Lecture6 Recurrent Neural Network

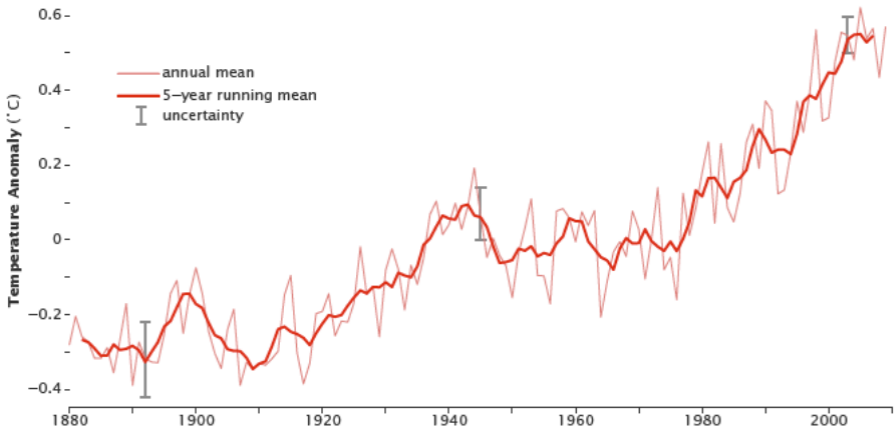
1. 通过序列学习

- 很多数据都是以序列的形式出现的
- 文本和语音等

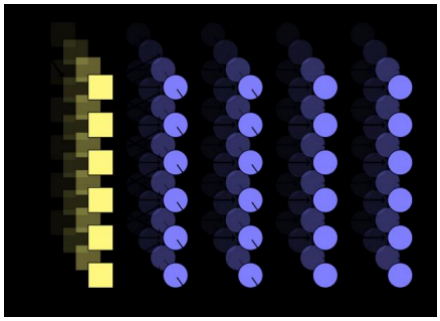
什么是循环神经网络 RNN

循环神经网络（RNN）是一类人工神经网络，其中节点之间的连接形成一个有向图沿时间序列。这允许它显示时间动态行为，与前馈神经网络不同，RNN 可以使用其内部状态（内存）来处理输入序列，这使得它们适用于无分割的、连接的手写识别或语音识别等任务

- 天气预测

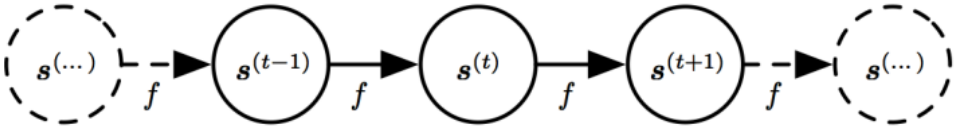


- 音乐创作



与图像一样，我们需要考虑结构（通常是时间依赖性），但要注意，在序列中，未来和过去不是“对称”处理的（但是可以是双向的）

此外，我们可以有**很长**的**时间依赖性**和**任意长度的输入/输出**，甚至无限长



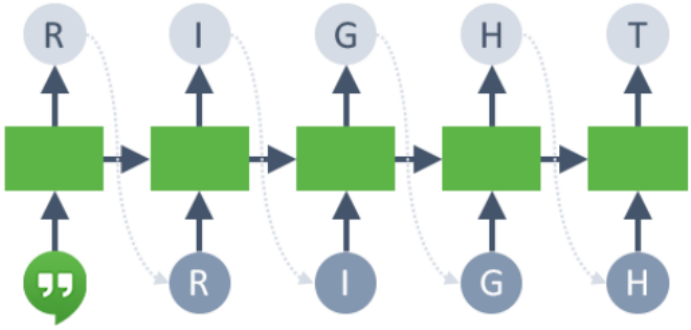
序列数据

- 很多数据都是以序列的形式出现的
  - 例如：1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3
- 如何对这个序列建模?
  - 建模时间依赖关系
  - 定义状态:三种状态：1，2，3，并做一些假设
  - 转移矩阵：$S_t = f(S_{t-1}, X_t)$，如果我们有输入 X_t
- 为什么转移矩阵不是 $S_t = f(S_{t-1})$
  - 我们不能预测有噪声的序列
  - 1.1, 2.2, 3.1, 1.3, 2.2, 3.3, 1.4, 2.4, 3.3, 1.1 ,2.0, 3.1, 1.2, 2.1, 3.0, 1.1, 2.3, 3.4, … …

使用 RNN 的示例

单词预测

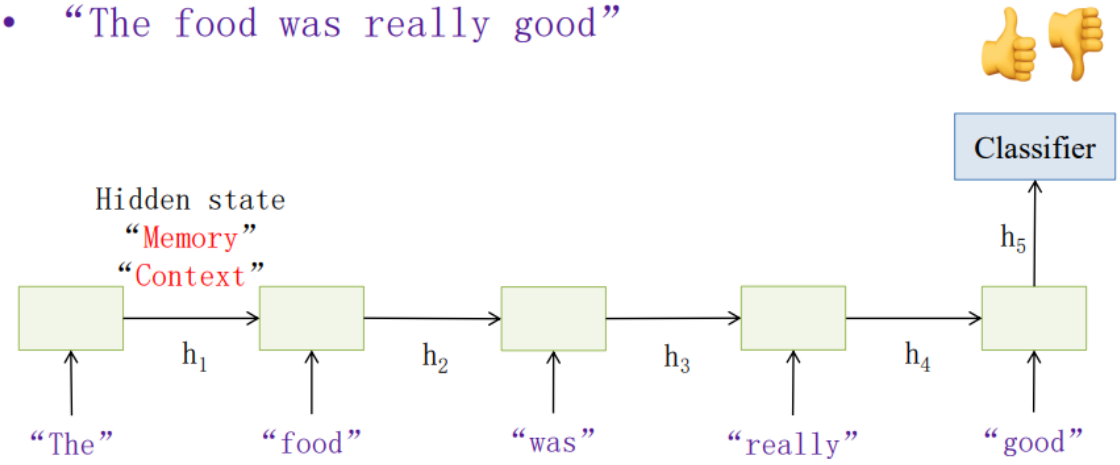
- 考虑单词 `mountain`
- 给定字符转 `mountai` 下一个字符是 `n` 的概率很大
- 这正是我们希望我们的神经网络能够建立的模型
- 更一般地说，我们需要对**上下文 context** 和**内存 memory** 进行建模



文本情感分类

- 对餐馆、电影或产品的评论进行正面或负面的分类
  - “The food was really good”
  - “The vacuum cleaner broke within two weeks”
  - “The movie had slow parts, but overall was worth watching”
- 我们可以用什么特征表示或预测结构来解决这个问题?

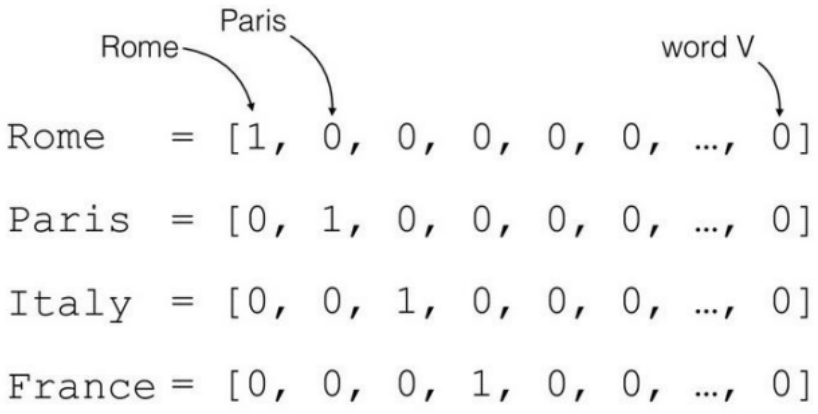
- “The food was really good”



输入的表示

- 下面我们假设我们的输入是字符（例如，网络的任务是预测下一个字符）或单词（例如，网络的任务是预测下一个单词）
- 一般来说，我们将在输入中有一个符号序列，可以从一个更大的词汇表中提取

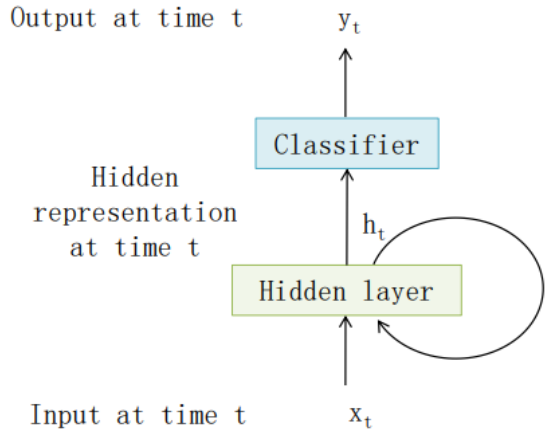
我们可以有如下的表示方法，**one-hot vector representation**



- 当然，如果输入是字符，我们也可以这样编码

2. 循环神经网络 RNN 介绍

单层单一时间视图

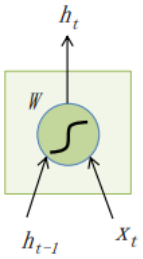


在时刻 t 时间输入 x_t，通过隐层计算得到输入 y_t，循环性的体现为

$$h_t = f_W(x_t, h_{t-1})$$

- h_t：新的状态
- f_W：权重计算
- x_t：在时刻 t 的输入
- h_{t-1}：上一个状态保存下来的信息

Vanilla RNN 单元



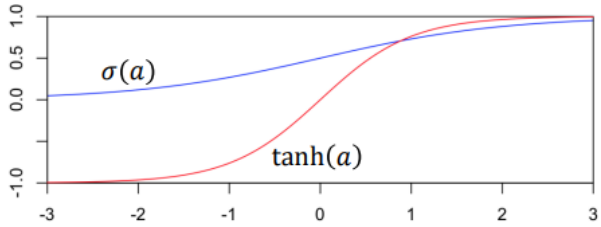
$$\begin{aligned} h_t &= f_W(x_t, h_{t-1}) \\ &= \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} \end{aligned}$$

- 标准的 RNN 循环单元
- 再经过激活函数 $tanh$

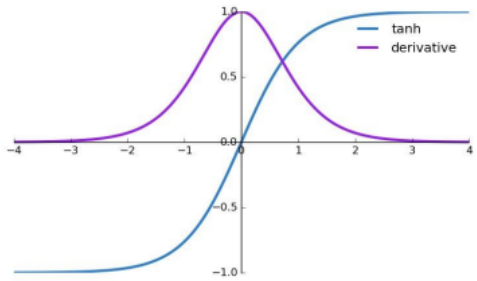
可得到计算公式如下，对于一个 Vanilla RNN 单元，每个时刻 t 的输出 h_t 为

$$\begin{aligned} h_t &= f_W\left(x_t, h_{t-1}\right) \\ &= \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} \\ &= \tanh\left(W_x x_t + W_h h_{t-1}\right) \end{aligned}$$

- W_x：处理输入 x_t 的参数
- W_h：处理上一个状态下传递信息 h_{t-1}

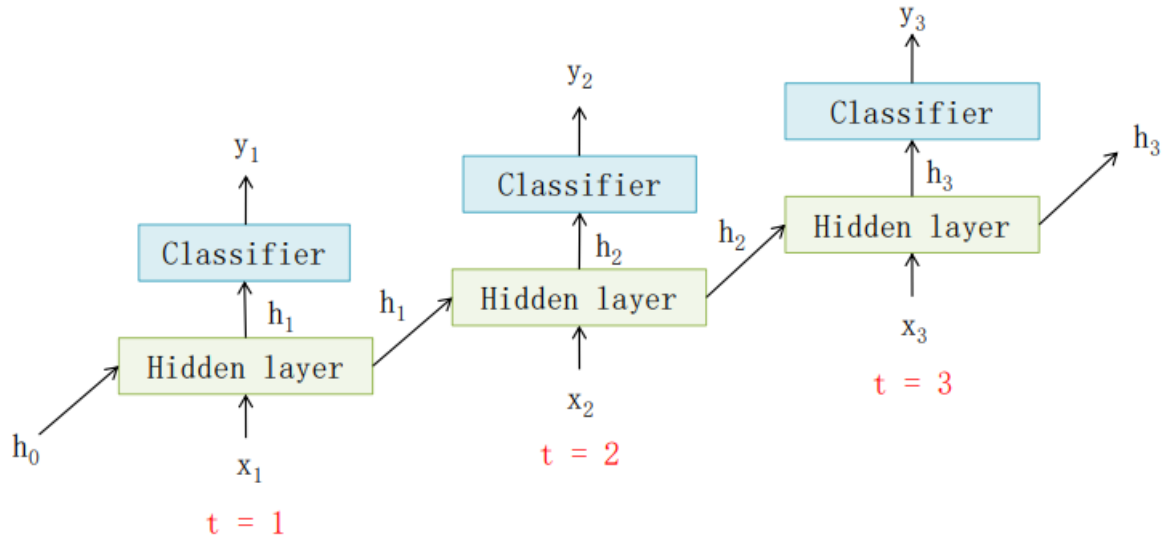


$$\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} = 2\sigma(2a) - 1$$

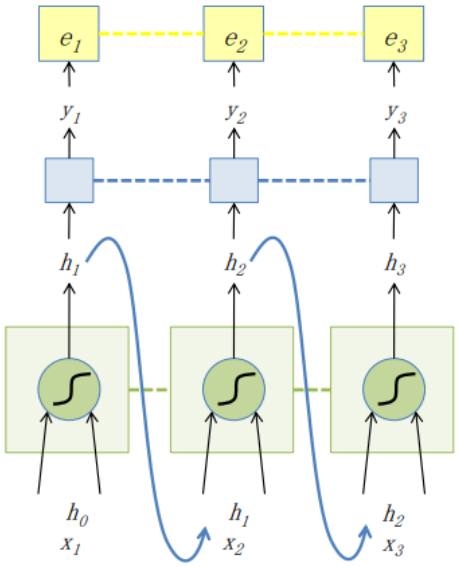


$$\frac{d}{da}\tanh(a) = 1 - \tanh^2(a)$$

展开 RNN 视图



RNN 前向传播



$$e_t = -\log(y_t(GT_t))$$

$$y_t = \text{softmax}(W_y h_t)$$

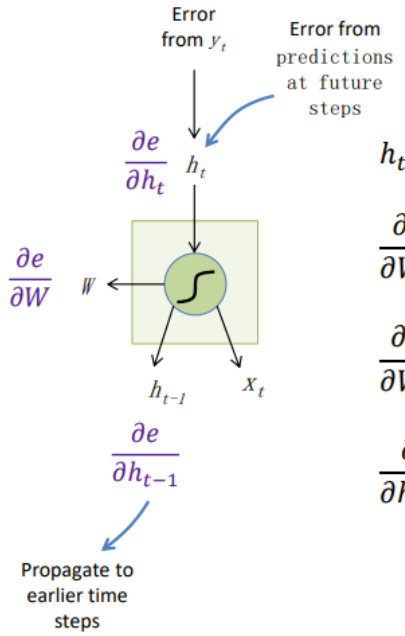
$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

- W_x，W_h，W_y 向前传递，参数共享

基于时间的反向传播 Backpropagation Through Time (BPTT)

- 最常用的训练 RNN 的方法
  - 将展开网络(前向传递时使用)看作一个接受整个时间序列作为输入的大前馈网络
  - 对展开网络中的每个副本进行权值更新计算，然后求和（或平均）并应用于 RNN 权值
  - 采用类似 MLP/CNN 的反向传播算法
  - 然而，这些层现在对应于不同的步骤/时间，因此我们称其为基于时间反向传播
- 对于**非常深**的网络，我们需要将许多不同的**梯度相乘**，对于长序列，这是一个问题
  - **梯度消失/梯度爆炸**
- 在实践中，使用**截断 BPTT**：运行RNN向前 k_1 时间步，向后传播 k_2 时间步

RNN 反向传播



$$h_t = \tanh(W_x x_t + W_h h_{t-1})$$

$$\frac{\partial e}{\partial W_h} = \frac{\partial e}{\partial h_t} \odot \left(1 - \tanh^2(W_x x_t + W_h h_{t-1})\right) h_{t-1}^T$$

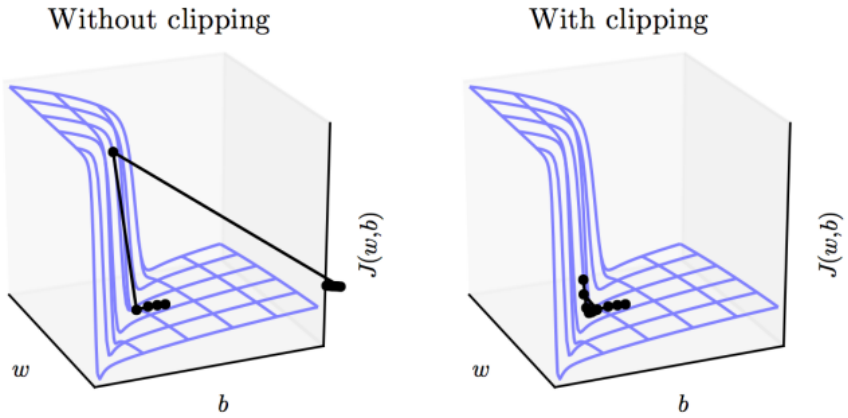
$$\frac{\partial e}{\partial W_x} = \frac{\partial e}{\partial h_t} \odot \left(1 - \tanh^2(W_x x_t + W_h h_{t-1})\right) x_t^T$$

$$\frac{\partial e}{\partial h_{t-1}} = W_h^T \left(1 - \tanh^2(W_x x_t + W_h h_{t-1})\right) \odot \frac{\partial e}{\partial h_t}$$

存在**梯度消失**的情况，考虑计算当 $k << n$ 时的 $\frac{\partial e_n}{\partial h_k}$

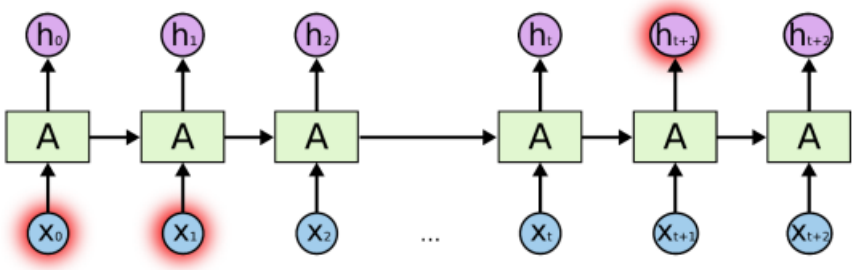
- 较大的 $tanh$ 激活会产生较小的梯度
- 如果 W_h 的最大奇异值小于 1，梯度可能消失

梯度折叠 Gradient Clipping

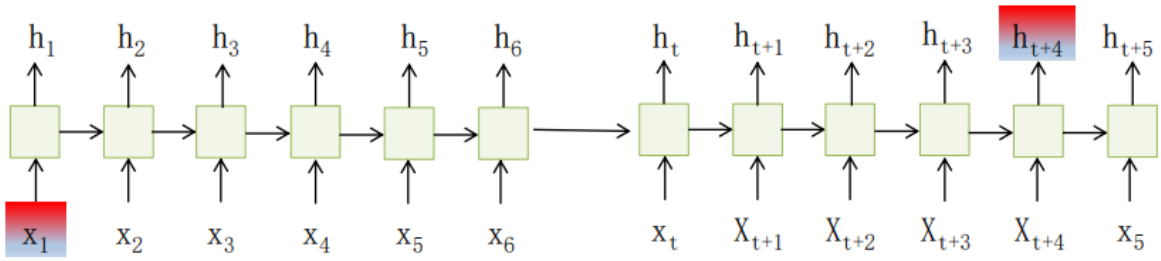


- 梯度爆炸很容易通过缩放或剪梯度变来解决
- 但是梯度消失呢？‘

长项依赖 Long-term dependencies



- 通过查看最近的信息来执行当前的任务
  - eg: 预测最后一个单词 the clouds are in the **sky**
- 在某些情况下，我们需要更多的上下文
  - eg: grew up in France… I speak fluent **French**

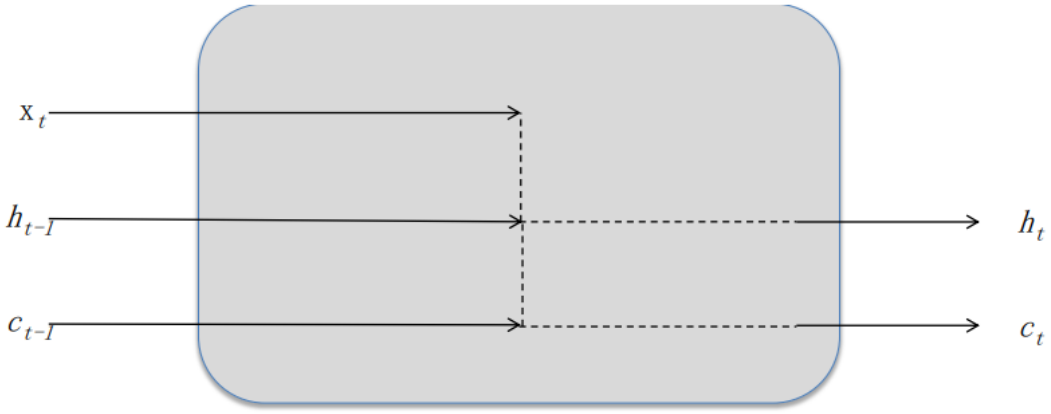


- 理论上，RNN 应该能够捕获长期的依赖关系，但实际上并非如此
- $\boldsymbol{h}^{(t)} = \boldsymbol{W}^\top \boldsymbol{h}^{(t-1)}$ 考虑一个由这个递归关系描述的非常简单的 RNN
  - 这可以简化为 $\boldsymbol{h}^{(t)} = \left(\boldsymbol{W}^t\right)^\top \boldsymbol{h}^{(0)}$
  - 已知 $\boldsymbol{W}$ 的矩阵分解，我们得到 $\boldsymbol{h}^{(t)} = \boldsymbol{Q}^\top \boldsymbol{\Lambda}^t \boldsymbol{Q} \boldsymbol{h}^{(0)}$，$\boldsymbol{W} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top$
  - 小于 1 的特征值造成梯度消失，大于 1 的特征值造成梯度爆炸
  - $h^{(0)}$ 的任何不与 $\boldsymbol{W}$ 的最大特征向量对齐的分量最终被丢弃
  - 注意，这个问题是 RNN 特有的，在 MLP 中，$\boldsymbol{W}$ 不是由所有层共享的
  - 不幸的是，我们无法避免带有爆炸/消失梯度的参数空间区域。我们需要去学习长期依赖
  - 在那里，训练不会停止，而是变得不可持续地缓慢，在实践中，我们不能训练 RNN 处理超过 10 的序列

3. 长短期记忆网络 LSTM Long Short-Term Memory

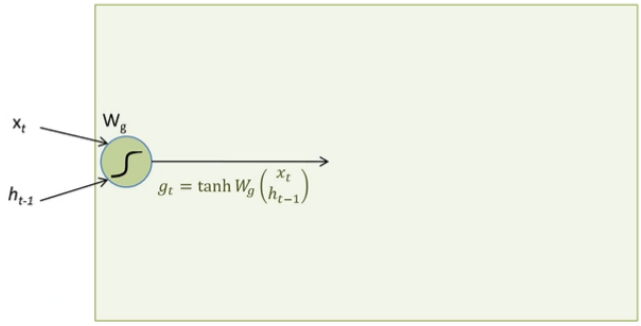
> S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Computation 9 (8), pp. 1735–1780, 1997

添加一个**不受矩阵乘法或压扁影响**的**存储单元**，从而避免梯度衰减

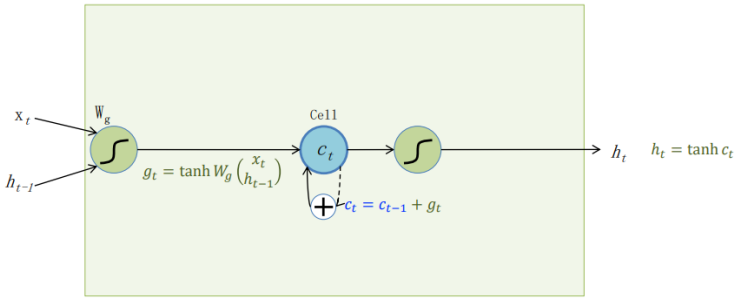


- x_t：在时刻 t 的输入
- h_{t-1}：上一个状态保存下来的信息
- c_{t-1}：上一个状态的记忆单元

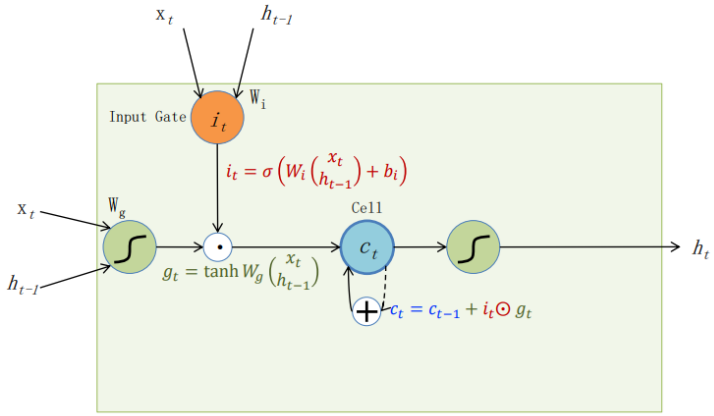
LSTM 单元



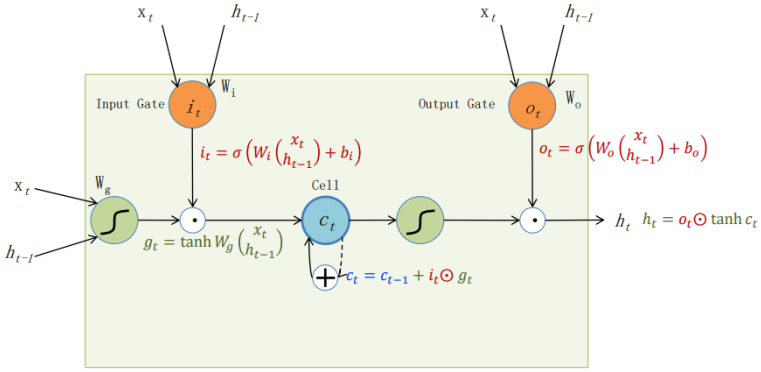
标准的 RNN 网络



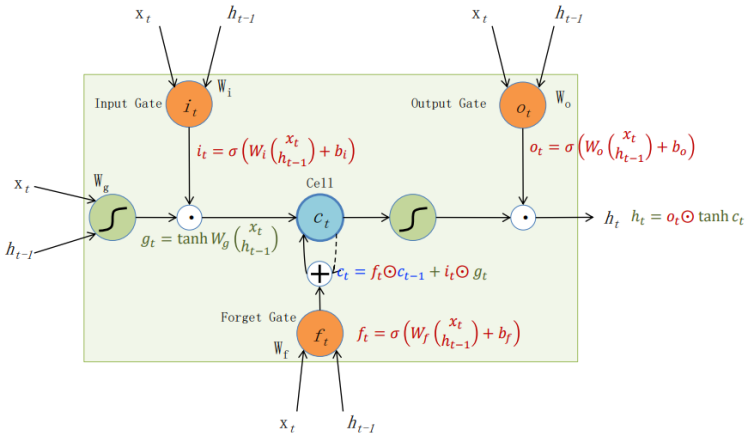
添加一个记忆单元 c_t
根据公式 $c_t = c_{t-1} + g_t$ 更新
再通过一个激活函数激活
$h_t = tanh\ c_t$



添加一个 input gate，控制输入的信息传入的量
使用 sigmoid 函数激活
c_t 的更新函数也会进行更改



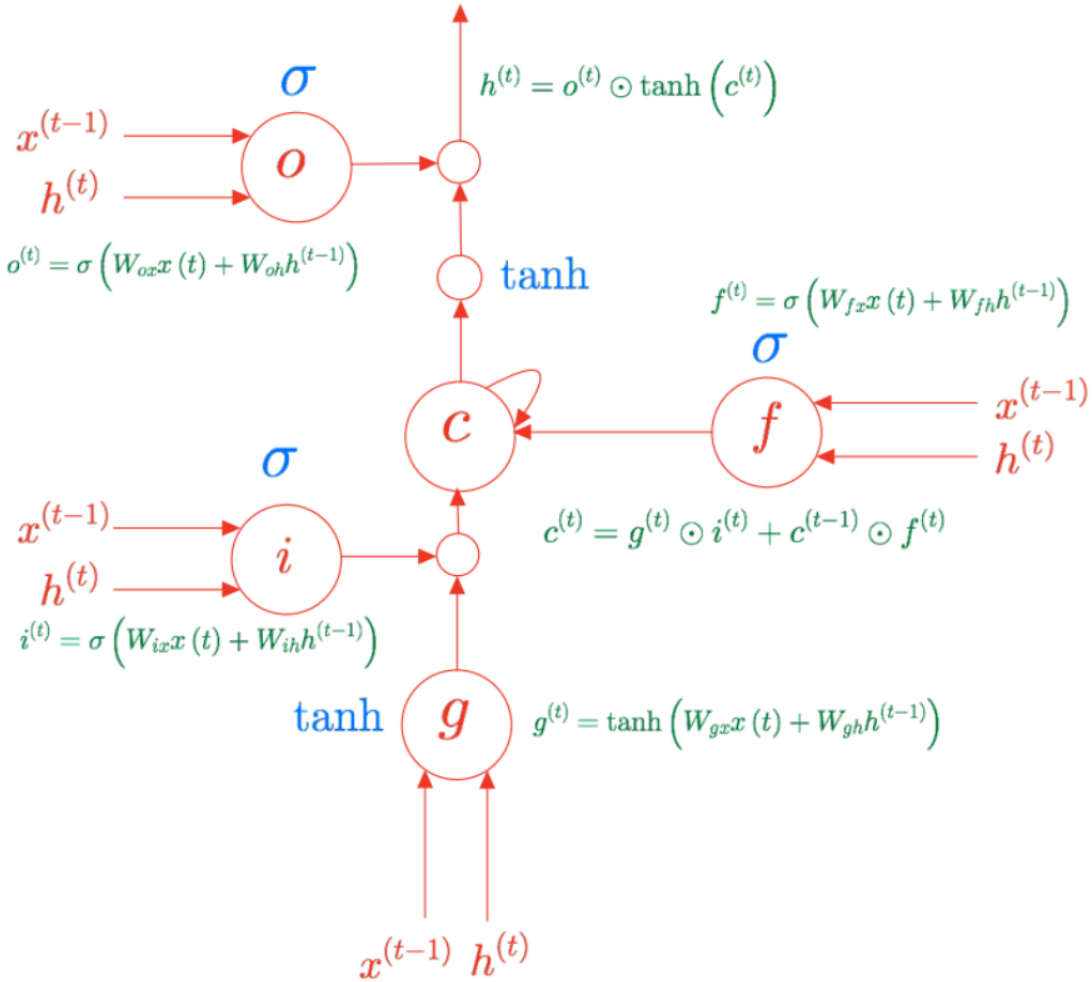
添加一个 output gate，控制输出信息传出的量
使用 sigmoid 函数激活
h_t 的更新函数也会进行更改



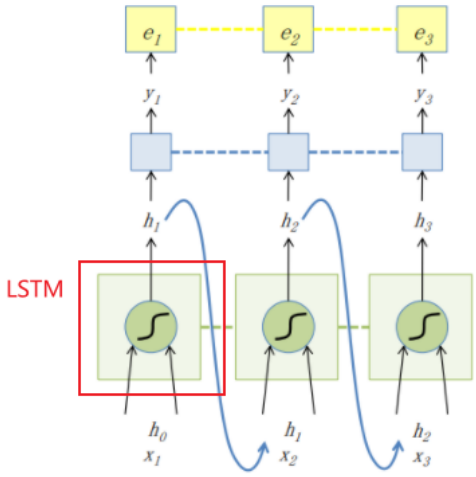
添加一个 forget gate，控制记忆单元的记忆程度

使用 signoid 函数激活

c_t 的更新函数也会进行更改



上面的图可以简化为 RNN 的一个小的 block

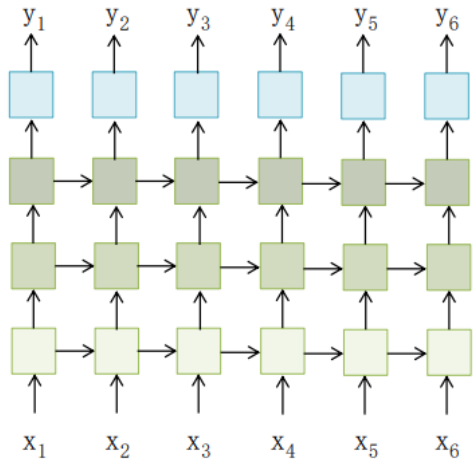


LSTM 反向传播

4. 其它 RNN

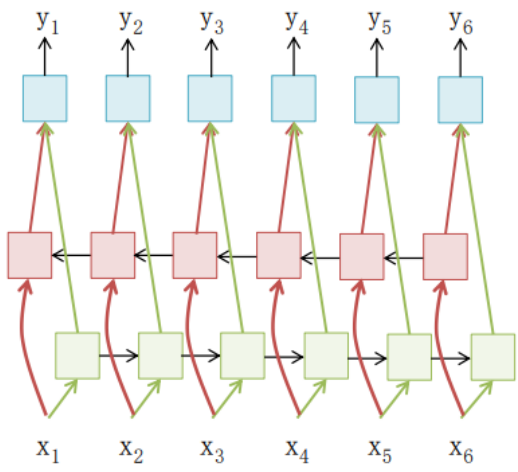
多层 RNN

我们当然可以设计具有多个隐藏层的 RNN



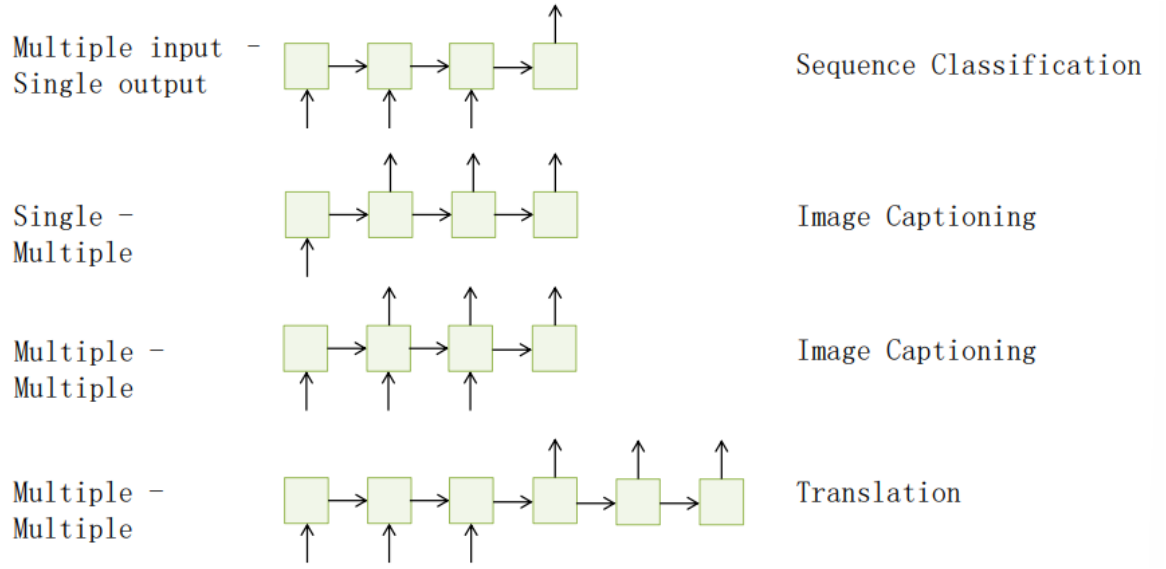
双向 RNN

RNN 可以对输入序列进行正向处理和反向处理



- 在语音识别中很流行

使用示例



LSTM 真的很厉害吗?

- LSTM 存在着无数的变化，不同的研究者提出了不同的 LSTM 单元排列
- 哪一个更好呢?
  - 没有：https://arxiv.org/pdf/1503.04069.pdf
- 此外，RNN 的性能可以优于 LSTM 和 GRU：http://proceedings.mlr.press/v37/jozefowicz15.pdf