循环神经网络
Recurrent Neural Network

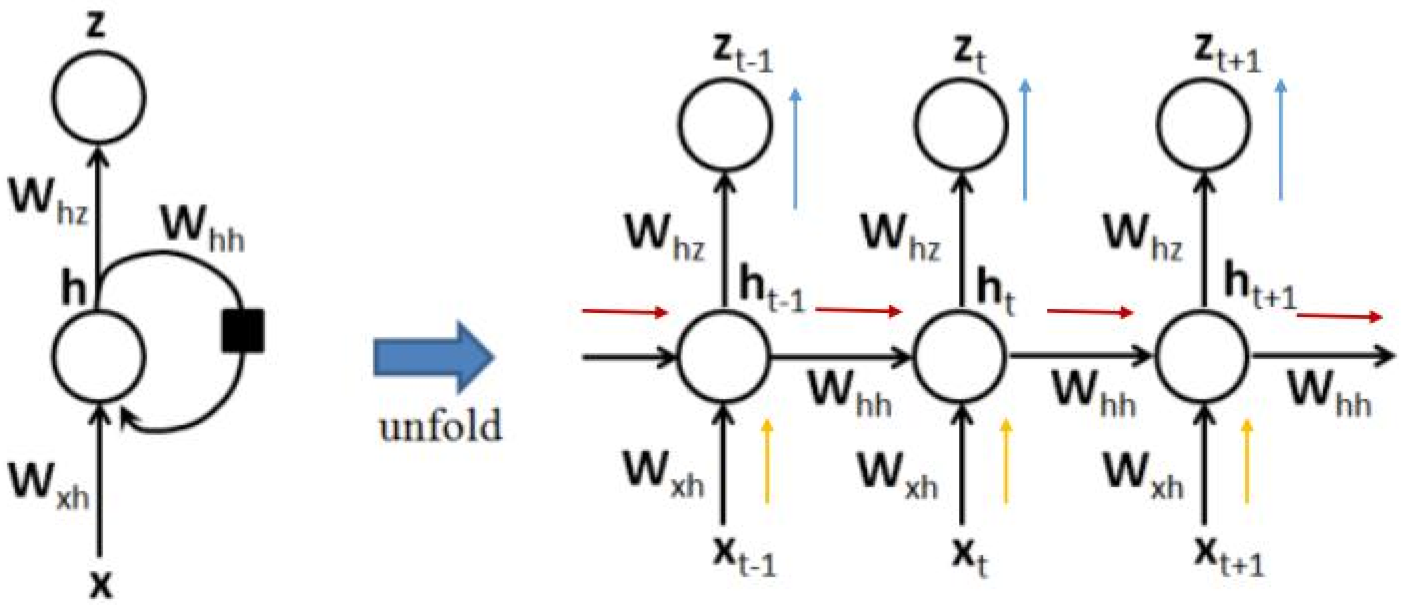
主讲TA：张祖胜

2020/10/30

引入

- RNN是一类扩展的人工神经网络，适用于对序列数据进行建模
  - 序列数据：文本，词语的序列；语音，音节的序列；视频，图像的序列
  - 核心思想：前后的样本在时序上存在相关性；通过神经网络在时序上的展开，可以捕捉样本之间的序列相关性

结构

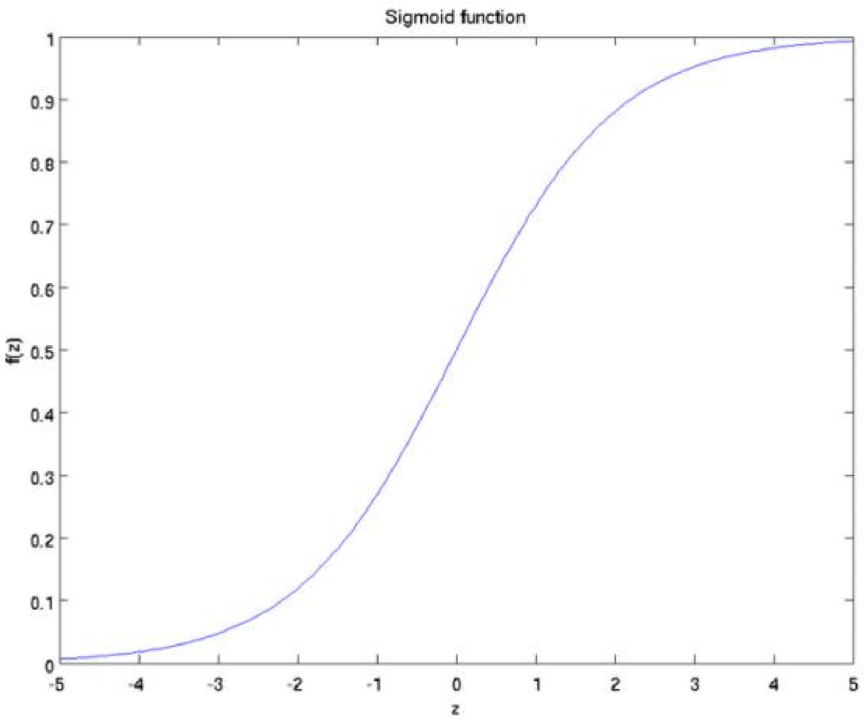


$$\mathbf{h}_t = \tanh(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h)$$

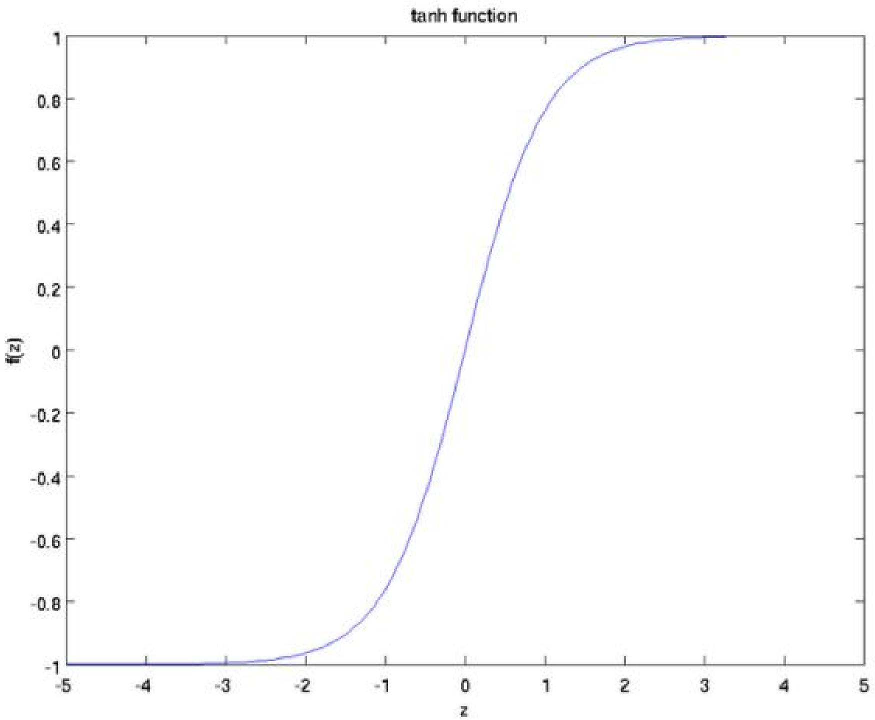
$$\mathbf{z}_t = \mathrm{softmax}(\mathbf{W}_{hz}\mathbf{h}_t + \mathbf{b}_z)$$

激活函数

- 常用的激活函数是sigmoid和tanh



$$f(z) = \frac{1}{1 + \exp(-z)}$$



$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

激活函数

- softmax是sigmoid的一个变种，通常用于多分类任务的输出层，将输入的特征向量转化成各个标签的概率

$$h_\theta\left(x^{(i)}\right)=\begin{bmatrix}p\left(y^{(i)}=1|x^{(i)};\theta\right)\\p\left(y^{(i)}=2|x^{(i)};\theta\right)\\\vdots\\p\left(y^{(i)}=k|x^{(i)};\theta\right)\end{bmatrix}=\frac{1}{\sum_{j=1}^{k}e^{\theta_j^T x^{(i)}}}\begin{bmatrix}e^{\theta_1^T x^{(i)}}\\e^{\theta_2^T x^{(i)}}\\\vdots\\e^{\theta_k^T x^{(i)}}\end{bmatrix}$$

训练

- 反向传播：通过链式法则求损失函数 E 对网络权重的梯度，利用梯度的反方向对权重进行更新；RNN的反向传播需要考虑时序演化过程，相对复杂
- 定义权重矩阵U，V，W
  - 每个时刻的隐状态和输出为：

$$
\begin{aligned}
h_t &= \tanh(Ux_t + Wh_{t-1}) \\
\hat{y}_t &= \mathrm{softmax}(Vh_t)
\end{aligned}
$$

  - 时序上的总损失为：

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

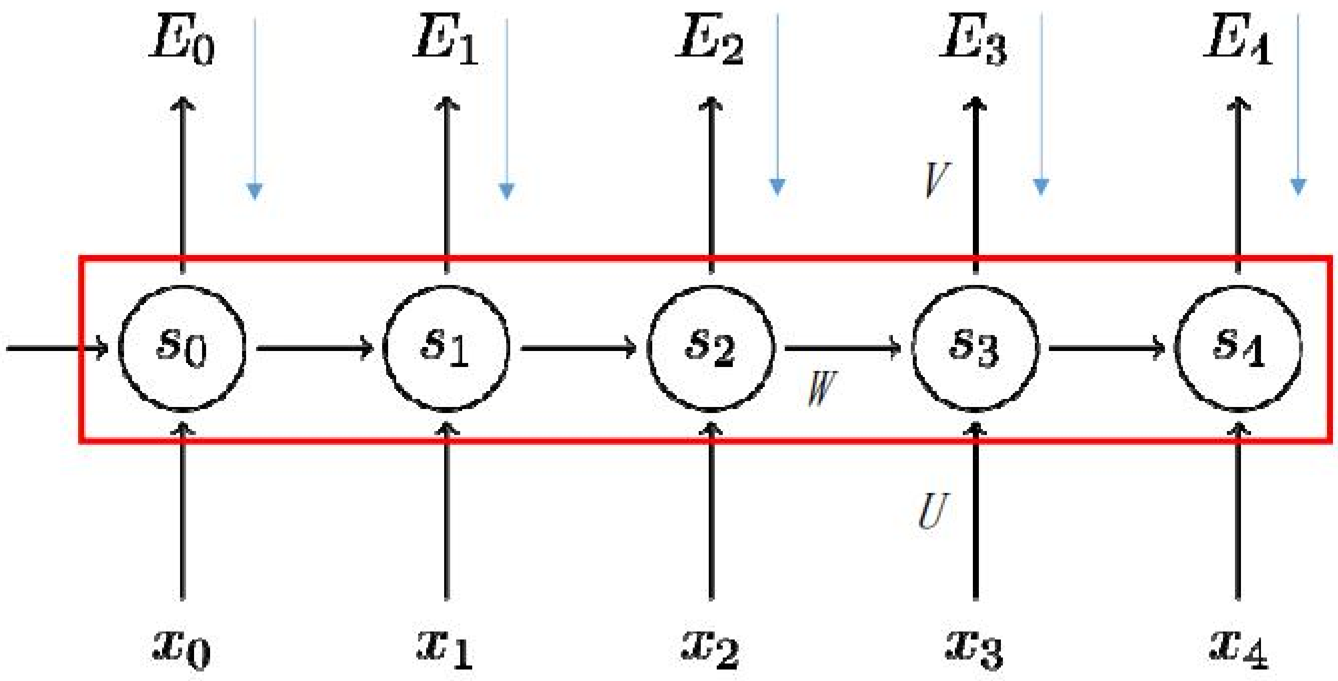
训练

- 目前的任务是求E对U，V，W的梯度；以V为例，E对V的总梯度为：

$$\frac{\partial E}{\partial V}=\sum_t \frac{\partial E_t}{\partial V}$$

- E_3对V的梯度（$z_3=Vs_3$）：

$$\begin{aligned}\frac{\partial E_3}{\partial V} &= \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial V} \\ &= \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial z_3}\frac{\partial z_3}{\partial V}\end{aligned}$$



训练

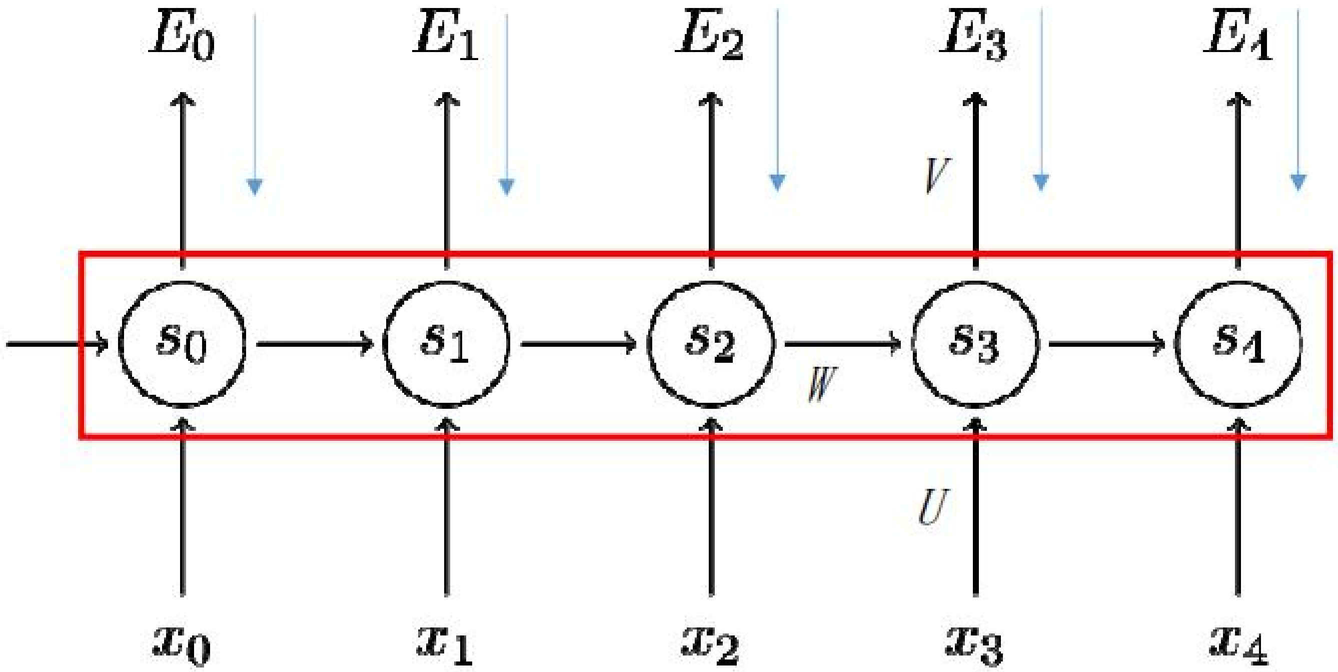
- E_3对W的梯度：

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

其中，

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

- 即s_3依赖于s_2，s_2依赖于s_1和W，依赖关系一直传递到t=0时刻；所以，当我们计算s3对W的梯度时，不能把s2看做常数项



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

训练

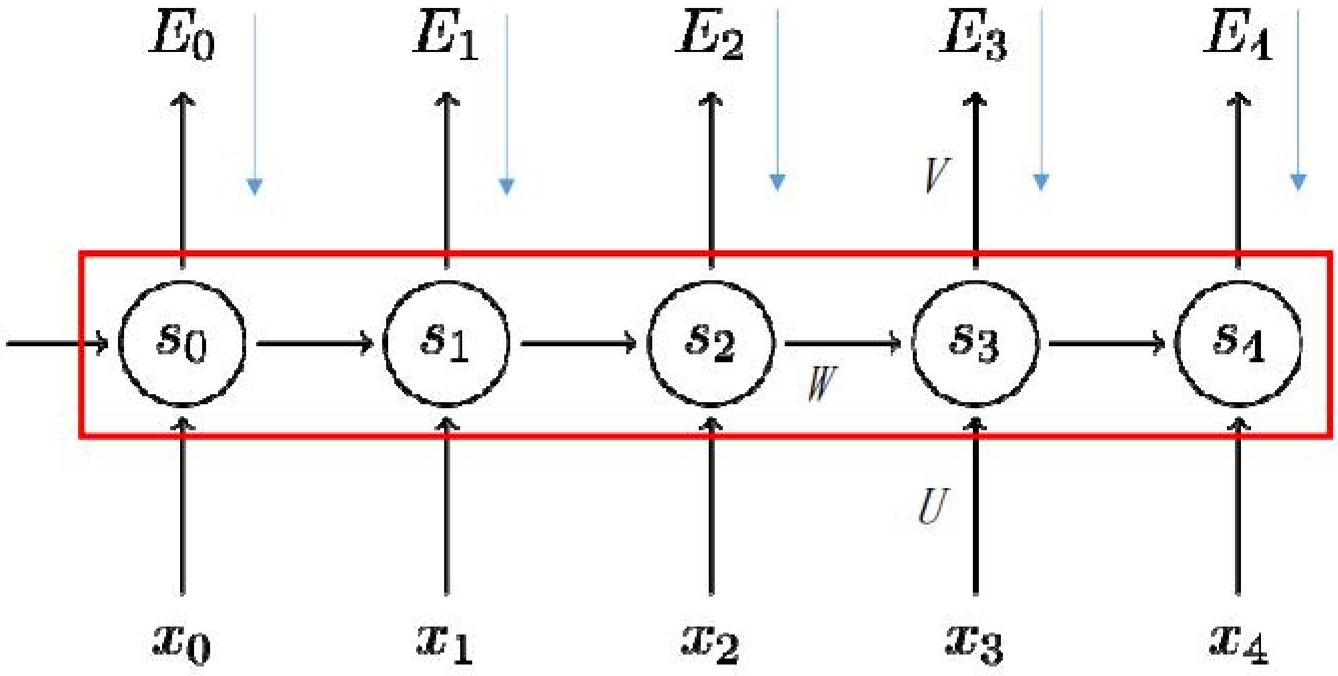
- E_3对U的梯度：

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

其中，

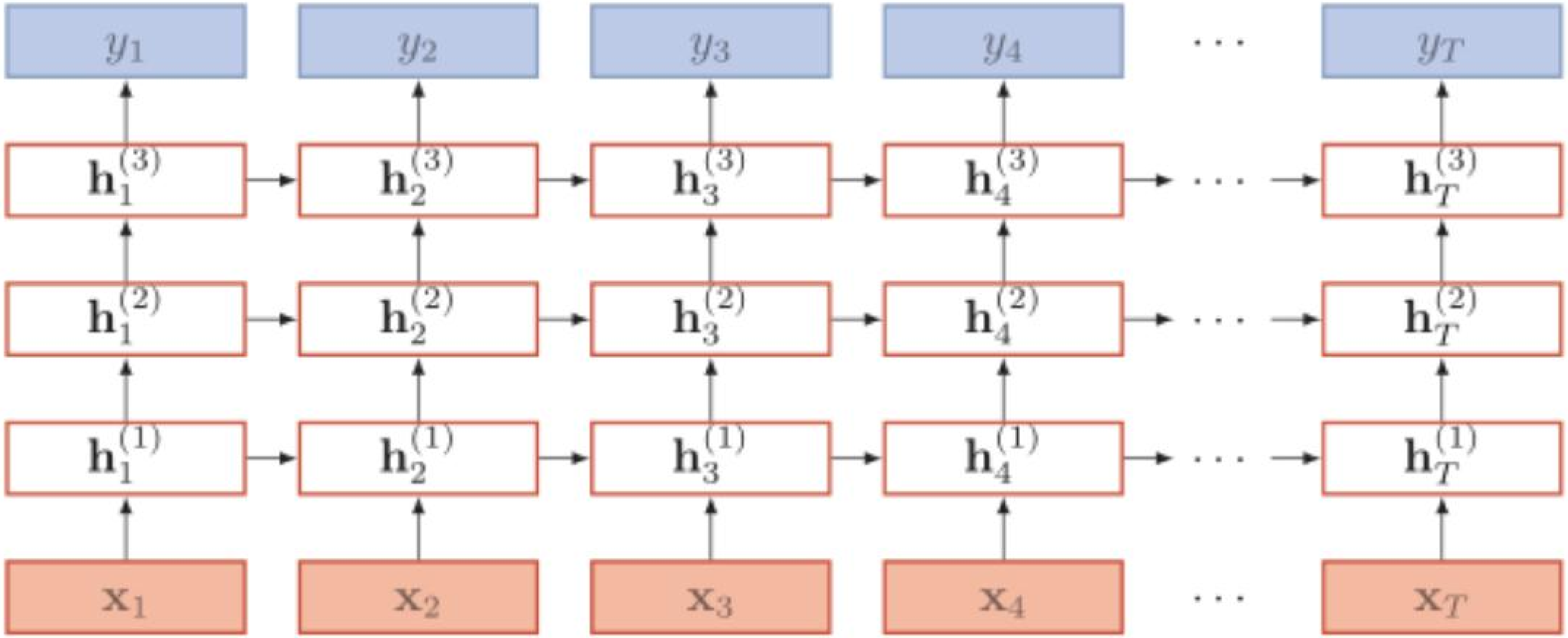
$$s_3=\tanh(Ux_t+Ws_2)$$

- 与前面一样，我们需要考虑依赖关系的传递



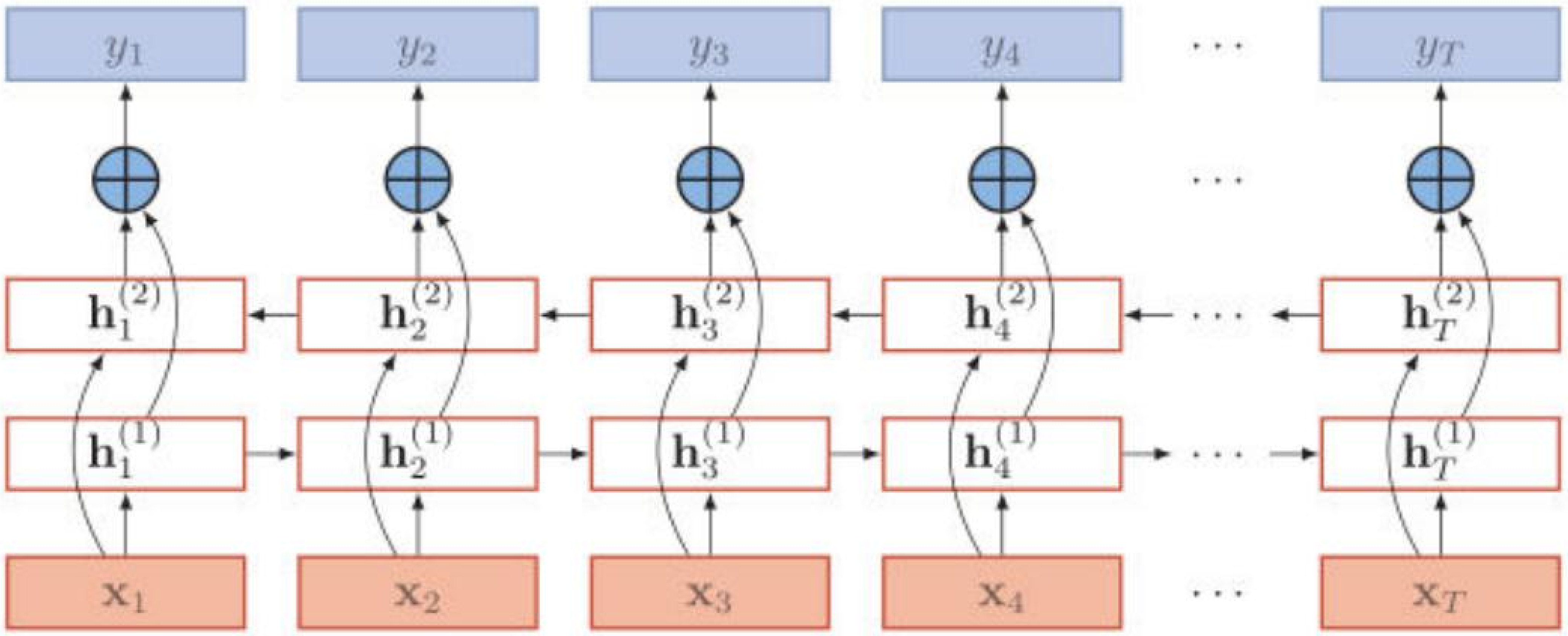
$$\boxed{\frac{\partial E_3}{\partial U}=\sum_{k=0}^{3}\frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\left(\prod_{t=k+1}^{3}\frac{\partial s_t}{\partial s_{t-1}}\right)\frac{\partial s_k}{\partial U}}$$

扩展-堆叠RNN



y_1
y_2
y_3
y_4
$\cdots$
y_T
$\mathbf{h}_1^{(3)}$
$\mathbf{h}_2^{(3)}$
$\mathbf{h}_3^{(3)}$
$\mathbf{h}_4^{(3)}$
$\cdots$
$\mathbf{h}_T^{(3)}$
$\mathbf{h}_1^{(2)}$
$\mathbf{h}_2^{(2)}$
$\mathbf{h}_3^{(2)}$
$\mathbf{h}_4^{(2)}$
$\cdots$
$\mathbf{h}_T^{(2)}$
$\mathbf{h}_1^{(1)}$
$\mathbf{h}_2^{(1)}$
$\mathbf{h}_3^{(1)}$
$\mathbf{h}_4^{(1)}$
$\cdots$
$\mathbf{h}_T^{(1)}$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
$\cdots$
$\mathbf{x}_T$

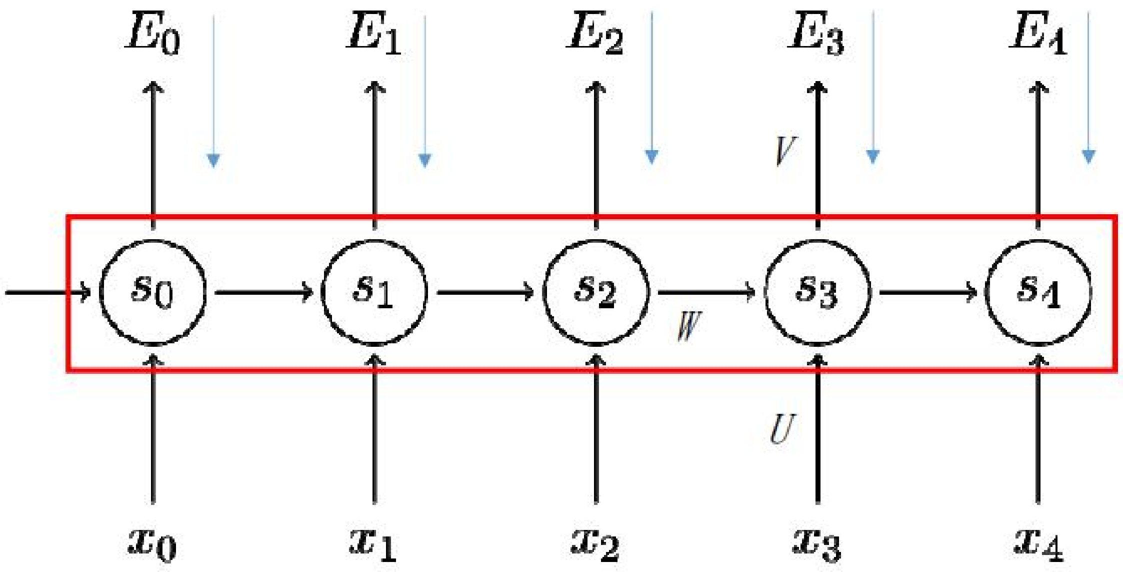
扩展-双向RNN



y_1
y_2
y_3
y_4
$\cdots$
y_T
$\mathbf{h}_1^{(2)}$
$\mathbf{h}_2^{(2)}$
$\mathbf{h}_3^{(2)}$
$\mathbf{h}_4^{(2)}$
$\mathbf{h}_T^{(2)}$
$\mathbf{h}_1^{(1)}$
$\mathbf{h}_2^{(1)}$
$\mathbf{h}_3^{(1)}$
$\mathbf{h}_4^{(1)}$
$\mathbf{h}_T^{(1)}$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
$\mathbf{x}_T$

问题

- 梯度消失；梯度爆炸



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

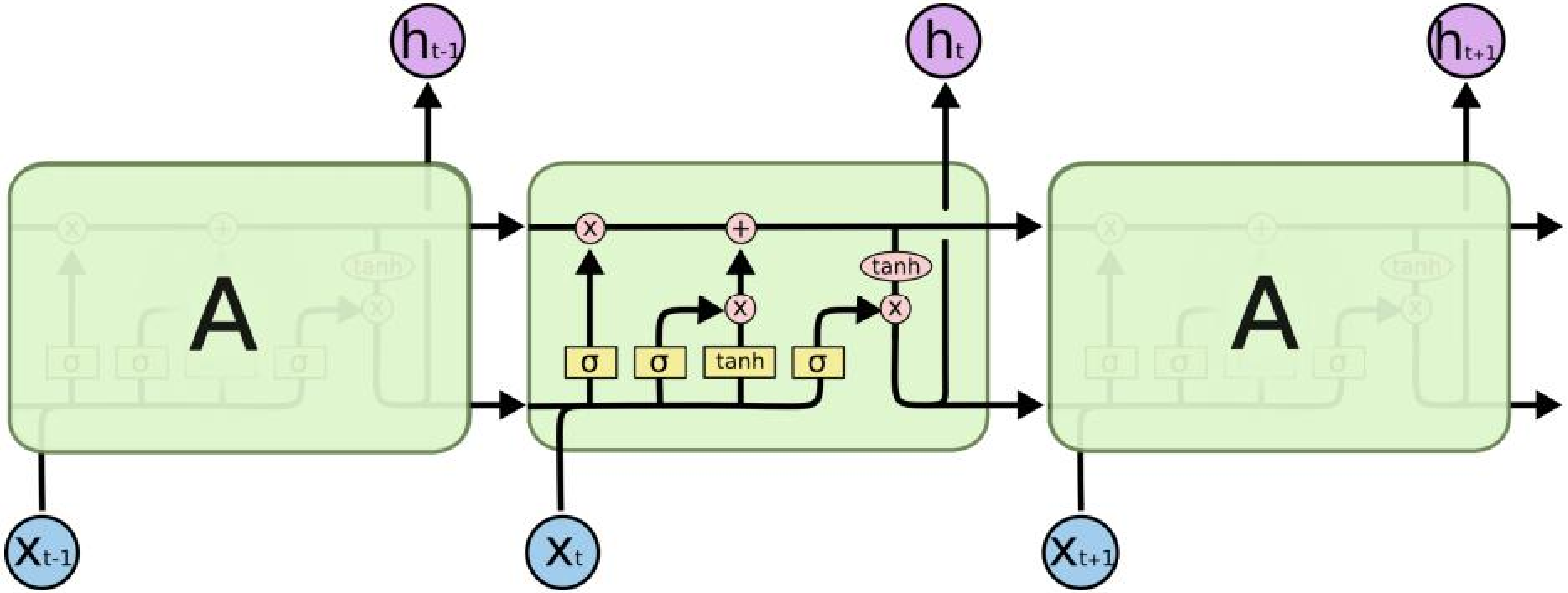
- 概念澄清：RNN里的梯度消失，指的是当前时刻的梯度由近距离的梯度所主导，而远距离的梯度由于连乘的存在而对当前时刻的梯度求和贡献不大

解决方案

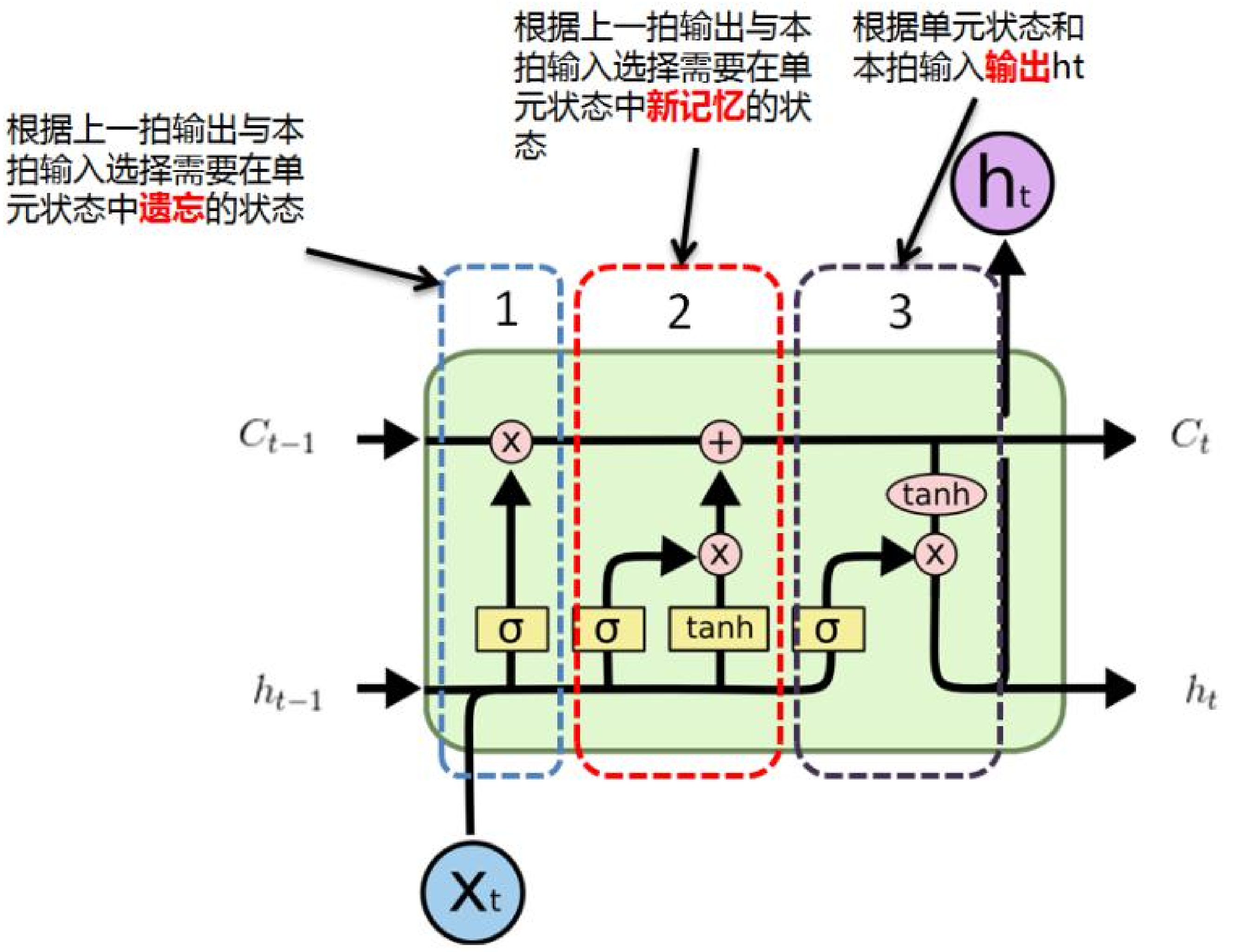
- 选择其他激活函数，如ReLU
- 改进网络结构，使用变种的RNN，如LSTM、GRU

LSTM

- 全称：Long Short-Term Memory Network，长短期记忆网络
- 通过引入门控机制，使得梯度能够进行远距离的流通，缓解梯度消失问题



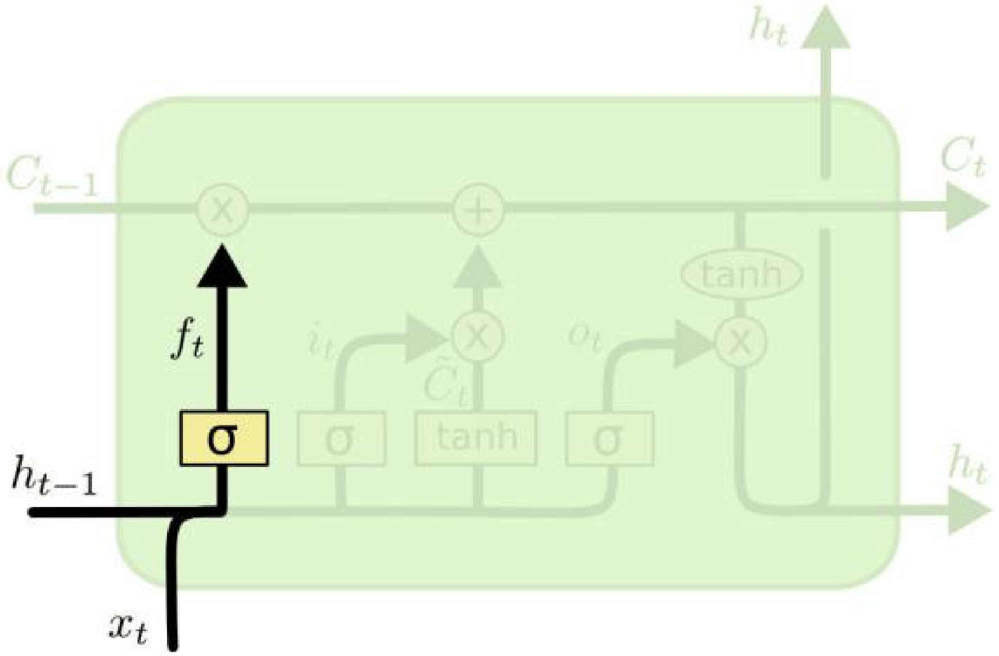
LSTM



C_{t-1}　C_t　单元状态 CellState　　　h_{t-1}　h_t　单元输出

LSTM-遗忘门

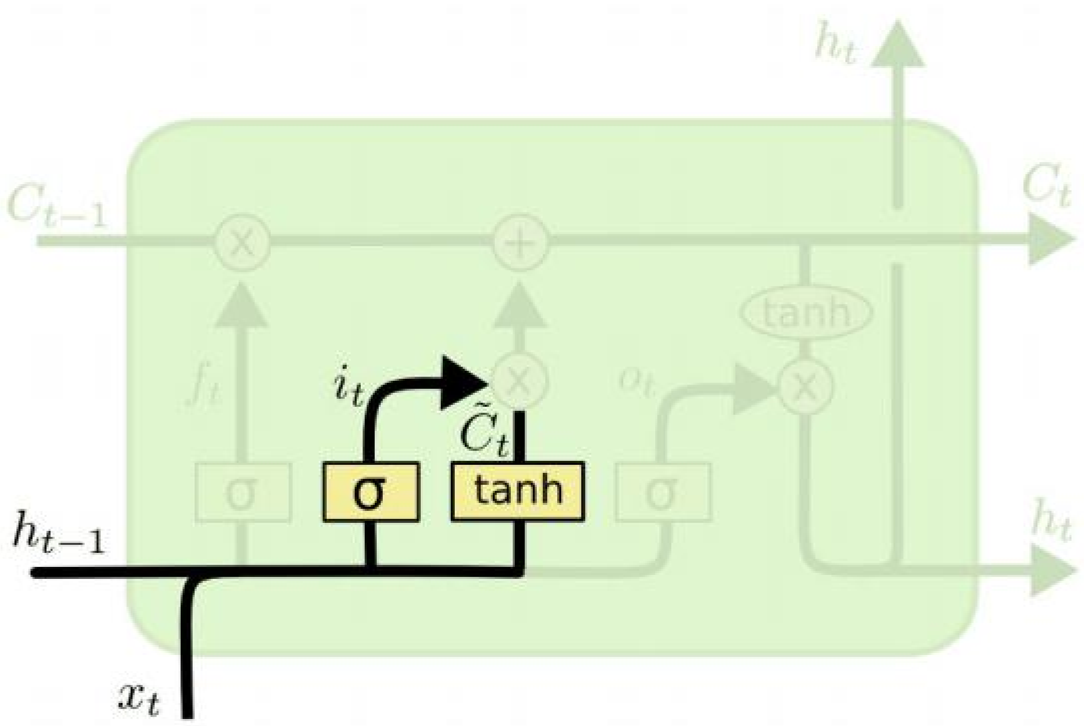
- 遗忘门：将当前时刻的输入x_t以及前一个时刻的输出h_{t-1}通过激活函数，在每个位置映射成一个0到1之间的数，然后和LSTM中用于保存历史信息的细胞状态**按位相乘**，控制每个位置的保留程度



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

LSTM-输入门

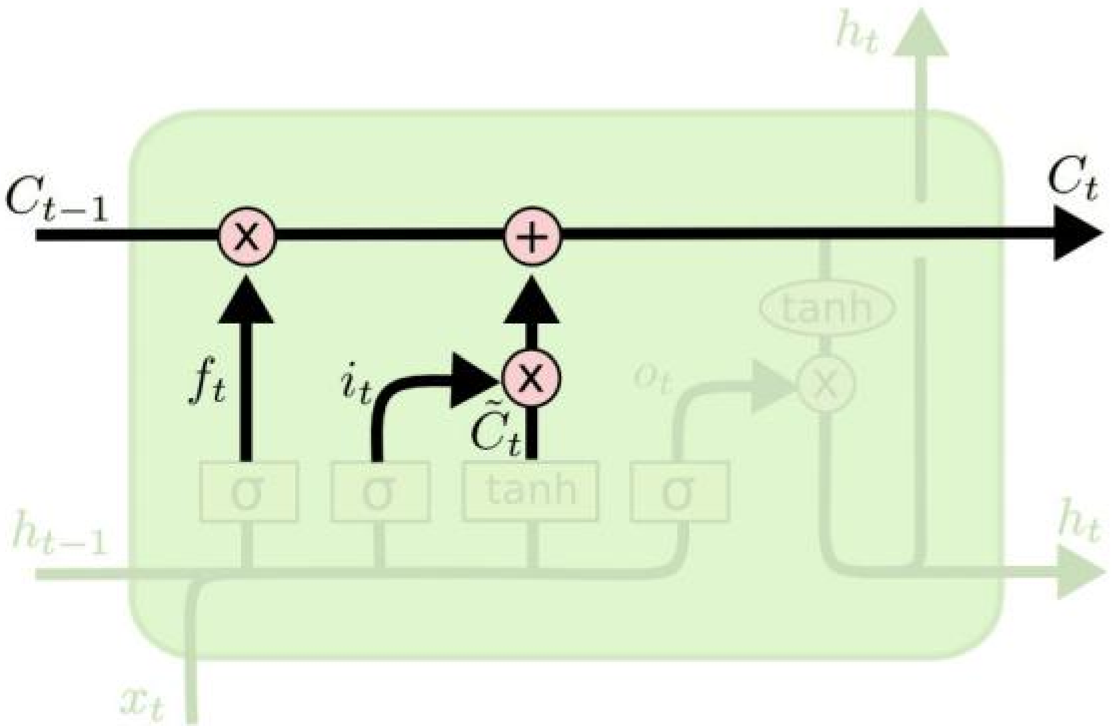
- 输入门：与遗忘门相类似，表示当前产生的隐状态有多少需要被保留并加入到历史信息中



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

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

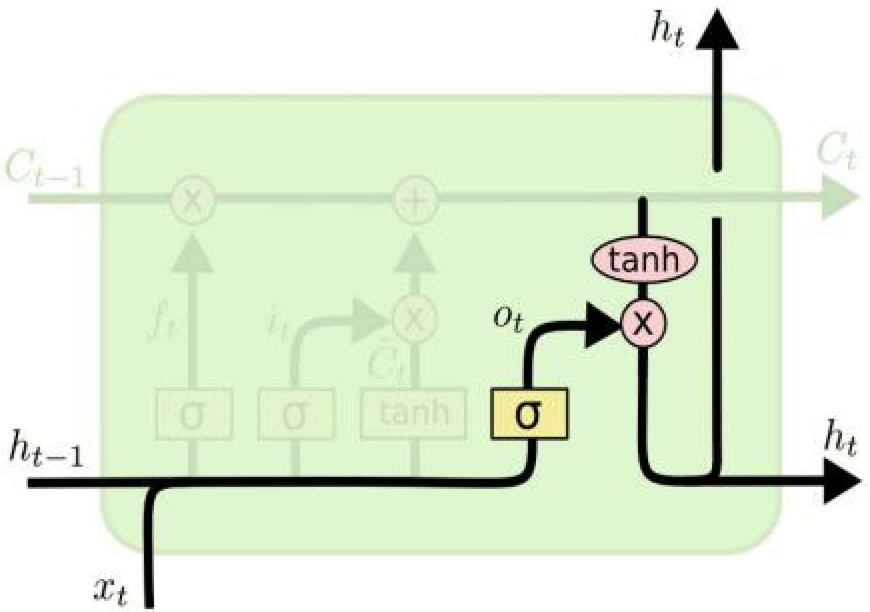
LSTM-细胞状态更新



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

LSTM-输出门

- 输入门：与遗忘门以及输入门相类似，控制当前时刻隐状态的输出



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

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

LSTM-总结

- LSTM的训练方式依然是时序上的反向传播，但参数比一般RNN要多

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

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

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

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

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

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

LSTM-关于缓解梯度消失

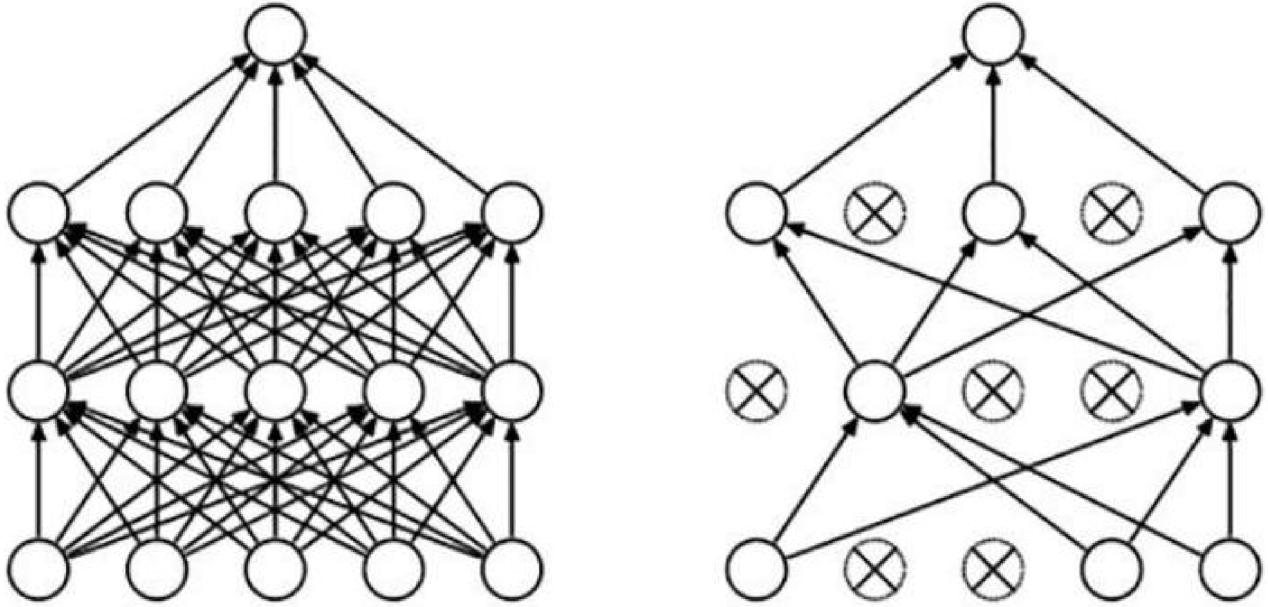
- 以遗忘门为例，定义t时刻的损失函数值为E_t，则E_t对遗忘门参数的梯度计算如下：

$$
\begin{aligned}
\frac{\partial E_t}{\partial W_f} = & \frac{\partial E_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial c_t} \cdot \frac{\partial c_t}{\partial f_t} \cdot \frac{\partial f_t}{\partial W_f} \\
& + \frac{\partial E_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial c_t} \cdot \boxed{\frac{\partial c_t}{\partial c_{t-1}} \cdot \frac{\partial c_{t-1}}{\partial f_{t-1}}} \cdot \frac{\partial f_{t-1}}{\partial W_f} \\
& + \ldots
\end{aligned}
$$

$$
\frac{\partial c_t}{\partial c_{t-1}} = f_t + \frac{\partial f_t}{\partial c_{t-1}} \cdot c_{t-1} + \ldots
$$

Dropout

- RNN参数众多，容易出现过拟合，需要有效的正则化方法；Dropout是其中的一种简单有效的方法

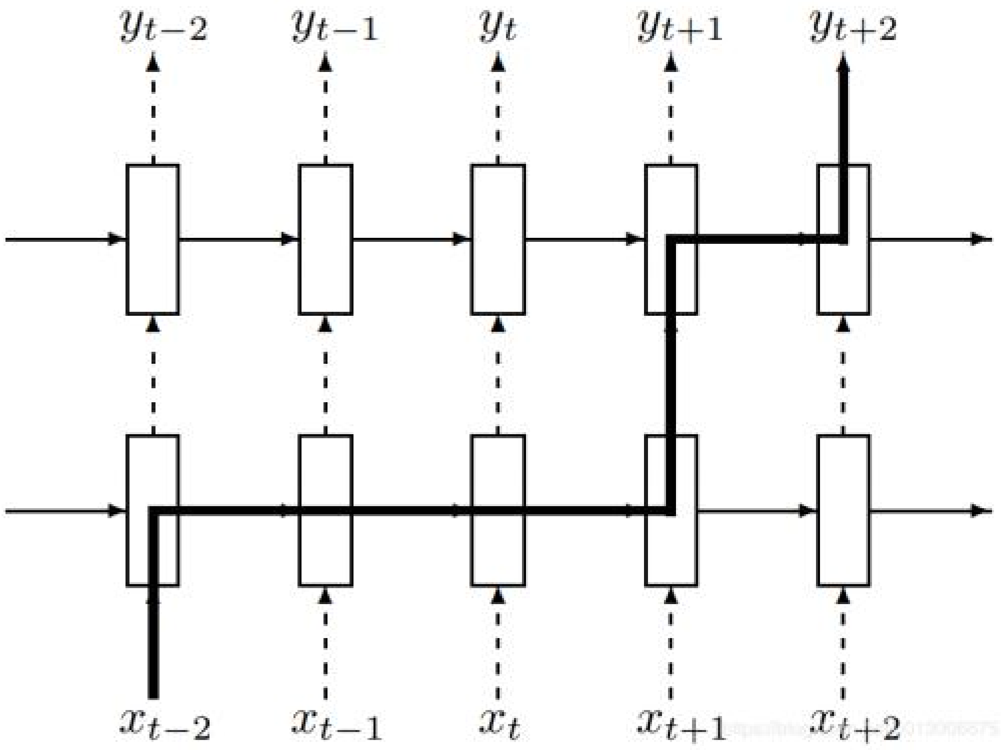


(a) Standard Neural Net　　(b) After applying dropout.

$$
\begin{aligned}
r_j^{(l)} &\sim \mathrm{Bernoulli}(p), \\
\widetilde{\mathbf{y}}^{(l)} &= \mathbf{r}^{(l)} * \mathbf{y}^{(l)}, \\
z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \widetilde{\mathbf{y}}^{l} + b_i^{(l+1)} \\
y_i^{(l+1)} &= f(z_i^{(l+1)}).
\end{aligned}
$$

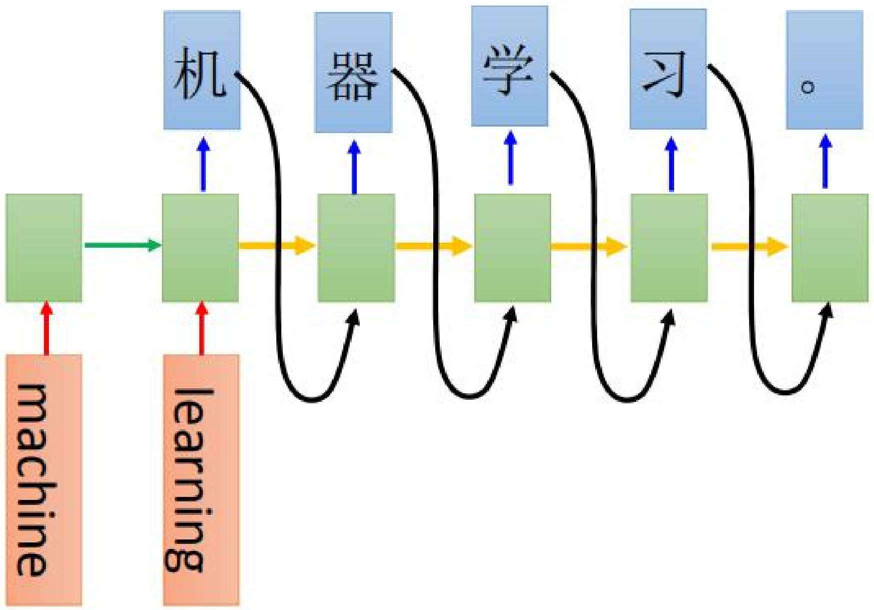
Dropout

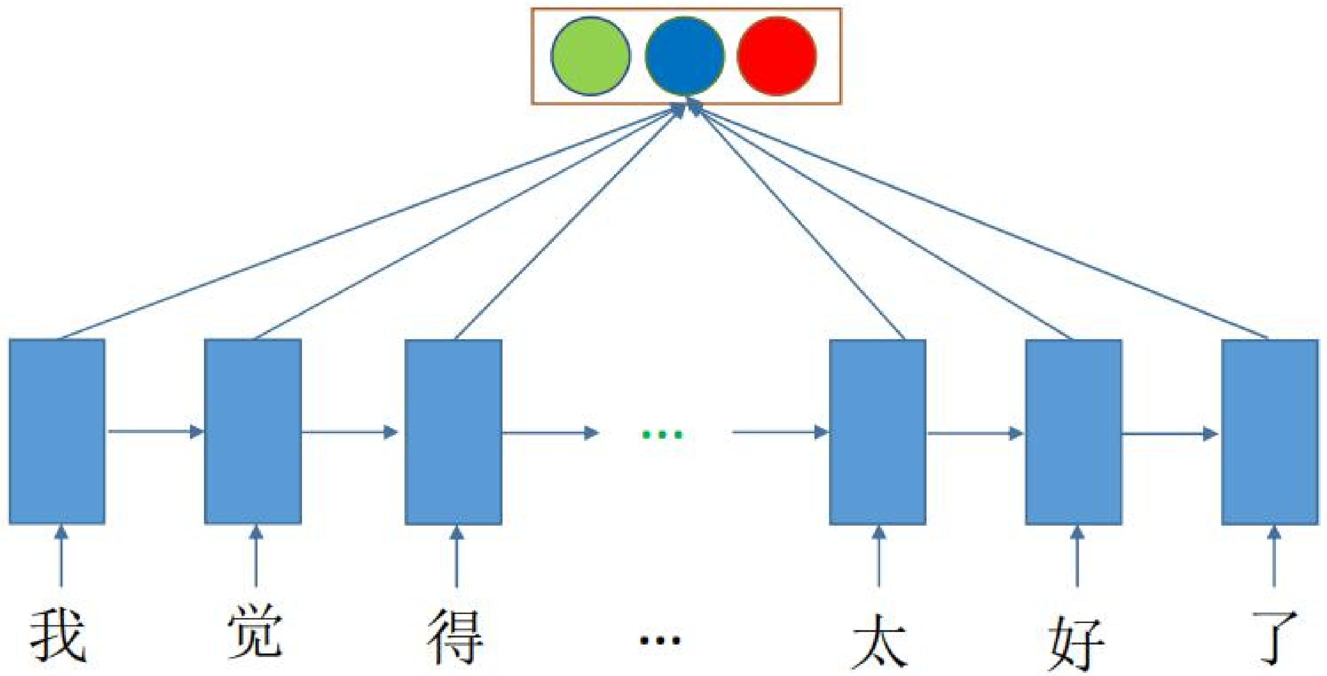
- 在RNN中，Dropout需要设置在网络的非循环部分，否则信息会因为循环的进行而逐渐丢失
- 例如，如果我们把Dropout设置在隐状态上（实线），那么每经过一次循环，信息就会被随机丢弃一部分；而如果设置在输入上（虚线），那么因Dropout造成的信息损失与循环步数无关，只与网络层数有关



应用

- RNN的应用：机器翻译、情感分类……





RNN实验

- 任务：关键词提取（Subtask1：Aspect Term Extraction）
- 数据集：SemEval-2014，Laptop
- 任务说明及数据集下载：http://alt.qcri.org/semeval2014/task4/

- 词向量（GloVe）：https://nlp.stanford.edu/projects/glove/
- 停用词处理：对应的词向量置为0，或者随机初始化

- 常用的深度学习框架：TensorFlow，Keras，Pytorch……

期中Project

- 实验内容
  - 使用CNN完成图片分类任务
  - 使用RNN完成关键词提取任务

- DDL
  - 实验报告：第十一周周四（11.12），晚上12:00
  - 验收：第十一周周五（11.13）