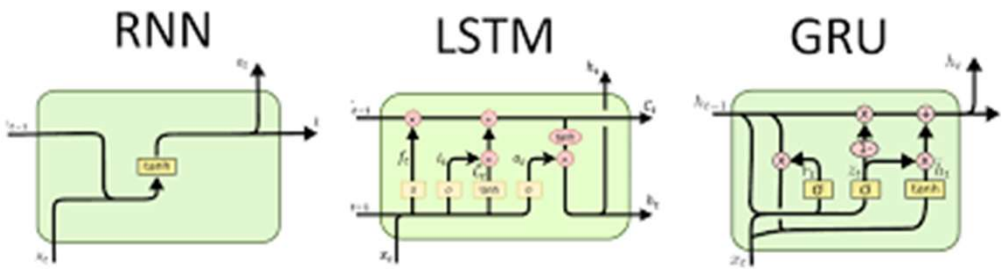
Recurrent Neural Network

循環神經網路介紹

Sean, 王家祥

2020/11/20

CONTENTS



/01

RNN 簡介

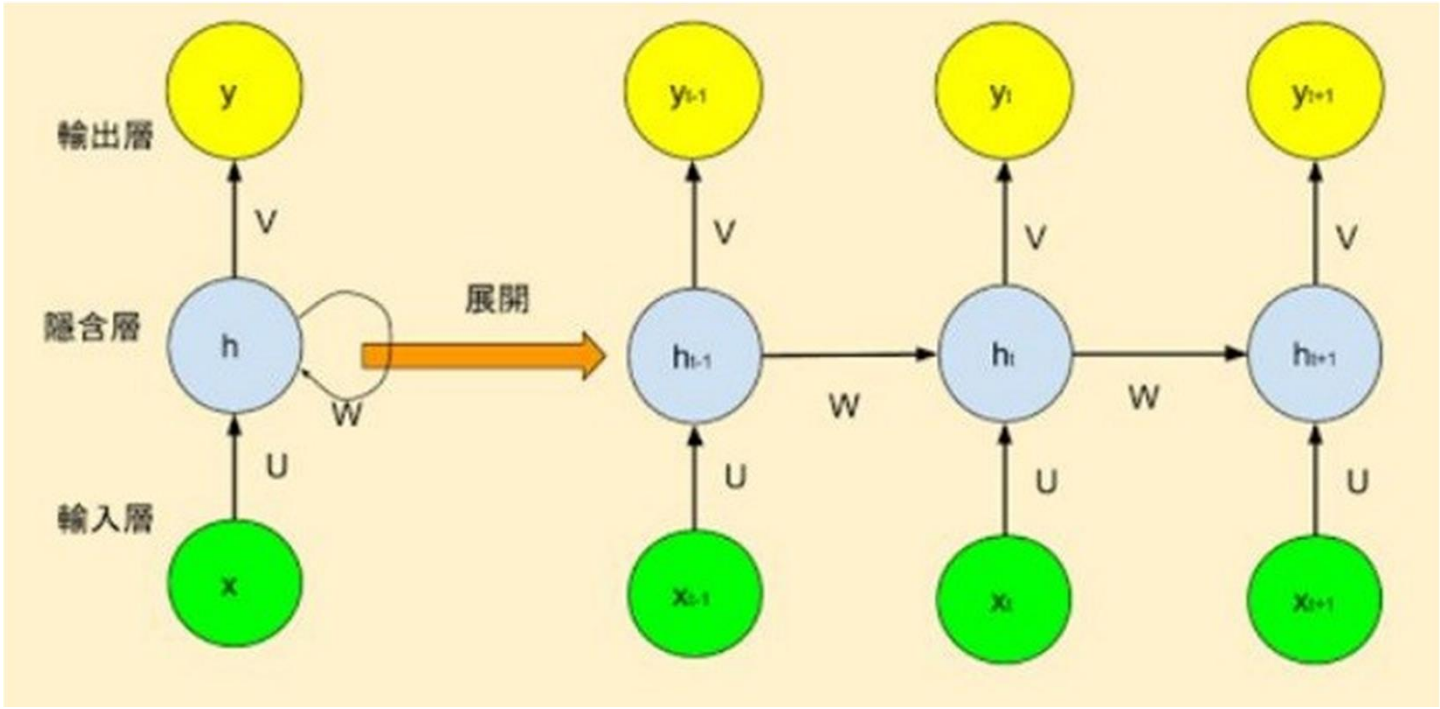
RNN 簡介

- 語言通常要考慮前言後語，以免斷章取義，也就是說，建立語言的相關模型，如果能額外考慮上下文的關係，準確率就會顯著提高，因此，學者提出『循環神經網路』(Recurrent Neural Network, RNN)演算法，它是『自然語言處理』領域最常使用的 Neural Network 模型
- 簡單的RNN模型(Vanilla RNN)額外考慮前文的關係，把簡單迴歸的模型 (y=W*x+b)，改為下列公式，其中h就是預測值(y)，同樣，將它加一個 Activation Function(通常使用 tanh)，就變成第二個公式

$$h_t = W * h_{t-1} + Ux_t + b$$

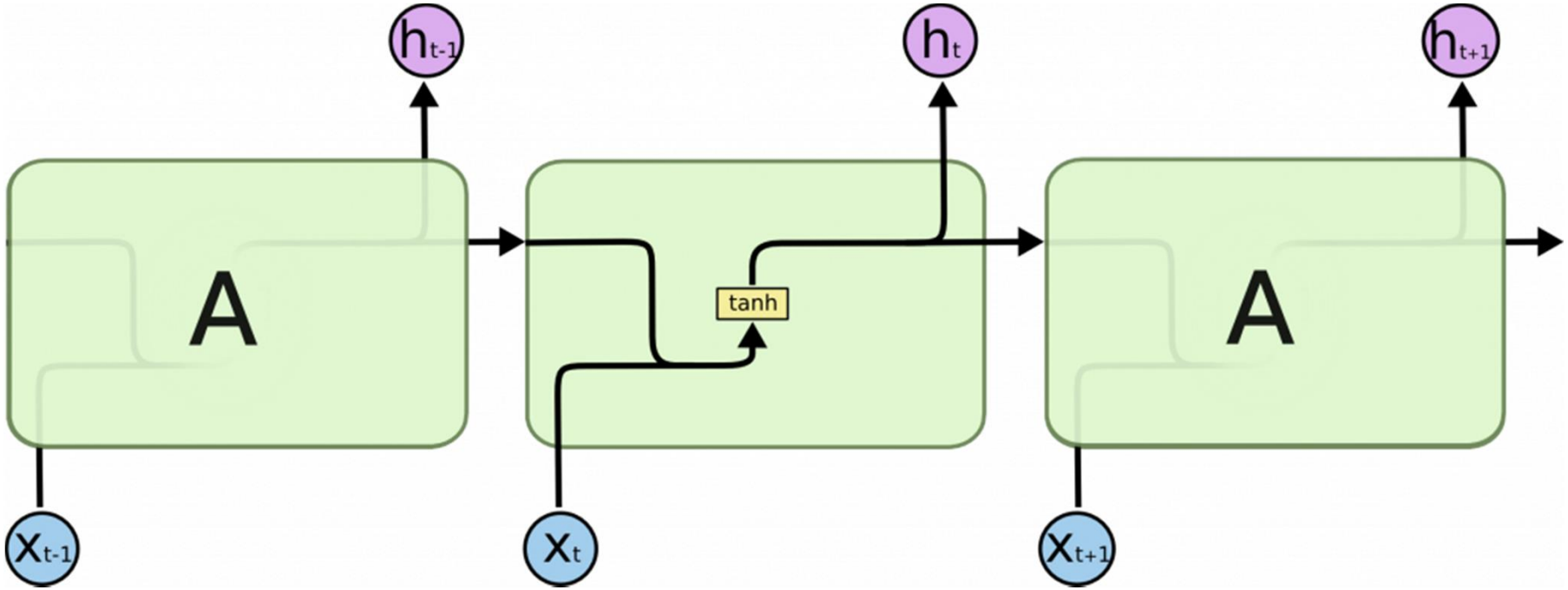
$$y_t = g(V * h_t)$$

RNN 簡介(單個神經元展開)

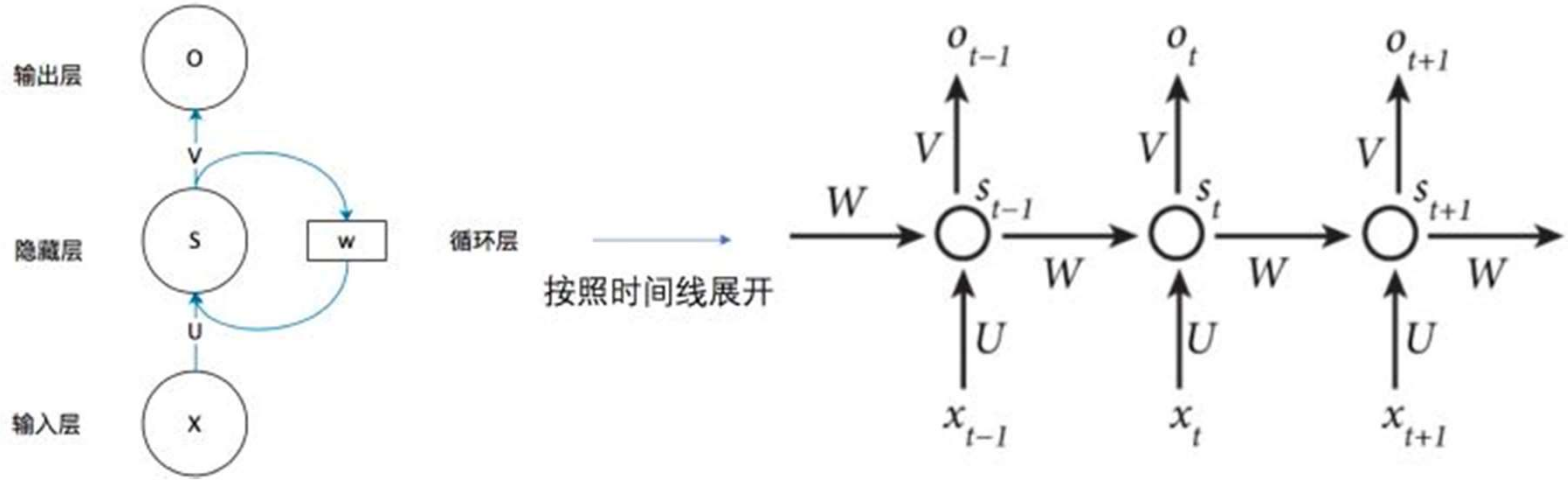


當前 output 不只受上一層輸入的影響，也受到同一層前一個 output 的影響(即前文)，類似統計學的『時間數列』(Time Series)

RNN 簡介(單個神經元展開)



RNN 簡介(單個神經元展開)



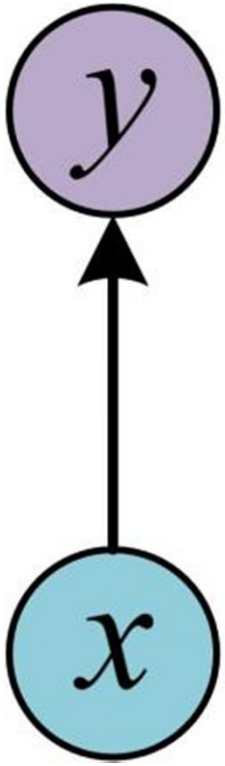
當前 output 不只受上一層輸入的影響，也受到同一層前一個 output 的影響(即前文)，類似統計學的『時間數列』(Time Series)

/02

RNN 圖解

RNN 圖解

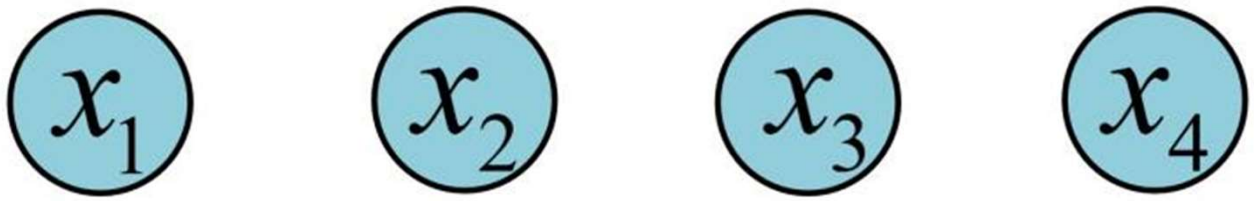
- 最基本的单层网络结构，输入是x，经过变换Wx+b和激活函数f得到输出y：



$$y = f(Wx + b)$$

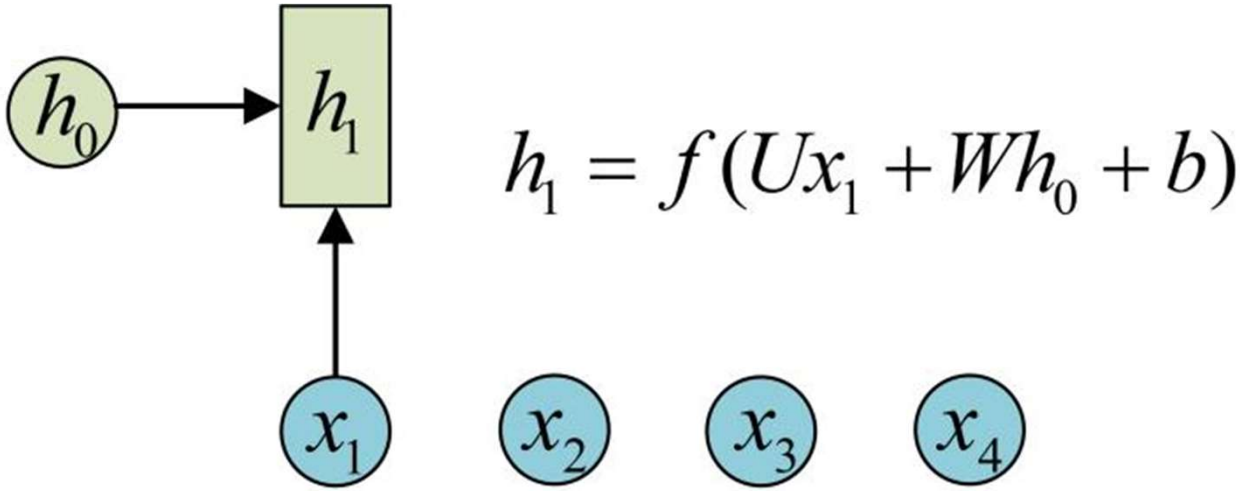
RNN 圖解

- 在实际应用中，我们还会遇到很多序列形的数据，如：
- 自然语言处理问题。x1可以看做是第一个单词，x2可以看做是第二个单词，依次类推。
- 语音处理。此时，x1、x2、x3……是每帧的声音信号。
- 时间序列问题。例如每天的股票价格等等。
- 其单个序列如下图所示：



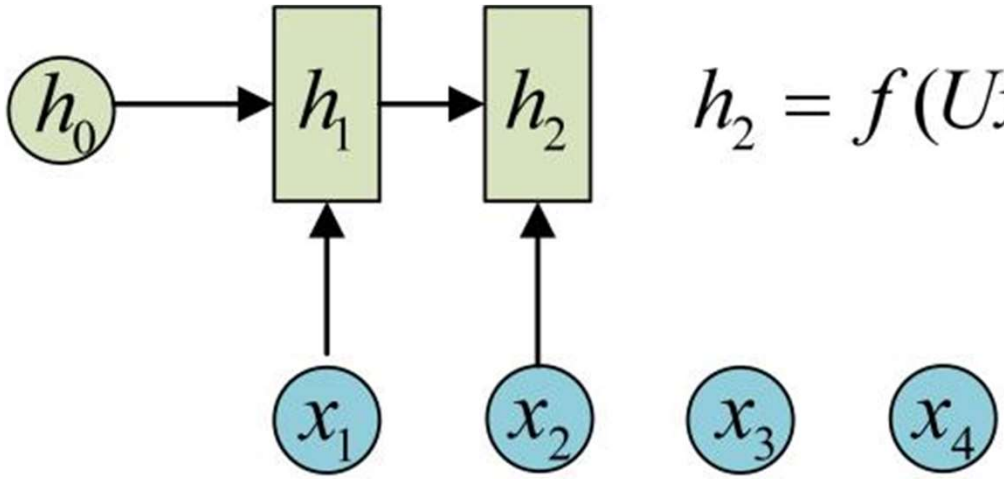
RNN 圖解

- 诸如此类的序列数据用原始的神经网络难以建模，
- 基于此，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。 为了便于理解，先计算*h*1•：



RNN 圖解

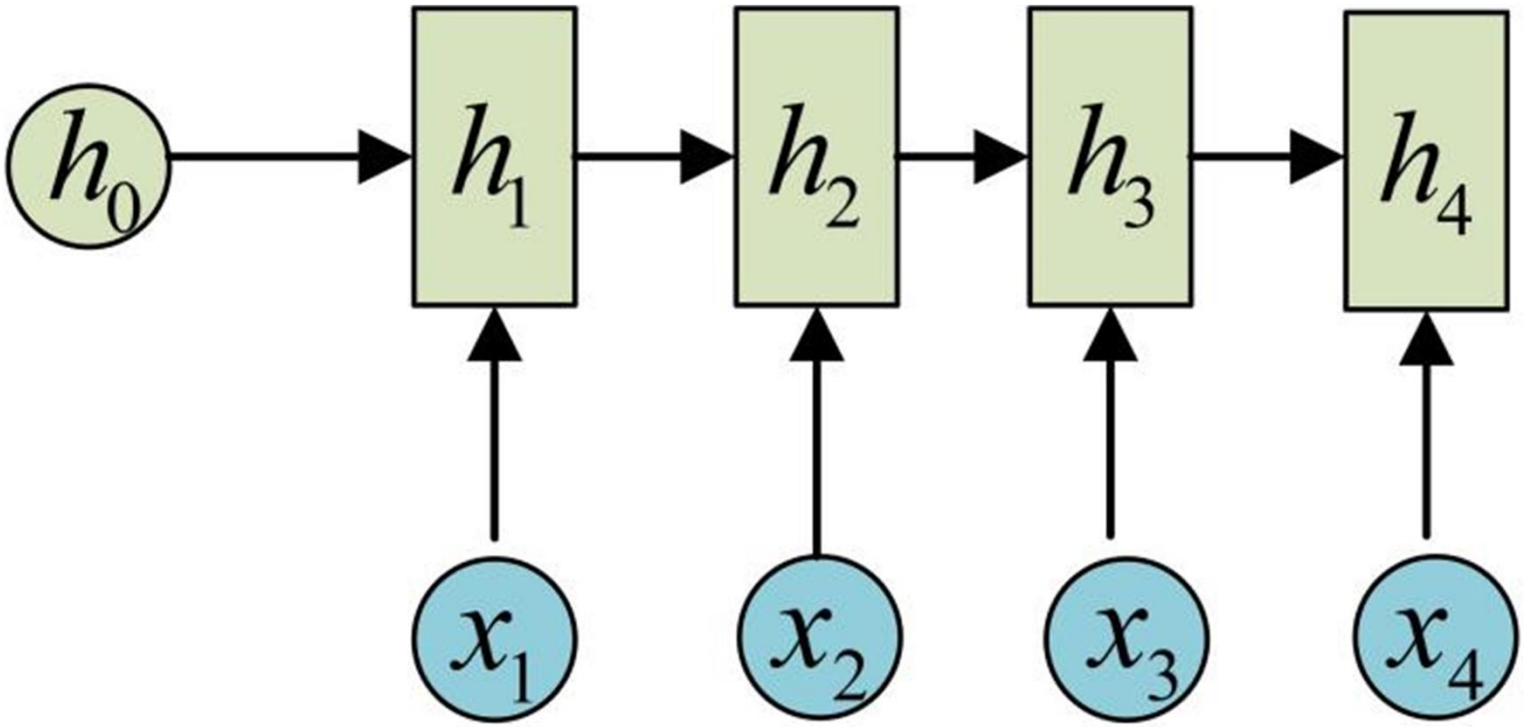
- 图中的圆圈表示向量，箭头表示对向量做变换。
- RNN中，每个步骤使用的参数`U,W,b`•相同，
- `h_2`的计算方式和`h_1•`类似，其计算结果如下：



$$h_2 = f(Ux_2 + Wh_1 + b)$$

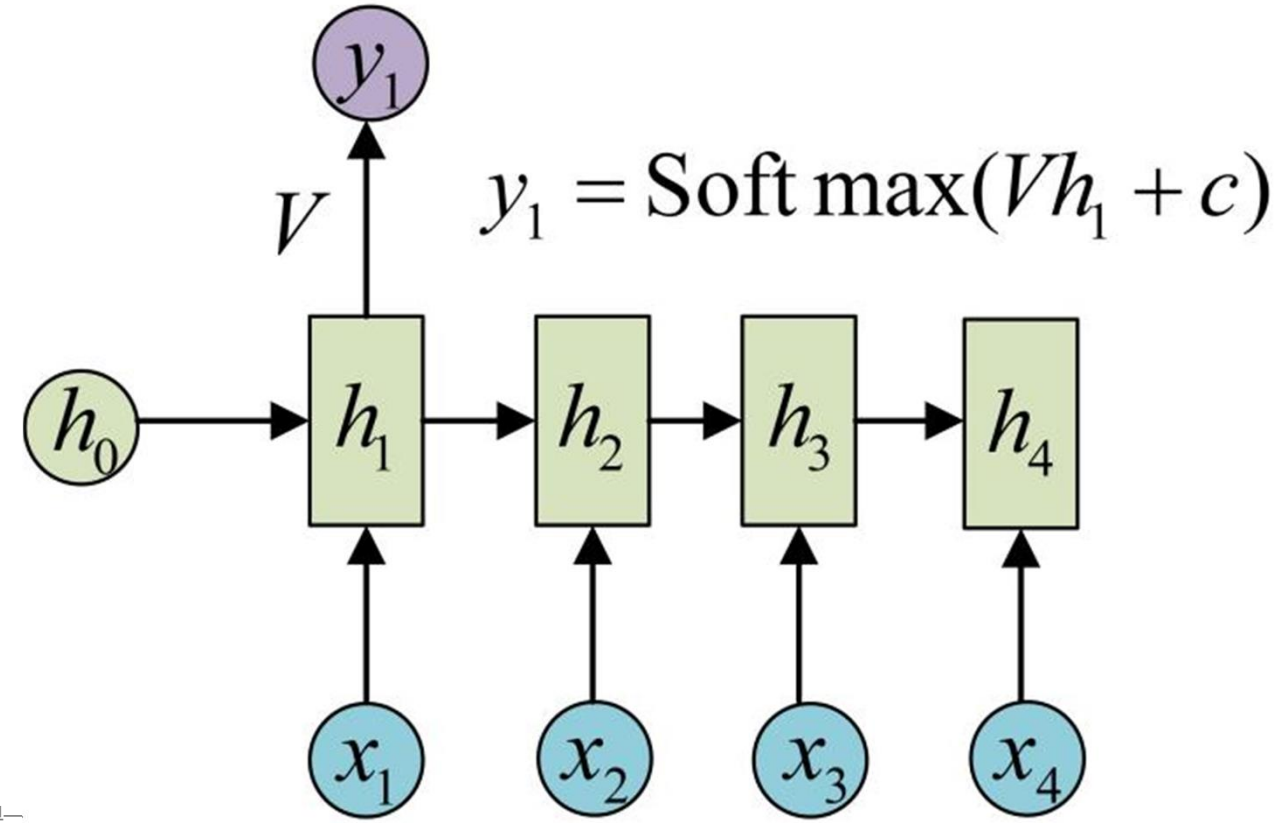
RNN 圖解

- 计算*h*3,*h*4•也相似，可得:



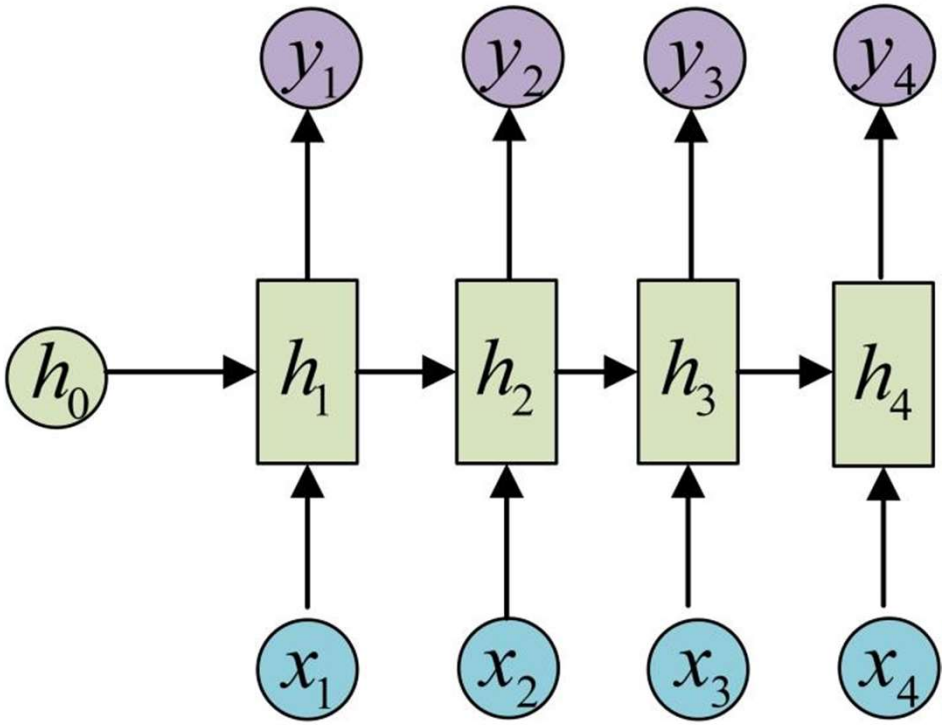
RNN 圖解

- 接下来，计算RNN的输出*y*1，采用*Softmax*作为激活函数，
- 根据*y***n=f*(*W***x*+*b*)，得*y*1•:



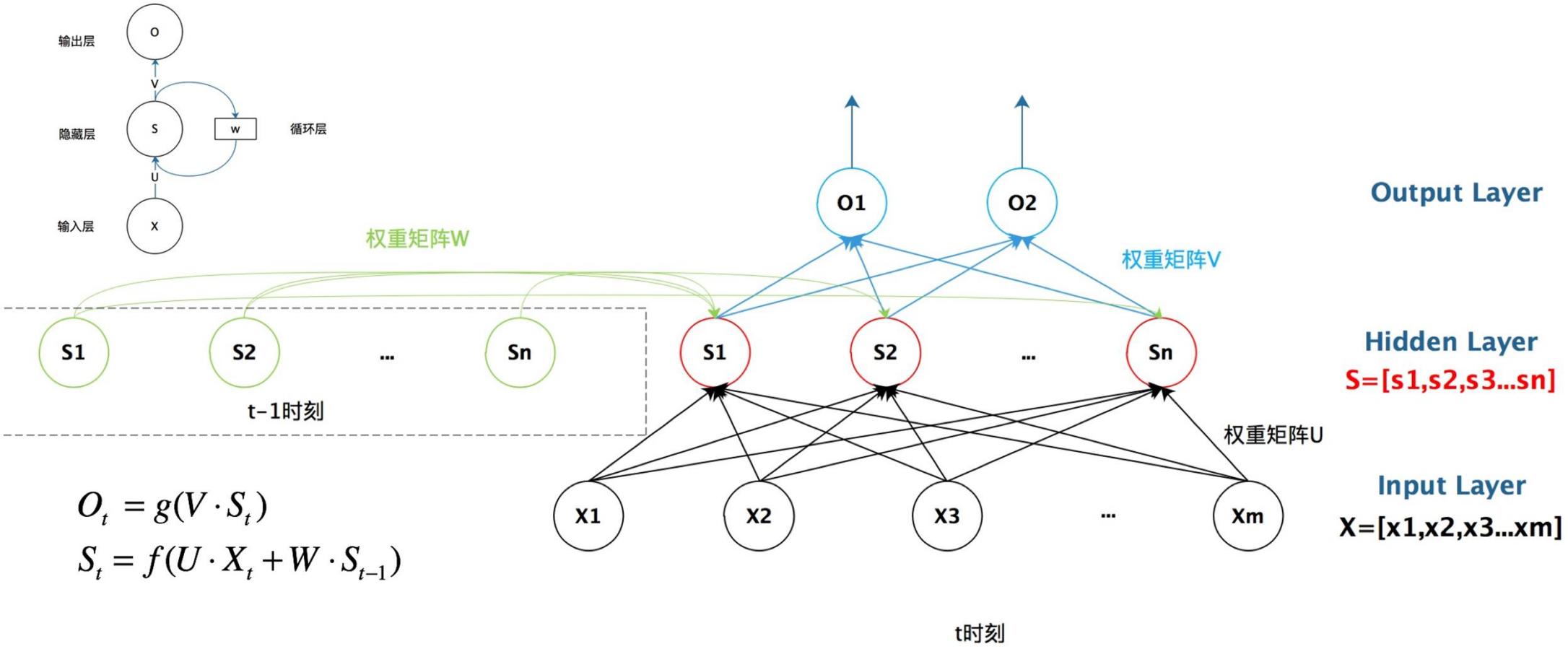
RNN 圖解

- 使用和*y*1•相同的参数*V*,*c*•，得到*y*1,*y*2,*y*3,*y*4•的输出结构：



以上即为最经典的RNN结构，其输入为x1,x2,x3,x4，输出为y1,y2,y3,y4，当然实际中最大值为y**n，这里为了便于理解和展示，只计算4个输入和输出。从以上结构可看出，RNN结构的输入和输出等长。

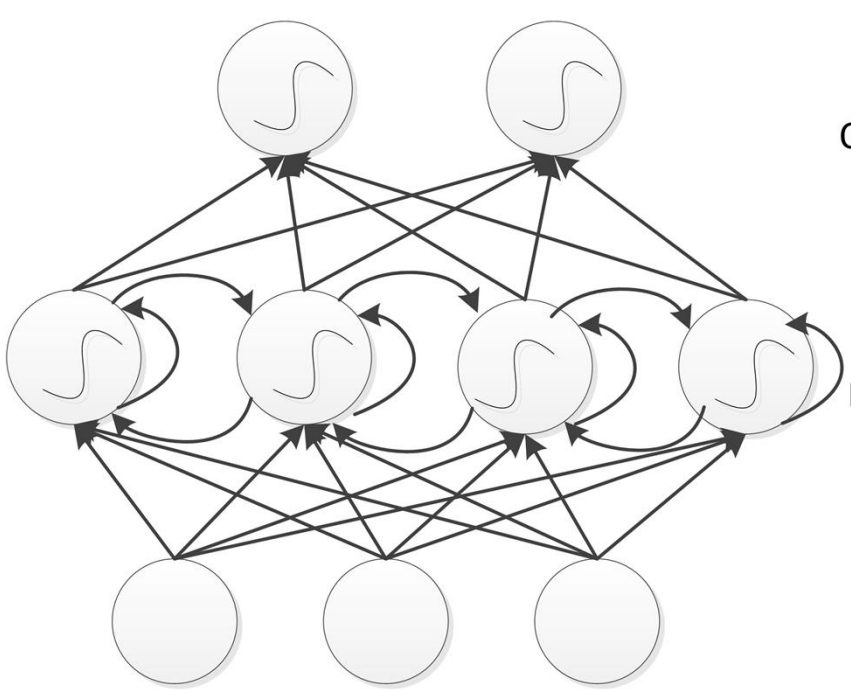
RNN 圖解

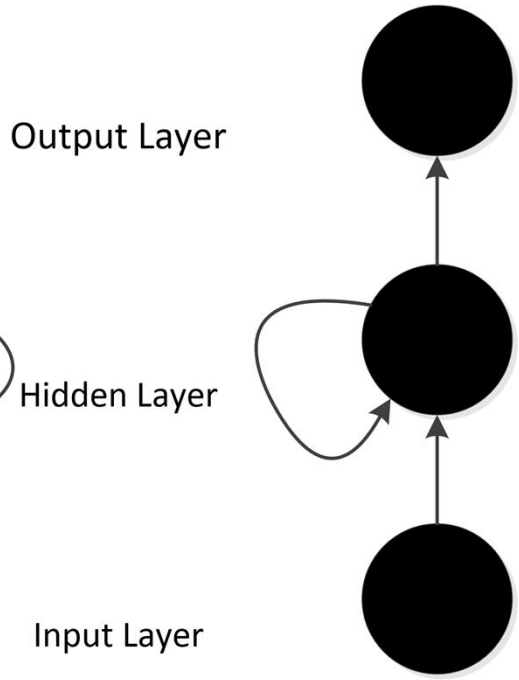


$$O_t = g(V \cdot S_t)$$

$$S_t = f(U \cdot X_t + W \cdot S_{t-1})$$

典型RNN模型





RNN特點

- 1.RNNs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。

- 2.RNNs中当前单元的输出与之前步骤输出也有关，因此称之为循环神经网络。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。隐藏层之间的节点连接起来，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。

- 3.标准的RNNs结构图，图中每个箭头代表做一次变换，也就是说箭头连接带有权值。

- 4.在标准的RNN结构中，隐层的神经元之间也是带有权值的，且权值共享。

- 5.理论上，RNNs能够对任何长度序列数据进行处理。但是在实践中，为了降低复杂度往往假设当前的状态只与之前某几个时刻状态相关

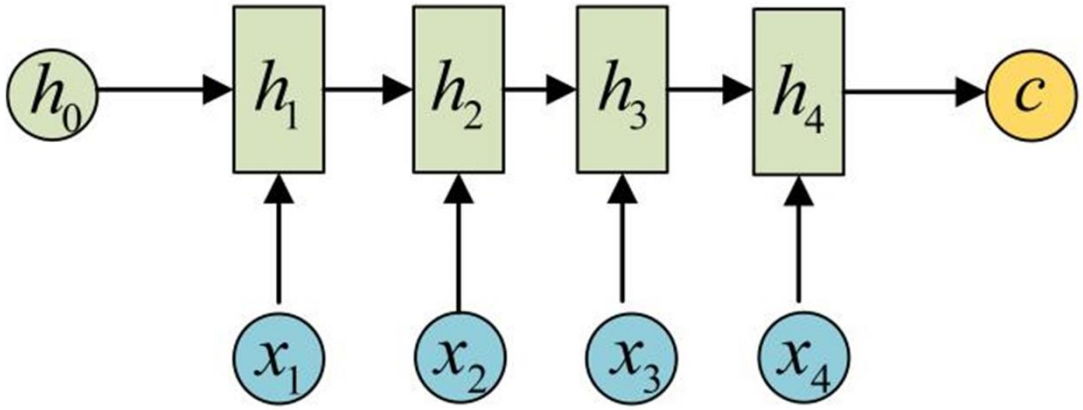
Encoder-Decoder 架構

- 我們遇到的大部分問題序列都是不等長的，如機器翻譯中，源語言和目標語言的句子往往並沒有相同的長度。
- 其建模步驟如下：
- 步驟一：將輸入數據編碼成一個上下文向量c，這部分稱為Encoder，得到c有多種方式，最簡單的方法就是把Encoder的最後一個隱狀態賦值給c，還可以對最後的隱狀態做一個變換得到c，也可以對所有的隱狀態做變換。其示意如下所示：

$$(1)\ c = h_4$$

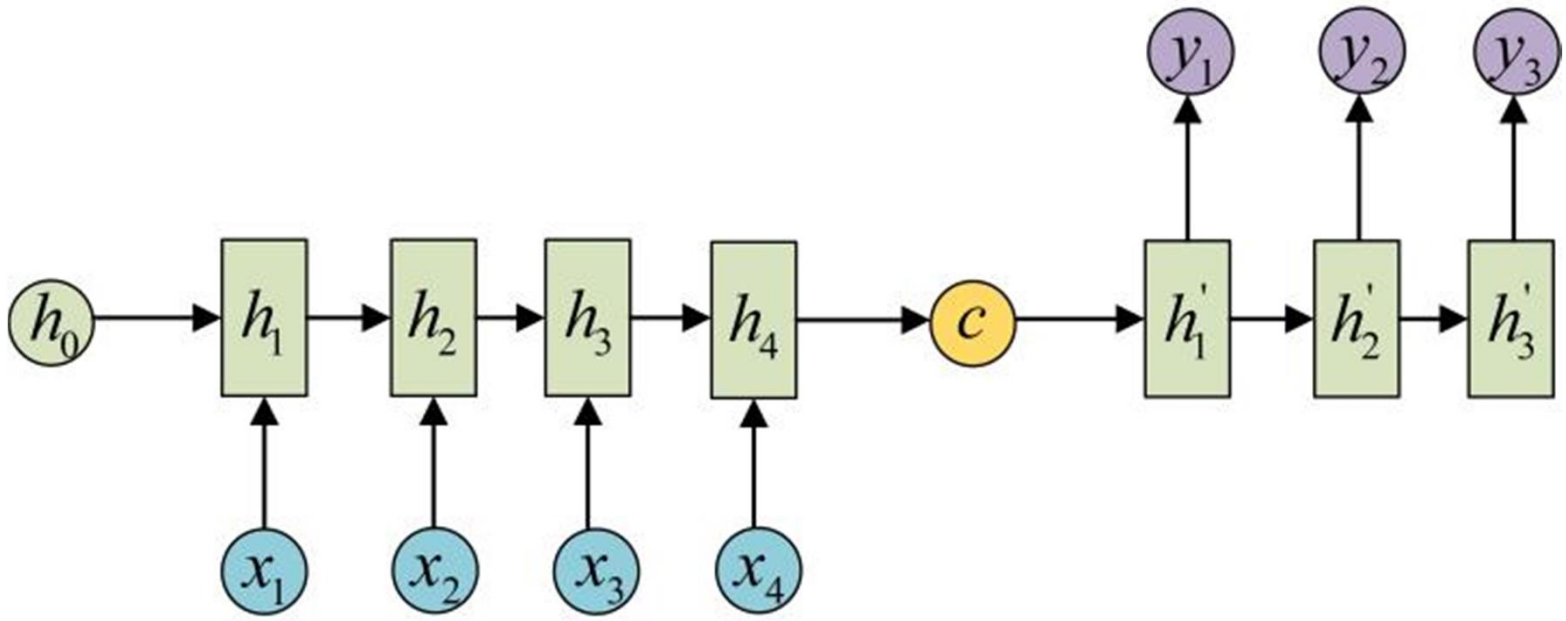
$$(2)\ c = q(h_4)$$

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

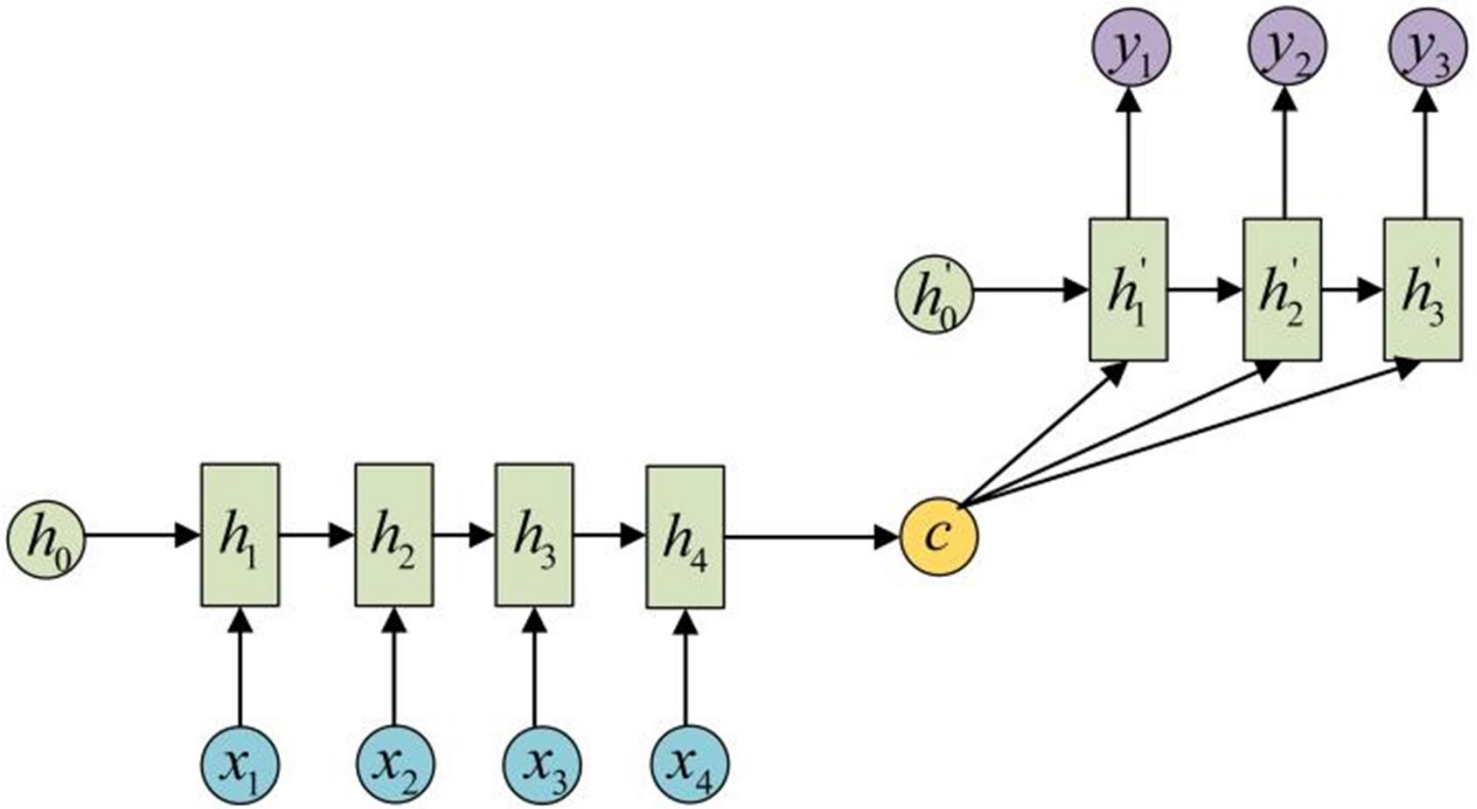


Encoder-Decoder 架構

- 步驟二：用另一個RNN網絡（我們將其稱為Decoder）對其進行編碼，方法一是將步驟一中的c•作為初始狀態輸入到Decoder，示意圖如下所示：
- 後續的新的神經網路架構有個以此為改善做出重大變革的 transformer model



Encoder-Decoder 架構



/03

SoftMax function

Softmax function

- 在机器学习尤其是深度学习中，softmax是个非常常用而且比较重要的函数，

尤其在多分类的场景中使用广泛。

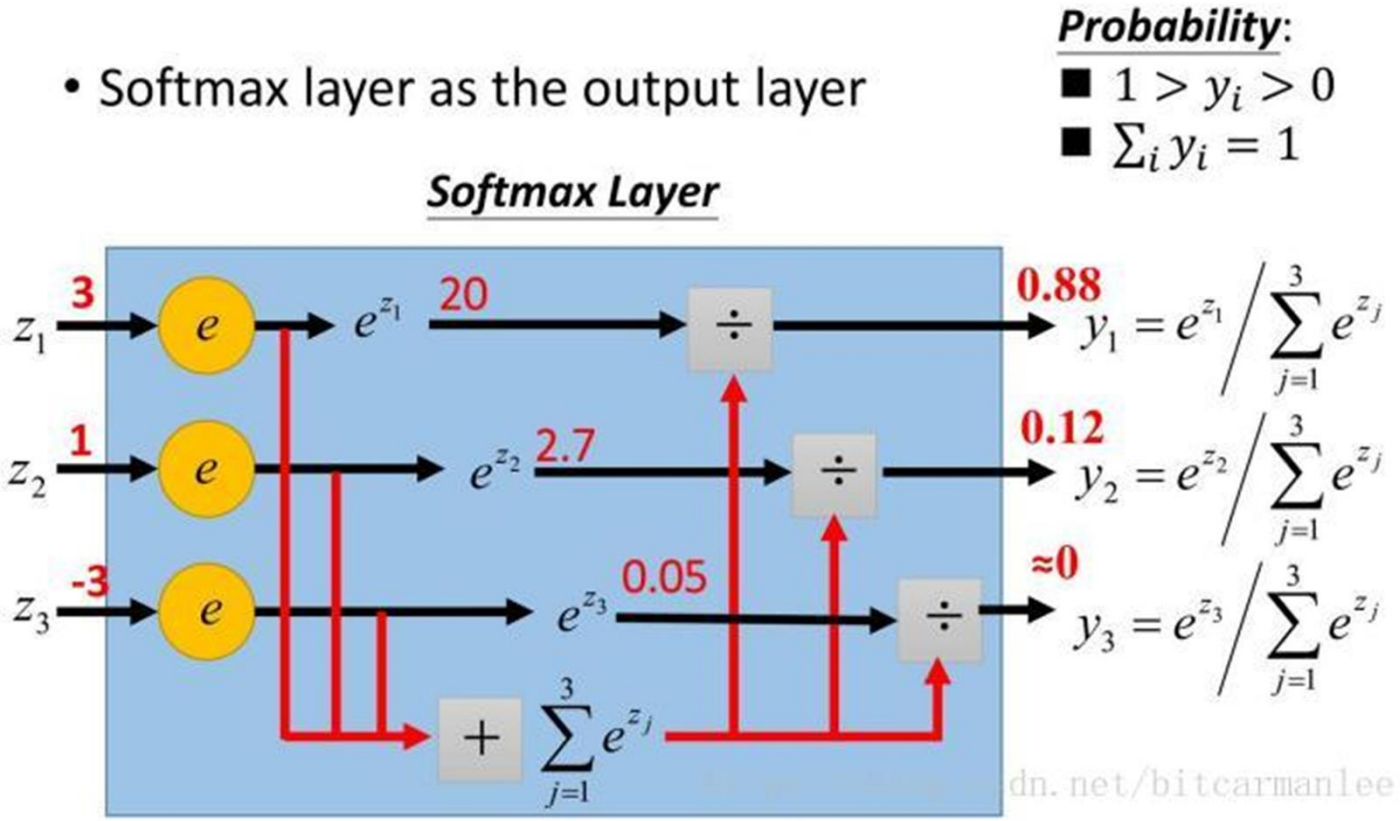
- 他把一些输入映射为0-1之间的实数，并且归一化保证和为1，因此多分类的概率之和也刚好为1

假设有一个数组V，V_i表示V中的第i个元素，那么这个元素的softmax值为:

$$S_i = \frac{e^i}{\sum_j e^j}$$

该元素的softmax值，就是该元素的指数与所有元素指数和的比值。

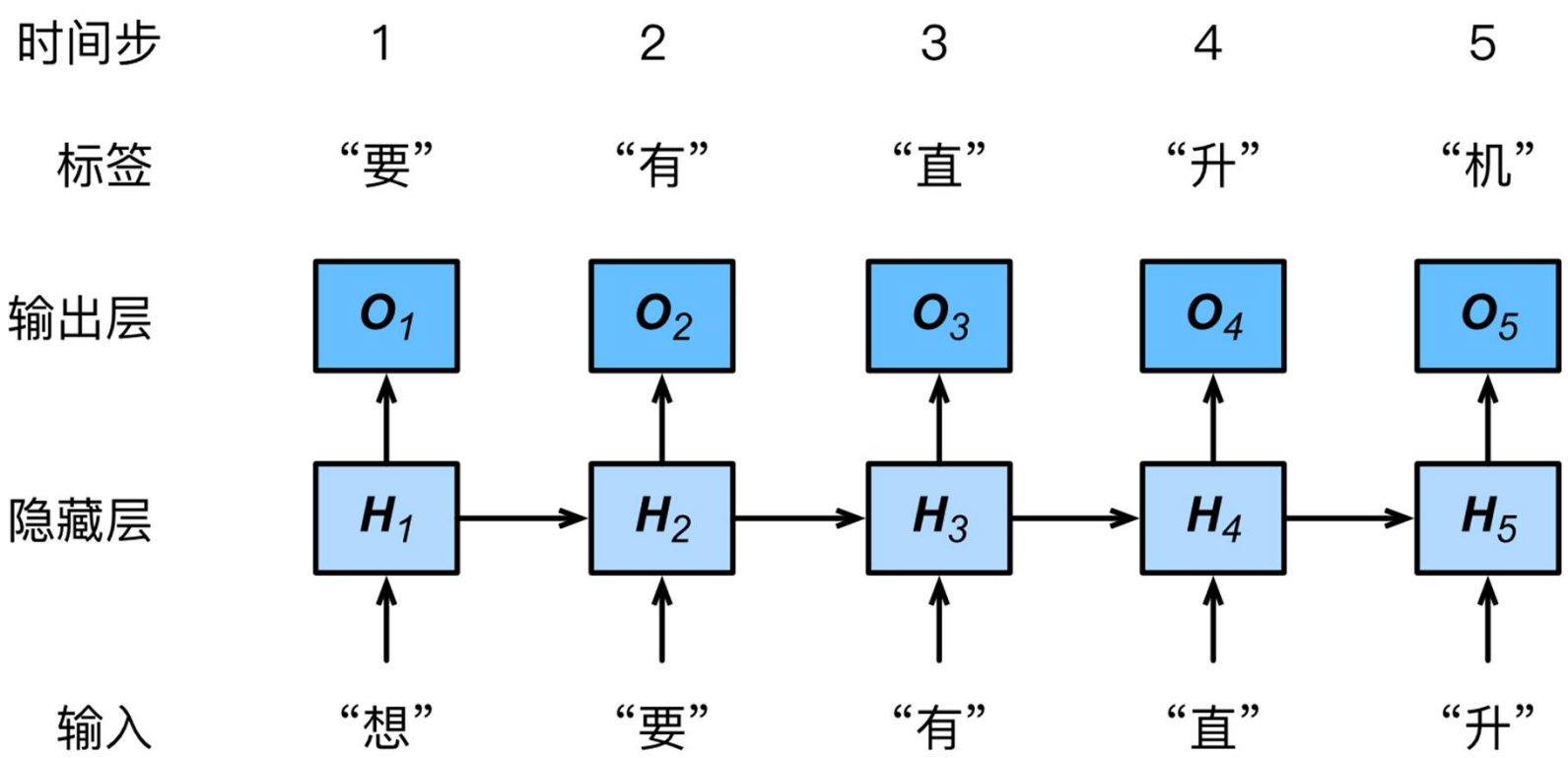
Softmax function



/04

RNN Demo

RNN Demo



RNN Demo

- http://www.gunniliang.com/notebooks/Delete/git_r/two_month_report/202011_2021_1/11_16_to_11_20_third/code/RNN_Keras.ipynb

- http://www.gunniliang.com/notebooks/Delete/git_r/two_month_report/202011_2021_1/11_16_to_11_20_third/code/RNN_Scratch.ipynb
- (代碼部分 有些部分尚待整理解修正)

thanks!

Complicated to be Simple

Sean