



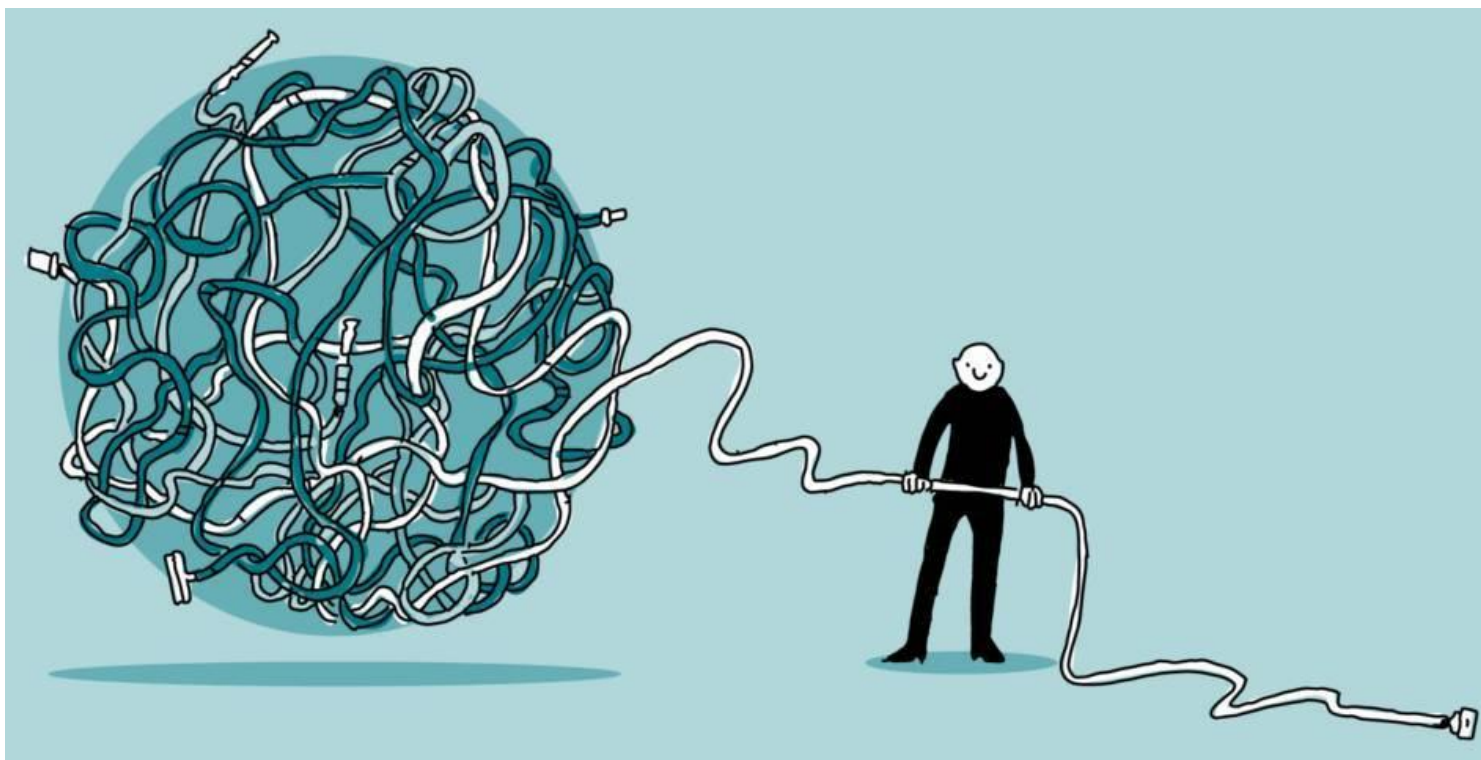
Recurrent Neural Network (RNN)

汇报人：扣扣



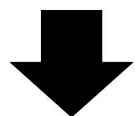
大纲

- 循环神经网络应用
- 简单版本RNN
- 简单版本RNN存在问题
- LSTM基本结构解析
- GRU基本结构解析
- LSTM代码示例
- 相关面试题
- 作业

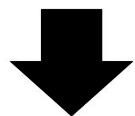


应用：情感分类

这家店的装修不错。

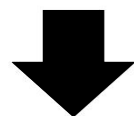


RNN

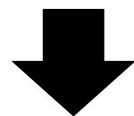


正面

这家店的装修不错，但牛排是过期的。



RNN

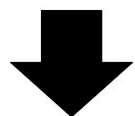


负面

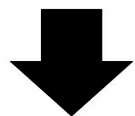
记忆能力

应用：chatbot

你好！

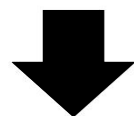


RNN

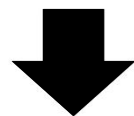


你好！

还是你好！



RNN

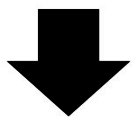


哈哈，是吧。

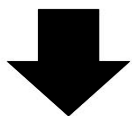
记忆能力

应用：诗词生成

锄

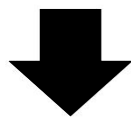


RNN

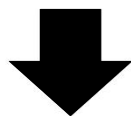


禾日当午

汗



RNN

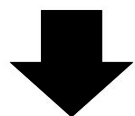


滴禾下土

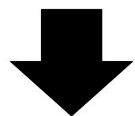
记忆能力

应用：流畅性评价

今天是周六

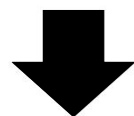


RNN

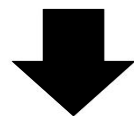


ppl: 100

是今天周六



RNN



ppl: 1000

$$\begin{aligned} \text{perplexity}(S) &= p(w_1, w_2, w_3, \dots, w_m)^{-1/m} \\ &= \sqrt[m]{\prod_{i=1}^m \frac{1}{p(w_i | w_1, w_2, \dots, w_{i-1})}} \end{aligned} \quad (1)$$

记忆能力

应用：填槽（slot）填充

我想订个票，明天晚上8点到上海。

RNN

Slot

目的地：上海
到达时间：明天晚上8点

我想订个票，明天晚上8点从上海走。

RNN

Slot

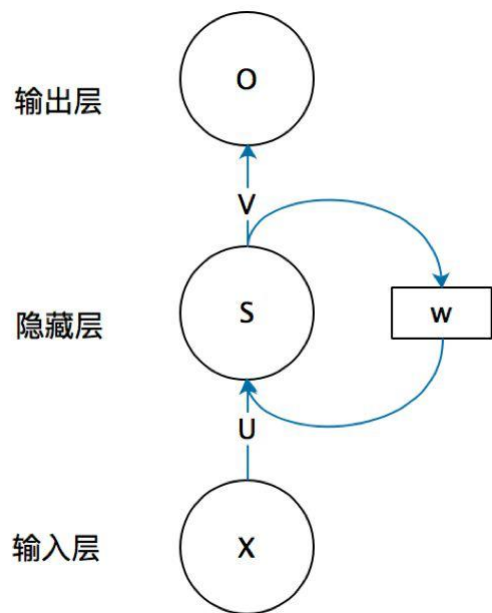
出发地：上海
出发时间：明天晚上8点

为什么需要RNN

某些任务需要能够更好地处理序列信息，
即前面的输入和后面的输入是有关系的，
因此需要模型具备记忆能力。

NeuHub
AI Links All

RNN



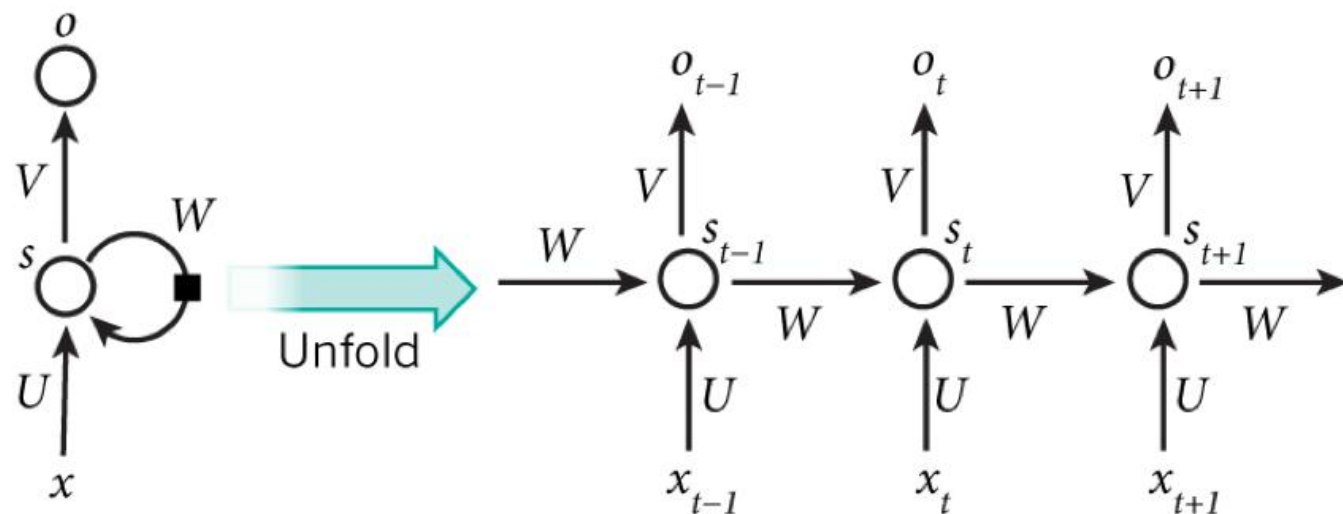
x 是一个向量，它表示某个时刻输入层的值（多个时刻的 x 组成一个序列）； s 是一个向量，它表示隐藏层的值；

U 是输入层到隐藏层的权重矩阵， o 也是一个向量，它表示输出层的值； V 是隐藏层到输出层的权重矩阵。

W 是什么：循环神经网络的隐藏层的值 s 不仅仅取决于当前这次的输入 x ，还取决于上一次隐藏层的值 s 。权重矩阵 W 就是隐藏层上一次的值作为这一次的输入的权重。

NeuHub
AI Links All

RNN



$$o_t = g(Vs_t) \quad (\text{式1})$$

$$s_t = f(Ux_t + Ws_{t-1}) \quad (\text{式2})$$

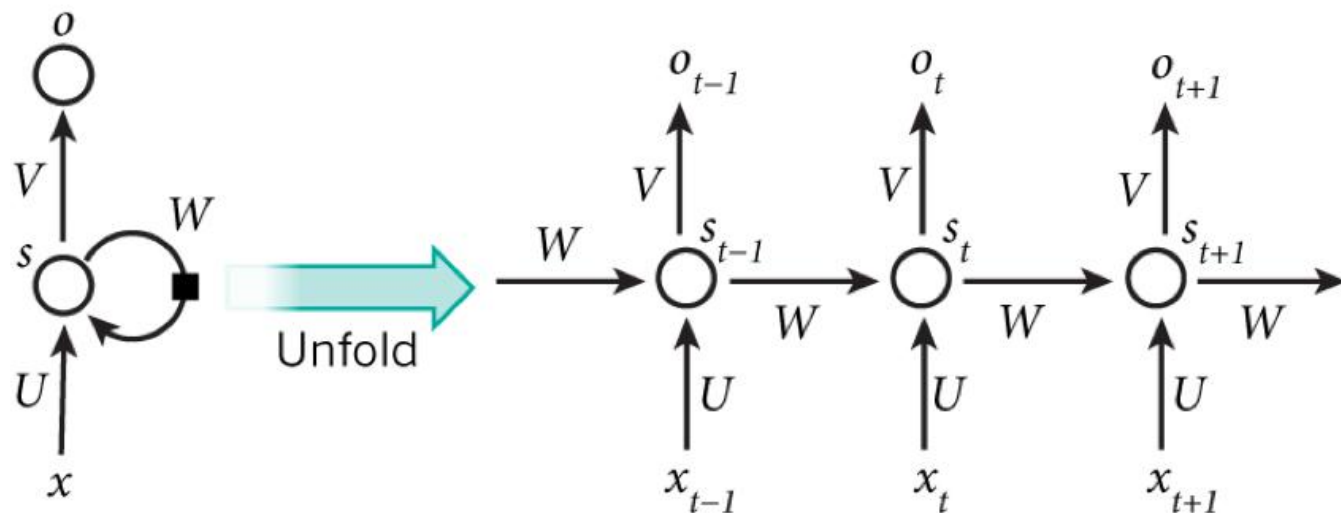
第一个式子是输出层的计算公式，输出层是一个全连接层，也就是它的每个节点都和隐藏层的每个节点相连。 V 是输出层的权重矩阵， g 是激活函数。

第二个式子是隐藏层的计算公式，它是循环层。 U 是输入 x 的权重矩阵， W 是上一次的 s 的权重矩阵， f 是激活函数。

从上面的公式我们可以看出，循环层和全连接层的区别就是循环层多了一个权重矩阵 W 。

注意： W 、 V 、 U 为共享权重。

RNN



思考:

- W, V, U有多少个?
- 假如x为4维向量, s为3维向量, U, W的大小分别为多少?

$$o_t = g(Vs_t) \quad (\text{式1})$$

$$s_t = f(Ux_t + Ws_{t-1}) \quad (\text{式2})$$

NeuHub
AI Links All

RNN

$$o_t = g(Vs_t) \quad (\text{式1})$$

$$s_t = f(Ux_t + Ws_{t-1}) \quad (\text{式2})$$

思考:

- W , V , U 有多少个?
- 假如 x 为 m 维向量, s 为 n 维向量, U, W 的大小分别为多少?

$$\begin{bmatrix} s_1^t \\ s_2^t \\ \vdots \\ s_n^t \end{bmatrix} = f\left(\begin{bmatrix} u_{11} & u_{12} & \dots & u_{1m} \\ u_{21} & u_{22} & \dots & u_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ u_{n1} & u_{n2} & \dots & u_{nm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} + \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \dots & w_{nn} \end{bmatrix} \begin{bmatrix} s_1^{t-1} \\ s_2^{t-1} \\ \vdots \\ s_n^{t-1} \end{bmatrix} \right)$$

RNN

$$o_t = g(Vs_t) \quad (\text{式1})$$

$$s_t = f(Ux_t + Ws_{t-1}) \quad (\text{式2})$$

$$\begin{aligned} o_t &= g(Vs_t) \\ &= Vf(Ux_t + Ws_{t-1}) \\ &= Vf(Ux_t + Wf(Ux_{t-1} + Ws_{t-2})) \\ &= Vf(Ux_t + Wf(Ux_{t-1} + Wf(Ux_{t-2} + Ws_{t-3}))) \\ &= Vf(Ux_t + Wf(Ux_{t-1} + Wf(Ux_{t-2} + Wf(Ux_{t-3} + \dots)))) \end{aligned}$$

总结：可以看出，循环神经网络的输出值，受前面历次输入值 x_1 、 $x_2 \dots x_{t-1}$ 影响，这就是为什么循环神经网络具备记忆能力的原因。

RNN

- 很多现实中的数据都是序列数据，前后相关，需要具备记忆能力的模型来建模。
- 比如，当我们在理解一句话意思时，孤立的理解这句话的每个词是不够的，我们需要处理这些词连接起来的整个序列；当我们处理视频的时候，我们也不能只单独的去分析每一帧，而要分析这些帧连接起来的整个序列。

NeuHub
AI Links All

双向RNN

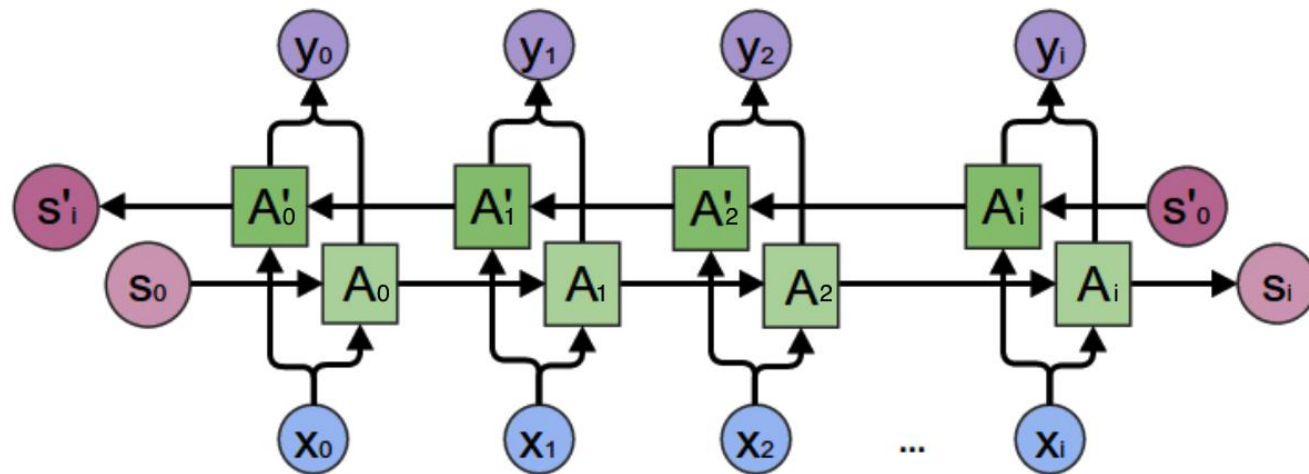
完形填空

Jim was a worker. One of his feet was bigger than ____1____. He couldn't ____2____ the right shoes ____3____ his feet. One day his friend Mike said to him "____4____ don't you go to a shoemaker? A good shoemaker can ____5____ you the right shoes." ____6____ Jim went to the shoemaker near Mike's home, very soon the shoemaker finished the work. Jim ____7____ the shoes and wasn't happy. He ____8____ the shoemaker. "You aren't a ____9____ shoemaker! I wanted you to make me one shoe bigger than the other, ____10____ you made me one shoe smaller than the other."

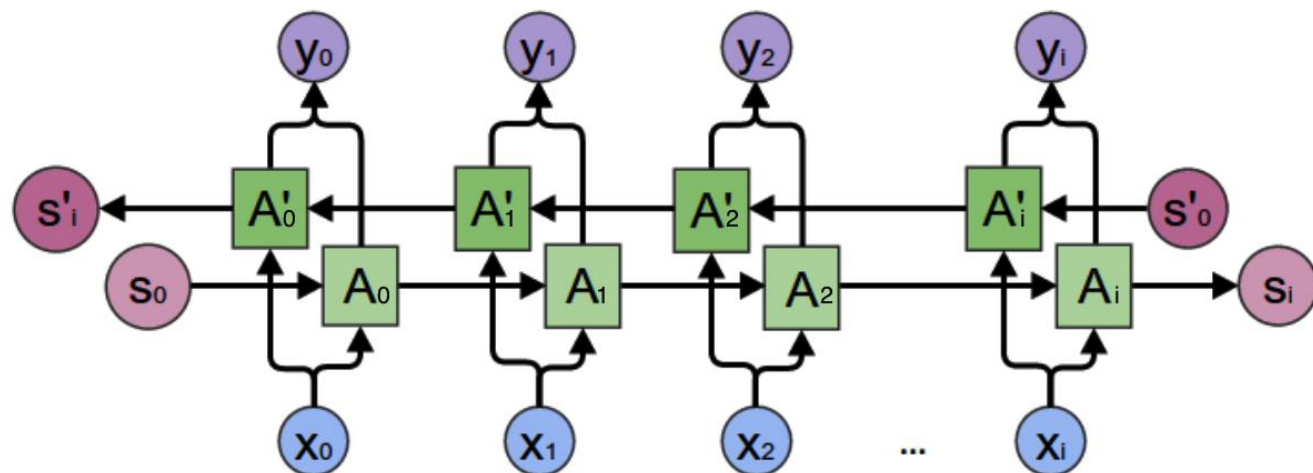
- ()1. A. other B. the other C. another D. that one
 ()2. A. see B. look for C. find D. find out
 ()3. A. for B. with C. on D. of
 ()4. A. When B. Where C. How D. Why
 ()5. A. make B. pass C. do D. give
 ()6. A. Then B. But C. So D. Because
 ()7. A. saw B. watched C. found D. looked at
 ()8. A. said B. said to C. spoke D. talked
 ()9. A. good B. bad C. right D. kind
 ()10. A. then B. and C. but D. so

阅读理解

简单版本的循环神经网络无法对此进行建模，因此，我们需要双向循环神经网络。



双向RNN



$$y_2 = g(VA_2 + V'A'_2)$$

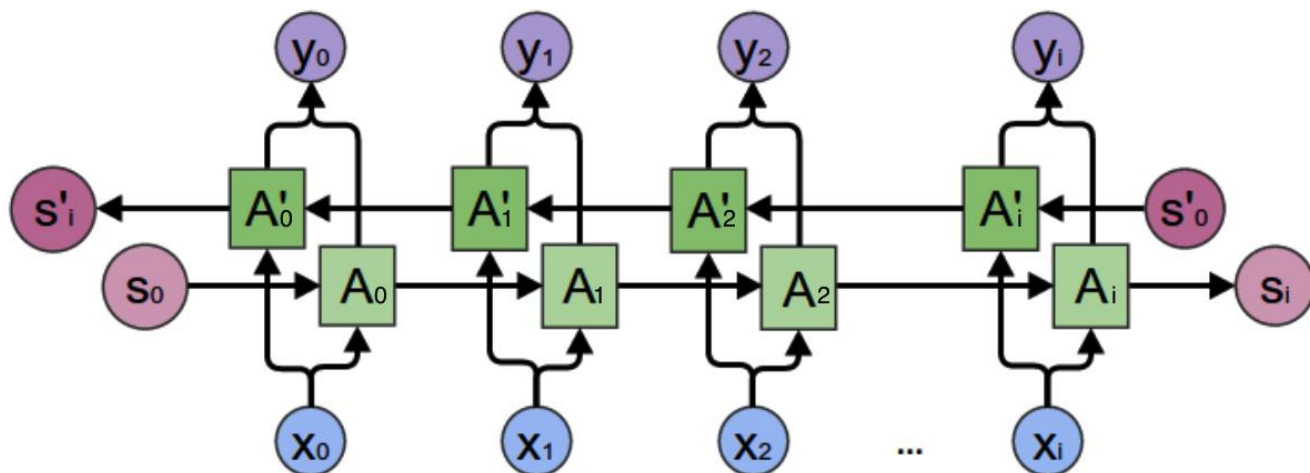
$$A_2 = f(WA_1 + Ux_2)$$

$$A'_2 = f(W'A'_3 + U'x_2)$$

从上图可看出，双向循环神经网络的隐藏层要保存两个值，一个A参与正向计算，另一个值A'参与反向计算。如，最终 y_2 的输出值取决于 A_2 和 A'_2 。

规律：正向计算时，隐藏层的值 A_t 与 A_{t-1} 有关；反向计算时，隐藏层的值 A'_t 与 A'_{t+1} 有关；最终的输出取决于正向和反向计算的加和。

双向RNN



正向计算和反向计算不共享权重，也就是说U和U'、W和W'、V和V'都是不同的权重矩阵。

$$y_2 = g(VA_2 + V'A'_2)$$

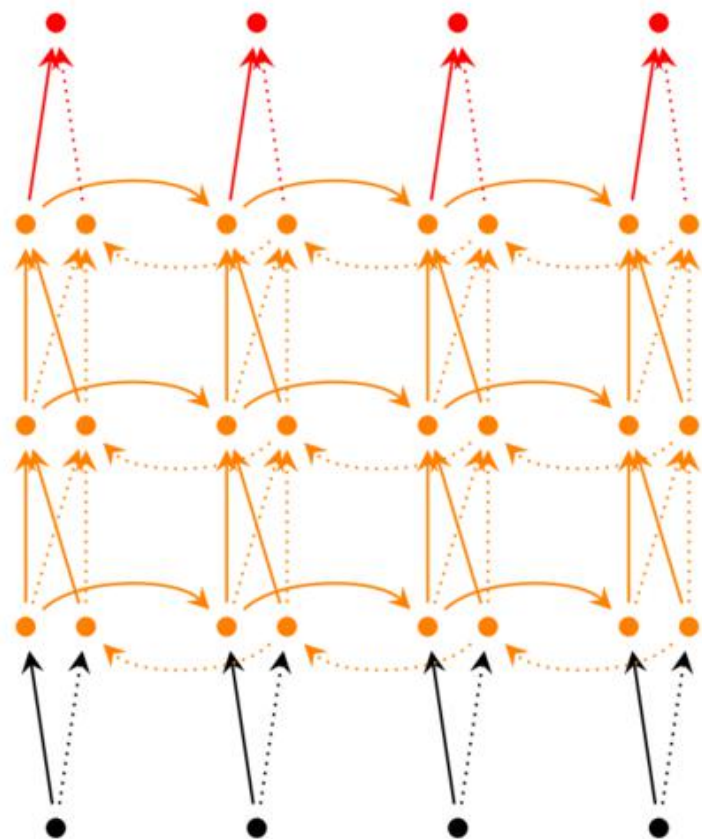
$$A_2 = f(WA_1 + Ux_2)$$

$$A'_2 = f(W'A'_3 + U'x_2)$$



$$o_t = g(Vs_t + V's'_t)$$

深度RNN



前面介绍的循环神经网络只有一个隐藏层，当然也可以根据任务及数据情况，堆叠两个以上的隐藏层，这样就得到了深度循环神经网络。

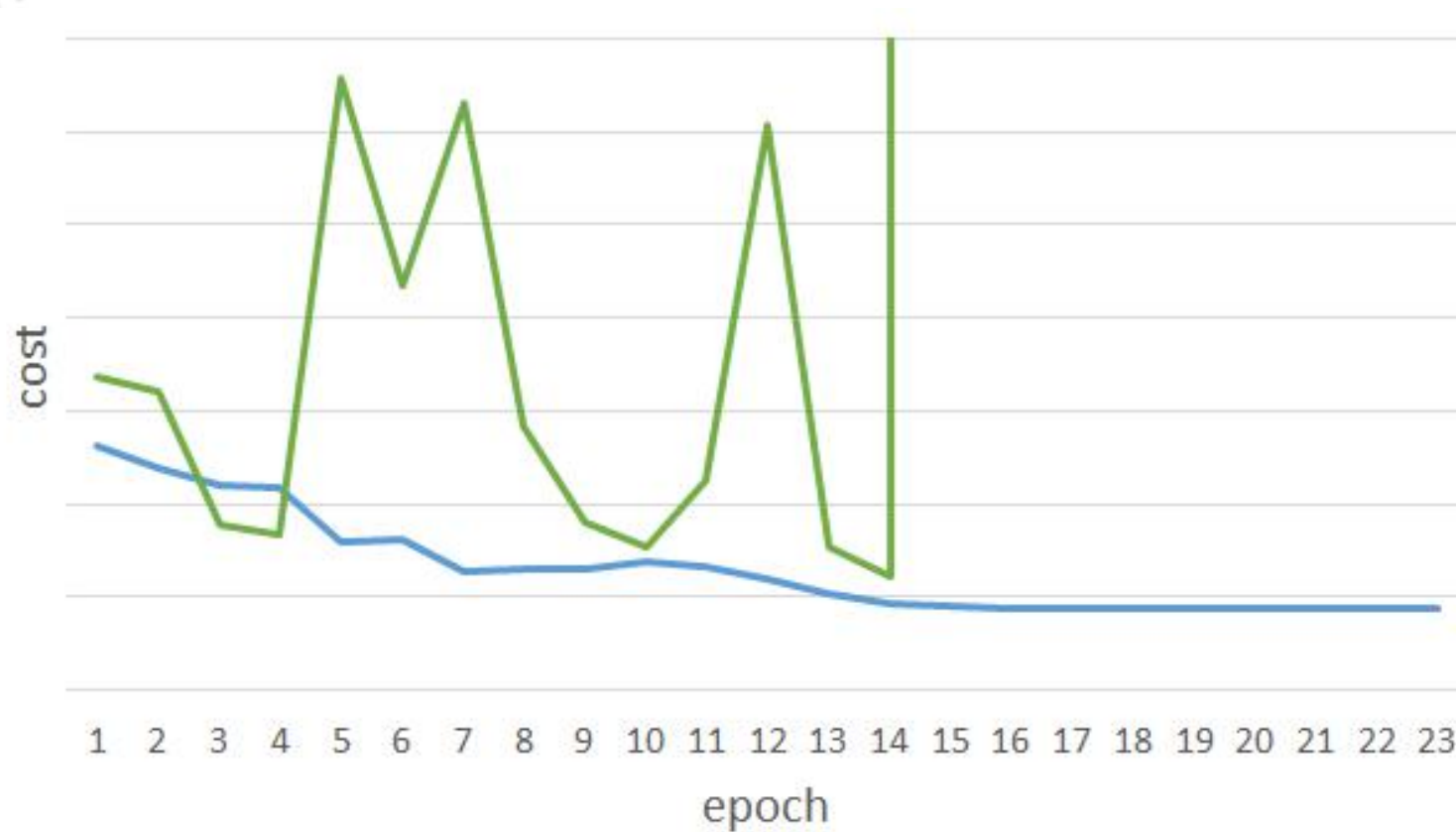
RNN与梯度消失、梯度爆炸



&



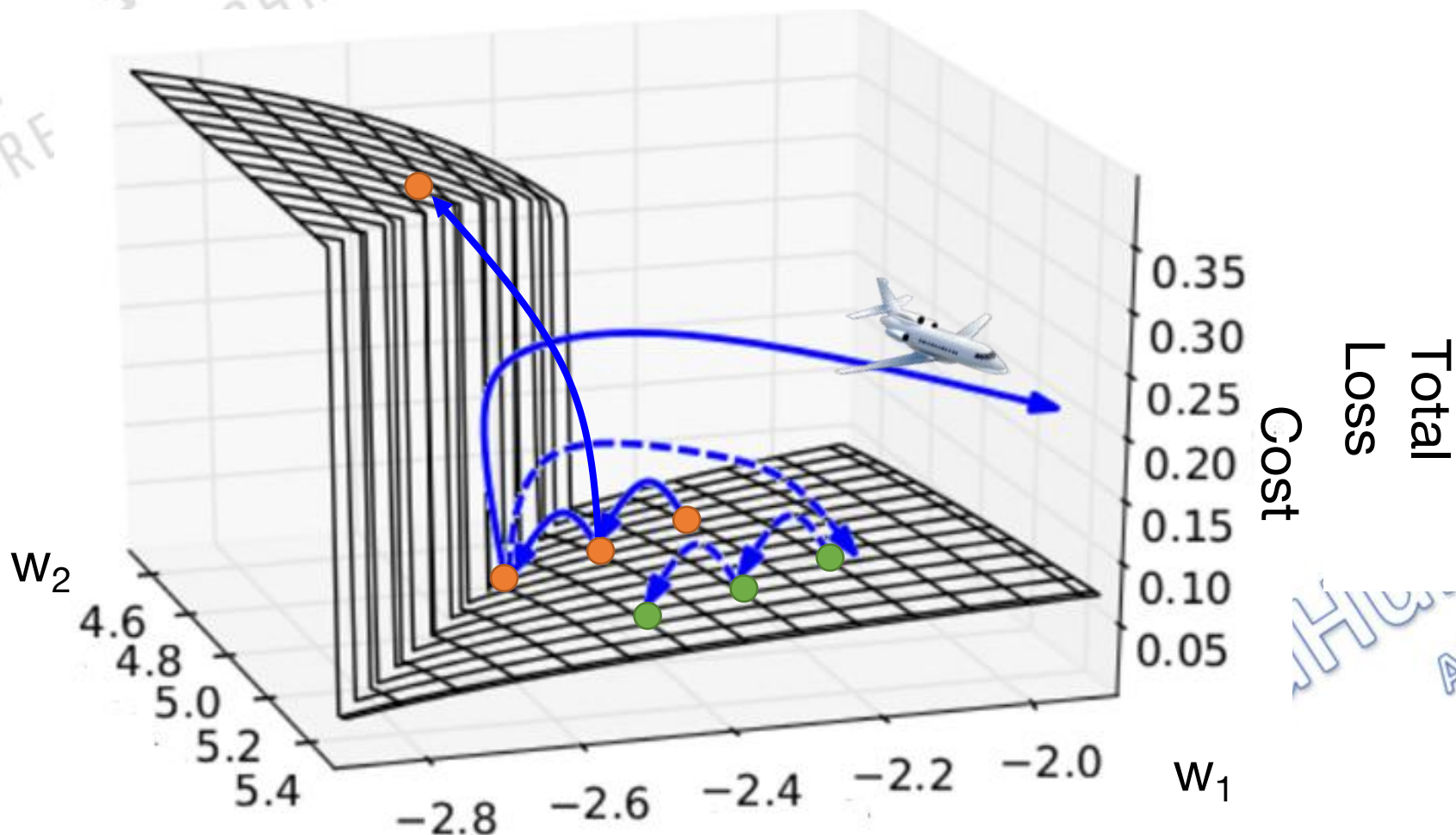
- RNN很难训练



RNN与梯度消失、梯度爆炸



&



RNN与梯度消失、梯度爆炸

RNN前向传导公式为：

$$s_t = g(Ux_t + Ws_{t-1})$$
$$o_t = f(Vs_t)$$

假设t 最大为 3

损失函数为：

$$L = q(o_3, y)$$

当 $t = 1$ ：

$$\text{状态： } s_1 = g(Ux_1 + Ws_0)$$

$$\text{输出： } o_1 = f(Vg(Ux_1 + Ws_0))$$

当 $t = 2$ ：

$$\text{状态： } s_2 = g(Ux_2 + Ws_1)$$

$$\text{输出： } o_2 = f(Vg(Ux_2 + Ws_1)) = f(Vg(Ux_2 + Wg(Ux_1 + Ws_0)))$$

当 $t = 3$ ：

$$\text{状态： } s_3 = g(Ux_3 + Ws_2)$$

$$\text{输出： } o_3 = f(Vg(Ux_3 + Wg(Ux_2 + Wg(Ux_1 + Ws_0))))$$

RNN与梯度消失、梯度爆炸

当 $t = 1$:

状态: $s_1 = g(Ux_1 + Ws_0)$

输出: $o_1 = f(Vg(Ux_1 + Ws_0))$

当 $t = 2$:

状态: $s_2 = g(Ux_2 + Ws_1)$

输出: $o_2 = f(Vg(Ux_2 + Ws_1)) = f(Vg(Ux_2 + Wg(Ux_1 + Ws_0)))$

当 $t = 3$:

状态: $s_3 = g(Ux_3 + Ws_2)$

输出: $o_3 = f(Vg(Ux_3 + Wg(Ux_2 + Wg(Ux_1 + Ws_0))))$

$$L = q(o_3, y)$$

$$\begin{aligned} \frac{\partial L}{\partial U} &= \frac{\partial L}{\partial o_3} \frac{\partial o_3}{\partial s_3} \frac{\partial s_3}{\partial U} + \frac{\partial L}{\partial o_3} \frac{\partial o_3}{\partial s_3} \frac{\partial s_3}{\partial s_2} \frac{\partial s_2}{\partial U} + \frac{\partial L}{\partial o_3} \frac{\partial o_3}{\partial s_3} \frac{\partial s_3}{\partial s_2} \frac{\partial s_2}{\partial s_1} \frac{\partial s_1}{\partial U} \\ &= \sum_{k=1}^3 \frac{\partial L}{\partial o_3} \frac{\partial o_3}{\partial s_3} \left(\prod_{j=k+1}^3 \frac{\partial s_j}{\partial s_{j-1}} \right) \frac{\partial s_k}{\partial U} \end{aligned}$$



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

$$\prod_{j=k+1}^3 \tanh'W$$

NeuHub
AI Links All

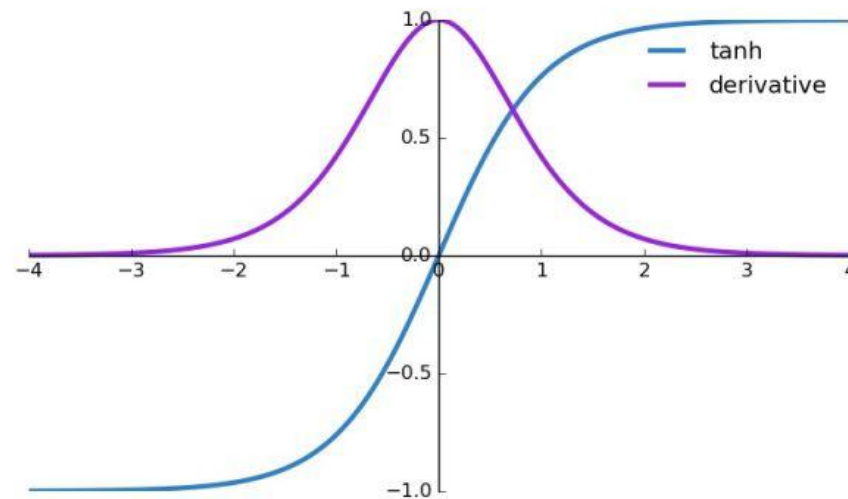
RNN与梯度消失、梯度爆炸

$$\begin{aligned}\frac{\partial L}{\partial U} &= \frac{\partial L}{\partial o_3} \frac{\partial o_3}{\partial s_3} \frac{\partial s_3}{\partial U} + \frac{\partial L}{\partial o_3} \frac{\partial o_3}{\partial s_3} \frac{\partial s_3}{\partial s_2} \frac{\partial s_2}{\partial U} + \frac{\partial L}{\partial o_3} \frac{\partial o_3}{\partial s_3} \frac{\partial s_3}{\partial s_2} \frac{\partial s_2}{\partial s_1} \frac{\partial s_1}{\partial U} \\ &= \sum_{k=1}^3 \frac{\partial L}{\partial o_3} \frac{\partial o_3}{\partial s_3} \left(\prod_{j=k+1}^3 \frac{\partial s_j}{\partial s_{j-1}} \right) \frac{\partial s_k}{\partial U}\end{aligned}$$



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

$$\prod_{j=k+1}^3 \tanh' * W$$



RNN与梯度消失、梯度爆炸

- 1、需要明确的是，RNN 中的梯度消失/梯度爆炸和普通的 MLP 或者深层 CNN 中梯度消失/梯度爆炸的含义不一样：MLP/CNN 中不同的层有不同的参数，各是各的梯度；而 RNN 中同样的权重在各个时间步共享，最终的梯度 g 等于各个时间步的梯度 g_t 的和。
- 2、由 1 中所述的原因，RNN 中总的梯度是不会消失的。即便梯度越传越弱，那也只是远距离的梯度消失，由于近距离的梯度不会消失，所有梯度之和并不会消失。RNN 所谓梯度消失的真正含义是，梯度被近距离梯度主导，导致模型难以学到远距离的依赖关系。
- 3、实践中梯度爆炸一般通过梯度裁剪来解决。

NeuHub
AI Links All



&



休息

NeuHub
AI Links All

引子：记忆机制

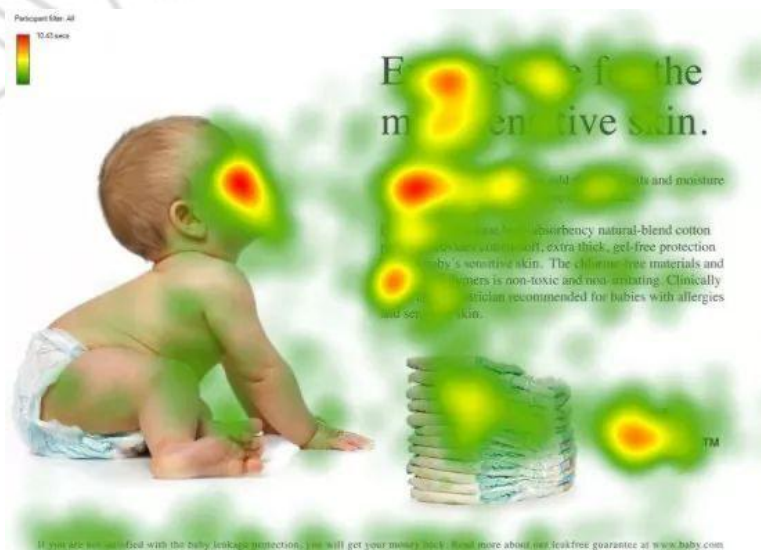
什么是记忆？

记忆是一种随时间而变化的状态。



引子：记忆机制

决定记忆状态的两大因素是什么？



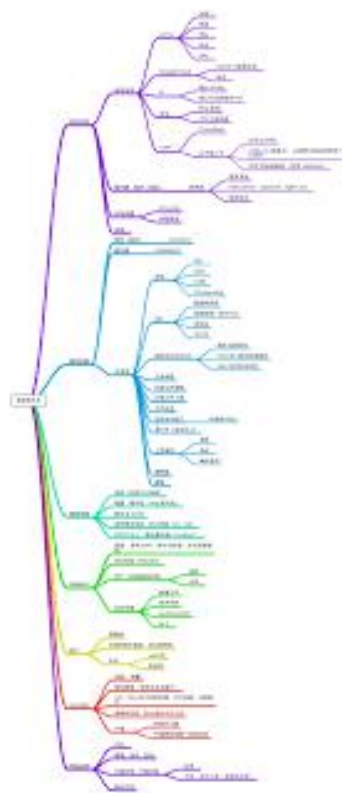
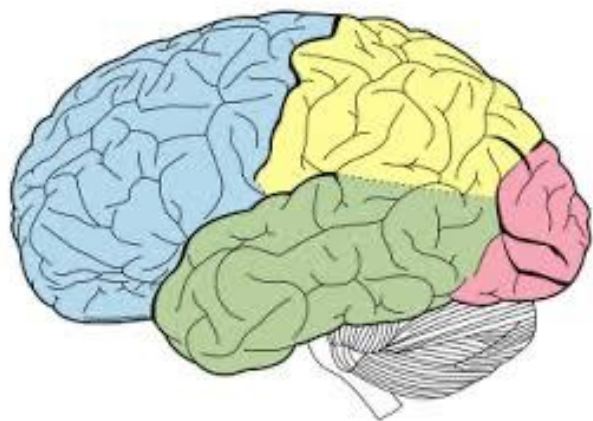
选择性的输入



选择性的遗忘

引子：记忆机制

在不同场景下的记忆输出不同。



→ task1

→ task2

→ task3

→ task4

...

LSTM中的记忆机制

- 记忆是一种随时间而变化的状态。
- 决定记忆状态的两大因素：选择性的输入、选择性的遗忘。
- 在不同场景下的记忆输出不同。

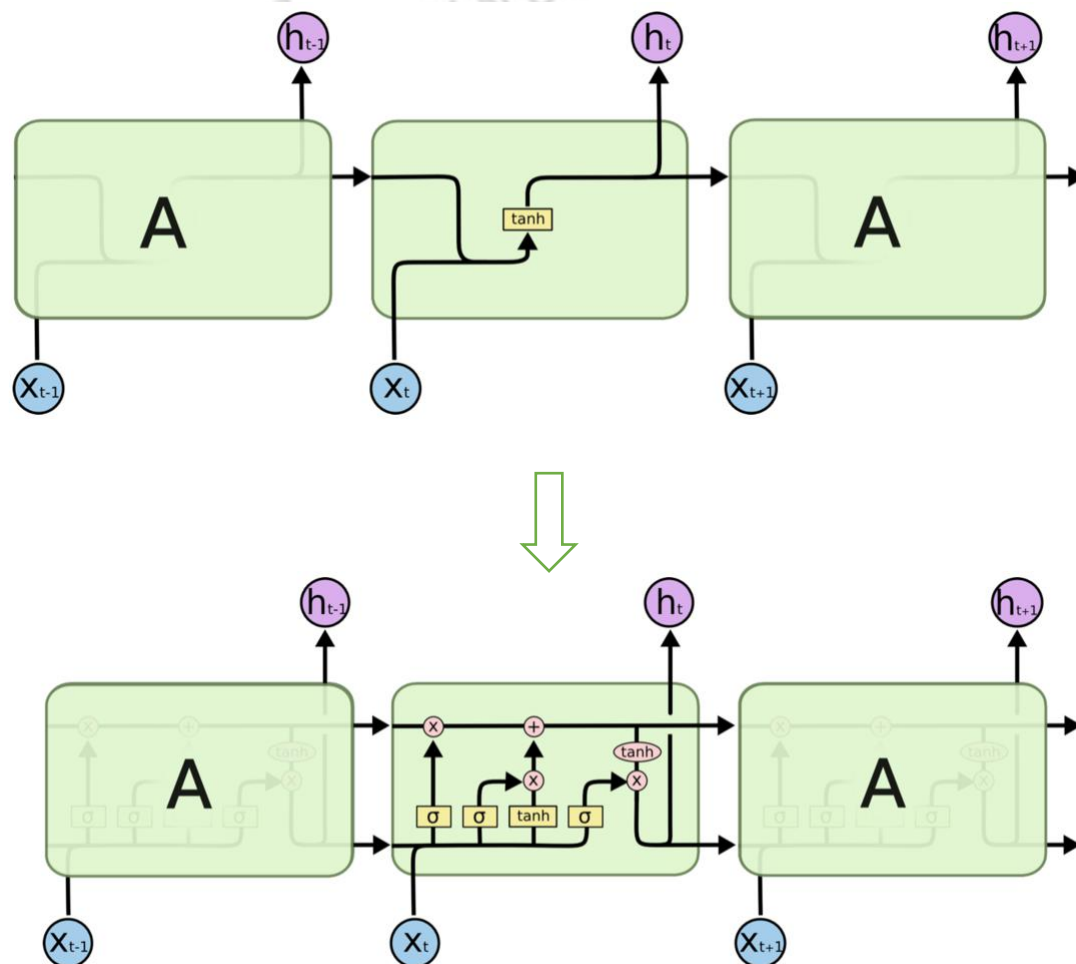


LSTM中的主要概念

- 细胞状态
- 输入门
- 遗忘门
- 输出门

输入有效的信息，遗忘无用的信息，输出与任务强相关的信息应用于任务处理。

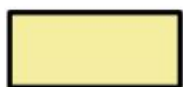
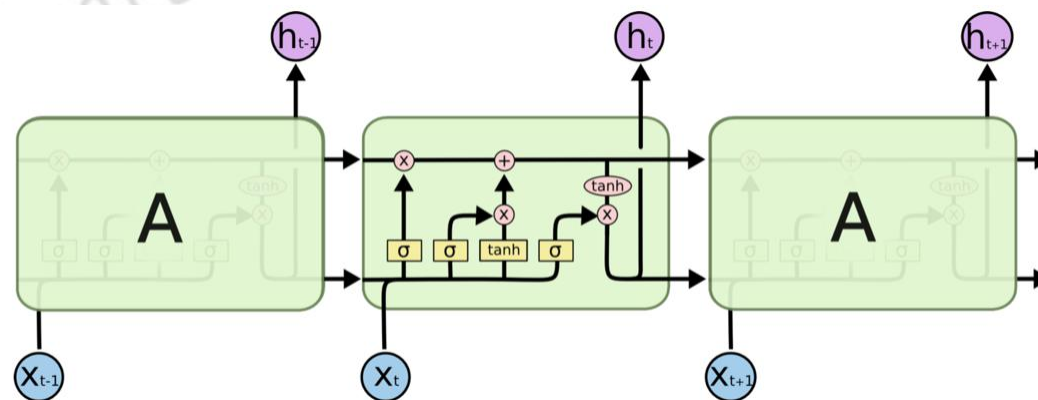
Long Short-term Memory (LSTM)



LSTM是一种特殊的RNN网络，该网络由 Hochreiter & Schmidhuber (1997)引入，并有许多人对其进行了改进和普及。他们的工作被用来解决了各种各样的问题，直到目前还被广泛应用。

NeuHub
AI Links All

Long Short-term Memory (LSTM)



Neural Network
Layer



Pointwise
Operation



Vector
Transfer



Concatenate

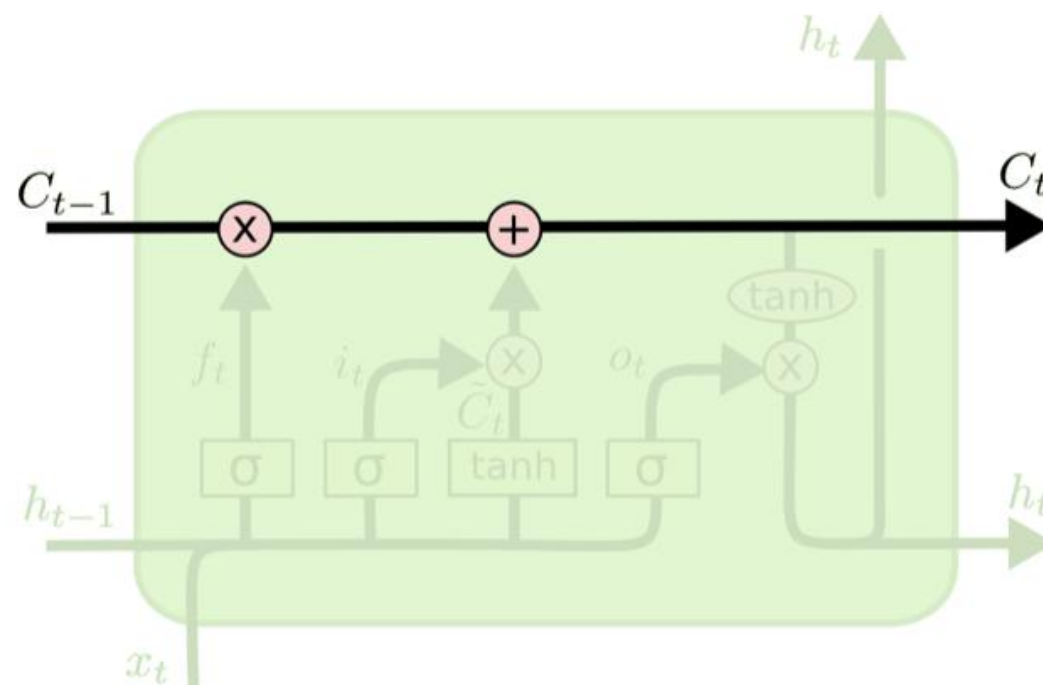


Copy

黄色方框表示激活函数操作，粉色圆圈表示点操作，单箭头表示数据流向，箭头合并表示向量的合并（concat）操作，箭头分叉表示向量的拷贝操作。

Long Short-term Memory (LSTM)

LSTMs的核心是细胞状态，用贯穿细胞的水平线表示。细胞状态像传送带一样，贯穿整个细胞却只有很少的分支，这样能保证信息不变的流过整个RNNs。（人的记忆状态贯穿人的一生。）



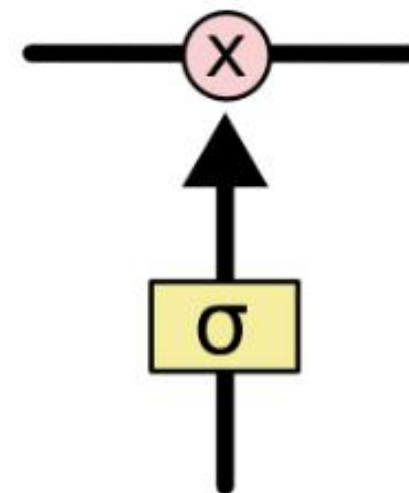
Long Short-term Memory (LSTM)

LSTM网络能通过一种被称为门的结构对细胞状态进行删除或者添加信息。

门能够有选择性地决定让哪些信息通过。其实门的结构很简单，就是一个sigmoid层和一个点乘操作的组合。

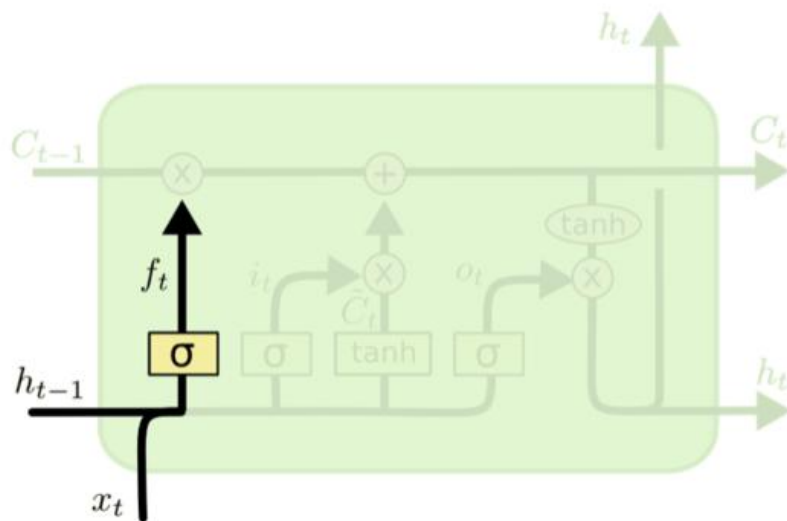
因为sigmoid层的输出是0-1的值，这代表有多少信息能够流过sigmoid层。0表示都不能通过，1表示都能通过。

一个LSTM里面包含三个门：忘记门、输入门和输出门。



New

LSTM：忘记门

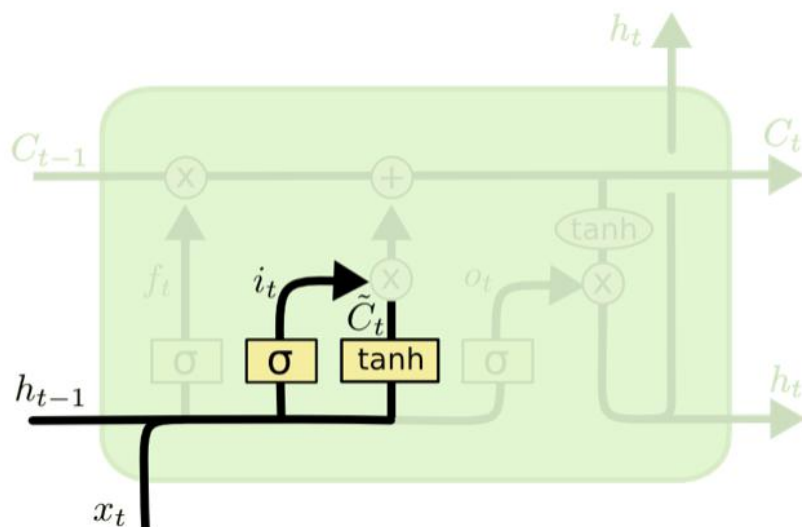


$$f^{(t)} = \sigma(W_f h^{(t-1)} + U_f x^{(t)} + b_f)$$

第一步：决定细胞状态需要**丢弃哪些信息**。

这部分操作是通过一个称为忘记门的sigmoid单元来处理，通过查看 h_{t-1} 和 x_t 信息来输出一个0-1之间的向量，该向量里面的0-1值表示细胞状态 C_{t-1} 中的哪些信息保留或丢弃多少。0表示不保留，1表示都保留。

LSTM：输入门

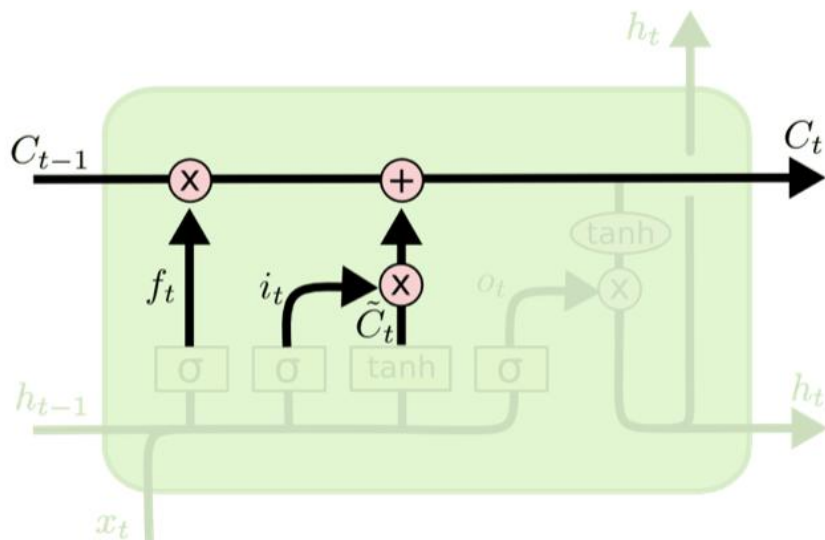


$$i^{(t)} = \sigma(W_i h^{(t-1)} + U_i x^{(t)} + b_i)$$

$$a^{(t)} = \tanh(W_a h^{(t-1)} + U_a x^{(t)} + b_a)$$

第二步：决定给细胞状态**添加哪些新的信息**。
 利用 h_{t-1} 和 x_t 通过一个称为输入门的操作来决定更新哪些信息。
 再利用 h_{t-1} 和 x_t 通过一个tanh层得到新的候选细胞信息 \tilde{c}_t （公式中的 $a(t)$ ），这些信息可能会被更新到细胞信息中。

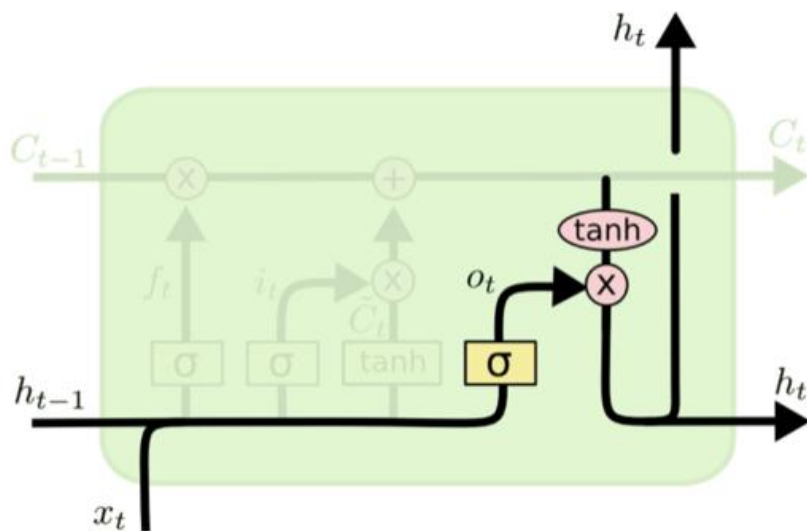
LSTM: 更新细胞状态C



$$C^{(t)} = C^{(t-1)} \odot f^{(t)} + i^{(t)} \odot a^{(t)}$$

第三步：更新旧的细胞信息 C_{t-1} ，变为新的细胞信息 C_t 。
更新的规则，通过忘记门选择忘记旧细胞信息的一部分，通过输入门选择添加候选细胞信息 \tilde{C}_t (公式中的 $a(t)$)的一部分得到新的细胞信息 C_t 。
注： \odot 为Hadamard积

LSTM：输出门

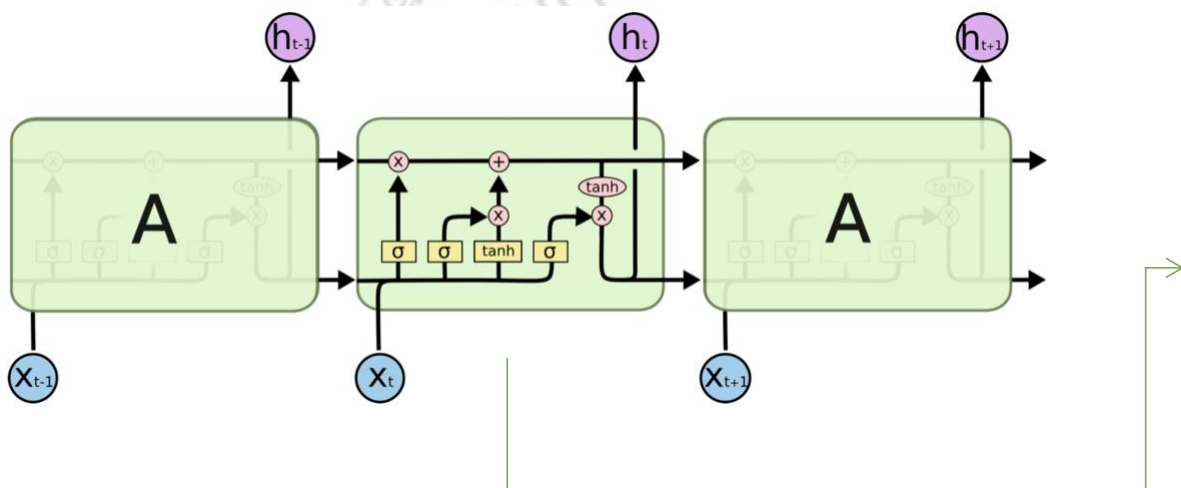


$$o^{(t)} = \sigma(W_o h^{(t-1)} + U_o x^{(t)} + b_o)$$

$$h^{(t)} = o^{(t)} \odot \tanh(C^{(t)})$$

第四步：根据输入的 h_{t-1} 和 x_t 来判断输出细胞的哪些状态特征。需要将输入经过一个称为输出门的sigmoid层得到判断条件，然后将细胞状态经过tanh层得到一个-1~1之间值的向量，该向量与输出门得到的判断条件相乘就得到了最终该RNN单元的输出。

总结



1) 更新遗忘门输出:

$$f^{(t)} = \sigma(W_f h^{(t-1)} + U_f x^{(t)} + b_f)$$

2) 更新输入门两部分输出:

$$i^{(t)} = \sigma(W_i h^{(t-1)} + U_i x^{(t)} + b_i)$$

$$a^{(t)} = \tanh(W_a h^{(t-1)} + U_a x^{(t)} + b_a)$$

3) 更新细胞状态:

$$C^{(t)} = C^{(t-1)} \odot f^{(t)} + i^{(t)} \odot a^{(t)}$$

4) 更新输出门输出:

$$o^{(t)} = \sigma(W_o h^{(t-1)} + U_o x^{(t)} + b_o)$$

$$h^{(t)} = o^{(t)} \odot \tanh(C^{(t)})$$

5) 更新当前序列索引预测输出:

$$\hat{y}^{(t)} = \sigma(V h^{(t)} + c)$$

LSTM：视频演示



&



<https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>

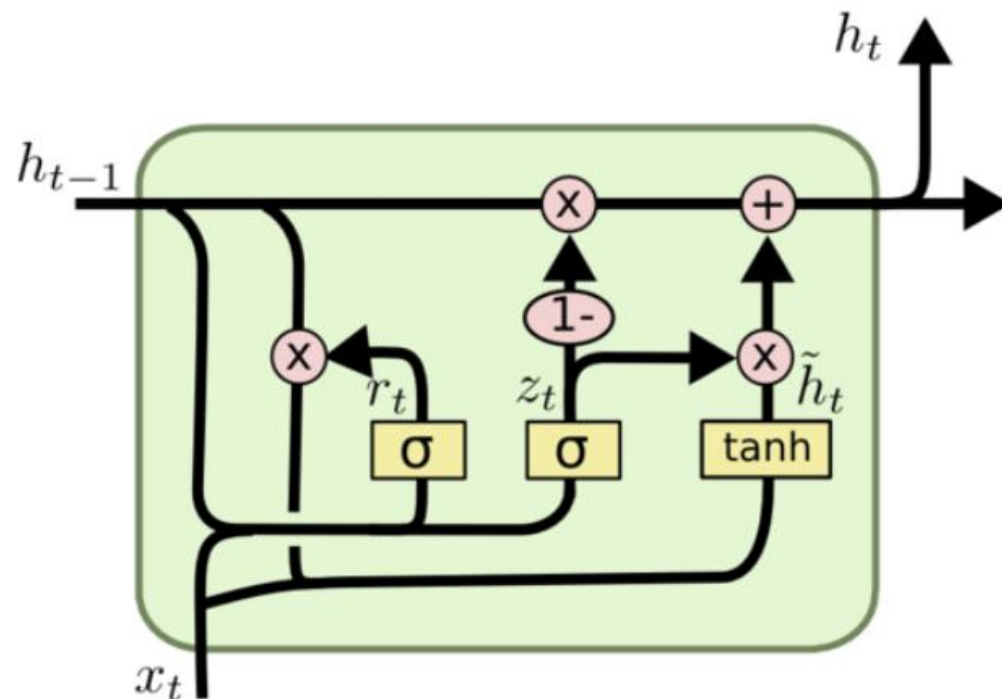
start from 6:40

NeuHub
AI Links All

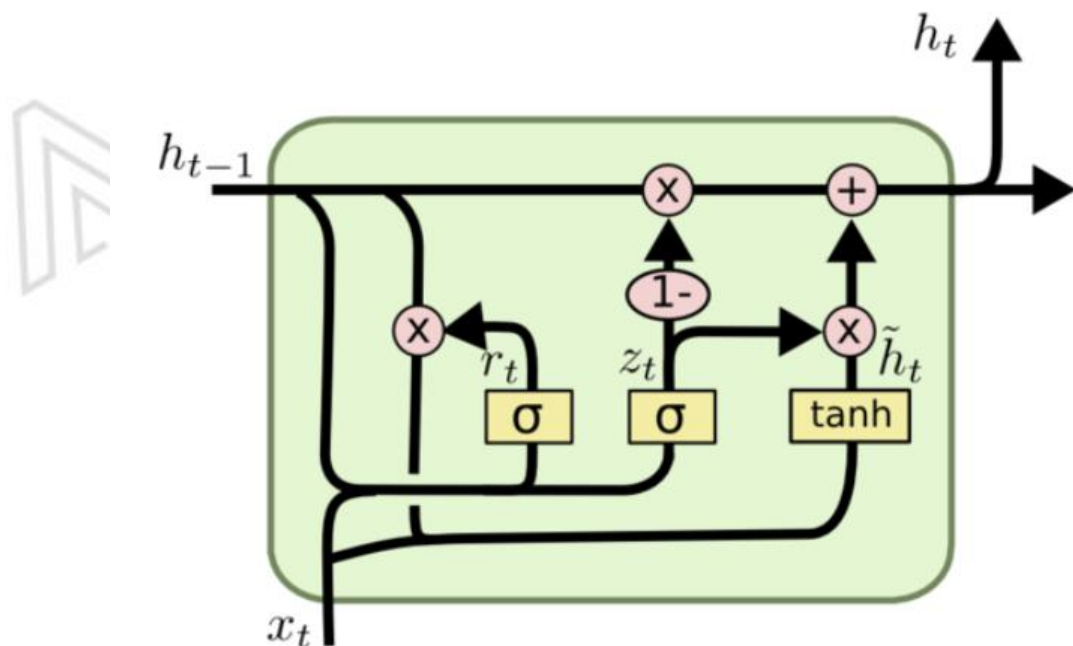
GRU

在GRU模型中只有两个门：

- 重置门（ r ）：控制前一状态有多少信息被写入到当前的候选状态 \tilde{h}_t 上的程度，重置门越小，前一状态的信息被写入的越少。
- 更新门（ z ）：控制前一时刻的状态信息与候选状态信息被带入到当前状态中的程度。



GRU



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

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

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

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



&



休息

NeuHub
AI Links All

LSTM代码示例

<https://colab.research.google.com/drive/1Zfvt9Vfs3PrJwSDF8jMvomz7CzU36RXk>

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9

NeuHub
AI Links All

LSTM代码示例

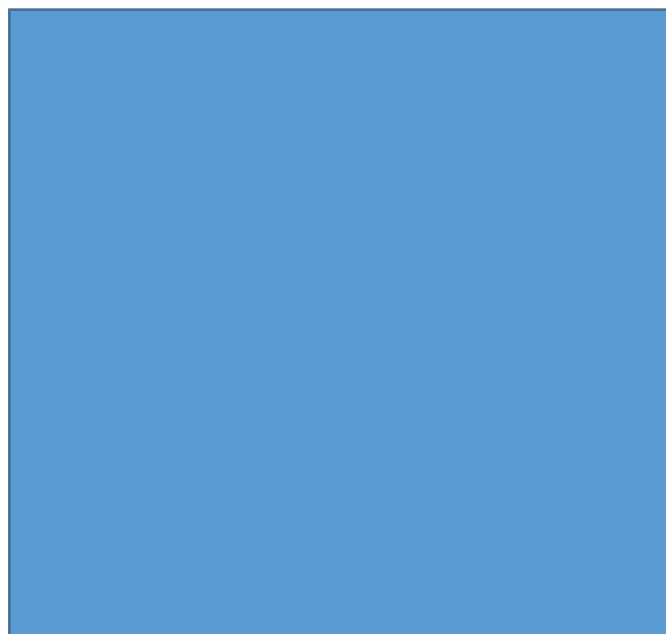
参数矩阵



X 4

x2h

28*128



X 4

h2h

128*128

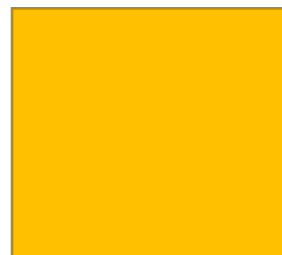
init cx

1*128

init hx

1*128

输入



图片28*28

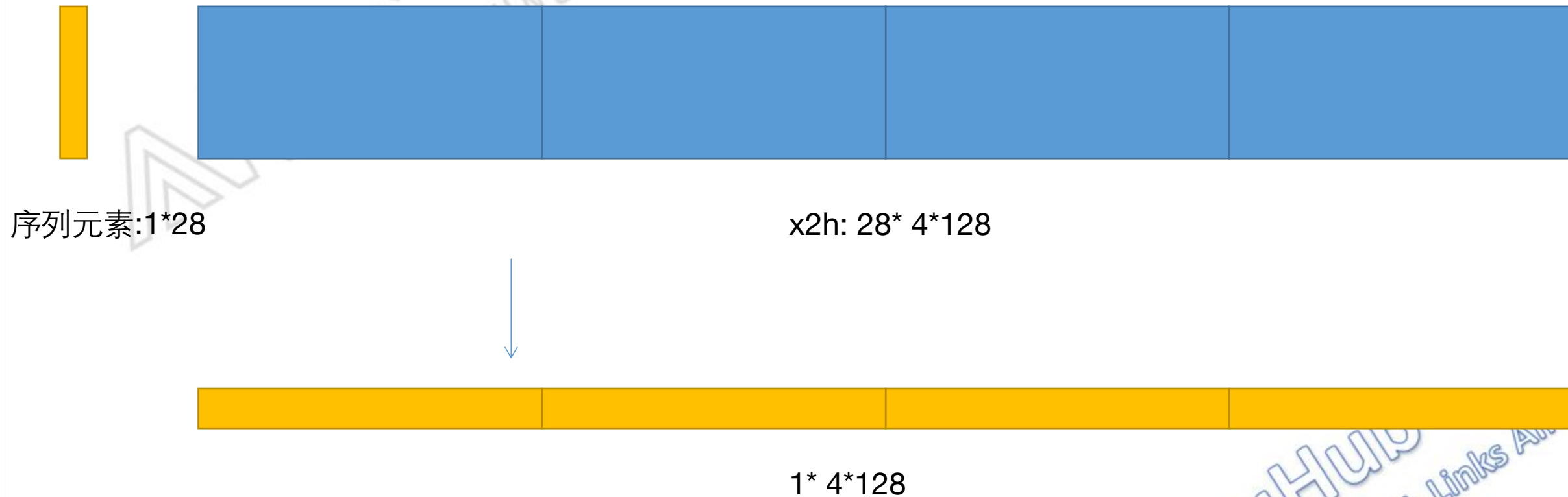


序列元素1*28

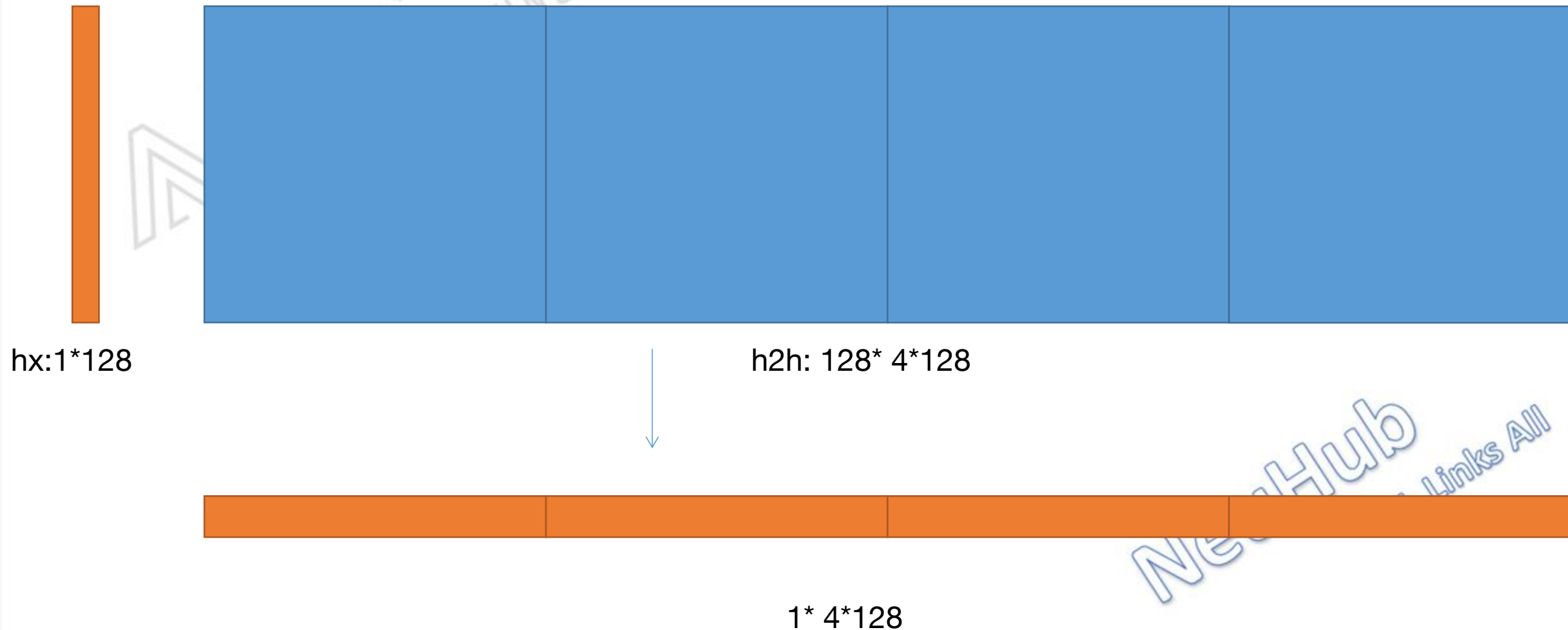


...

LSTM代码示例



LSTM代码示例



LSTM代码示例



&



+



$1 * 4 * 128$

NeuHub
AI Links All

LSTM代码示例



chunk

ingate



forgetgate



cellgate



outgate



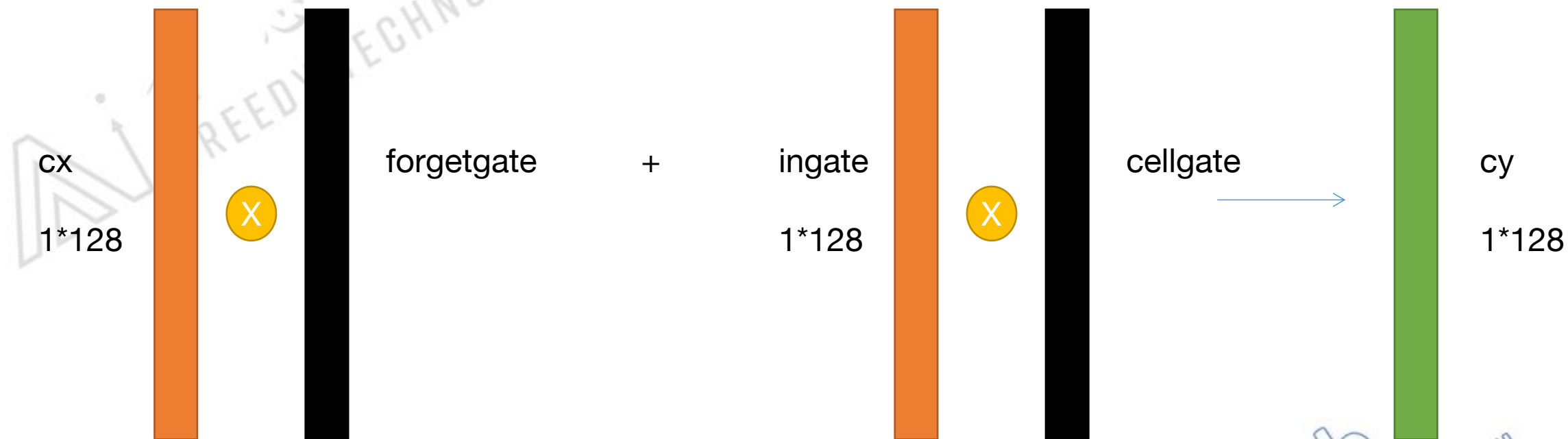
sigmoid or tanh



LSTM代码示例



&

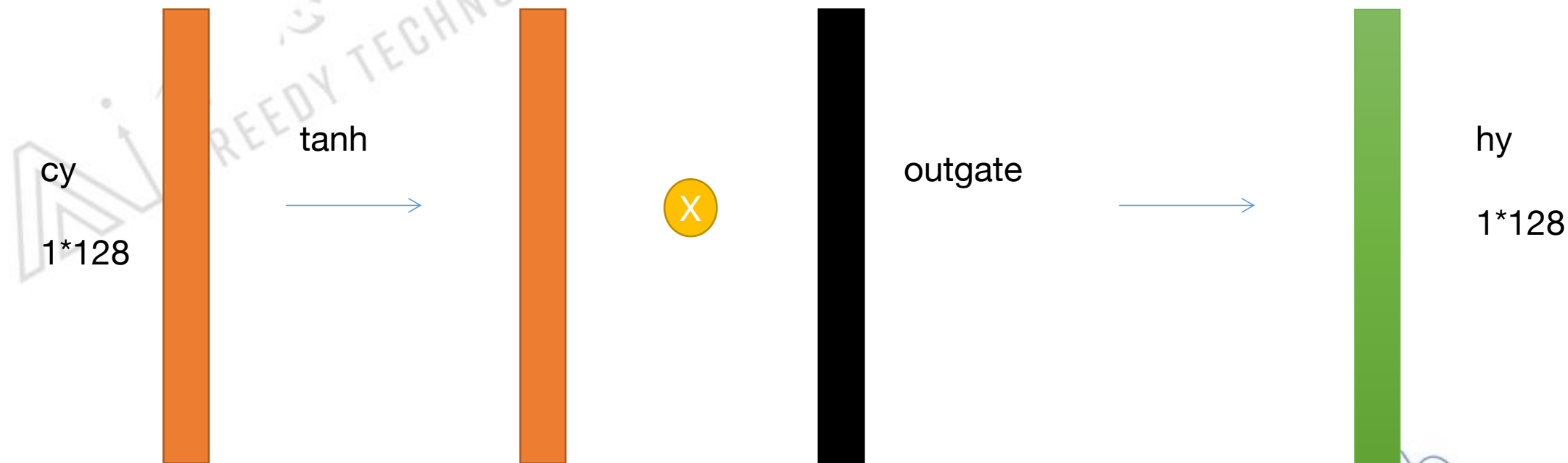


NeuHub
AI Links All

LSTM代码示例



&

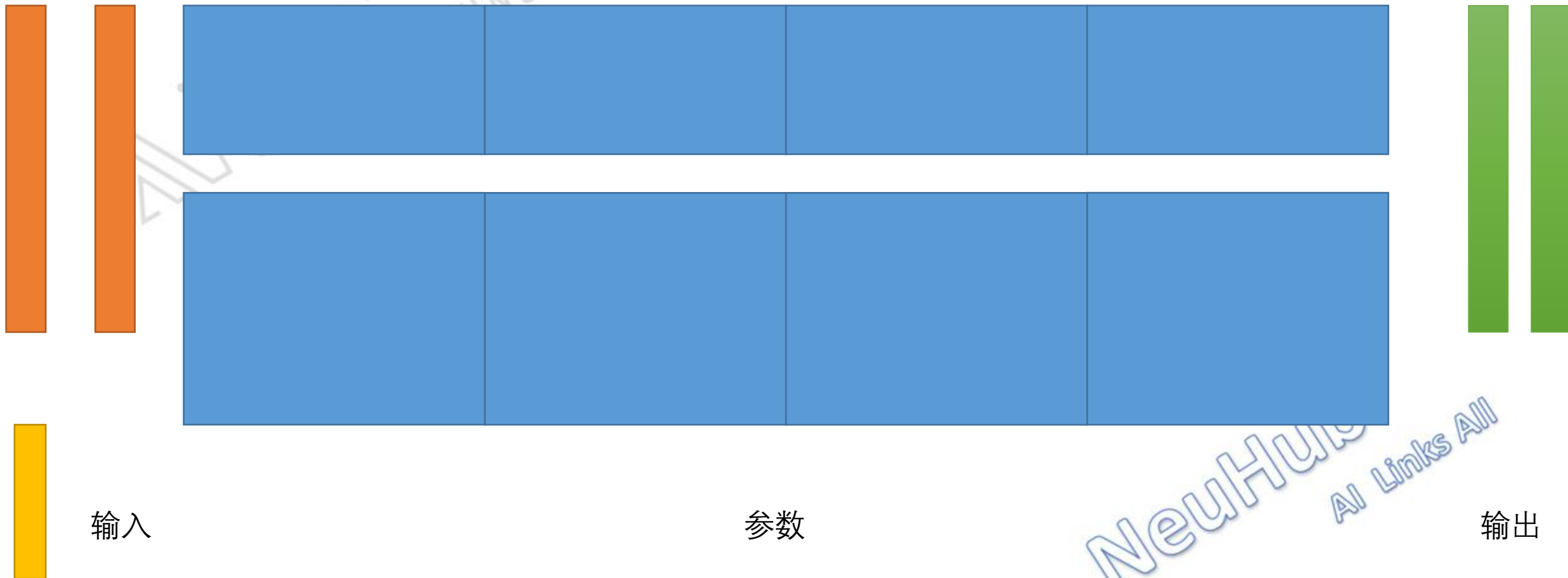


NeuHub
AI Links All

LSTM代码示例



&



典型面试题



&



- RNN在训练过程中存在什么问题？如何解决？
- LSTM与RNN相比，有哪些特点？
- 请画出LSTM的基本结构。
- 循环神经网络可以解决哪些问题？

NeuHub
AI Links All

作业



- <https://medium.com/datadriveninvestor/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577>
- 参考以上文章写一篇blog
- 主题：为什么LSTM可以解决RNN梯度消失的问题

NeuHub
AI Links All

参考

RNN

- <https://zybuluo.com/hanbingtao/note/541458>

LSTM

- <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- <https://www.youtube.com/watch?v=9zhrxE5PQgY&feature=youtu.be>
- <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>
- <https://medium.com/datadriveninvestor/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577>