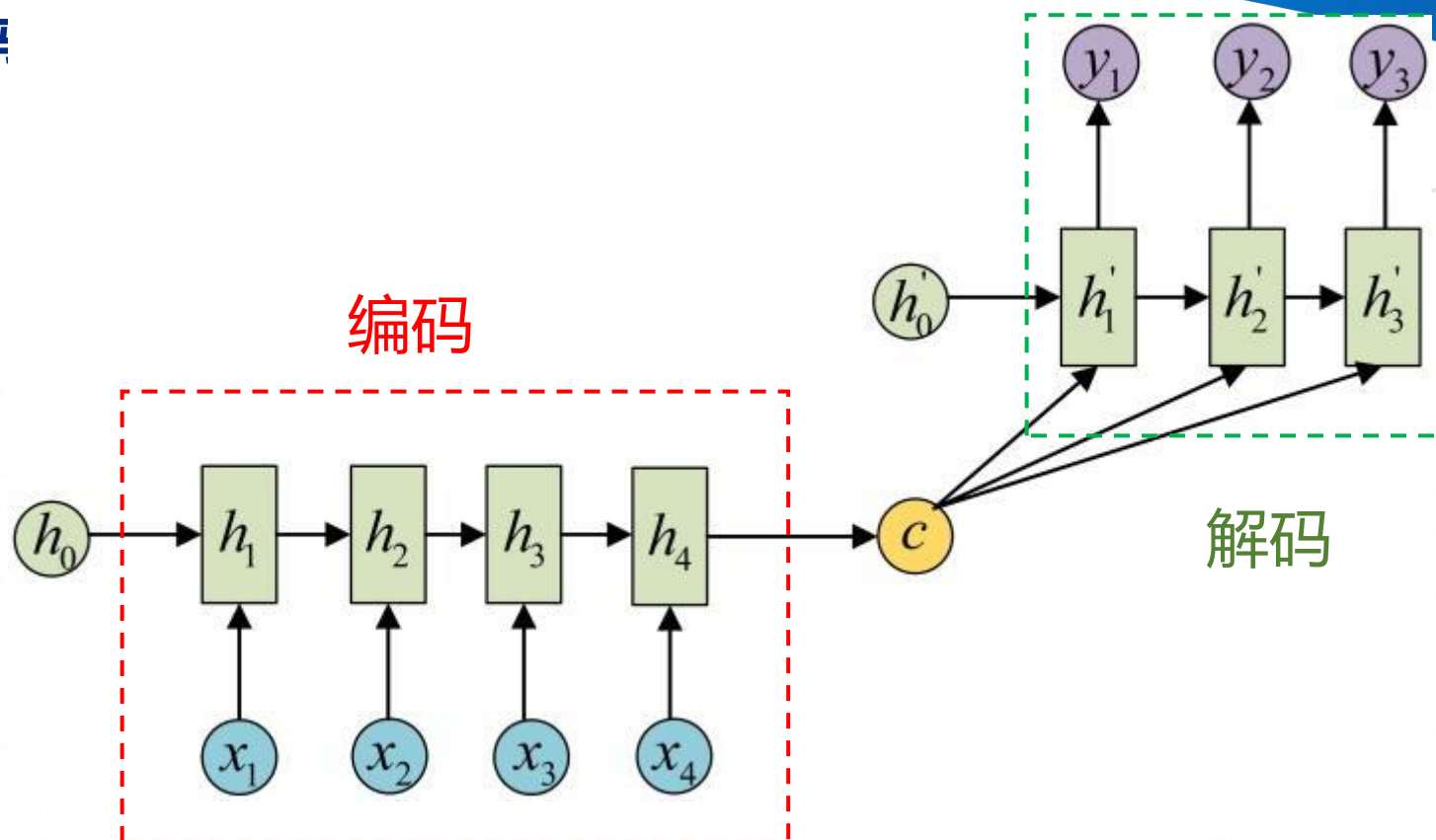
- 将数据编码成一个上下文向量 c ，该部分称为 E_{encoder} ；
- 有多种方式：(1)把最后一个隐状态赋值给 c ；(2)对最后的隐状态做一个变换；(3)对所有的隐状态做变换。

3. 编码器-解码器结构



- 用另一个RNN网络（称为 D_{decoder} ）对其进行解码；
- 方式1：将上下文向量 c 作为初始状态输入到 D_{decoder} 。

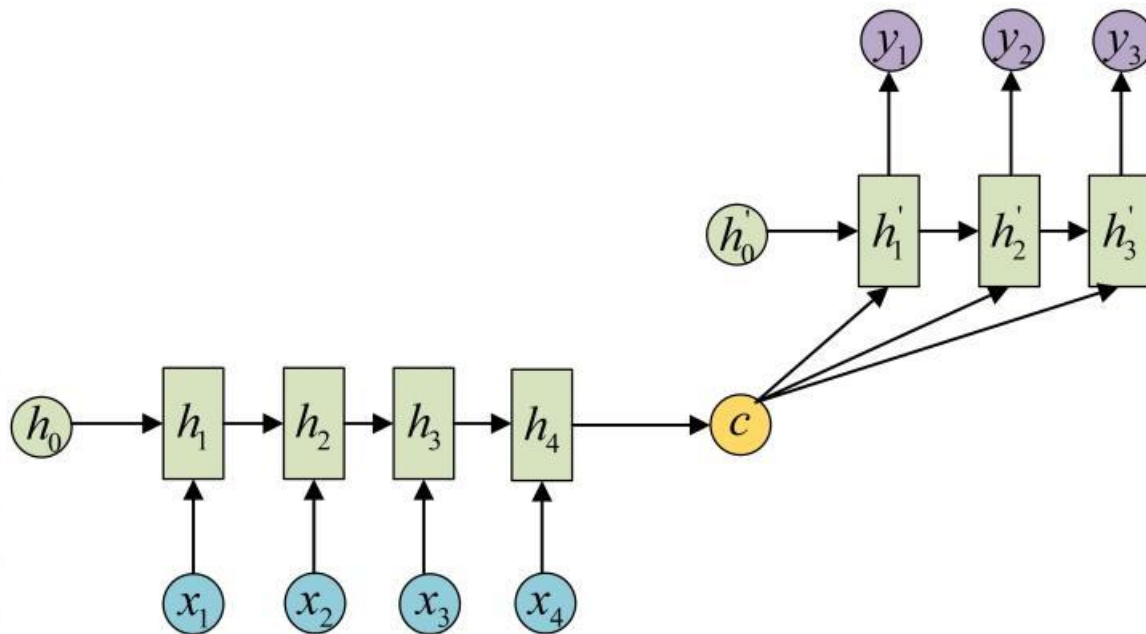
3. 编码器-解码



- 用另一个RNN网络（称为 D_{ecoder} ）对其进行解码；
- 方式2：将 c 作为 D_{ecoder} 的每一步输入。

3. 编码器-解码器结构

N vs M



应用场景:

- **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;
- **文本摘要**: 输入文本序列, 输出这段文本序列摘要;
- **阅读理解**: 输入文章, 输出问题答案;
- **语音识别**: 输入语音序列信息, 输出文字序列。

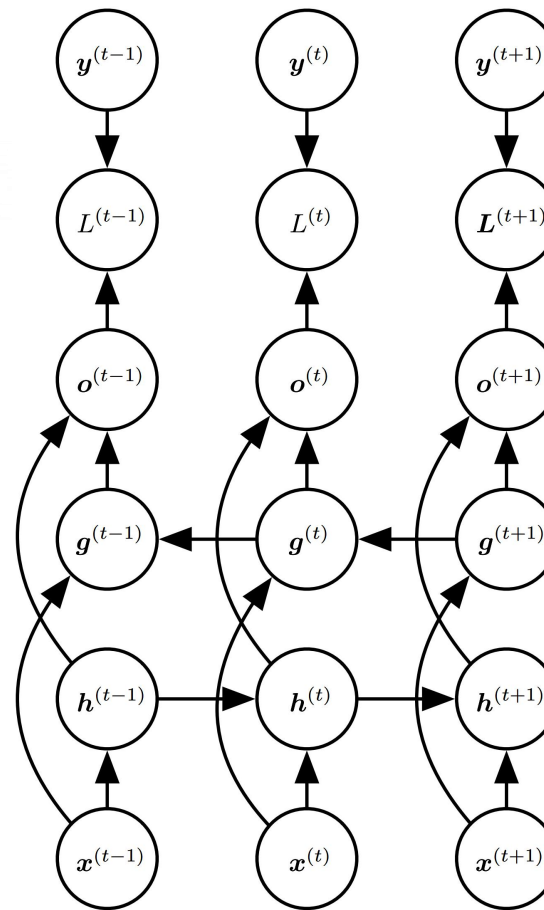
3. 编码器-解码器结构

双向RNN:

预测可能依赖于整个输入序列。

应用举例:

- 语音识别: 存在协同发音, 对于当前音素的解释可能依赖未来的一些单词。



在Encoder-Decoder结构中，Encoder把所有的输入序列都编码成一个**统一**的**语义特征c**再解码。

因此，c中必须包含原始序列中的所有信息，**c的长度**就成了限制模型性能的**瓶颈**。



以机器翻译为例，当要翻译的句子较长时，一个c可能存不下那么多信息，就会造成翻译精度的下降。



Attention机制通过在每个时间输入不同的c来解决此问题。

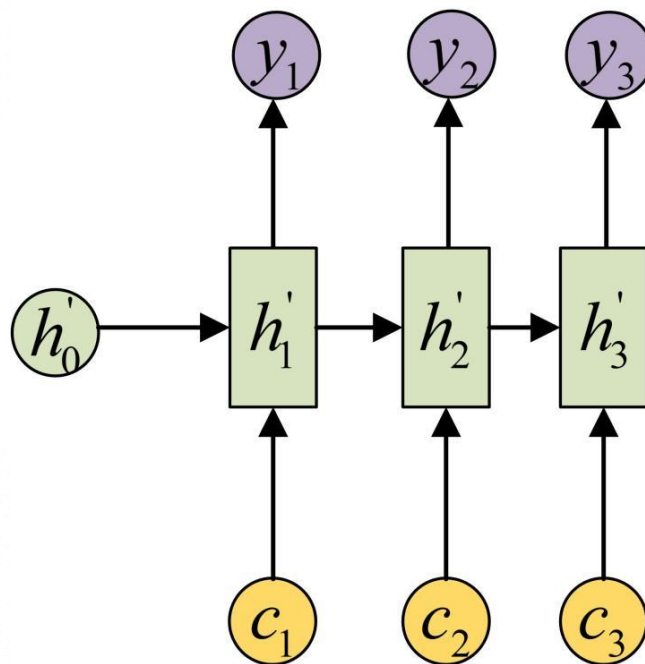
4. 注意力机制

Attention机制：
用a衡量Encoder中h在解码时的相关性，a从模型中学习得到。

我		爱		中		国												
h_1	$*$	a_{11}	$+$	h_2	$*$	a_{12}	$+$	h_3	$*$	a_{13}	$+$	h_4	$*$	a_{14}	$=$	c_1	$\longrightarrow$	I
h_1	$*$	a_{21}	$+$	h_2	$*$	a_{22}	$+$	h_3	$*$	a_{23}	$+$	h_4	$*$	a_{24}	$=$	c_2	$\longrightarrow$	Love
h_1	$*$	a_{31}		h_2	$*$	a_{32}	$+$	h_3	$*$	a_{33}	$+$	h_4	$*$	a_{34}	$=$	c_3	$\longrightarrow$	China

4. 注意力机制

Decoder中，每个c会自动选择与当前输出最匹配的上下文信息。



- RNN主要用于**处理序列数据**。

对于传统神经网络模型，从输入层到隐含层再到输出层，层与层之间一般为全连接，每层之间神经元是无连接的。但是传统神经网络无法处理数据间的前后关联问题。

例如，为了预测句子的下一个单词，一般需要该词之前的语义信息。这是因为一个句子中前后单词是存在语义联系的。

- RNN中当前单元的**输出与之前步骤输出也有关**。

具体的表现形式为当前单元会对之前步骤信息进行储存，并应用于当前输出的计算中。

隐藏层当前输出由当前时刻输入向量和之前时刻隐藏层状态共同决定。

RNN vs CNN

类别	特点描述
相同点	1. 传统神经网络的扩展。 2. 前向计算产生结果，反向计算模型更新。 3. 每层神经网络横向可以多个神经元共存，纵向可以有多个神经网络连接。
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 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



02 / PART ONE

L

S

T

M

- RNN存在“**长期依赖**”问题：
当神经网络的节点经过许多阶段的计算后，之前比较长的时间片的特征已经被覆盖，即**难以记住先前的信息**。
- RNN在处理长期依赖（时间序列上距离较远的节点）时会造成**梯度消失或者梯度膨胀**的现象。

计算中包含一条反复与矩阵 W 相乘的路径。那么 t 步后，相当于乘以 W^t ，特征值分解得：

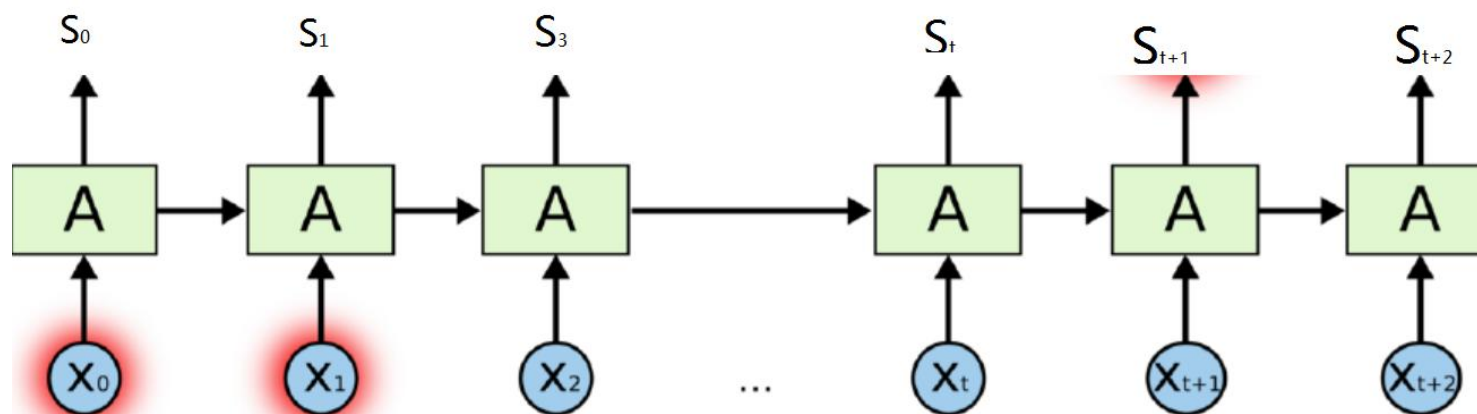
$$W^t = (V \text{diag}(\lambda) V^{-1})^t = V \text{diag}(\lambda)^t V^{-1}$$

当特征值 λ 不在 1 附近时，若在量级上大于 1 则会爆炸；若小于 1 时则会消失。

1. LSTM

梯度消失

$$1.01^{365} = 37.8$$
$$0.99^{365} = 0.03$$



“I grew up in France... I speak fluent French”

RNN缺陷：长期依赖(Long Term Dependencies)问题，产生长跨度依赖（短时记忆）的问题。

梯度爆炸

- 当梯度很大时，可以采用梯度截断的方法，将梯度约束在一个范围内。

$$w_1 = w_1 - \alpha \frac{\partial J(\mathbf{w})}{\partial w_1}$$
$$w_2 = w_2 - \alpha \frac{\partial J(\mathbf{w})}{\partial w_2}$$

让 $g_1 = \frac{\partial J(\mathbf{w})}{\partial w_1}$, $g_2 = \frac{\partial J(\mathbf{w})}{\partial w_2}$, 设定裁剪阈值为 c , $\|\mathbf{g}\|_2 = \sqrt{g_1^2 + g_2^2}$ 。

当 $\|\mathbf{g}\|_2$ 大于 c 时:

$$\mathbf{g} = \frac{c}{\|\mathbf{g}\|_2} \cdot \mathbf{g}$$

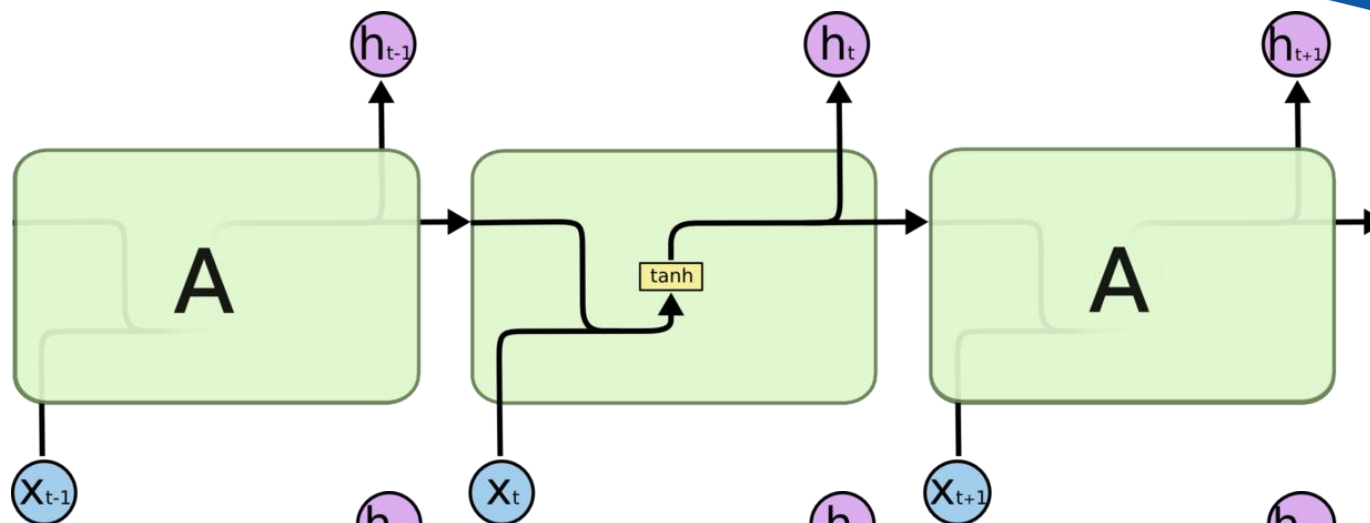
当 $\|\mathbf{g}\|_2$ 小于等于 c 时: $\mathbf{g}$ 不变。

- “长期依赖” 问题;
- 梯度消失或者梯度膨胀。

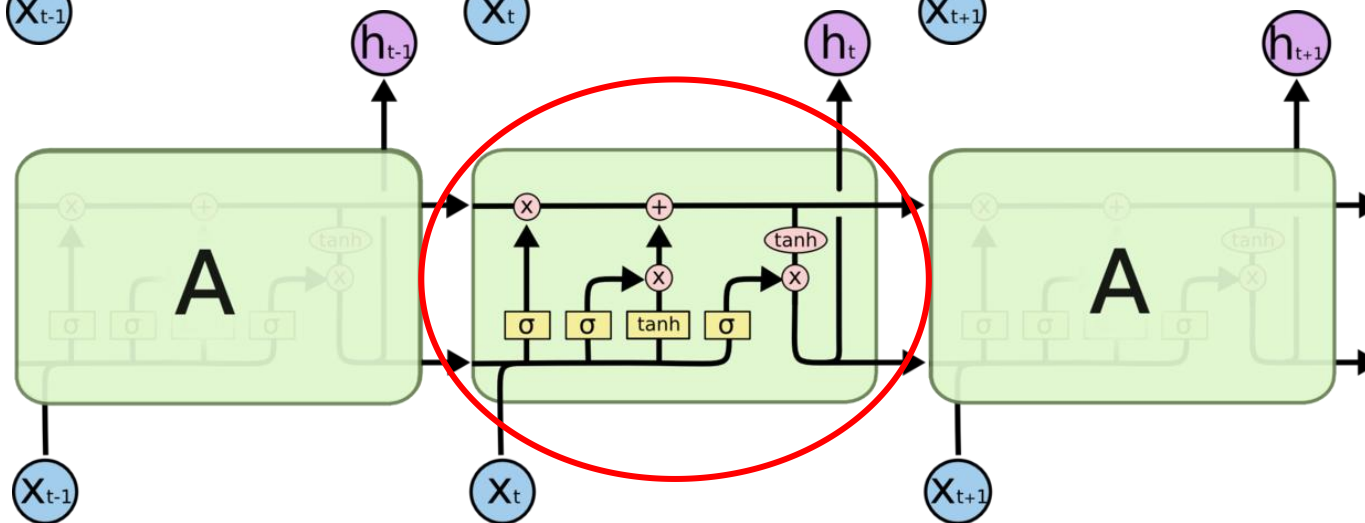
为了解决上述问题，研究人员提出了许多解决办法，其中最具有代表性的是Long Short-Term Memory (LSTM) 长短期记忆网络。

1. LSTM

RNN



LSTM



Neural Network
Layer



Pointwise
Operation



Vector
Transfer



Concatenate



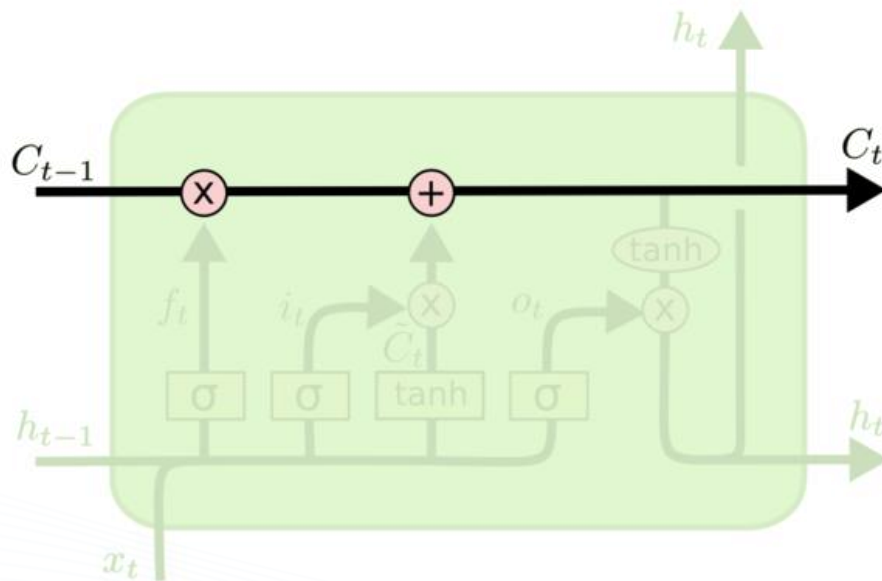
Copy

1. LSTM

LSTM的关键就是**细胞 (cell) 状态**，水平线在图上方贯穿运行。细胞状态类似于传送带。直接在整个链上运行，只有一些少量的线性交互。**信息在上面流传保持不变会很容易。**

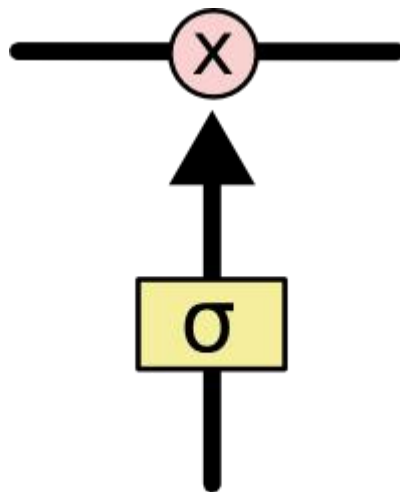


记忆长期信息



LSTM有通过精心设计的称作为“门”的结构来去除或者增加信息到细胞状态的能力。

门是一种让信息选择式通过的方法。他们包含一个 sigmoid 神经网络层和一个 pointwise 乘法操作。

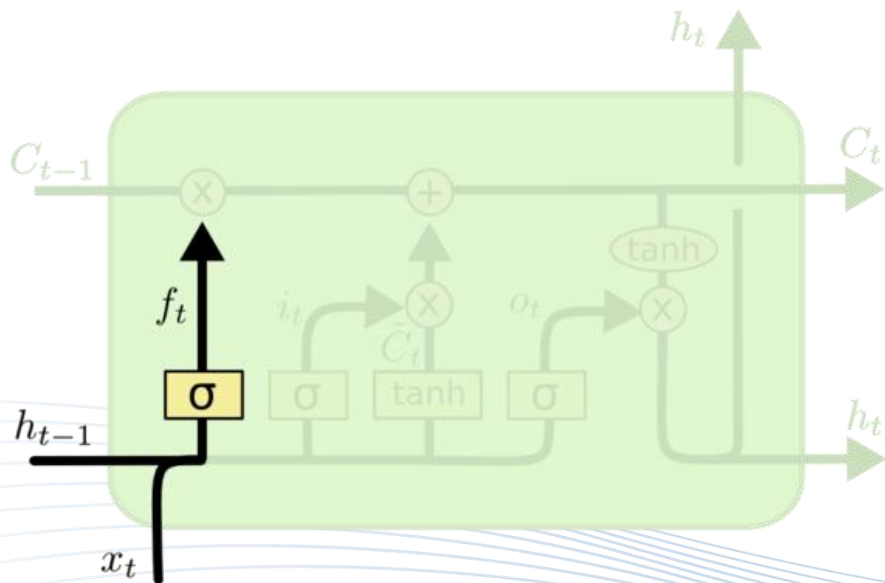


遗忘门

作用对象：细胞状态。

作用：将细胞状态中的信息选择性的遗忘。

操作步骤：该门会读取 h 和 x ，输出一个在 0 到 1 之间的数值给每个在细胞状态 C 中的数字。1 表示“完全保留”，0 表示“完全舍弃”。



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

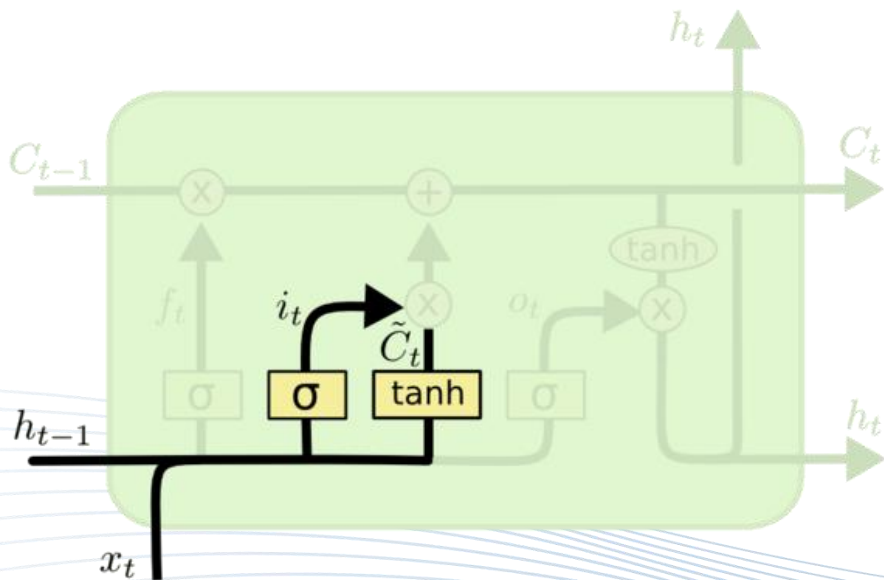
输入门

作用对象：细胞状态。

作用：将新的信息选择性的记录到细胞状态中。

操作步骤：

1. sigmoid层作为“输入门层”，决定什么值将要更新；
2. tanh层创建一个新的候选值向量 $\tilde{C}_t$ 加入到状态中。

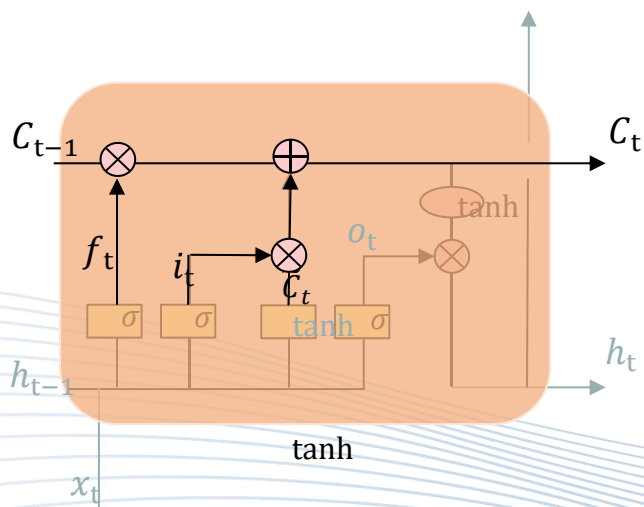


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

更新信息

- 更新旧细胞状态的时间: C_{t-1} 更新为 C_t 。
- 我们把旧状态与 f_t 相乘, 丢弃掉我们确定需要丢弃的信息。接着加上 $i_t * C_t$ 。这就是新的候选值, 根据我们决定更新每个状态的程度进行变化。



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

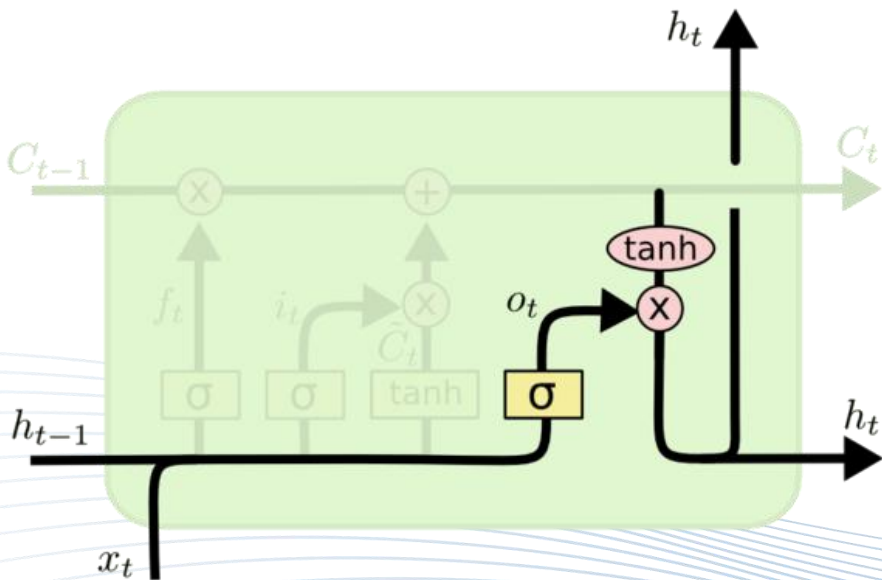
输出门

作用对象：隐层。

作用：确定输出什么值。

操作步骤：

1. 通过sigmoid层来**确定**细胞状态的**哪个部分将输出**；
2. 把细胞状态通过 tanh 进行处理，并将它和 sigmoid 门的输出相乘，最终仅仅会输出确定输出的那部分。



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

$$h_t = o_t * \tanh(C_t)$$

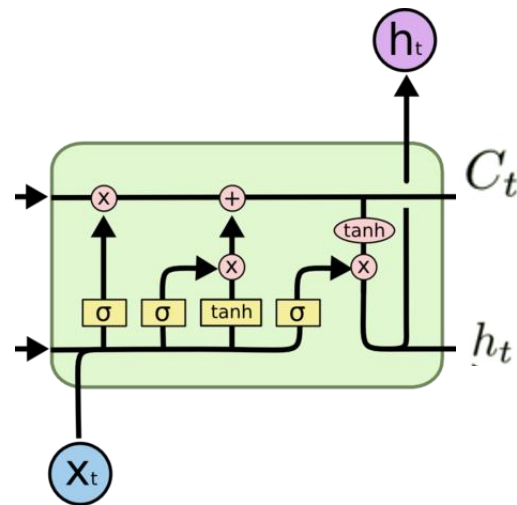
小结

1. 过程:

可以类比为人的记忆容量有限，每个时刻遗忘掉一些旧的知识，再输入（记住）一些新的知识，最后根据需要输出一些知识。遗忘、记忆和输出由门控制。

2. C和h的理解。

- 上面的细胞C代表了长时记忆；
- 下面的 h 代表了工作记忆或短时记忆。
- 长期状态C由Tanh函数激活，然后通过输出门后得到短期状态h；
- 短期状态h用来输入到全连接层计算模型输出。





03 / PART ONE

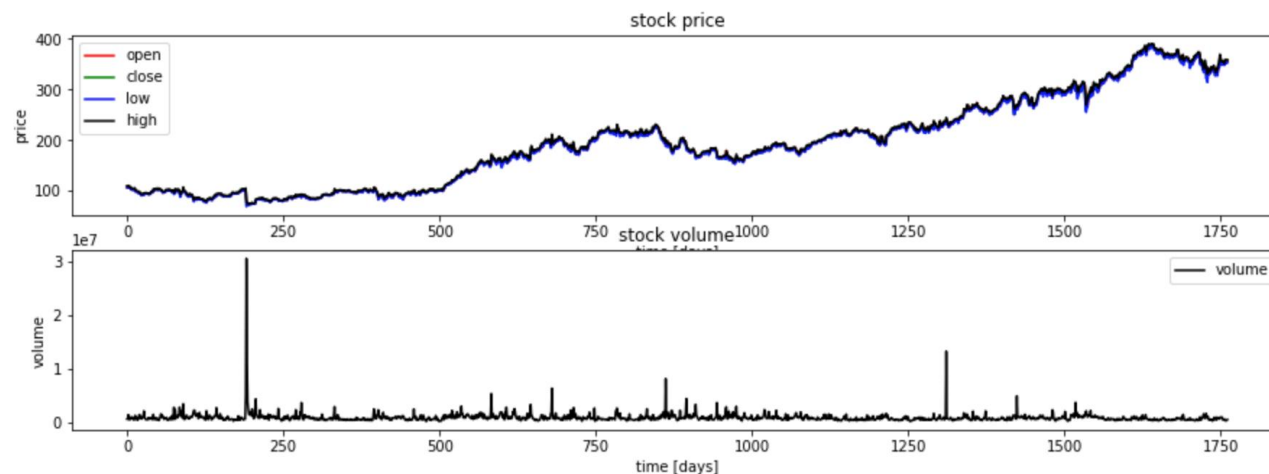
项目案例实践

1. 基于LSTM的股票预测

							open	close	low	high	volume	
							count	851264.000000	851264.000000	851264.000000	851264.000000	8.512640e+05
							mean	64.993618	65.011913	64.336541	65.639748	5.415113e+06
							std	75.203893	75.201216	74.459518	75.906861	1.249468e+07
date	symbol	open	close	low	high	volume	min	1.660000	1.590000	1.500000	1.810000	0.000000e+00
2016-12-30	ZBH	103.309998	103.199997	102.849998	103.930000	973800.0	25%	31.270000	31.292776	30.940001	31.620001	1.221500e+06
2016-12-30	ZION	43.070000	43.040001	42.689999	43.310001	1938100.0	50%	48.459999	48.480000	47.970001	48.959999	2.476250e+06
2016-12-30	ZTS	53.639999	53.529999	53.270000	53.740002	1701200.0	75%	75.120003	75.139999	74.400002	75.849998	5.222500e+06
2016-12-30	AIV	44.730000	45.450001	44.410000	45.590000	1380900.0	max	1584.439941	1578.130005	1549.939941	1600.930054	8.596434e+08
2016-12-30	FTV	54.200001	53.630001	53.389999	54.480000	705100.0						

指标数据:

- 开盘
- 收盘
- 最低
- 最高
- 成交量



1. 基于LSTM的股票预测

```
# min-max 归一化
```

```
def normalize_data(df):
```

```
    min_max_scaler = sklearn.preprocessing.MinMaxScaler()
```

```
    df['open'] = min_max_scaler.fit_transform(df.open.values.reshape(-1,1))
```

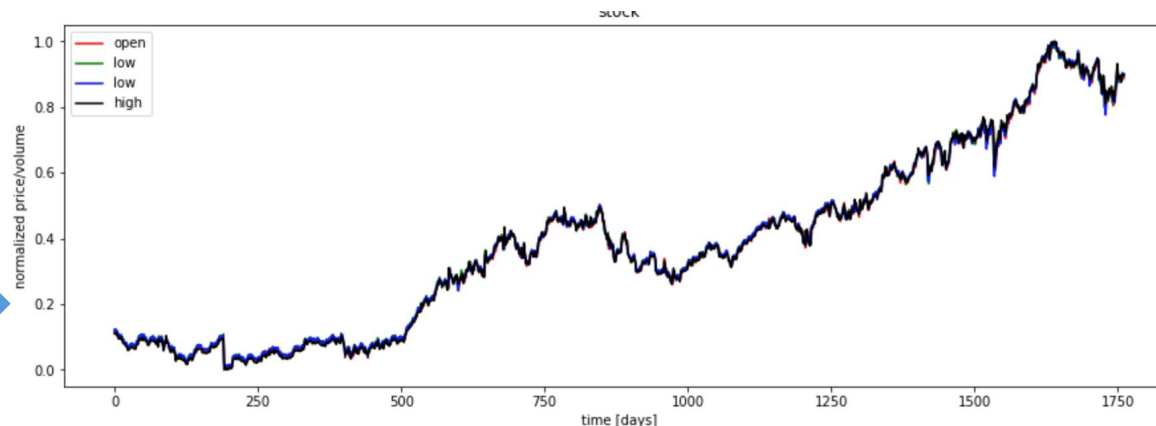
```
    df['high'] = min_max_scaler.fit_transform(df.high.values.reshape(-1,1))
```

```
    df['low'] = min_max_scaler.fit_transform(df.low.values.reshape(-1,1))
```

```
    df['close'] = min_max_scaler.fit_transform(df['close'].values.reshape(-1,1))
```

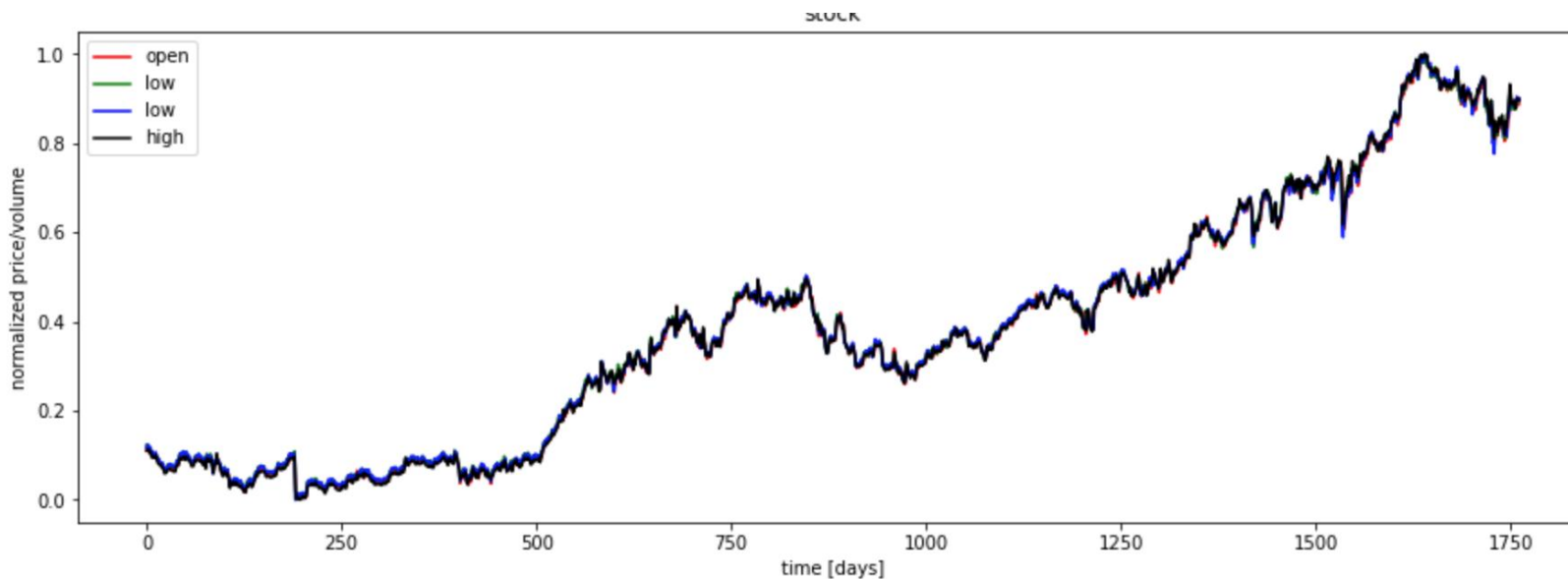
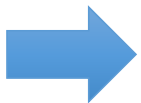
```
    return df
```

数据预处理（归一化）
之后结果：



1. 基于LSTM的股票预测

加窗
(window)



样本结果标记：窗口后涨跌作为样本标记, 即窗口后的交易结果作为预测结果

模型采用MSE作为损失函数, 对预测结果进行评价

股票走势转化为回归问题进行预测

1. 基于LSTM的股票预测

```
X = tf.placeholder(tf.float32, [None, n_steps, n_inputs])
y = tf.placeholder(tf.float32, [None, n_outputs])

layers = [tf.contrib.rnn.GRUCell(num_units=n_neurons, activation=tf.nn.leaky_relu)
          for layer in range(n_layers)]

multi_layer_cell = tf.contrib.rnn.MultiRNNCell(layers)
rnn_outputs, states = tf.nn.dynamic_rnn(multi_layer_cell, X, dtype=tf.float32)

stacked_rnn_outputs = tf.reshape(rnn_outputs, [-1, n_neurons])
stacked_outputs = tf.layers.dense(stacked_rnn_outputs, n_outputs)
outputs = tf.reshape(stacked_outputs, [-1, n_steps, n_outputs])
outputs = outputs[:,n_steps-1,:] # keep only last output of sequence

loss = tf.reduce_mean(tf.square(outputs - y)) # loss function = mean squared error
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate)
training_op = optimizer.minimize(loss)
```

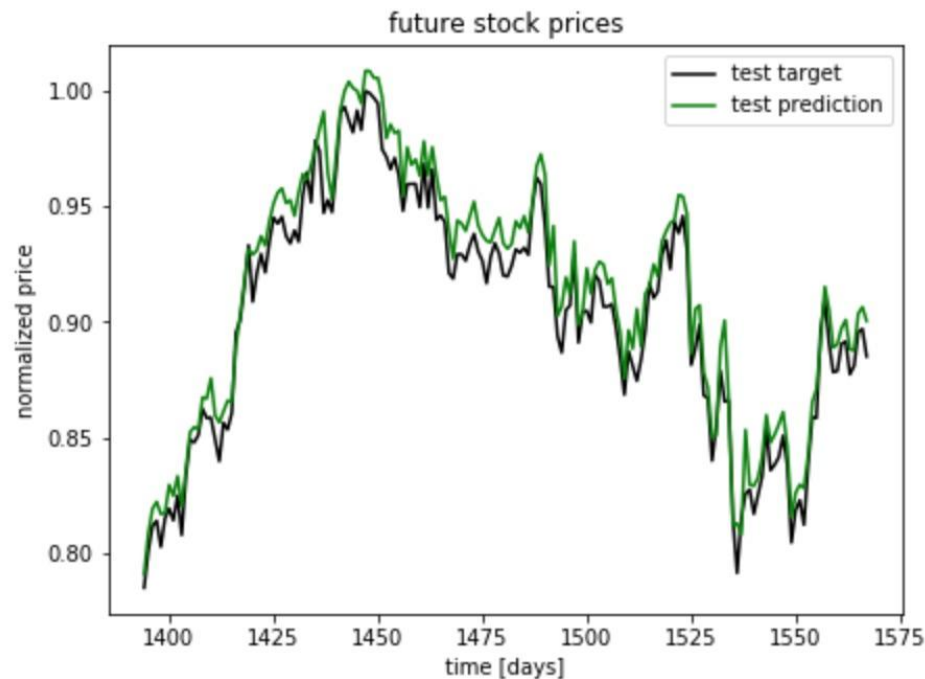
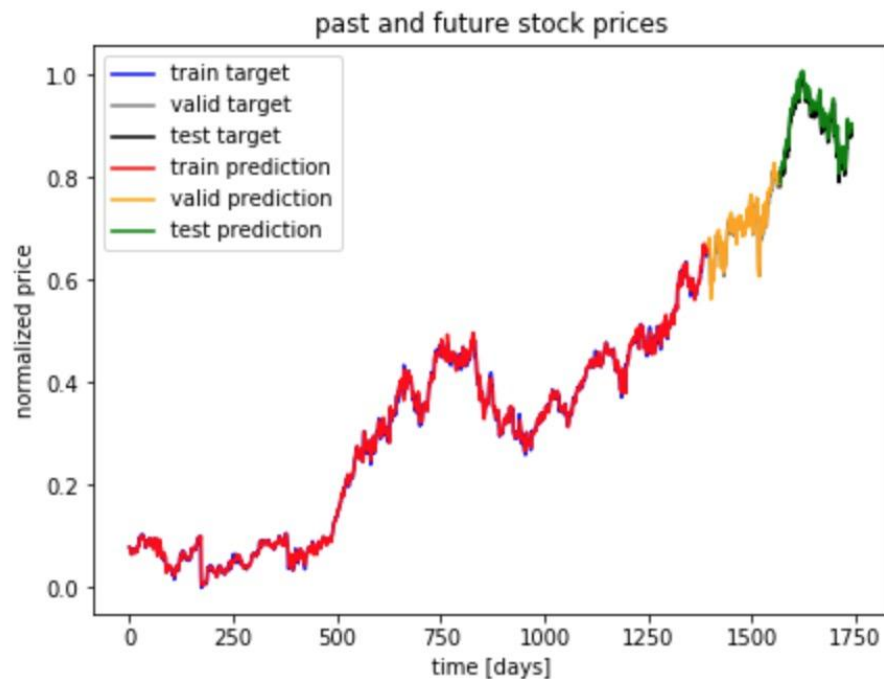
2层GRU (n_layers=2)

损失函数MSE

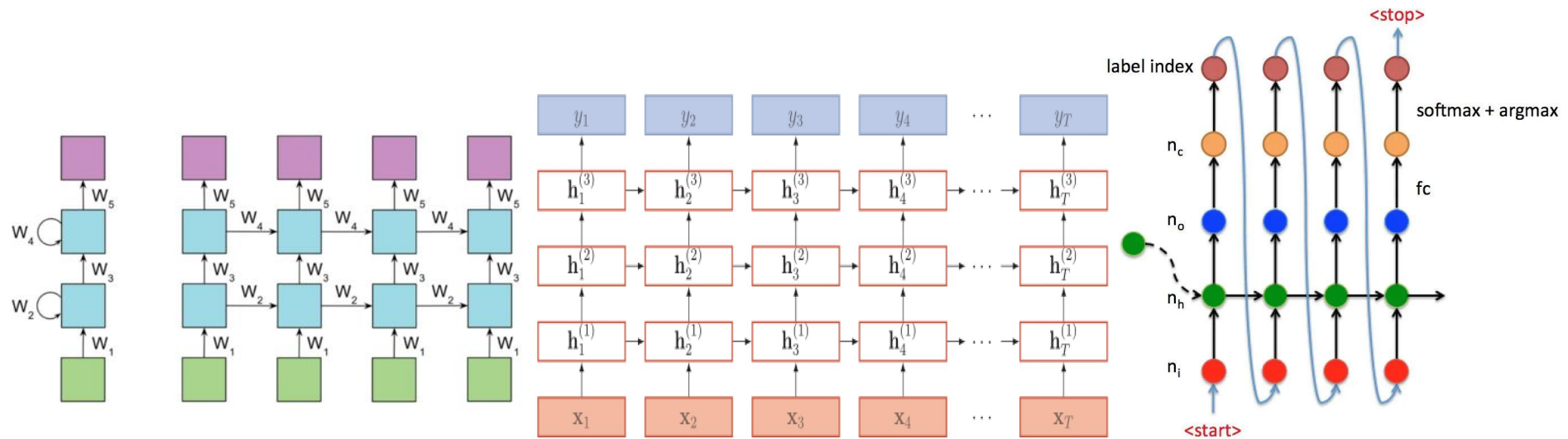
1. 基于LSTM的股票预测

❑ 训练epoch次数：10

❑ 训练集：1300+ 验证集：170+ 测试集170+

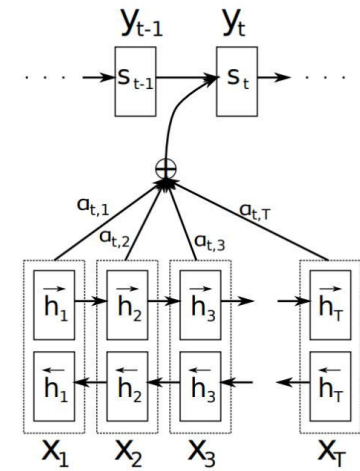
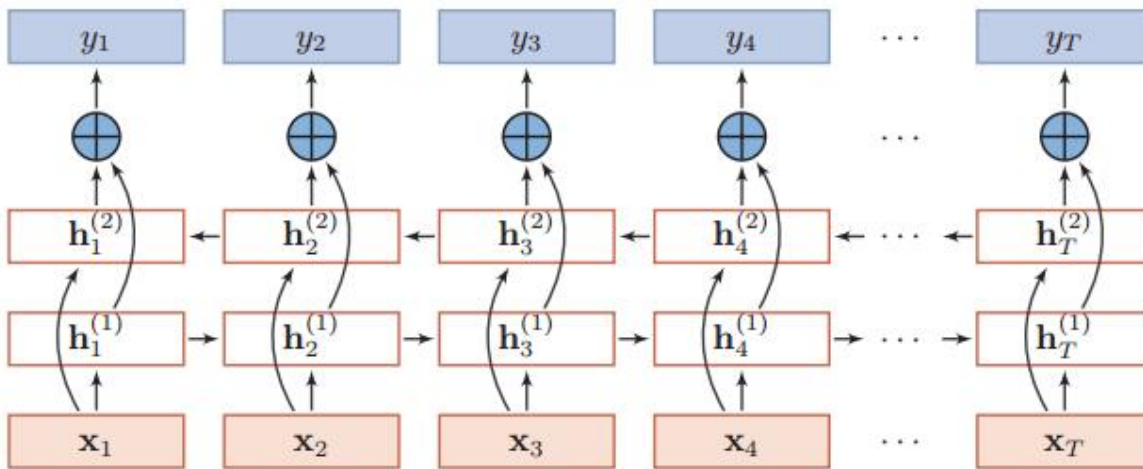
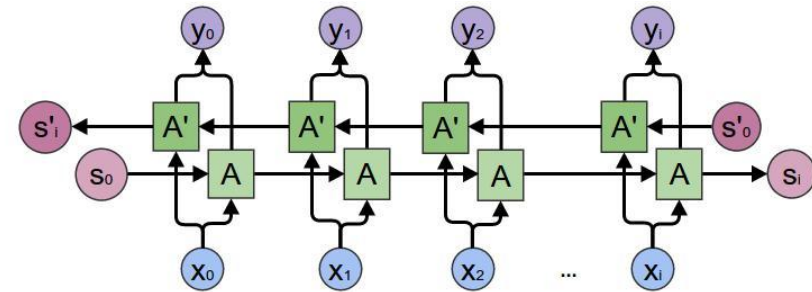


多隐层LSTM(堆叠)



2. 模型变体

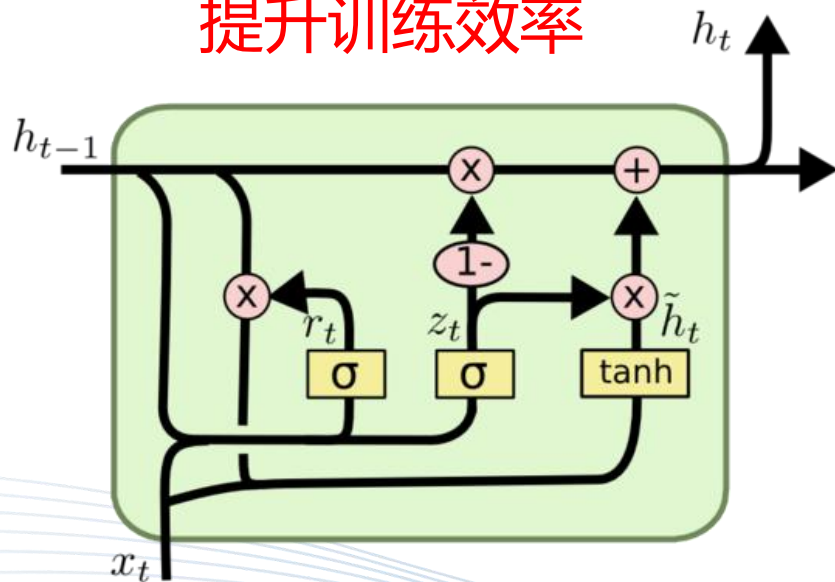
双向循环LSTM



Gated Recurrent Unit (GRU)

- 将忘记门和输入门合成了一个单一的更新门;
- 混合了细胞状态C和隐藏状态h;
- 是最流行的LSTM变体。

简化模型
提升训练效率



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

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

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

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

3. LSTM在语音识别中的应用



一、端到端语音识别系统

1. Deep Speech系列（如百度、微软）：

- 采用LSTM与CTC（Connectionist Temporal Classification）结合，直接学习音频频谱（如MFCC或梅尔谱）到文本的映射。
- 例：百度Deep Speech 2使用双向LSTM堆叠结构，在噪声环境下实现高鲁棒性识别。

2. 混合模型应用：

- 将CNN与LSTM结合，先用CNN提取语音频谱的局部特征（如频域信息），再用LSTM建模时序关系。

二、语音唤醒与关键词检测

1. 智能设备唤醒：

- 华为HiAI引擎采用轻量化LSTM模型，通过检测特定唤醒词（如"小艺小艺"）触发设备。
- 技术特点：采用低功耗设计，支持离线运行，LSTM层处理实时音频流检测。

三、说话人识别与验证

1. 声纹特征建模：

- 阿里云声纹识别系统使用LSTM提取说话人特征向量（如x-vector），通过时序建模捕捉发音习惯差异。
- 应用场景：电话银行身份验证，错误率低于0.5%。

四、多模态融合应用

1. 唇语识别辅助系统:

- 牛津大学研究团队将LSTM用于唇部运动视频与语音信号的跨模态对齐，提升嘈杂环境下的识别率。

五、前沿技术演进

1. Transformer-LSTM混合架构:

- 谷歌最新语音识别系统采用self-attention机制与LSTM结合，在长语音转录任务中减少15%的CER（字符错误率）。



河海大学
HOHAI UNIVERSITY

谢 谢!