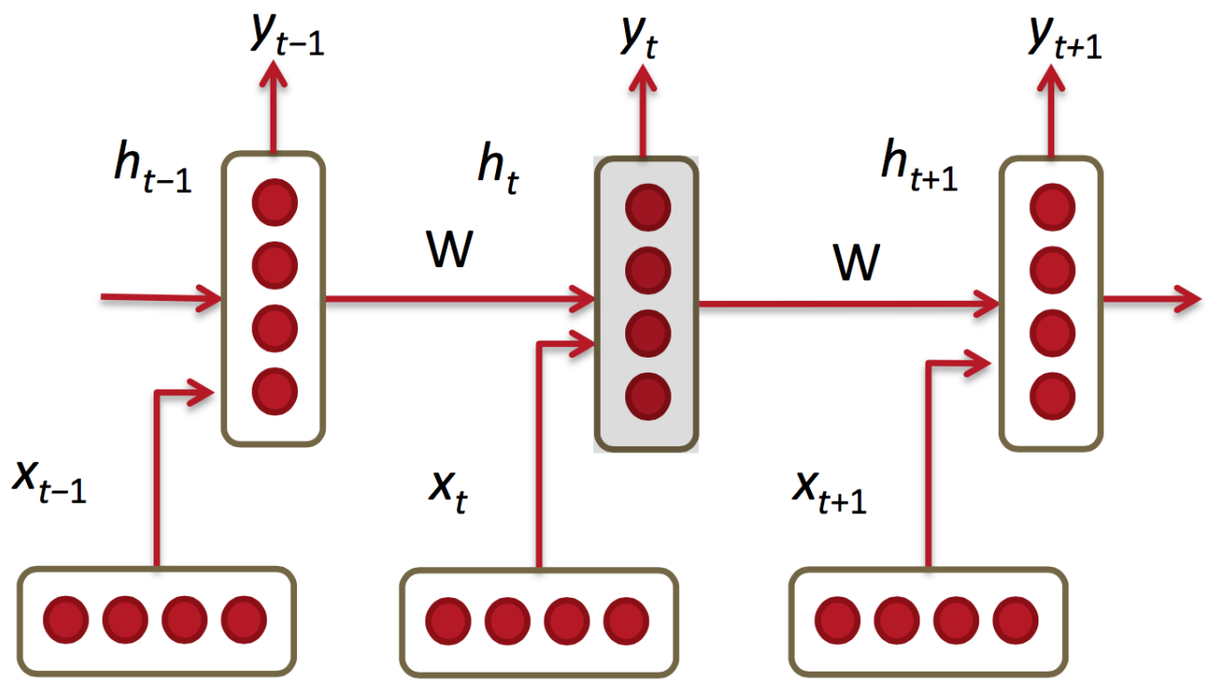
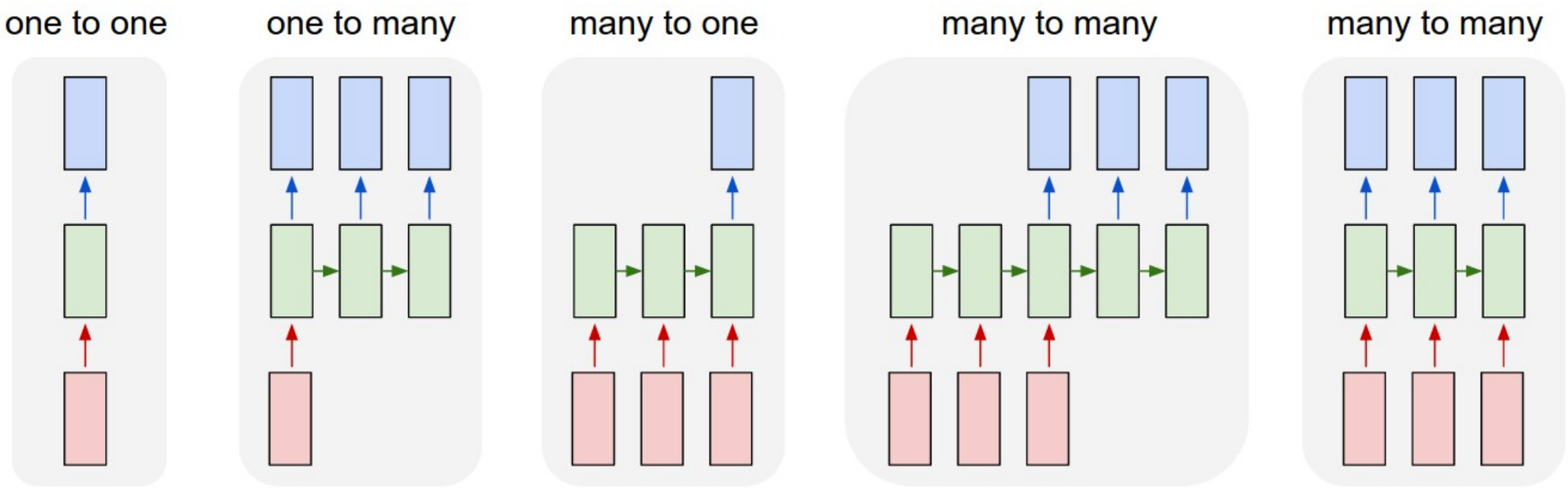
7. RNN 介绍

状态和模型

- IID 数据
  - 分类问题
  - 回归问题
  - 特征表达
- 更多的数据不满足IID
  - 序列分析（Tagging, Annotation)
  - 序列生成，如语言翻译，自动文本生成
  - 内容提取（Content Extraction），如图像描述

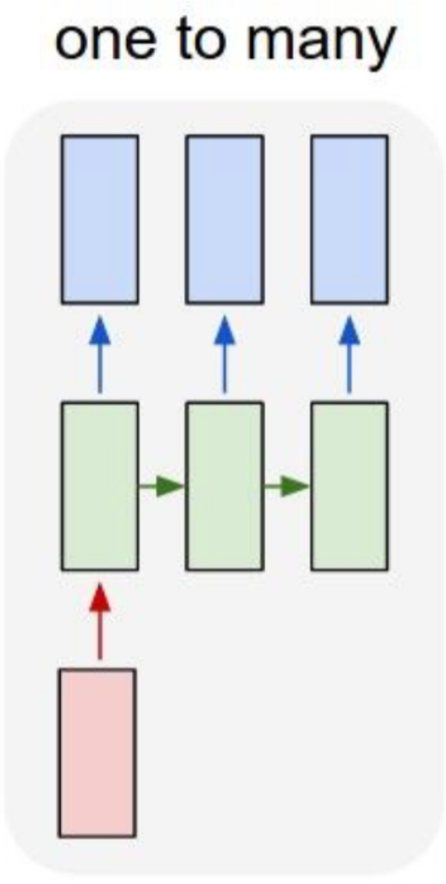


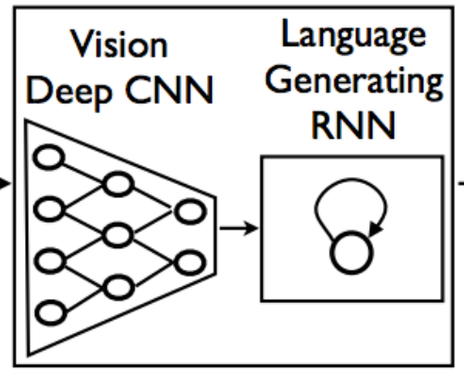
序列样本



- RNN不仅仅能够处理序列输出，也能得到序列输出，这里序列指的是向量的序列。
- RNN学习出来的是程序（状态机），不是函数

典型应用



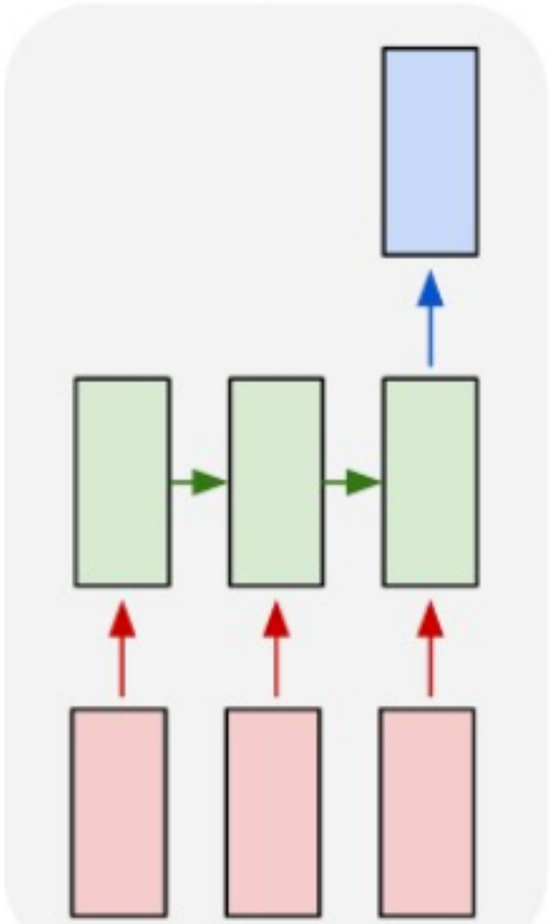


A group of people shopping at an outdoor market.

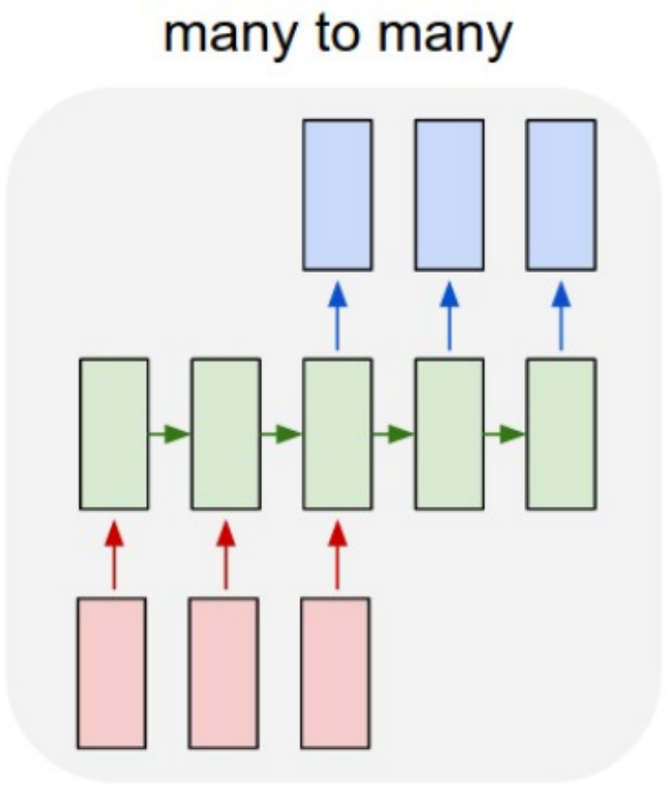
There are many vegetables at the fruit stand.

https://github.com/karpathy/neuraltalk2

many to one



http://vlg.cs.dartmouth.edu/c3d/



Google

翻译

关闭即时翻译

英语 中文 德语 检测语言

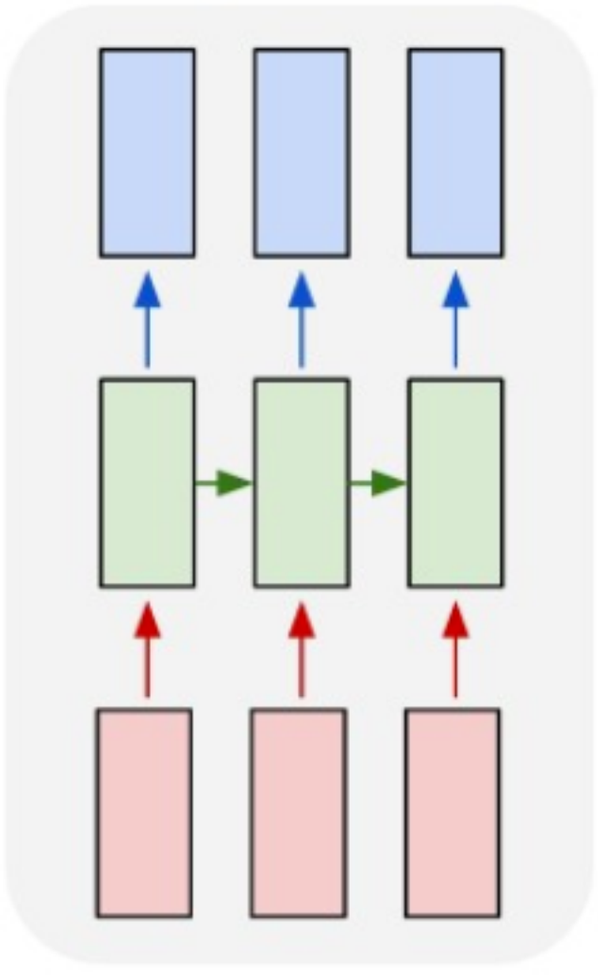
中文(简体) 英语 日语 翻译

too young too simple

太年轻，太单纯

修改建议

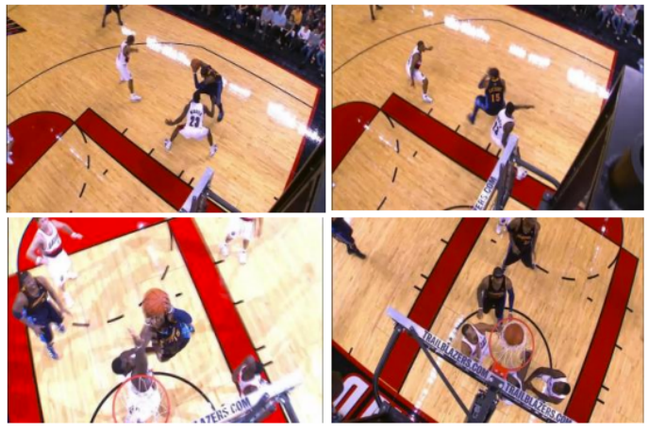
many to many



1. A woman giving speech on news channel.
2. Hillary Clinton gives a speech.
3. Hillary Clinton is making a speech at the conference of mayors.
4. A woman is giving a speech on stage.
5. A lady speak some news on TV.

1. A child is cooking in the kitchen.
2. A girl is putting her finger into a plastic cup containing an egg.
3. Children boil water and get egg whites ready.
4. People make food in a kitchen.
5. A group of people are making food in a kitchen.

1. A white car is drifting.
2. Cars racing on a road surrounded by lots of people.
3. Cars are racing down a narrow road.
4. A race car races along a track.
5. A car is drifting in a fast speed.



1. A player is putting the basketball into the post from distance.
2. The player makes a three-pointer.
3. People are playing basketball.
4. A 3 point shot by someone in a basketball race.
5. A basketball team is playing in front of speculators.

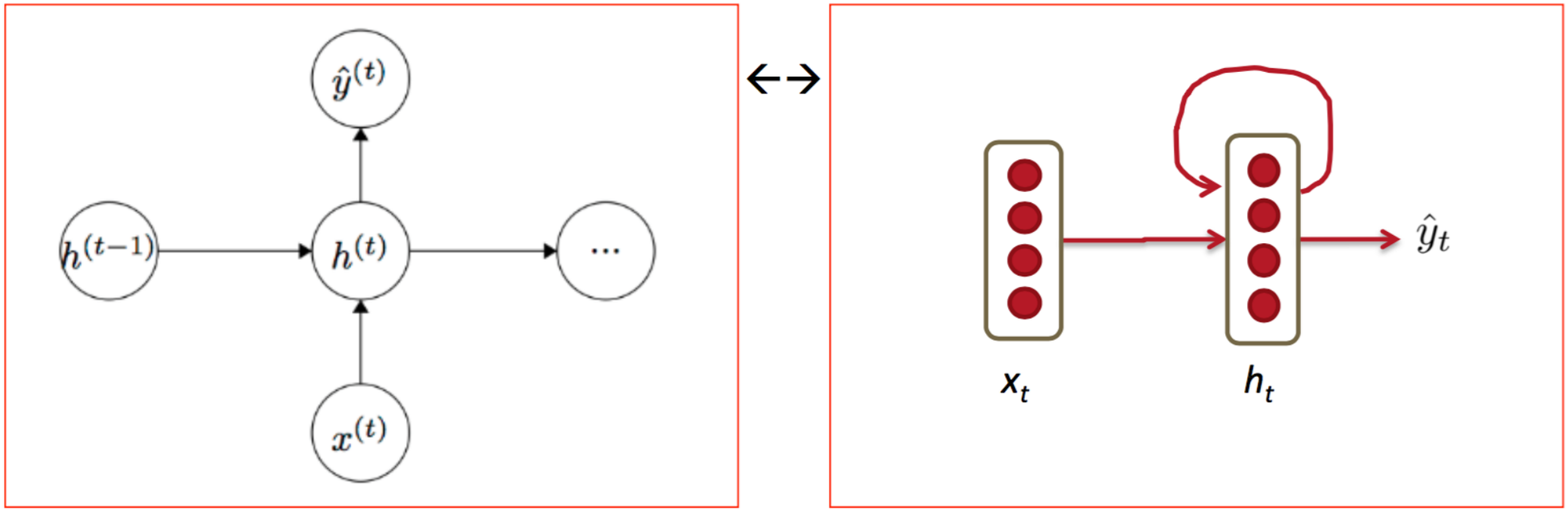
http://research.microsoft.com/apps/pubs/default.aspx?id=264836

序列预测

- 输入的是时间变化向量序列： $x_{t-2}, x_{t-1}, x_t, x_{t+1}, x_{t+2}$
- 在t时刻通过模型来估计 $x_{t+1} = f(x_t, \ldots, x_{t-\tau})$
- 问题：
  - 对内部状态难以建模和观察
  - 对长时间范围的场景(context)难以建模和观察
- 解决方案：引入内部隐含状态变量

$$x_{t+1} = f(x_t, \ldots, x_{t-\tau}, z_t, \ldots, z_{t-\tau})$$
$$z_{t+1} = g(x_{t+1}, \ldots, x_{t-\tau}, z_t, \ldots, z_{t-\tau})$$

序列预测模型



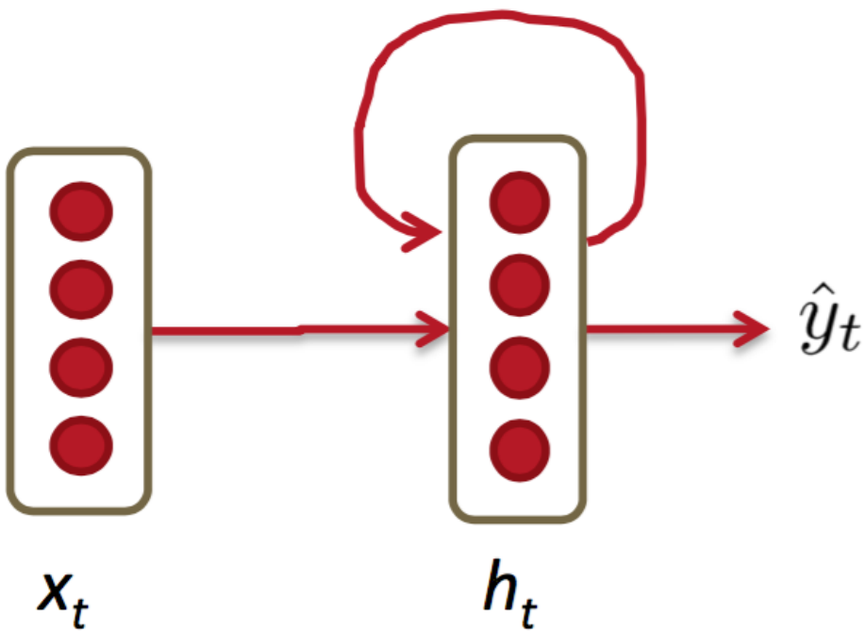
- 输入离散列序列 $x_1, \ldots, x_{t-1}, x_t, x_{t+1}, \ldots, x_T$
- 在时间t的更新计算：

$$h_t = \sigma\left(W^{(hh)} h_{t-1} + W^{(hx)} x_{[t]}\right)$$

$$\hat{y}_t = \text{softmax}\left(W^{(S)} h_t\right)$$

- 预测计算 $\hat{P}(x_{t+1} = v_j \mid x_t, \ldots, x_1) = \hat{y}_{t,j}$

序列预测模型



$$h_t = \sigma\left(W^{(hh)}h_{t-1} + W^{(hx)}x_{[t]}\right)$$

$$\hat{y}_t = \text{softmax}\left(W^{(S)}h_t\right)$$

$$\hat{P}(x_{t+1} = v_j \mid x_t, \ldots, x_1) = \hat{y}_{t,j}$$

- 在整个计算过程中，W保持不变
- h_0在0时刻初始化

$$h_0 \in \mathbb{R}^{D_h}$$

$$W^{(hh)} \in \mathbb{R}^{D_h \times D_h}$$

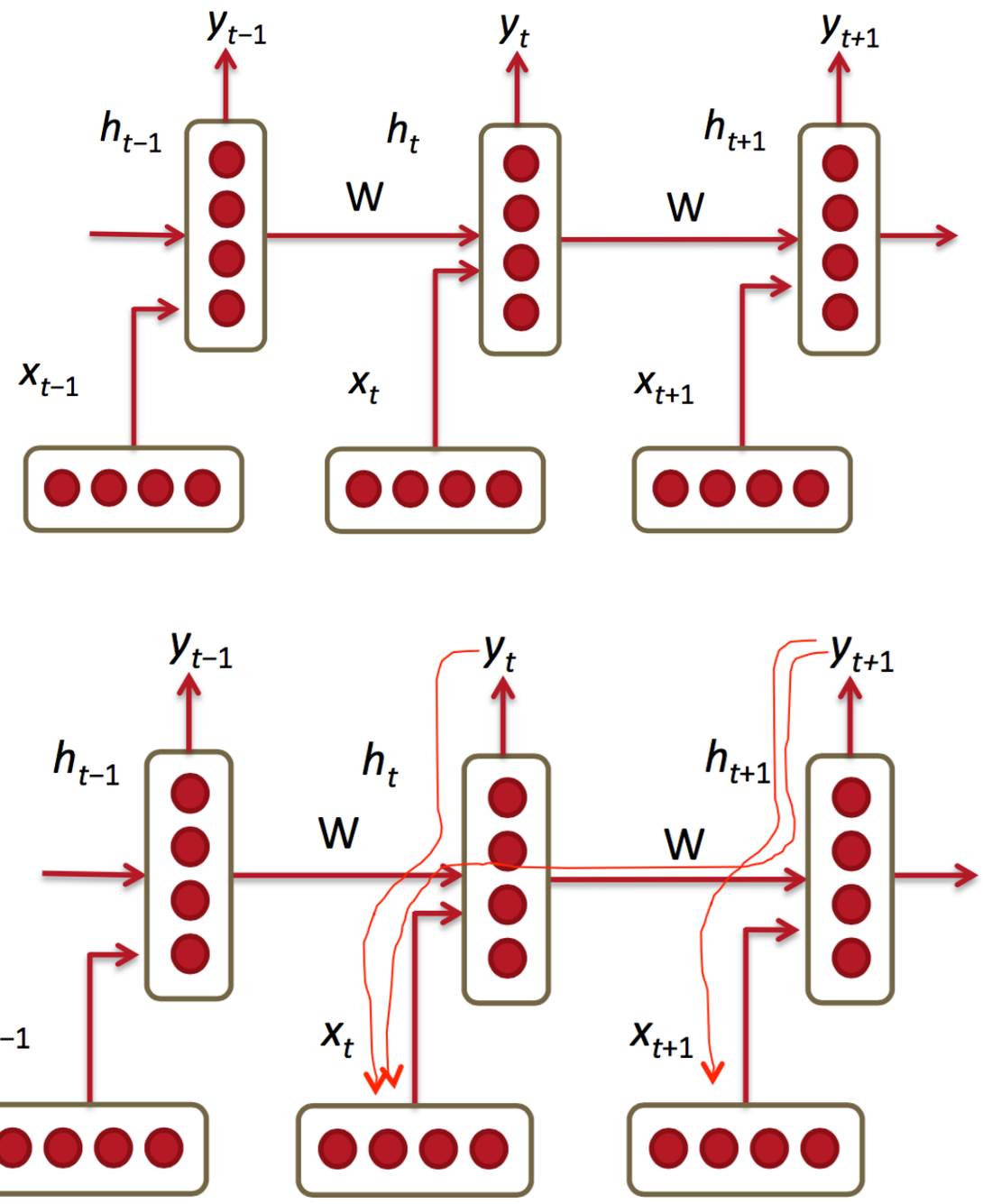
$$W^{(hx)} \in \mathbb{R}^{D_h \times d}$$

$$W^{(S)} \in \mathbb{R}^{|V| \times D_h}$$

$$\hat{y} \in \mathbb{R}^{|V|}$$

RNN训练（1）

- 前向计算，相同的W矩阵需要乘以多次
- 多步之前的输入x，会影响当前的输出
- 在后向计算的时候，同样相同的矩阵也会乘以多次



BPTT算法－BackProp Through Time

- RNN前向计算

$$
\begin{aligned}
h_t &= W f(h_{t-1}) + W^{(hx)} x_{[t]} \\
\hat{y}_t &= W^{(S)} f(h_t)
\end{aligned}
$$

- 计算W的偏导，需要把所有Time Step加起来

$$
\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W}
$$

- 应用链式规则

$$
\frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}
$$

BPTT算法：计算实现

- 计算目标：

$$\frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}$$

- 已知：

$$h_t = W f(h_{t-1}) + W^{(hx)} x_{[t]}$$

- 因此：

$$\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}}$$

$$\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} = \prod_{j=k+1}^{t} W^T \mathrm{diag}[f'(h_{j-1})]$$

$$\mathrm{diag}(z) = \begin{pmatrix} z_1 & & & \\ & z_2 & & 0 \\ & & \ddots & \\ & 0 & & z_{n-1} \\ & & & & z_n \end{pmatrix}$$

这里使用是向量微分

BPTT算法：梯度 vanishing/exploding 现象分析

- 已知：$$\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} = \prod_{j=k+1}^{t} W^T \text{diag}[f'(h_{j-1})]$$

- 根据 ‖XY‖ ≤ ‖X‖ ‖Y‖，可以知道：

$$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| \leq \|W^T\| \|\text{diag}[f'(h_{j-1})]\| \leq \beta_W \beta_h$$

- 其中Beta代表上限，因此：

$$\left\| \frac{\partial h_t}{\partial h_k} \right\| = \left\| \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} \right\| \leq (\beta_W \beta_h)^{t-k}$$

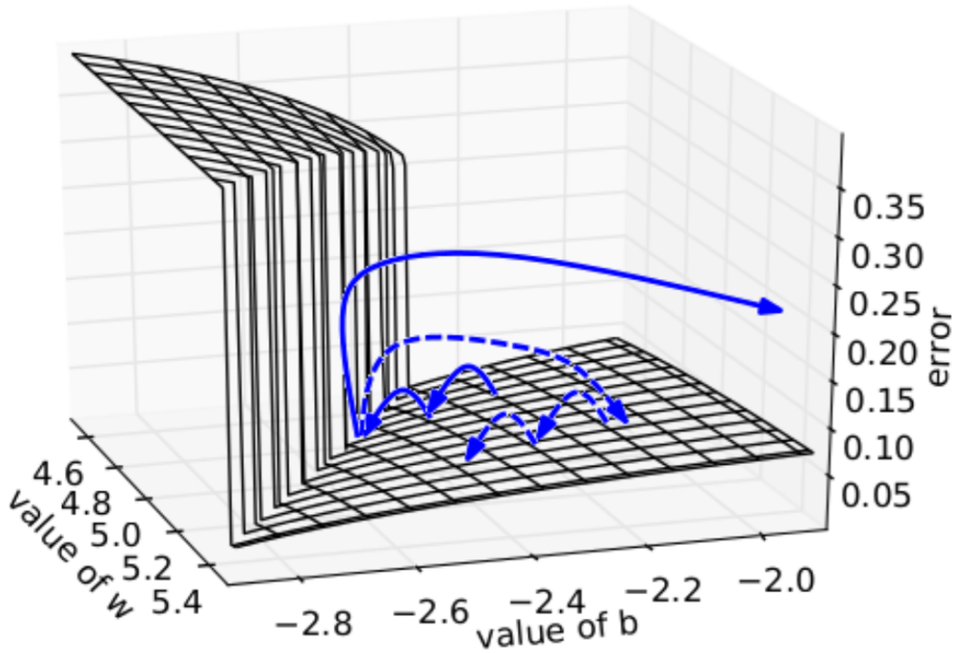
This can become very small or very large quickly [Bengio et al 1994], and the locality assumption of gradient descent breaks down. → **Vanishing or exploding gradient**

BPTT算法：解决方案

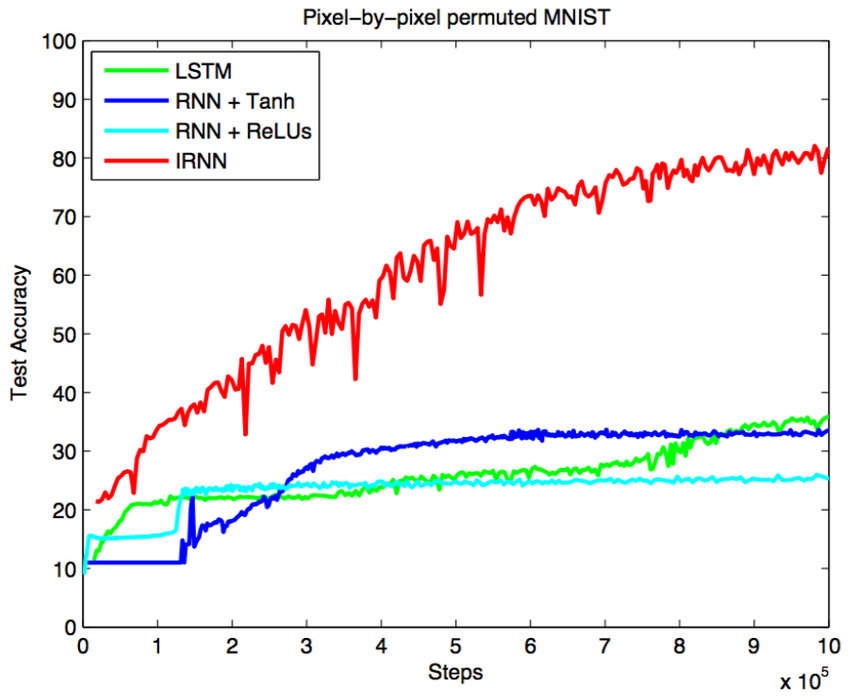
- Clipping

Algorithm 1 Pseudo-code for norm clipping the gradients whenever they explode

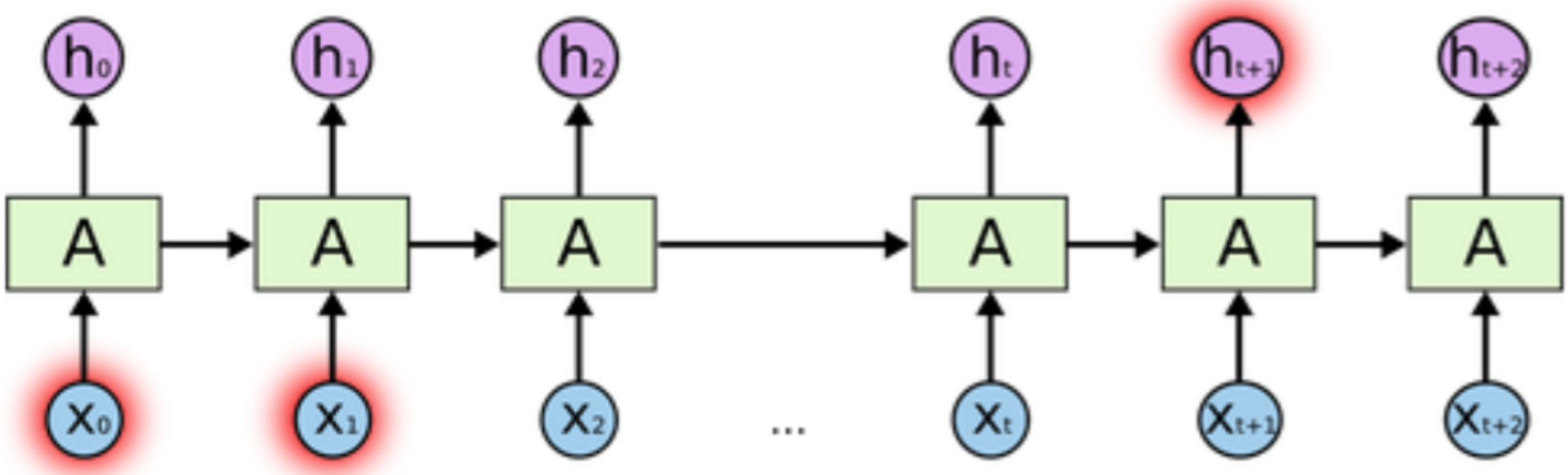
$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$
end if



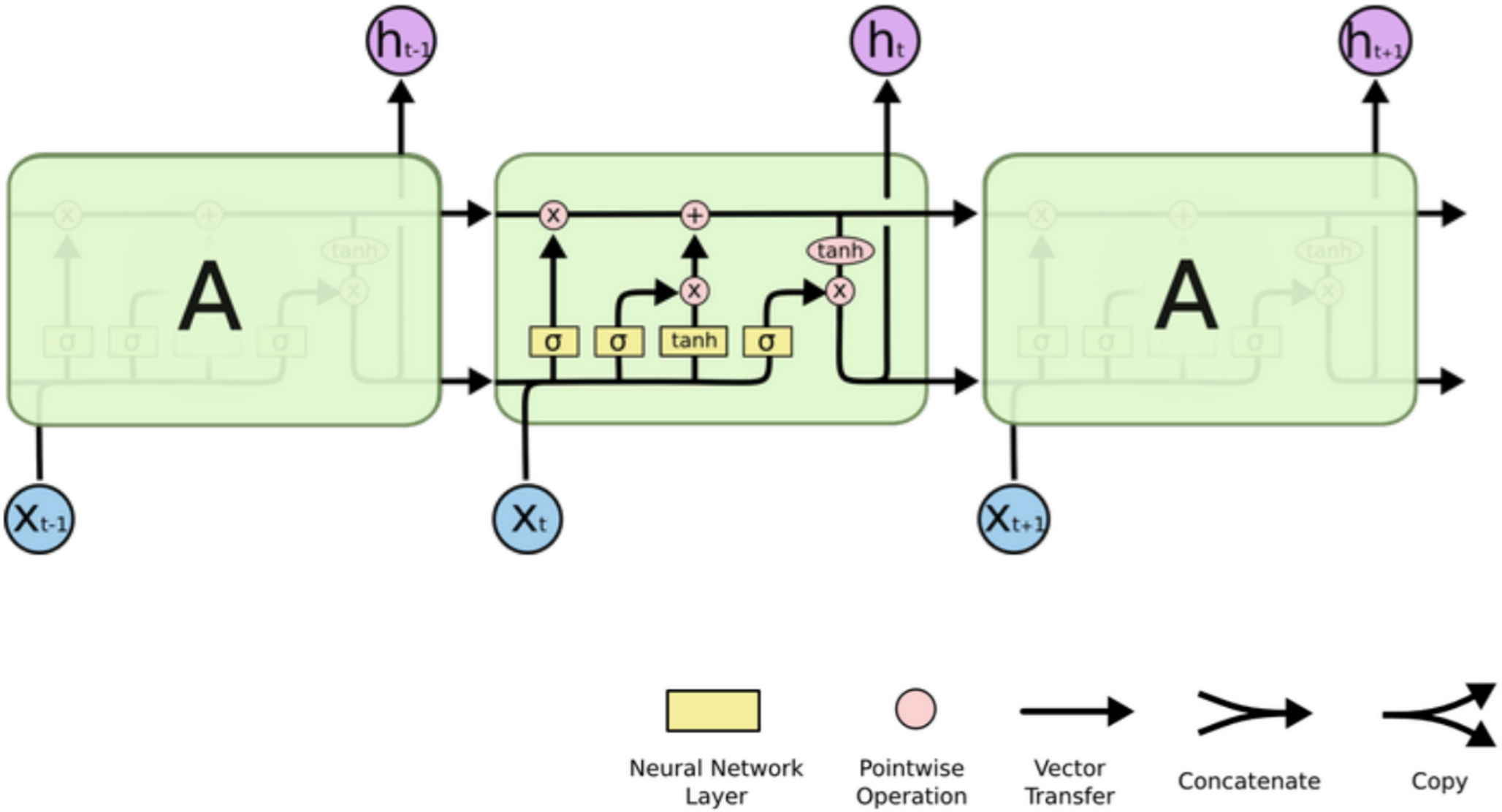
- W初始化为I，使用Relu替换Tanh



LSTM （Long Short Term Memory) Cell



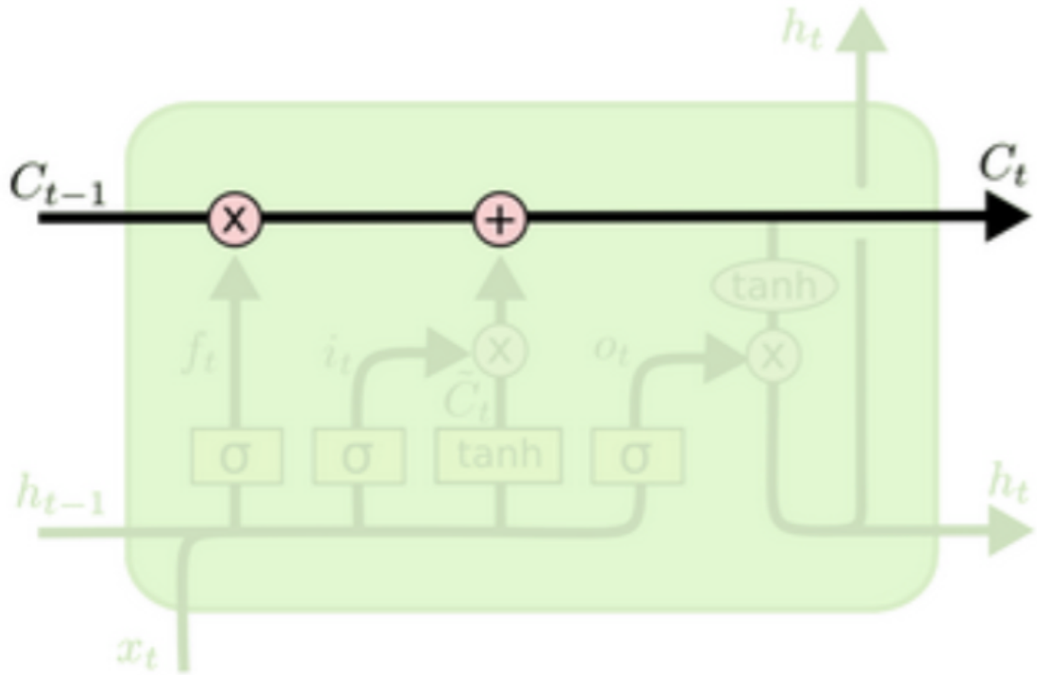
LSTM （Long Short Term Memory) Cell



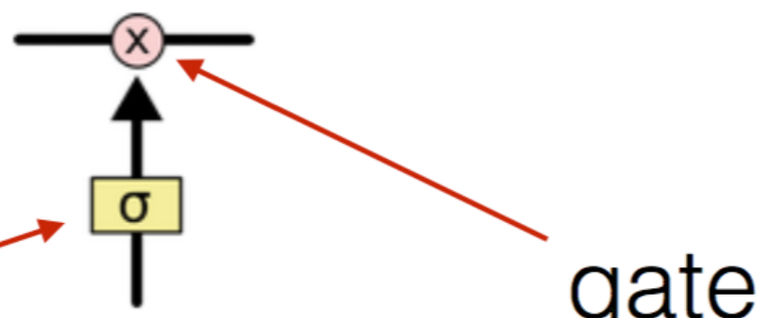
应用最为广泛、成功的RNN

LSTM：cell state

- 可以长期保存某个状态，cell state值通过forget gate控制实现保留多少“老”的状态

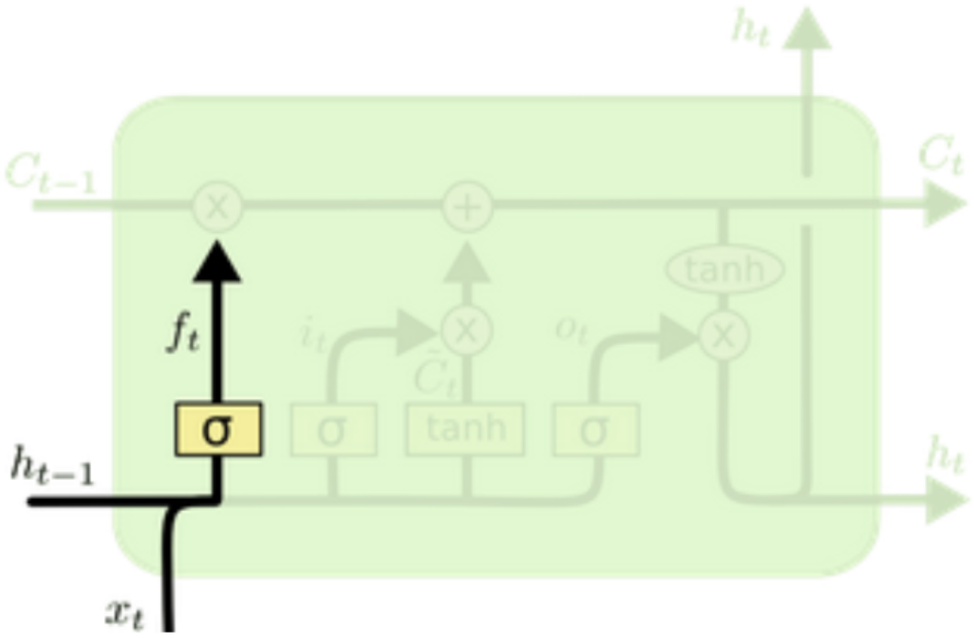


- Layer 把 输入维度x变成输出维度h

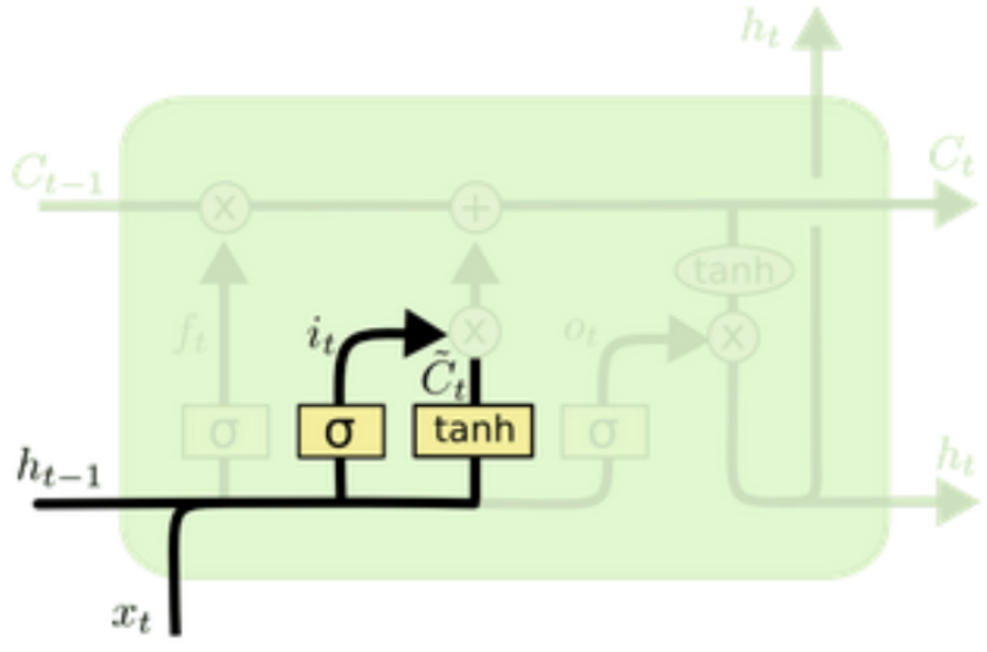


sigmoid **Layer** with weights and bias

LSTM: forget / input unit



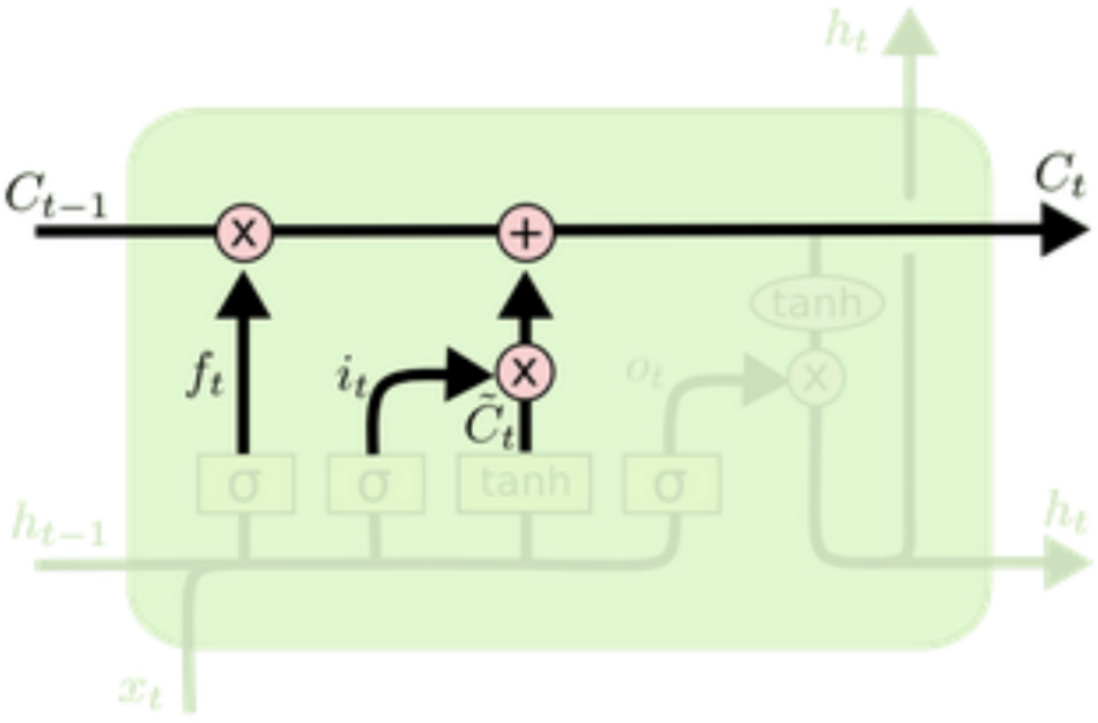
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$



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

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

LSTM: update cell



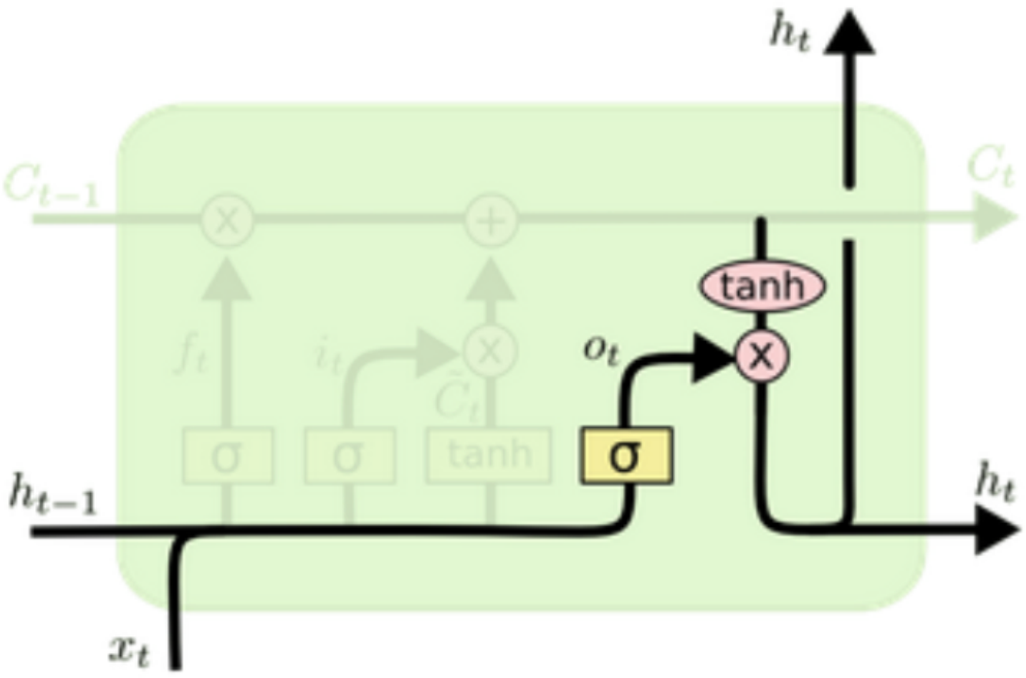
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$f_t \sim (0,1)$$

$$i_t \sim (0,1)$$

$$C_t \sim (-1,1)$$

LSTM: output



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

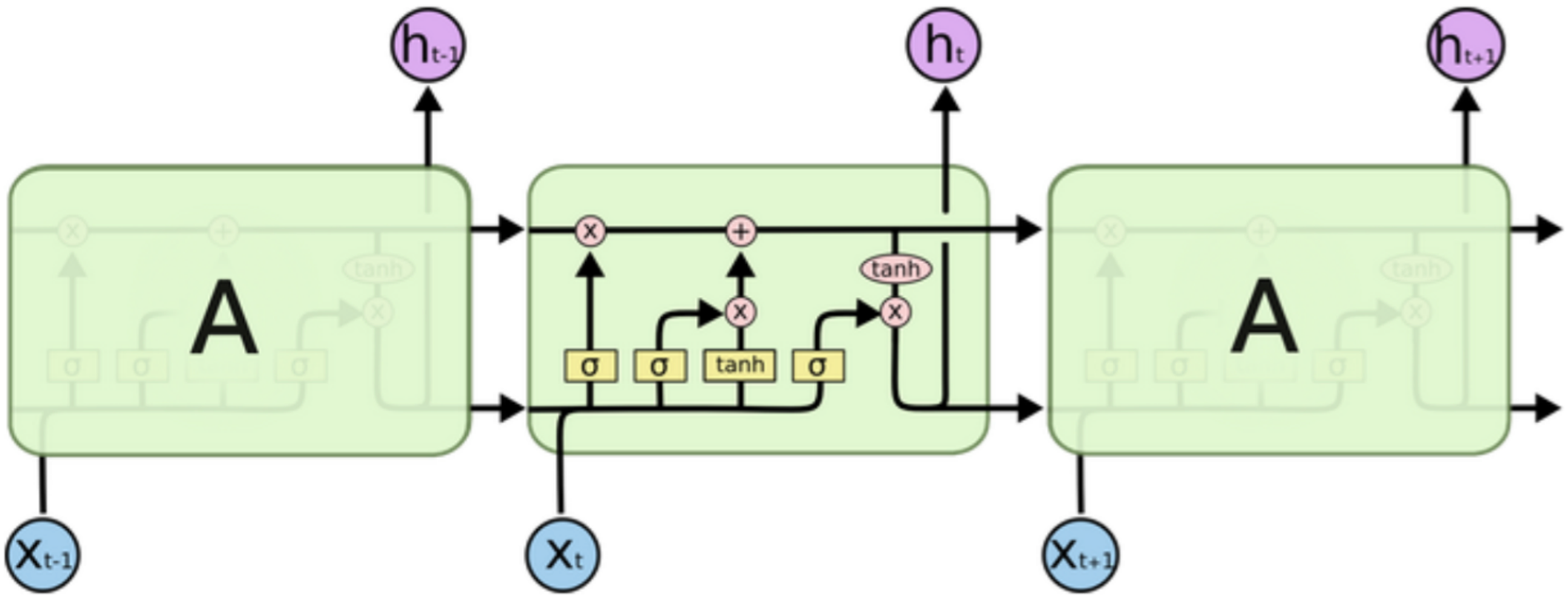
$$h_t = o_t * \tanh\left(C_t\right)$$

o_t~ (0,1)

C_t~ (-1,1)

h_t~ (-0.761,0.761)

LSTM



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

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

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

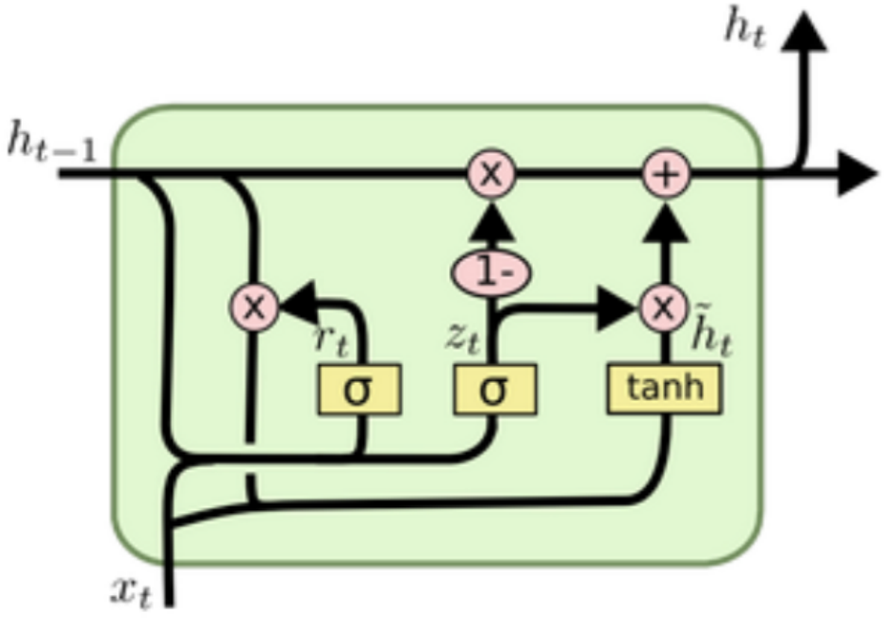
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

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

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

LSTM 其它变形

- Gated Recurrent Unit



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

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

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

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

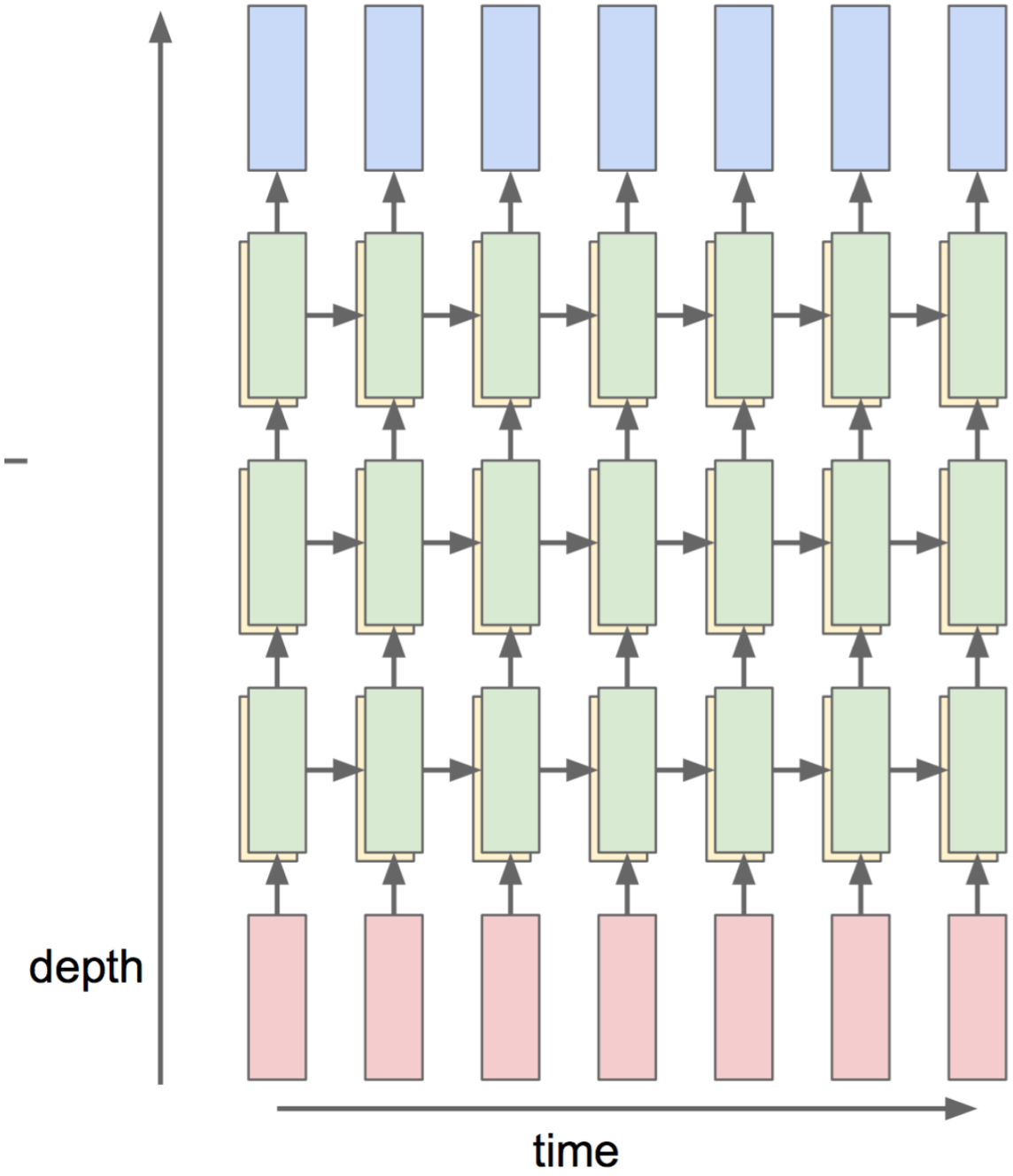
LSTM的训练

- 不需要记忆复杂的BPTT公式，利用时序展开，构造层次关系，可以开发复杂的BPTT算法

- LSTM 具备 定抑制梯度 vanishing/exploding 特性

使用LSTM

- 将多个LSTM单元组合为层
- 网络中有多层
- 复杂的结构能够处理更大范围的动态性



LSTM网络模拟一个有限状态机

这是一个非常简单的序列样本：在一个4x4的数字矩阵里，有一个位置是空数字。 输入序列就是移动这个空位置，输出就是被移动的数字。

如图所示：

输入序列，表示移动方向，输出表示移动的数字，如：

```
LEFT: 15 UP: 11 LEFT: 10 RIGHT: 10 DOWN: 11
```

Next: 8.RNN应用