循环神经网络RNN

雍宾宾

yongbb@lzu.edu.cn

RNN应用

姚寨沟

陇云西望远

南雁北来愁

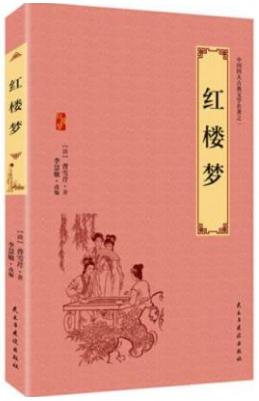
一片天边月

中流到岭头

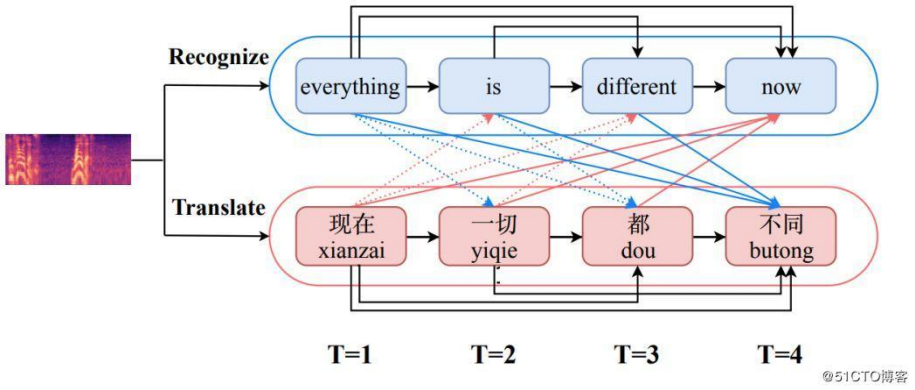
九歌：
http://jiuge.thunlp.org/

循环神经网络

Natural Language Processing, NLP



1. 文档分类：新闻分类



2. Seq2Seq：英汉互译

【买家差评】穿上后大家都说像大妈，根本没有商品图片上那个女的好看。

【卖家解释】这是哪的话！你哪里像大妈，你简直就是天使，要不是降临到地上的时候头先着地了，你穿什么都好看！

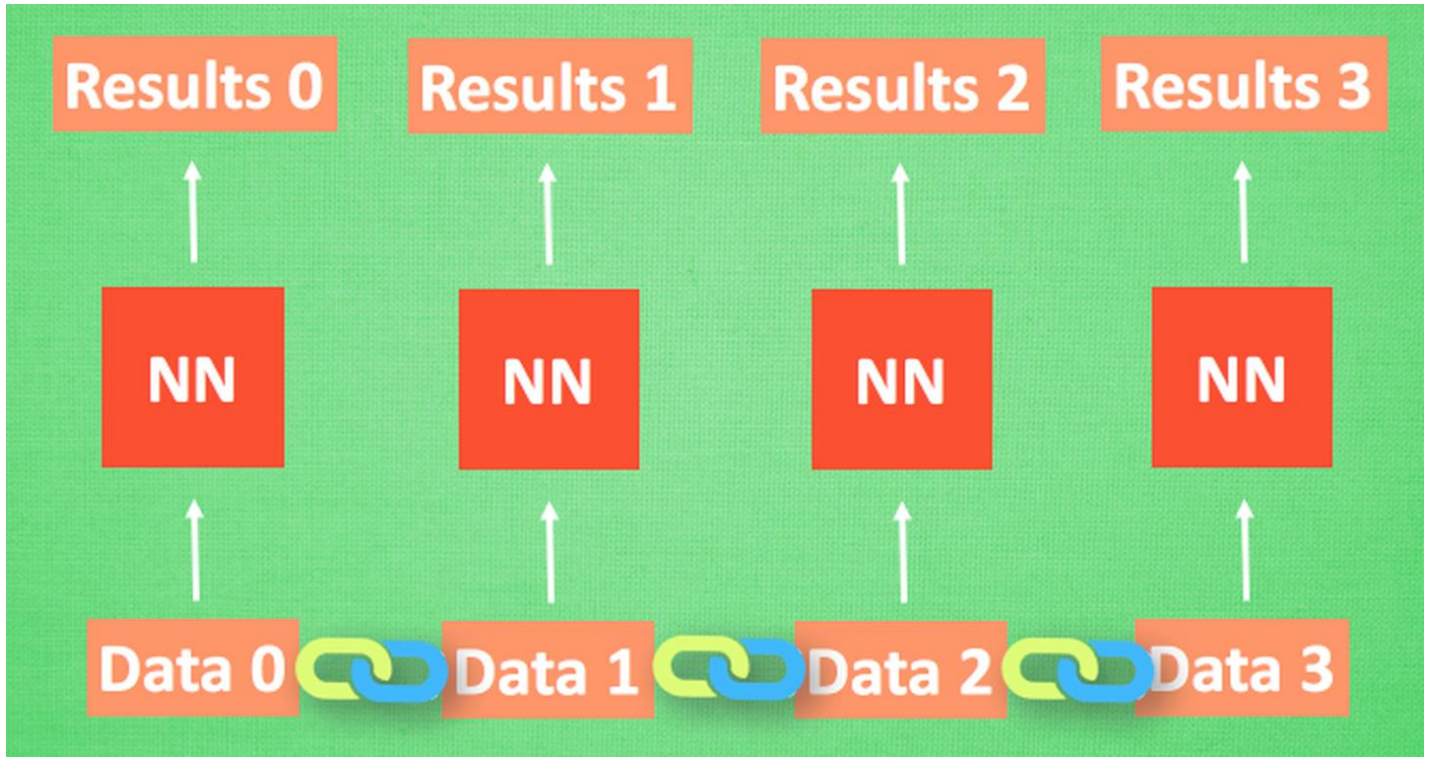
3. 情感分析：电影淘宝评论（正面或负面）

4. 时间序列预测

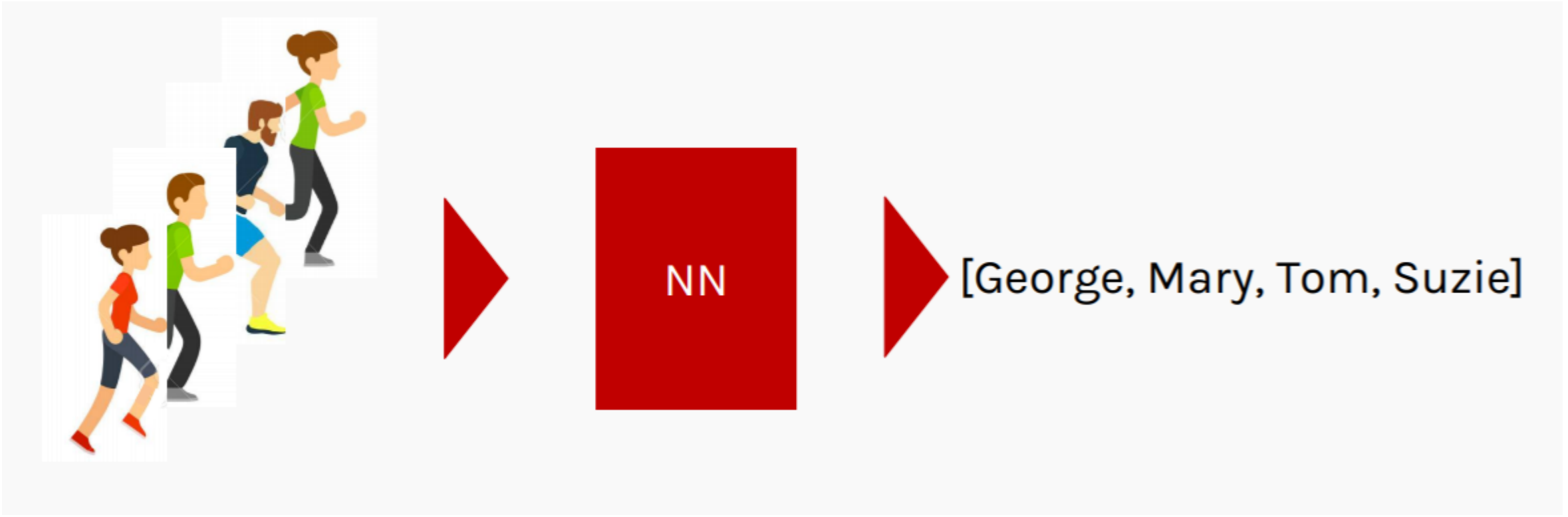
循环神经网络（Recurrent Neural Network, RNN）

我昨天上学迟到了，老师批评了____。

我的手机坏了，我打算____一部新手机。

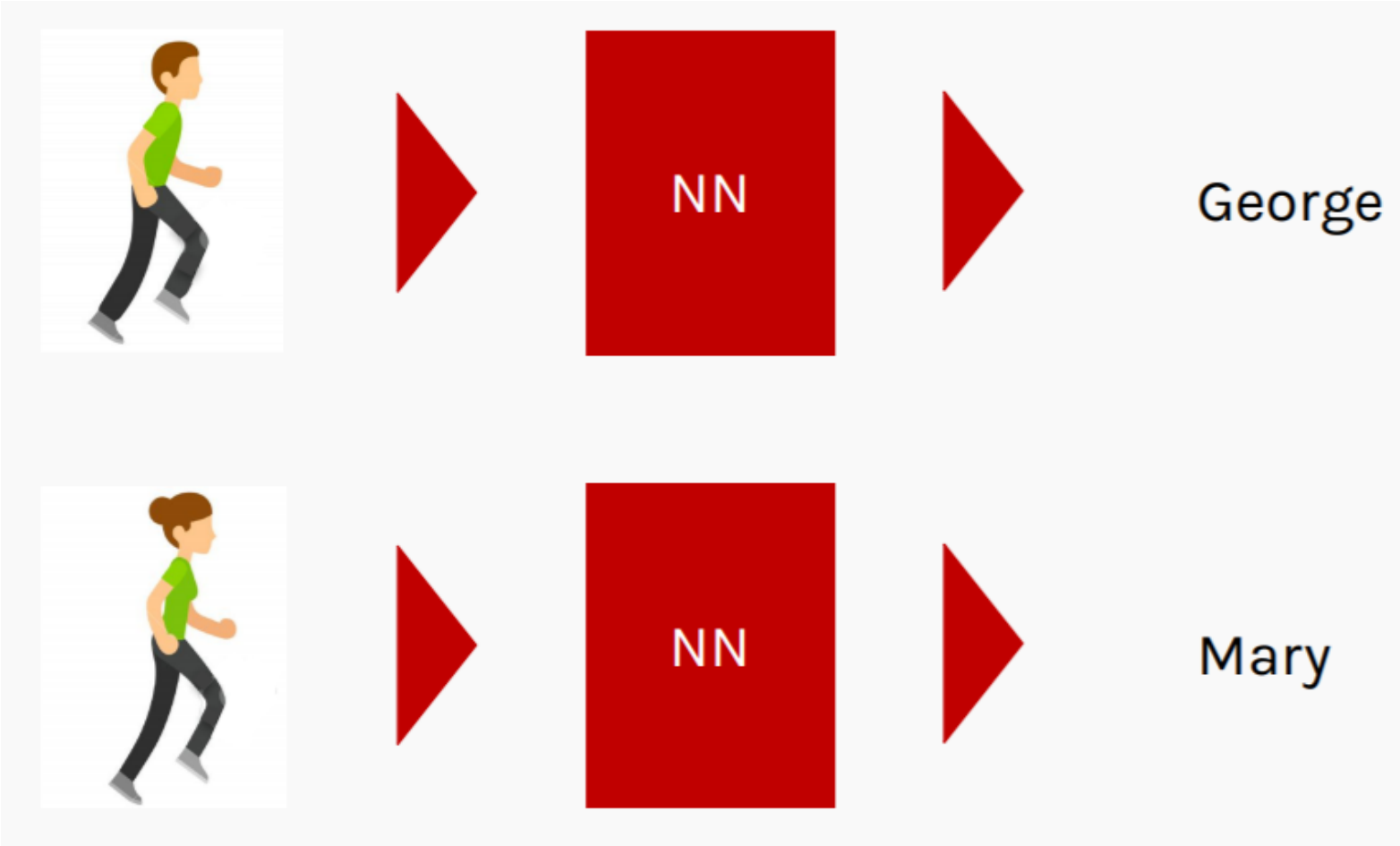


传统NN



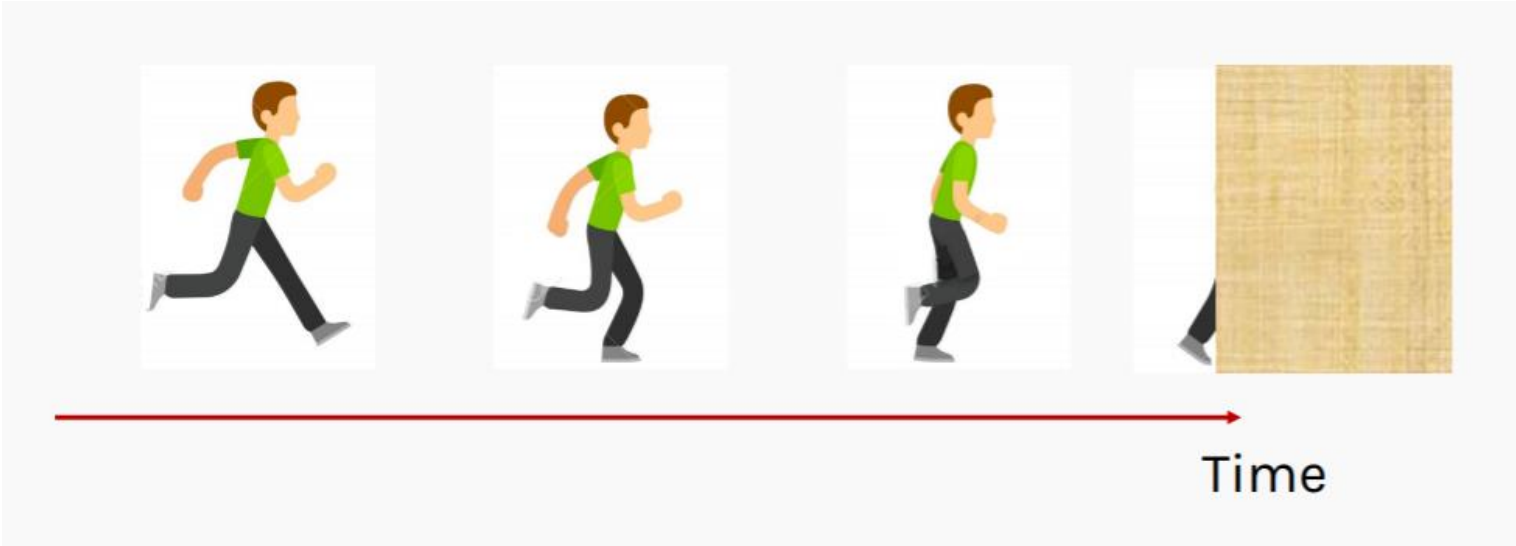
传统NN问题

训练样本

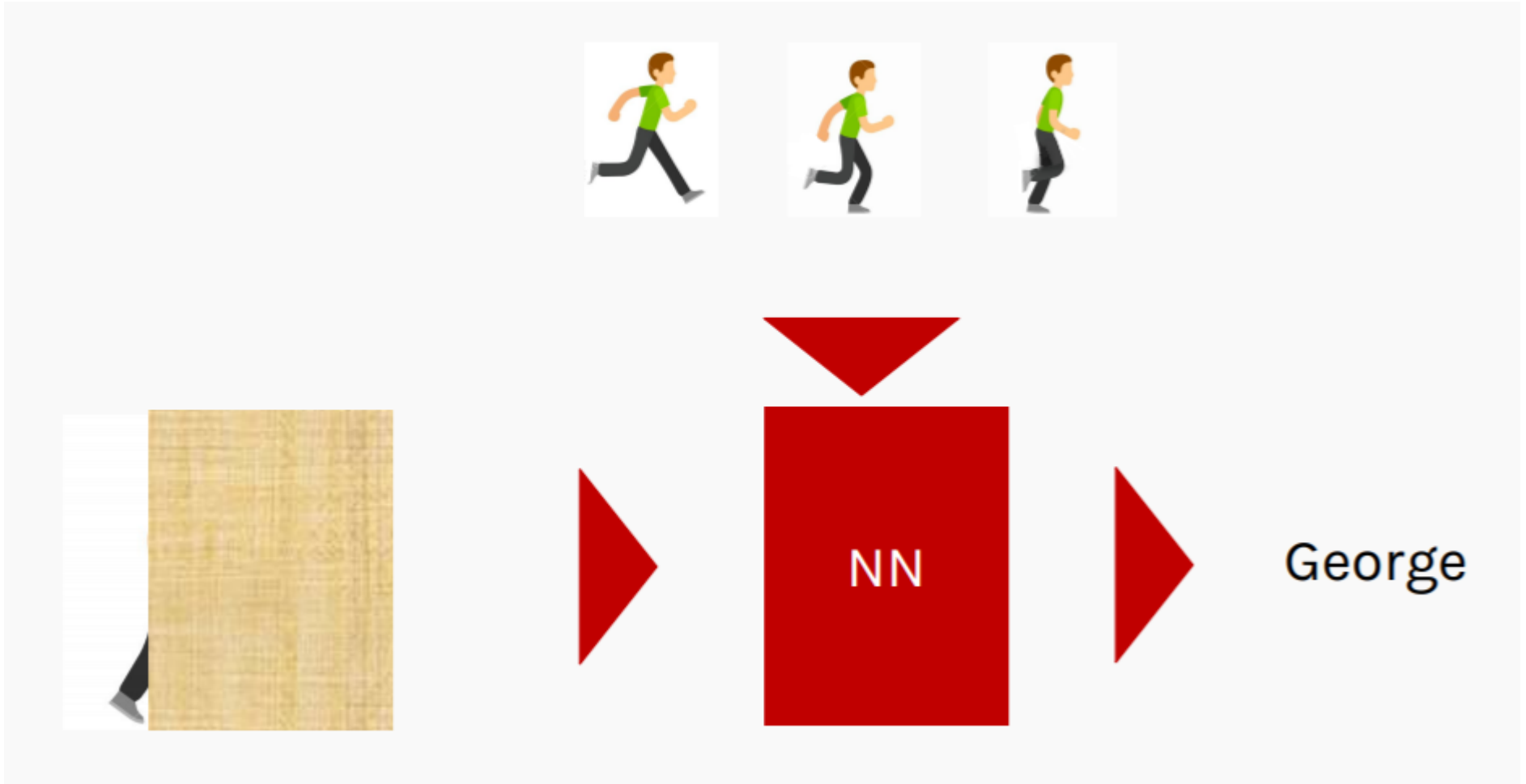


正确识别

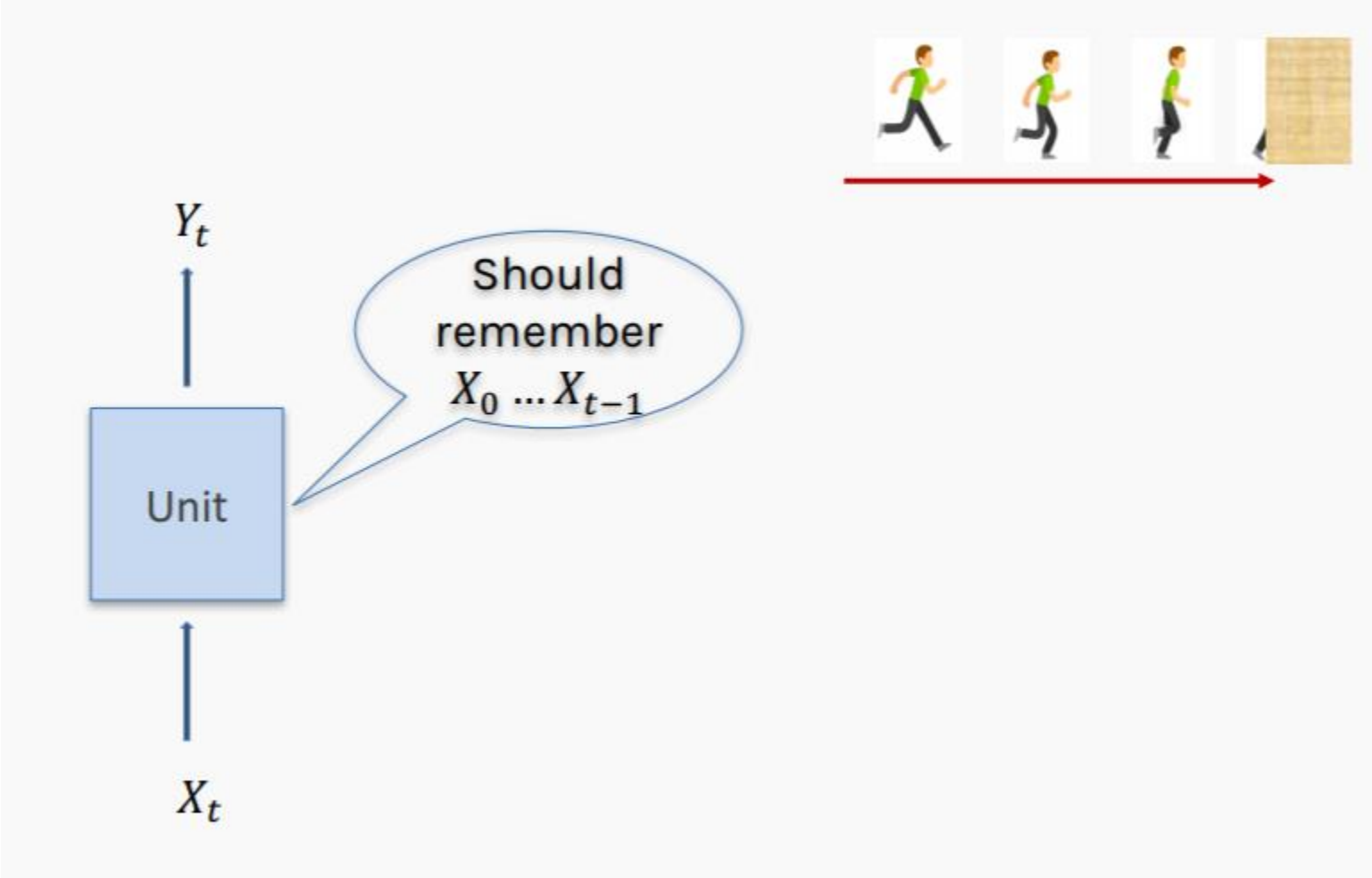
RNN-传统NN问题



RNN

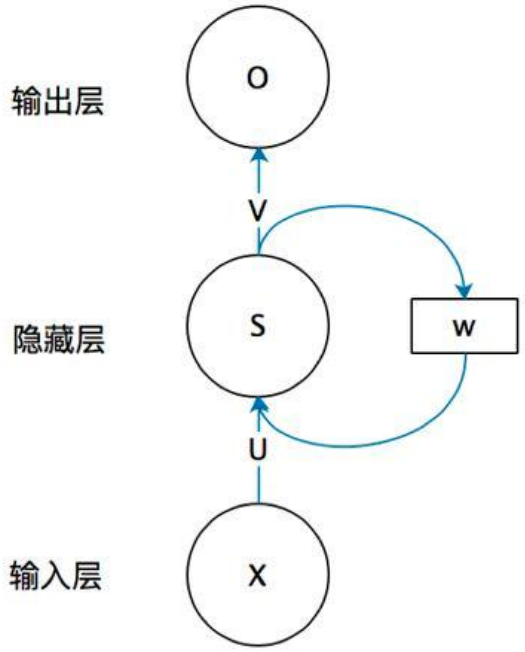


RNN-解决方法



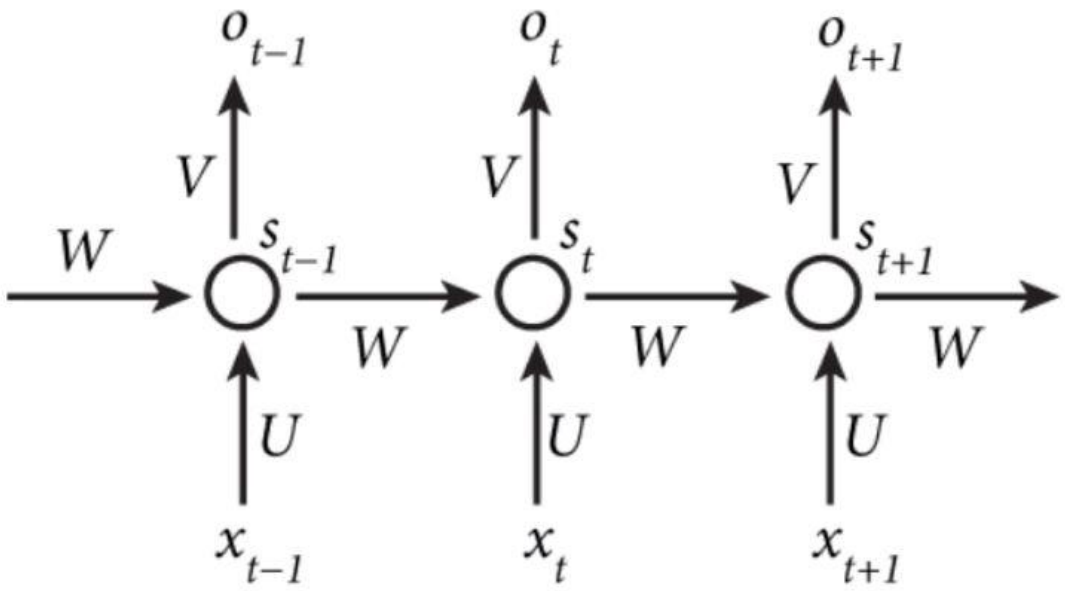
RNN模型

1990年，Jeffrey Elman

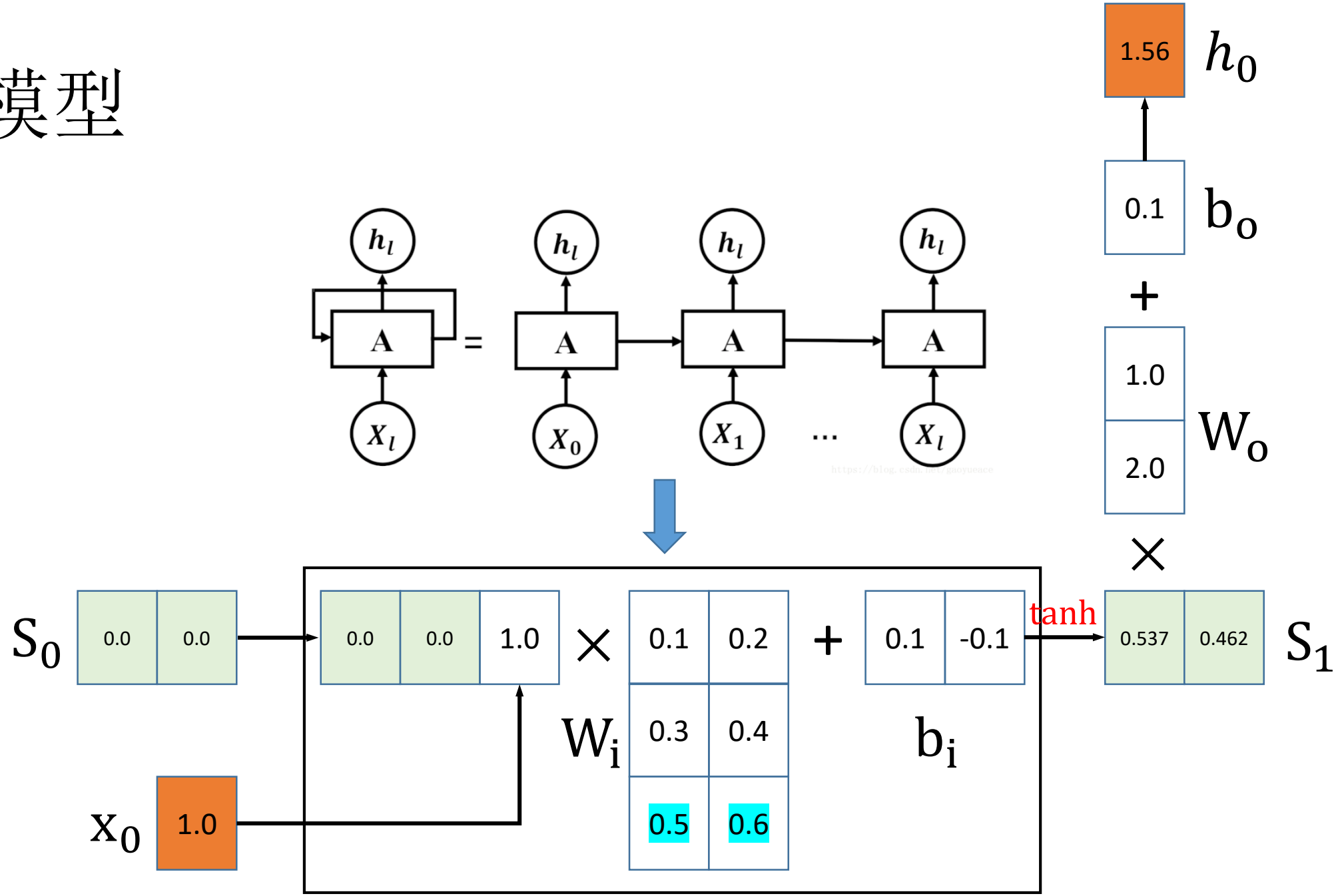


循环层

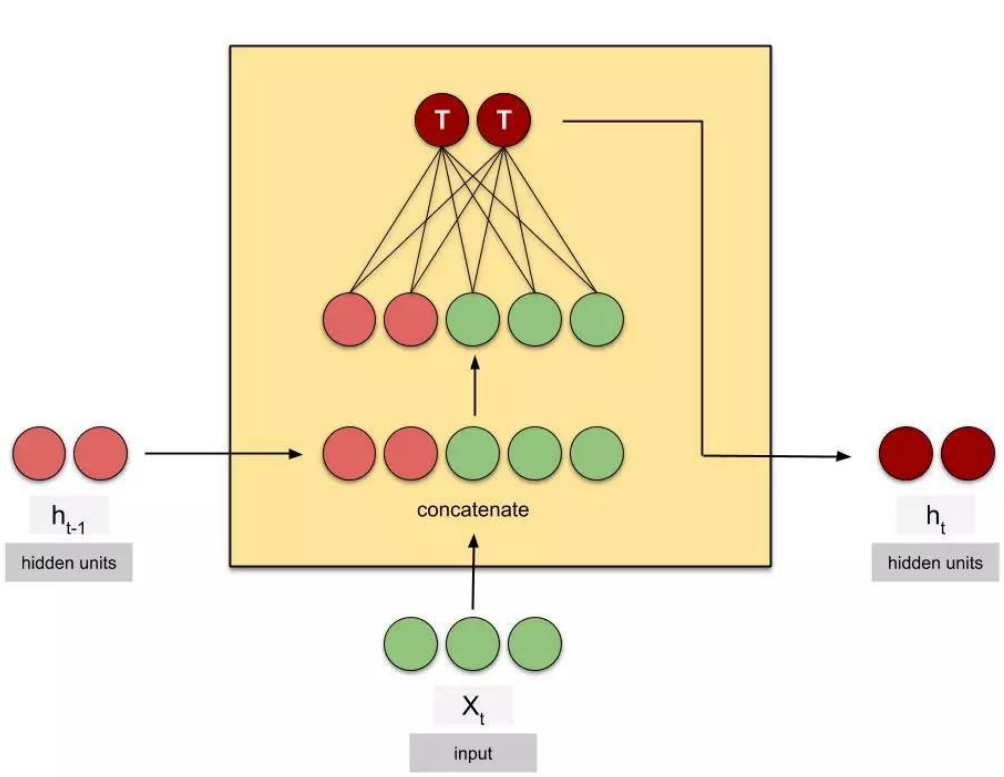
按照时间线展开

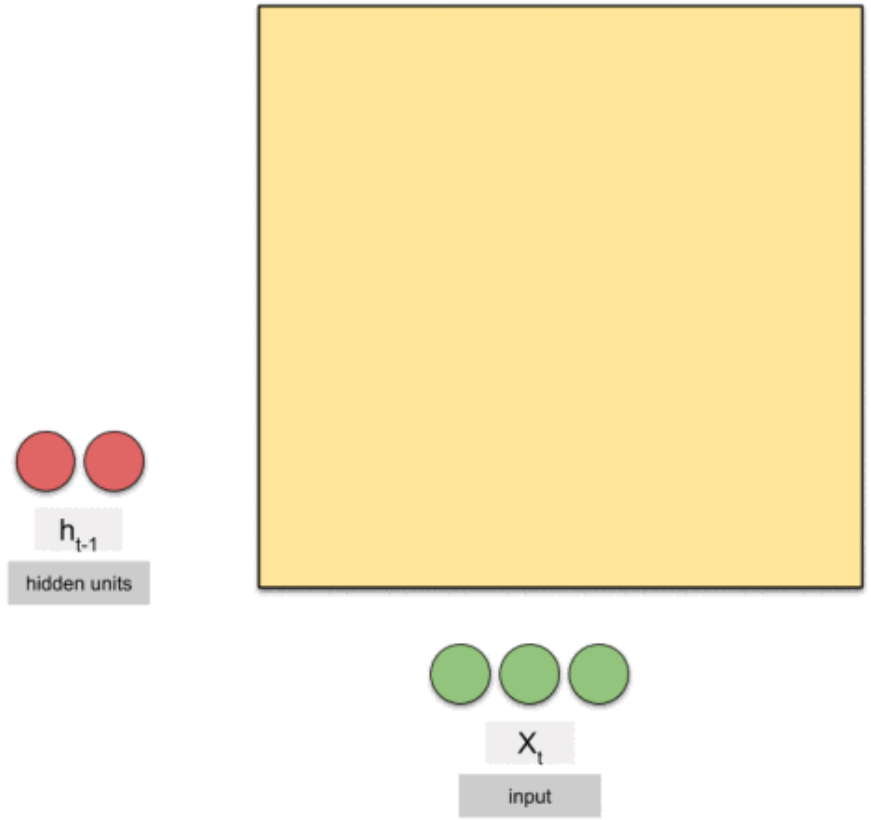


RNN模型

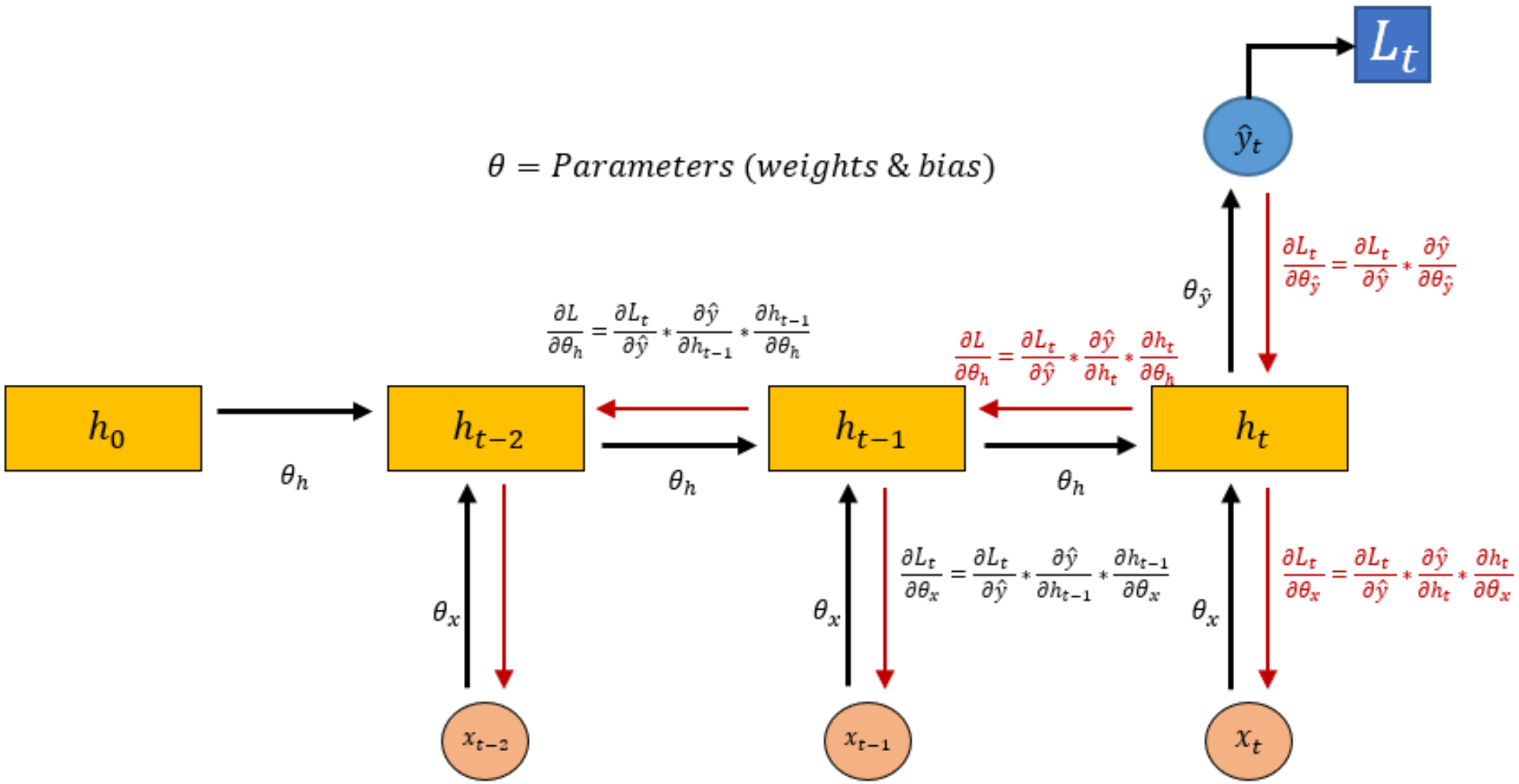

1.56
h_0
0.1
b_o
+
1.0
2.0
W_o
×
h_l
A
=
X_l
X_0
X_1
...
S_0
0.0
0.0
0.0
0.0
1.0
×
0.1
0.2
0.3
0.4
0.5
0.6
W_i
+
0.1
-0.1
b_i
tanh
0.537
0.462
S_1
x_0
1.0

RNN模型


T
T
concatenate
h_{t-1}
hidden units
h_t
hidden units
X_t
input


h_{t-1}
hidden units
X_t
input

RNN模型-反向传播

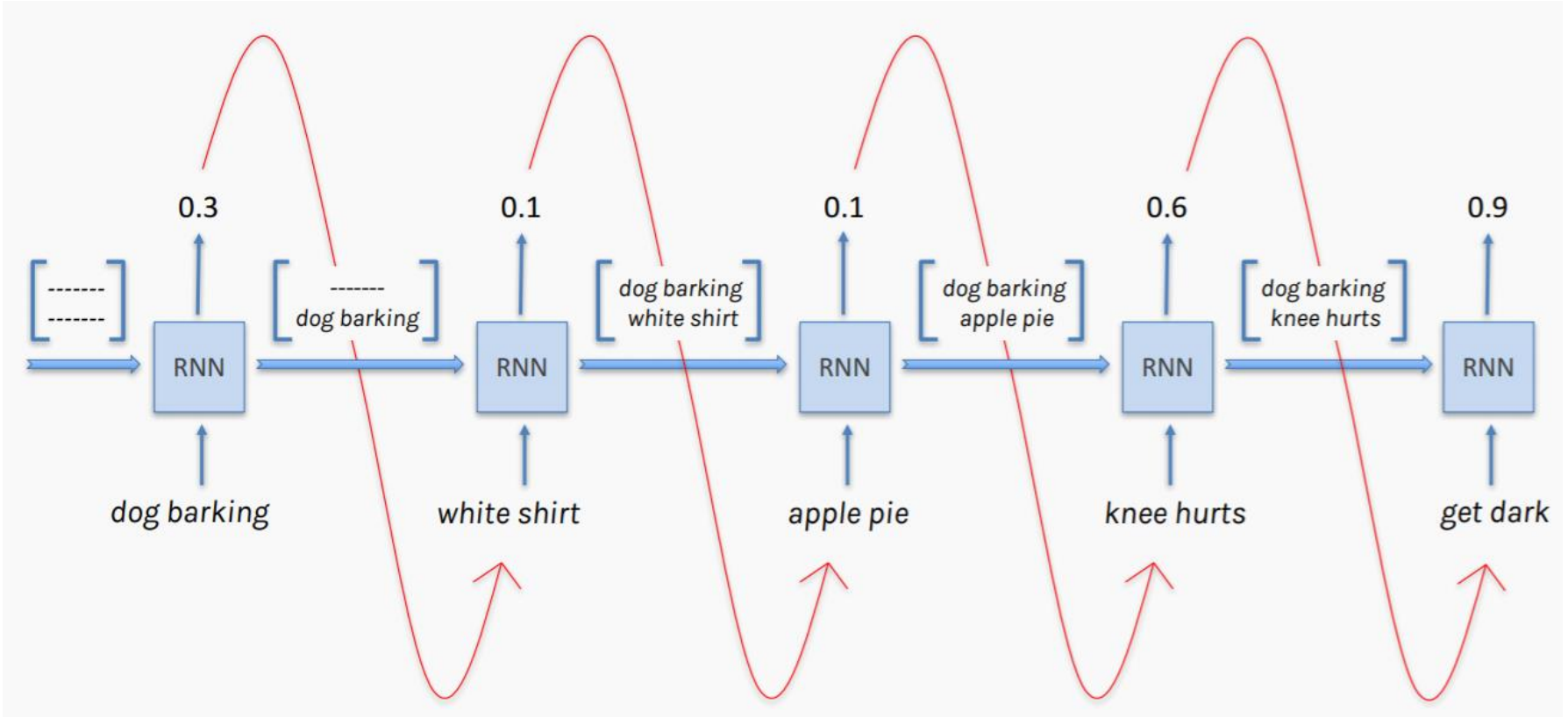

L_t
$\hat{y}_t$
$\theta = Parameters\ (weights\ \&\ bias)$
$\frac{\partial L_t}{\partial \theta_{\hat{y}}} = \frac{\partial L_t}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial \theta_{\hat{y}}}$
$\theta_{\hat{y}}$
$\frac{\partial L}{\partial \theta_h} = \frac{\partial L_t}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial h_{t-1}} * \frac{\partial h_{t-1}}{\partial \theta_h}$
$\frac{\partial L}{\partial \theta_h} = \frac{\partial L_t}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial h_t} * \frac{\partial h_t}{\partial \theta_h}$
h_0
h_{t-2}
h_{t-1}
h_t
θ_h
θ_h
θ_h
θ_x
θ_x
θ_x
$\frac{\partial L_t}{\partial \theta_x} = \frac{\partial L_t}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial h_{t-1}} * \frac{\partial h_{t-1}}{\partial \theta_x}$
$\frac{\partial L_t}{\partial \theta_x} = \frac{\partial L_t}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial h_t} * \frac{\partial h_t}{\partial \theta_x}$
x_{t-2}
x_{t-1}
x_t

imdb

```
from keras.datasets import imdb
from keras.preprocessing import sequence
from keras.layers import *
from keras.models import  *

max_features=10000
maxlen = 500
batch_size=32
(input_train,y_train),(input_test,y_test) =imdb.load_data(num_words=max_features)
input_train = sequence.pad_sequences(input_train,maxlen=maxlen)
input_test = sequence.pad_sequences(input_test,maxlen=maxlen)

model = Sequential()
model.add(Embedding(max_features,32))
model.add(SimpleRNN(32))
model.add(Dense(1,activation='sigmoid'))
model.compile(optimizer='rmsprop',loss='binary_crossentropy',metrics=['acc'])

history=model.fit(input_train,y_train,epochs=20,batch_size=128,validation_split=0.2)
print(model.evaluate(input_test,y_test))
```

RNN模型

图灵完备 [Siegelmann et al., 1991]：所有的图灵机都可以被一个由使用 *Sigmoid* 型激活函数的神经元构成的全连接循环网络来进行模拟。

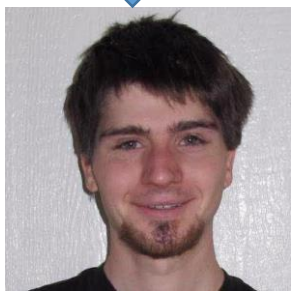


- 仅仅统计特征
- 没有理解语言含义
- 存在梯度消失问题

LSTM (long short-term memory)模型

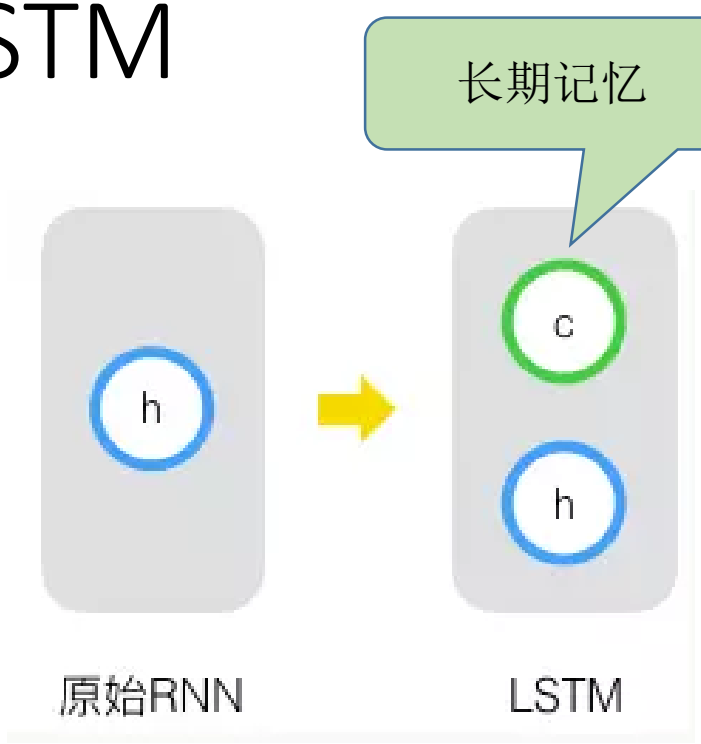
2018年图灵奖

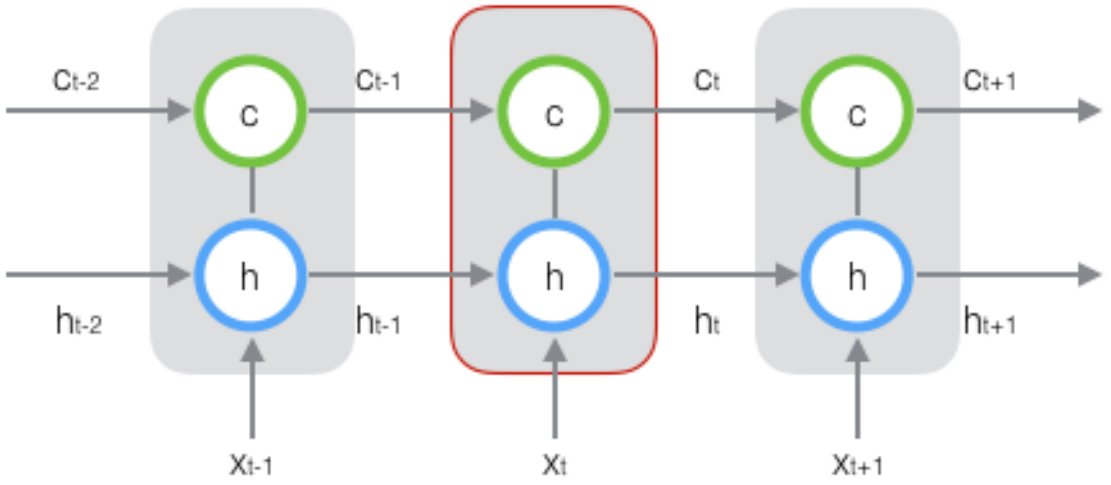
1997年Schmidhuber提出LSTM，1992年提出PM模型



Ian Goodfellow

LSTM





$$O_t = x_t + c_{t-1} + h_{t-1}$$

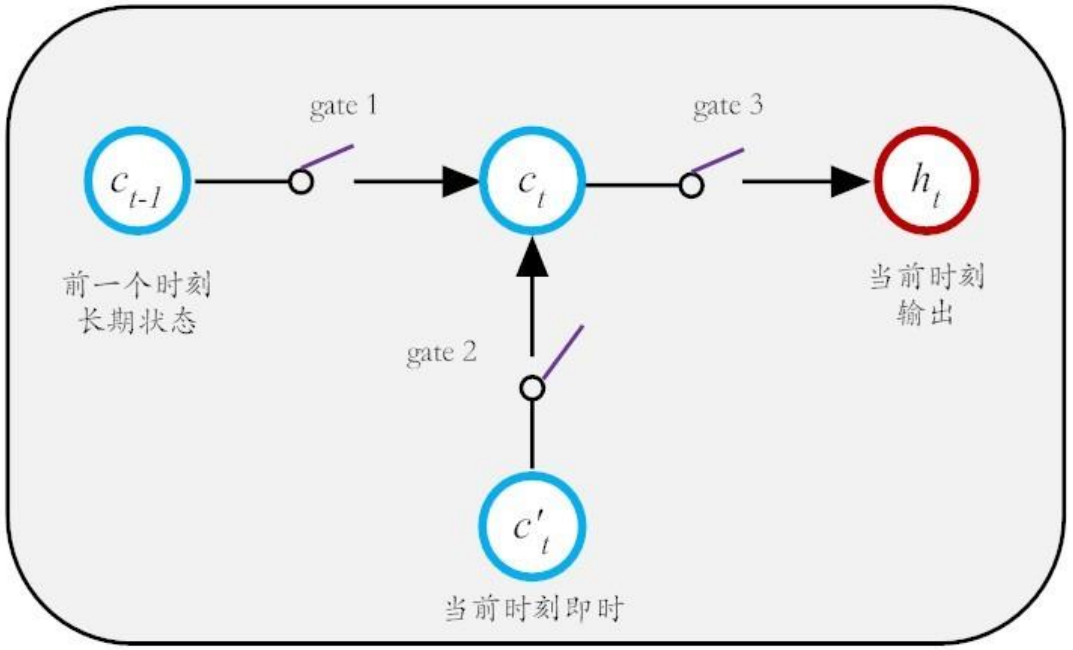
如何控制长期记忆Cell？

如何加入记忆？

如何删除记忆？

如何输出记忆？

LSTM



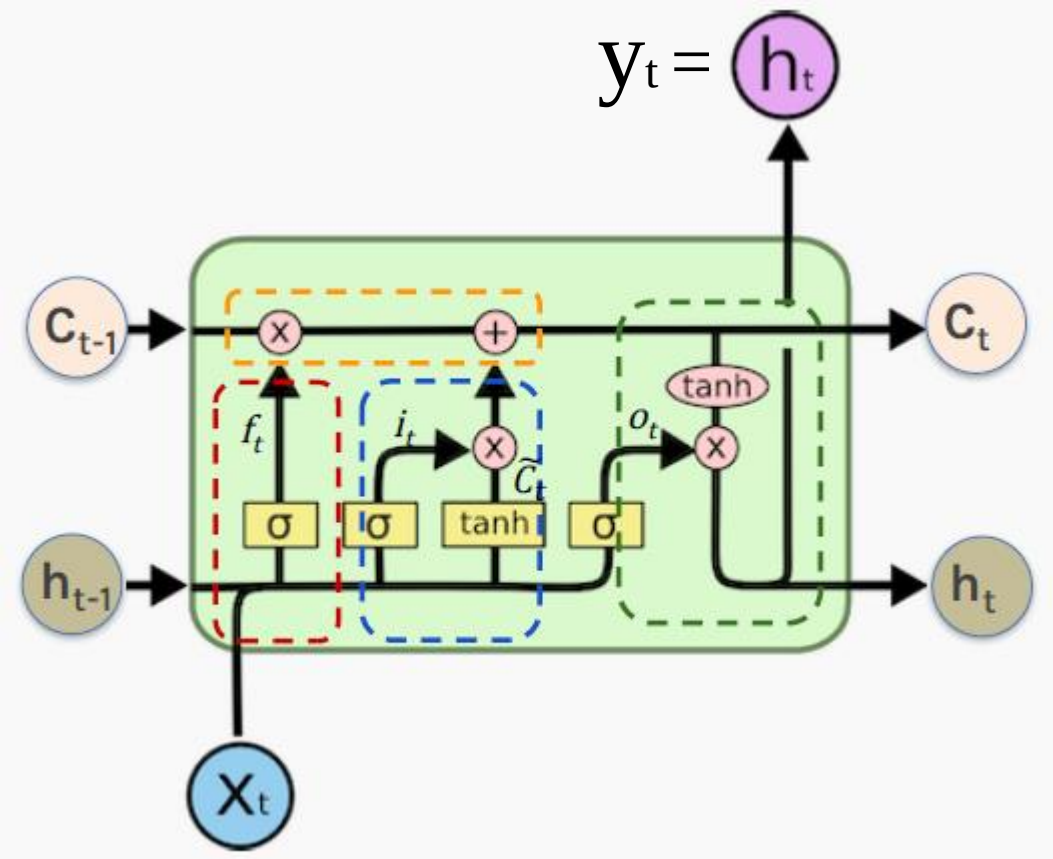
遗忘门 gate1：它决定了上一时刻的单元状态 c_{t-1} 有多少保留到当前时刻 c_t

输入门 gate2：它决定了当前时刻网络的输入 x_t 有多少保存到单元状态 c_t

输出门 gate3：控制单元状态 c_t 有多少输出到 LSTM 的当前输出值 h_t

三打白骨精，赶走孙悟空

LSTM(long short-term memory)模型

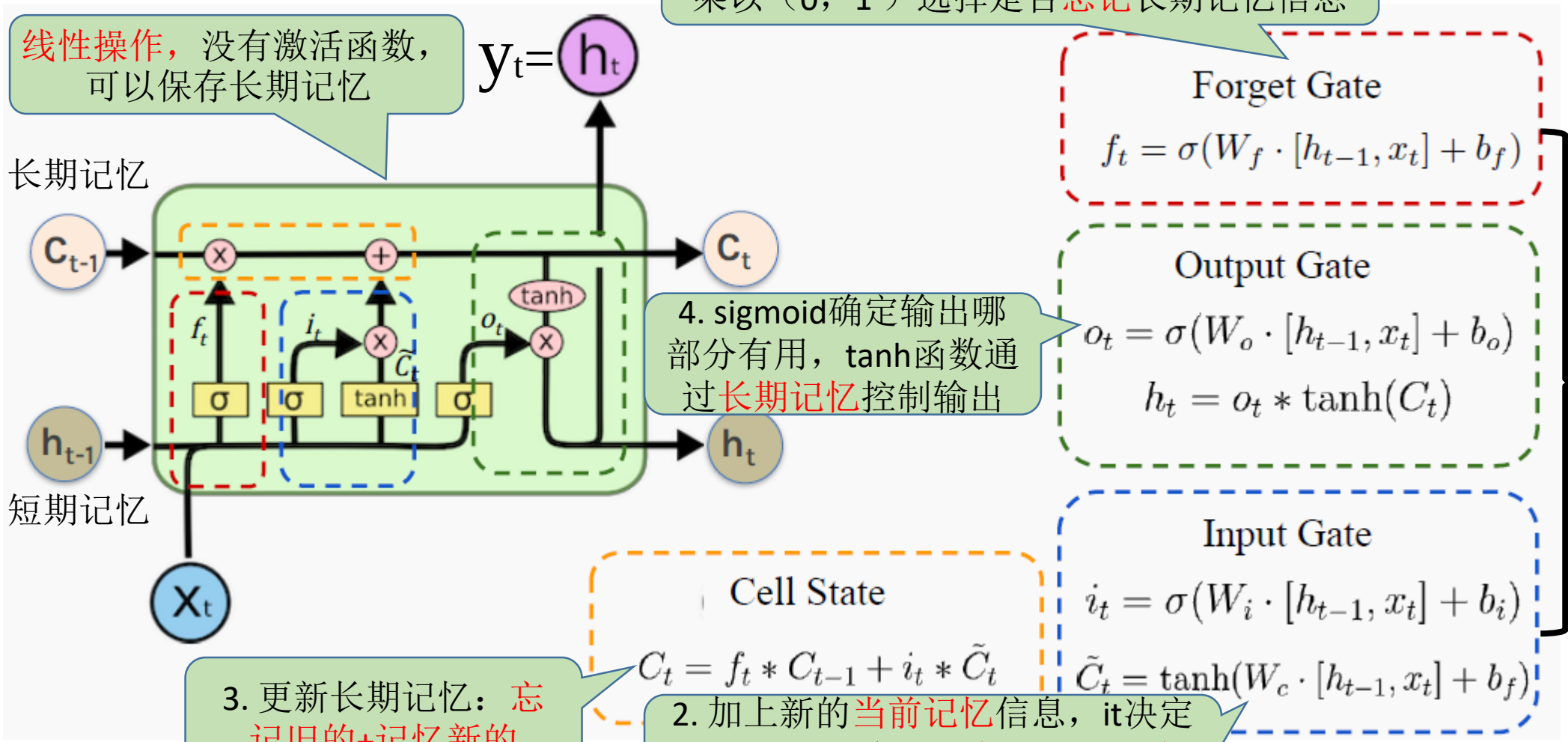


$$\begin{bmatrix} \tilde{\boldsymbol{c}}_t \\ \boldsymbol{o}_t \\ \boldsymbol{i}_t \\ \boldsymbol{f}_t \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(\boldsymbol{W} \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{h}_{t-1} \end{bmatrix} + \boldsymbol{b} \right)$$

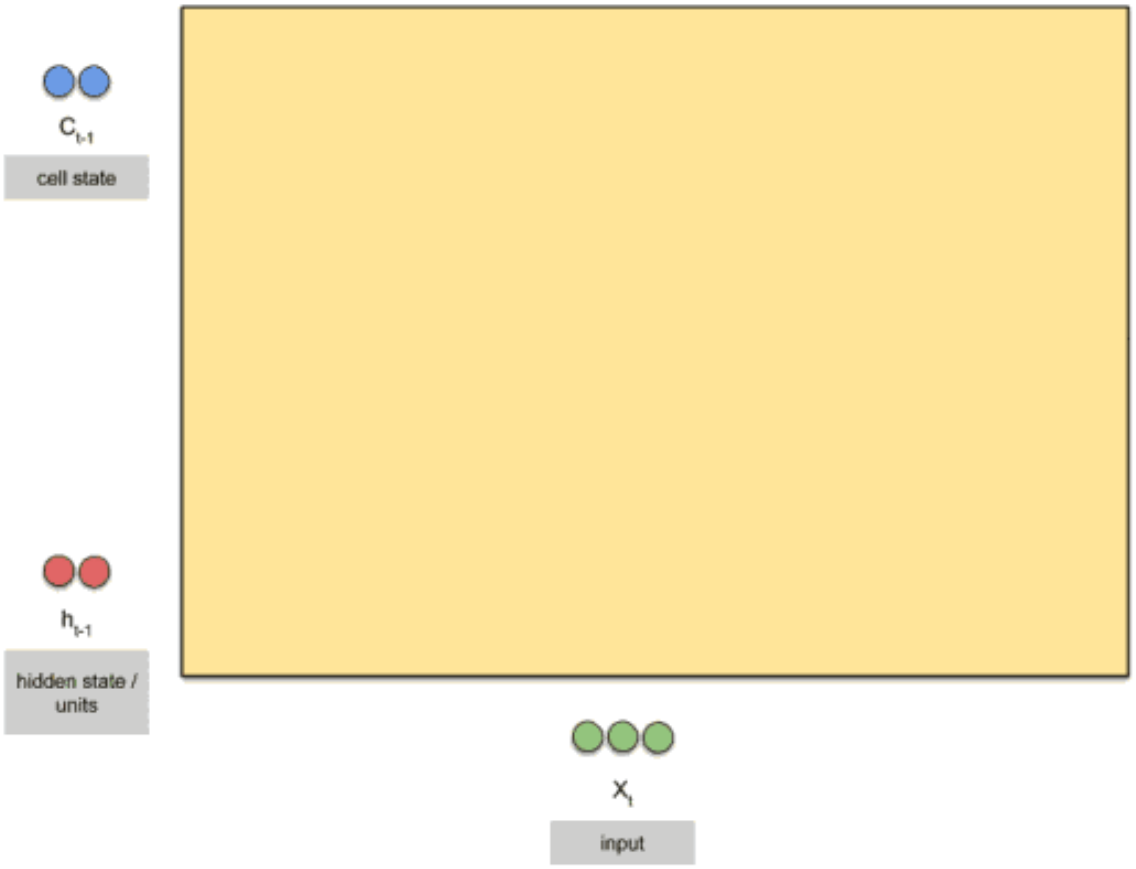
$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t,$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t),$$

LSTM


1. 查看前一个输出和当前输入，
乘以（0，1）选择是否忘记长期记忆信息
线性操作，没有激活函数，
可以保存长期记忆
$y_t = h_t$
长期记忆
短期记忆
C_{t-1}
C_t
h_{t-1}
h_t
X_t
f_t
i_t
$\tilde{C}_t$
o_t
σ
tanh
Forget Gate
$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$
Output Gate
$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$
$h_t = o_t * \tanh(C_t)$
4. sigmoid确定输出哪部分有用，tanh函数通过长期记忆控制输出
Input Gate
$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_f)$
Cell State
$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$
3. 更新长期记忆：忘记旧的+记忆新的
2. 加上新的当前记忆信息，it决定更新哪些信息（避免不重要信息进入记忆），tanh生成候选记忆
W_f
W_o
W_i
W_c
维度均为隐藏层节点数

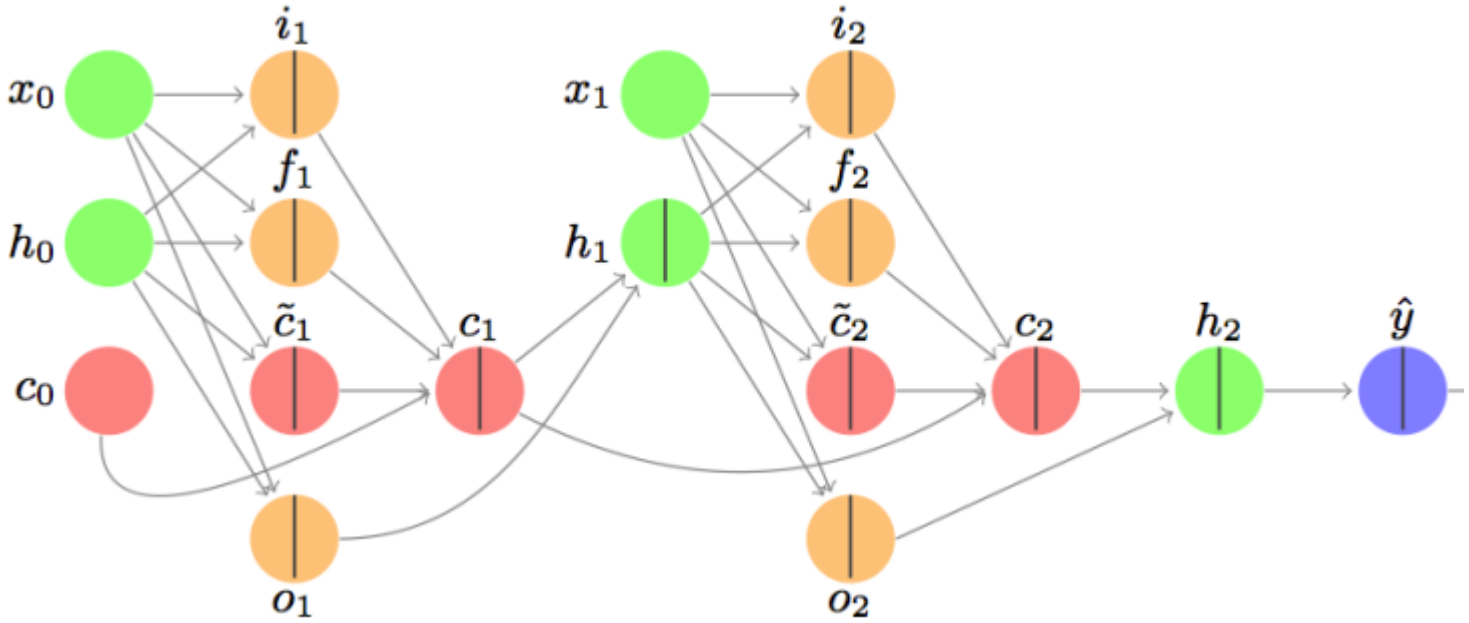
LSTM(long short-term memory)模型

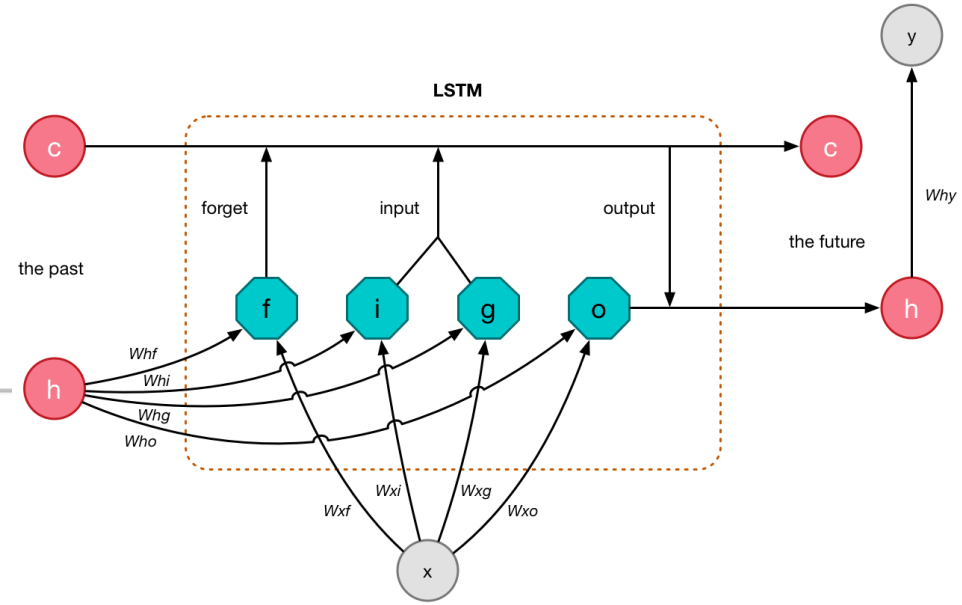


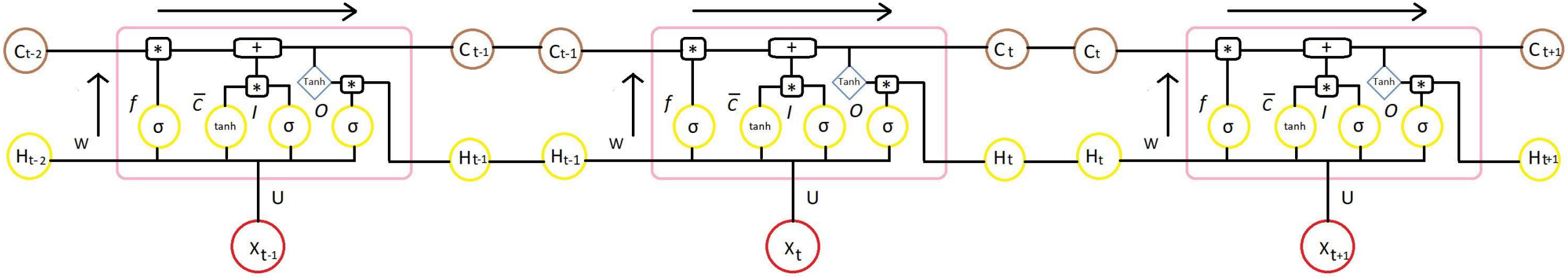

```
from keras.layers import LSTM
LSTM(32, input_length=10, input_dim=64, return_sequences=False)

model = Sequential()
model.add(LSTM(32, batch_input_shape=(None, 10, 64)))
# (Batch_size, Time_step, Input_Sizes)
 model.add(LSTM(32, input_length=10, input_dim=64))
```

输入门和遗忘门是互补关系，具有一定的冗余性

LSTM(long short-term memory)权值

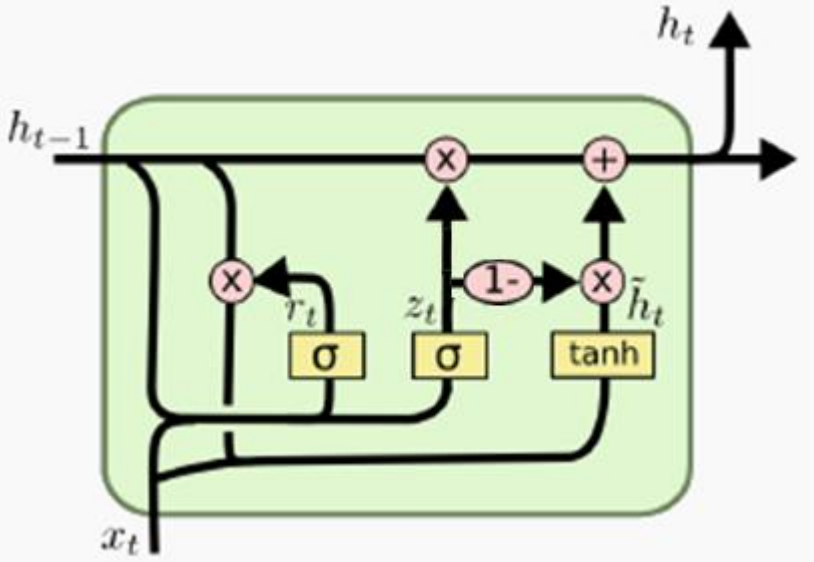






GRU(Gated Recurrent Unit)模型

Kyunghyun Cho等，2014



- GRU不引入额外的记忆单元
- 使用一个门z_t来控制输入和遗忘之间的平衡
- 简化计算快，与LSTM精度相当

$$\boldsymbol{z}_t = \sigma\left(\boldsymbol{W}_z\boldsymbol{x}_t + \boldsymbol{U}_z\boldsymbol{h}_{t-1} + \boldsymbol{b}_z\right)$$

$$\boldsymbol{r}_t = \sigma\left(\boldsymbol{W}_r\boldsymbol{x}_t + \boldsymbol{U}_r\boldsymbol{h}_{t-1} + \boldsymbol{b}_r\right)$$

$$\tilde{\boldsymbol{h}}_t = \tanh\left(\boldsymbol{W}_h\boldsymbol{x}_t + \boldsymbol{U}_h(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}) + \boldsymbol{b}_h\right)$$

$$\boldsymbol{h}_t = \boldsymbol{z}_t \odot \boldsymbol{h}_{t-1} + (1 - \boldsymbol{z}_t) \odot \tilde{\boldsymbol{h}}_t$$

线性操作，保存长期记忆

非线性操作，新记忆

$\boldsymbol{z}_t = 0, \boldsymbol{r} = 1$：退化为简单RNN模型

$\boldsymbol{z}_t = 0, \boldsymbol{r} = 0$：当前状态$\boldsymbol{h}_t$只和当前输入$\boldsymbol{x}_t$相关，和历史状态$\boldsymbol{h}_{t-1}$无关

$\boldsymbol{z}_t = 1$：当前状态$\boldsymbol{h}_t$只和历史状态$\boldsymbol{h}_{t-1}$有关，与当前输入$\boldsymbol{x}_t$无关，

imdb

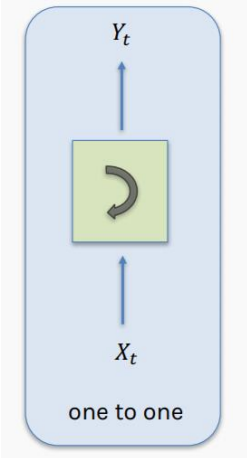
```
from keras.datasets import imdb
from keras.preprocessing import sequence
from keras.layers import *
from keras.models import  *

max_features=10000
maxlen = 500
batch_size=32
(input_train,y_train),(input_test,y_test) =imdb.load_data(num_words=max_features)
input_train = sequence.pad_sequences(input_train,maxlen=maxlen)
input_test = sequence.pad_sequences(input_test,maxlen=maxlen)

model = Sequential()
model.add(Embedding(max_features,32))
model.add(LSTM(32,return_sequences=True))
model.add(GRU(32))
model.add(Dense(1,activation='sigmoid'))
model.compile(optimizer='rmsprop',loss='binary_crossentropy',metrics=['acc'])

history=model.fit(input_train,y_train,epochs=20,batch_size=128,validation_split=0.2)
print(model.evaluate(input_test,y_test))
```

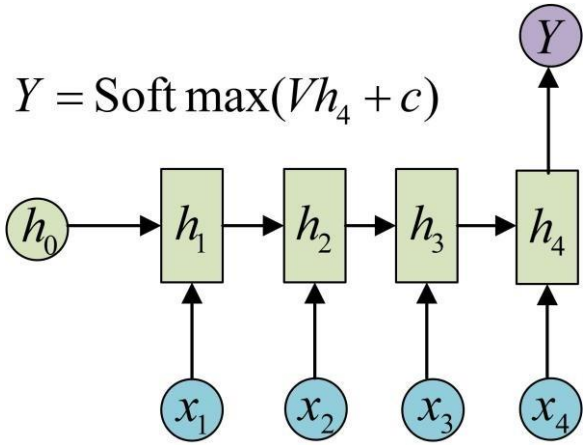
```
input_train=input_train.reshape(input_train.shape+(1,))
input_test=input_test.reshape(input_test.shape+(1,))
model.add(LSTM(32,input_shape=(500,1)))
```

imdb

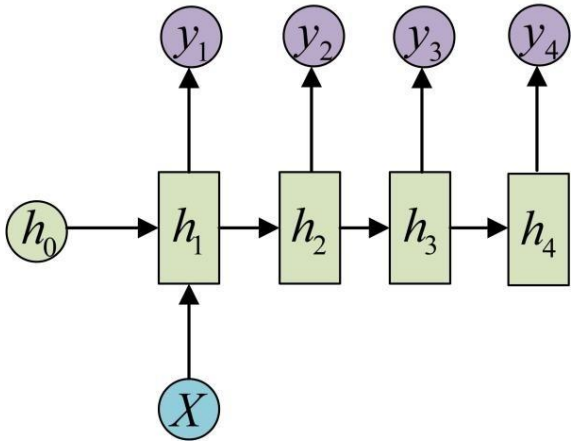
LSTM(No Embedding)	0.5778
SimpleRNN(64)	0.7372
LSTM(32)	0.8496
GRU(32)	0.8348
GRU(64)	0.8111
LSTM(32)+GRU(32)	0.8622

RNN模型结构

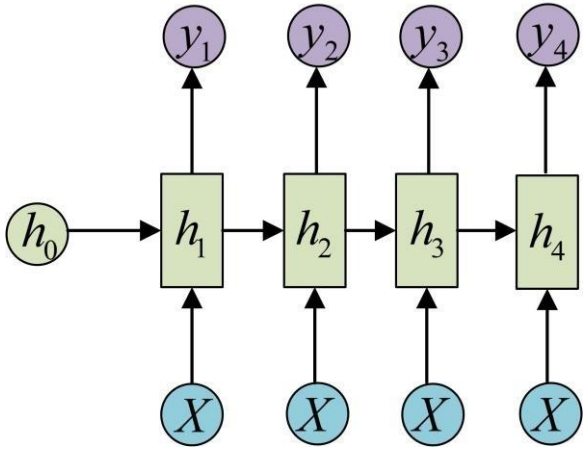




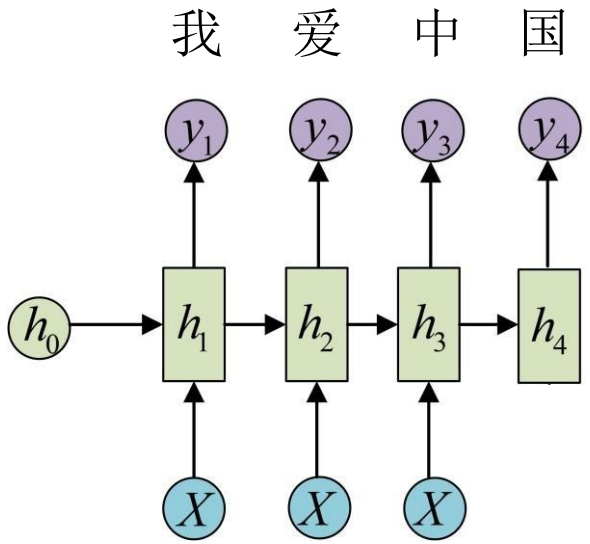
N *to* 1



1 *to* N



N *to* N

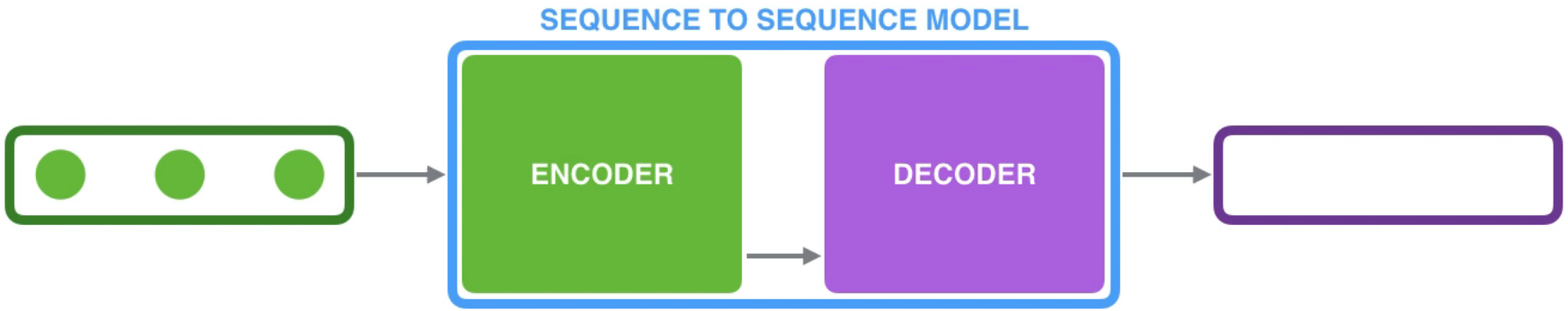


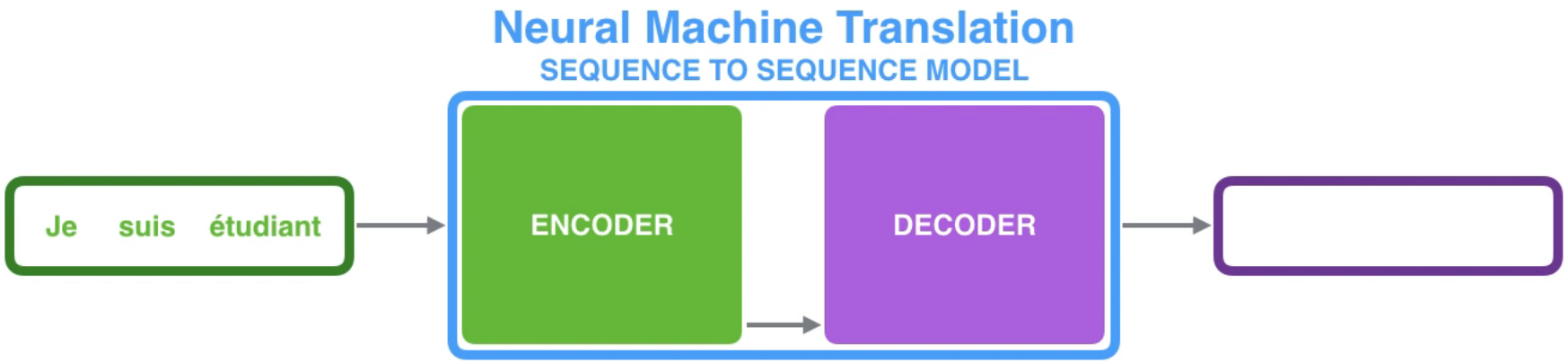
I love China

N *to* M

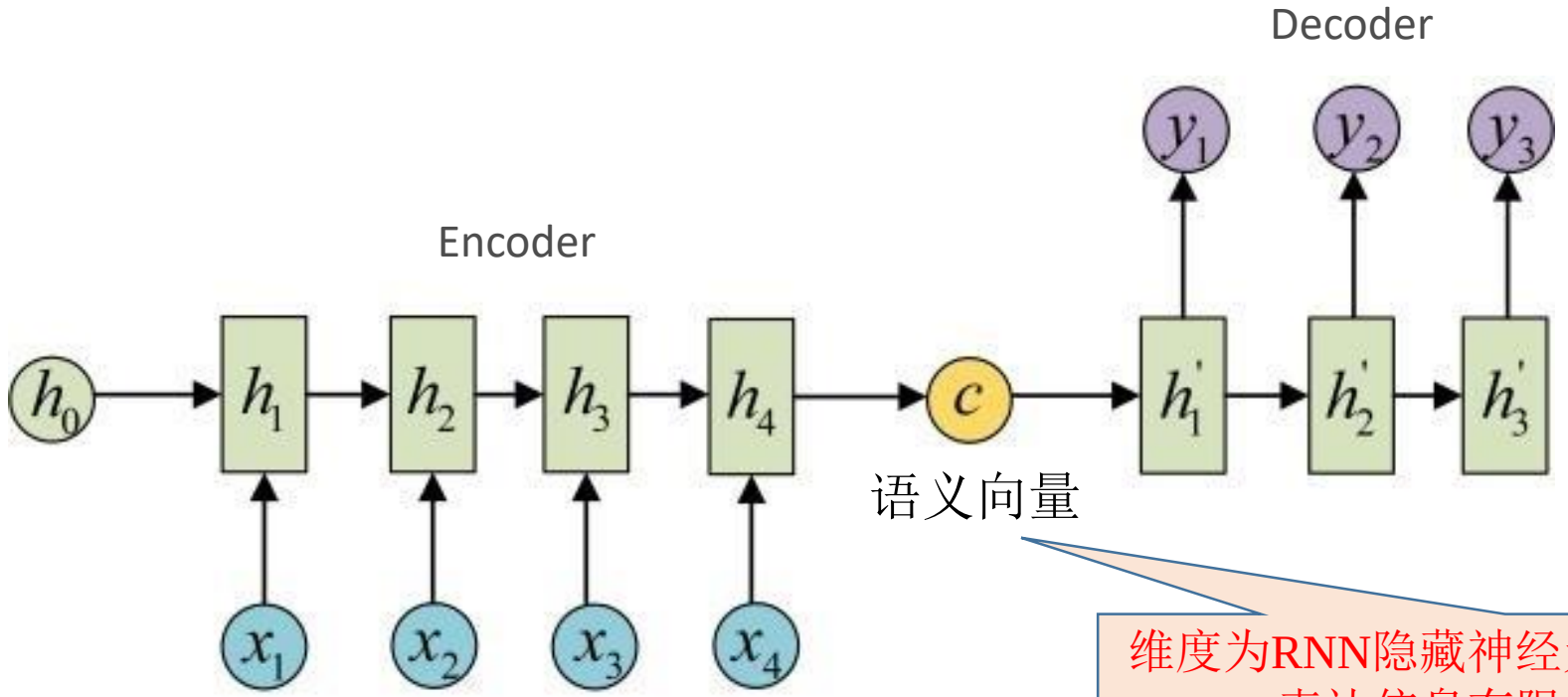
Seq2Seq： N *to* M

https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/





Seq2Seq： N *to* M



维度为RNN隐藏神经元个数256, 512, or 1024
表达信息有限，长句精度下降

机器翻译的实践表明，输入的序列越长，模型的翻译质量越差。

Input

Je

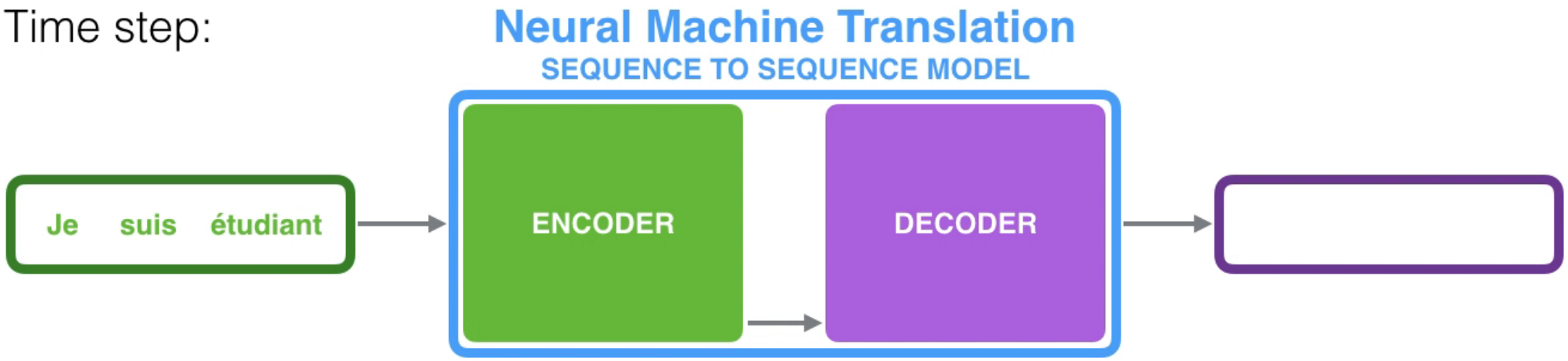
suis

étudiant

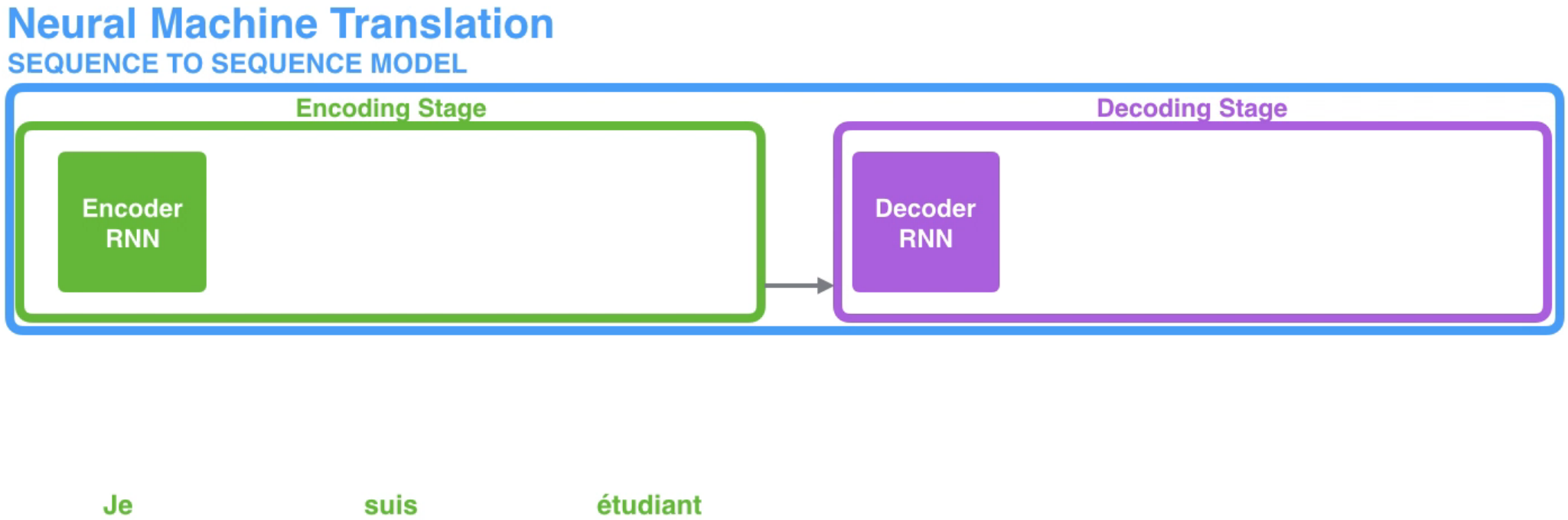
0.901	-0.651	-0.194	-0.822
-0.351	0.123	0.435	-0.200
0.081	0.458	-0.400	0.480

word embedding： 长度200 or 300

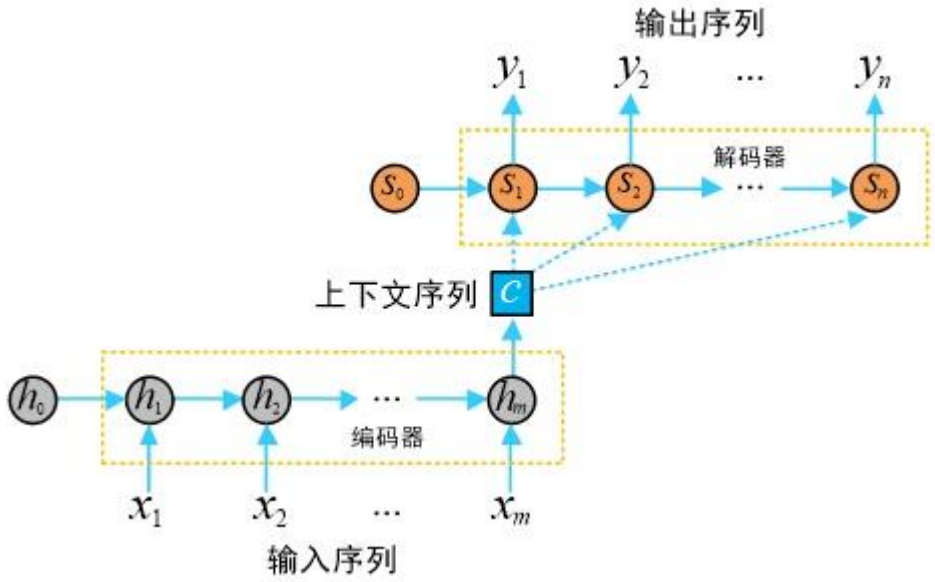
Seq2Seq： N *to* M



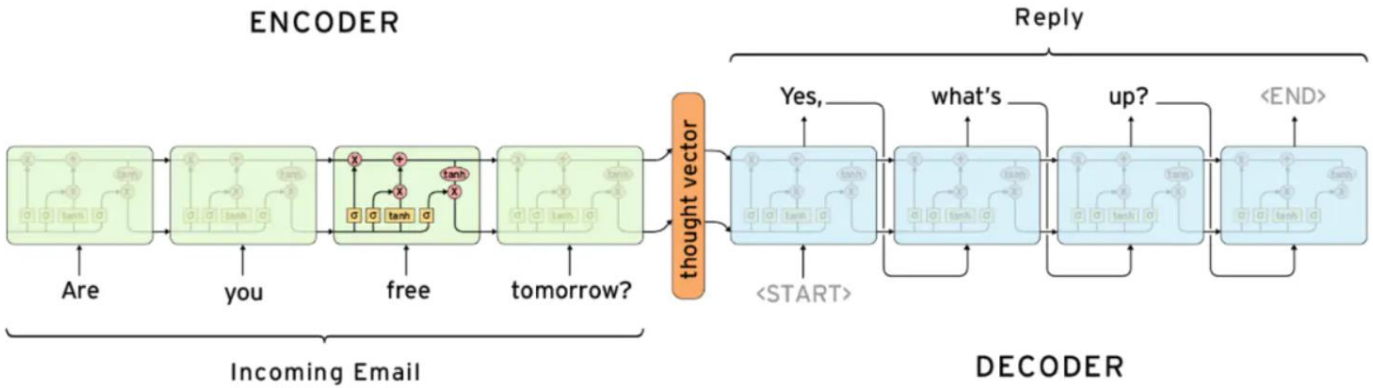
Seq2Seq： N *to* M



Seq2Seq： N *to* M



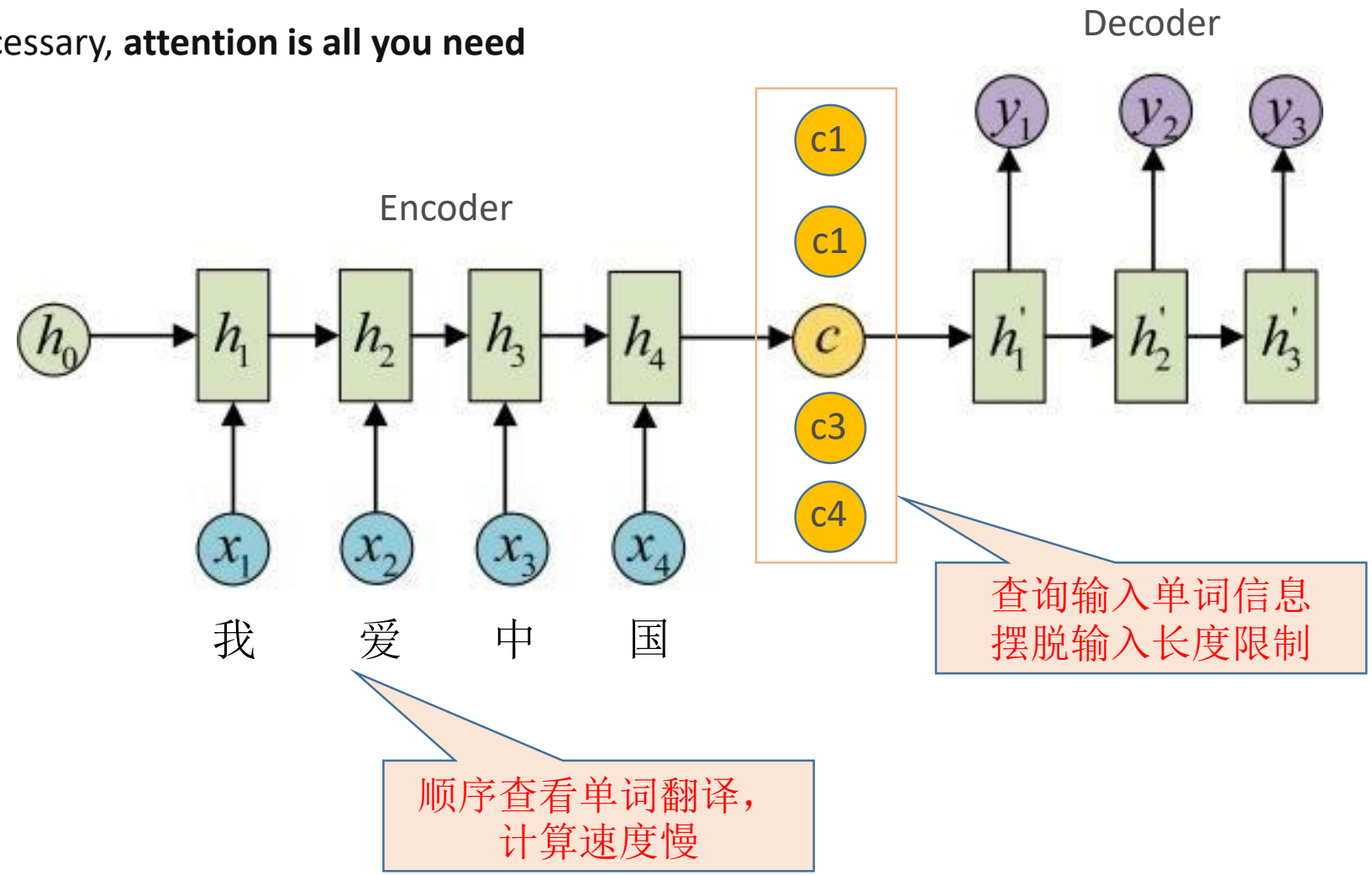
Context作为解码器输入



decoder 每一个时刻的输出作为下一个时刻的输入，
隐藏层状态贯穿了整个 LSTM

Attention机制

RNN is unnecessary, **attention is all you need**

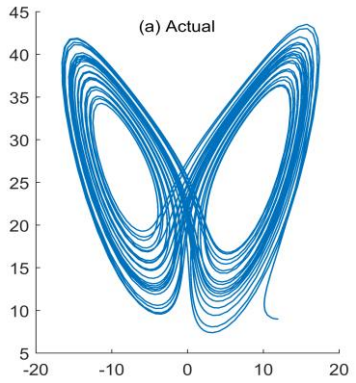


时间序列预测

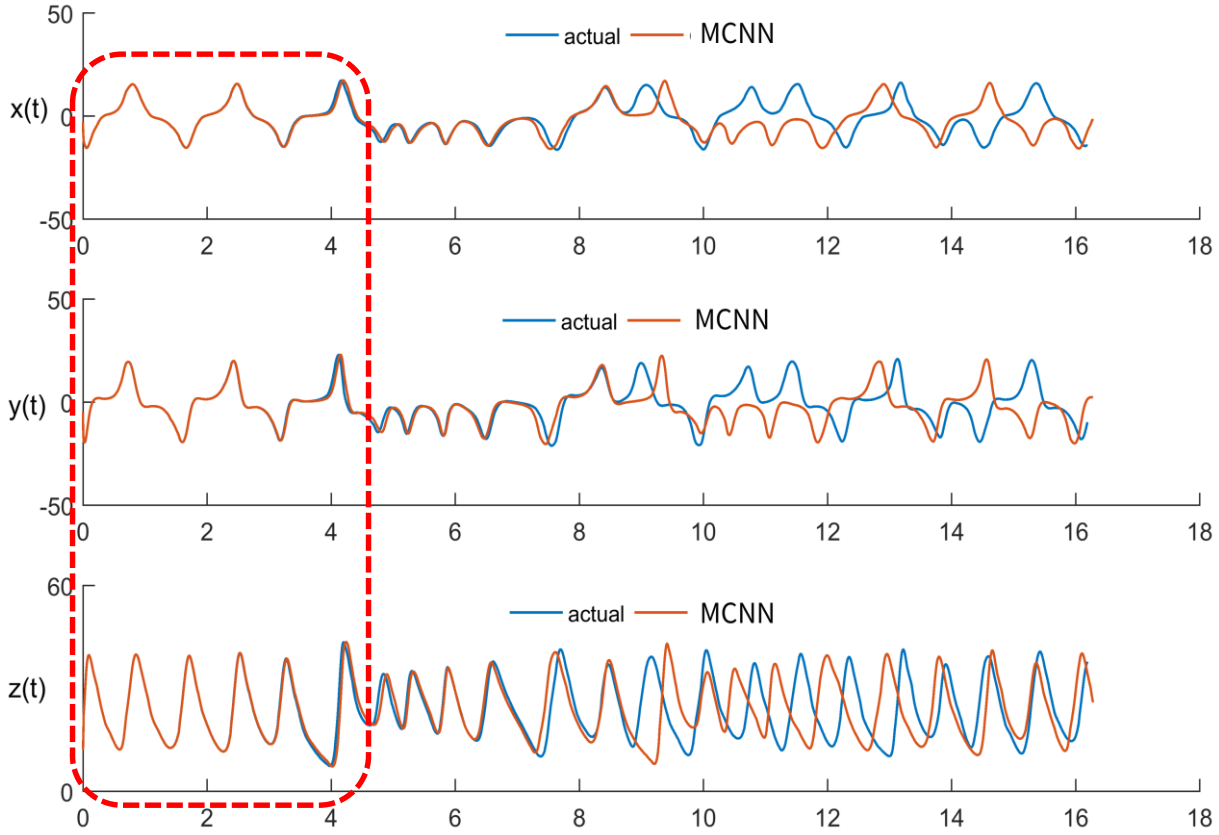
世界上没有两片完全
相同的树叶。

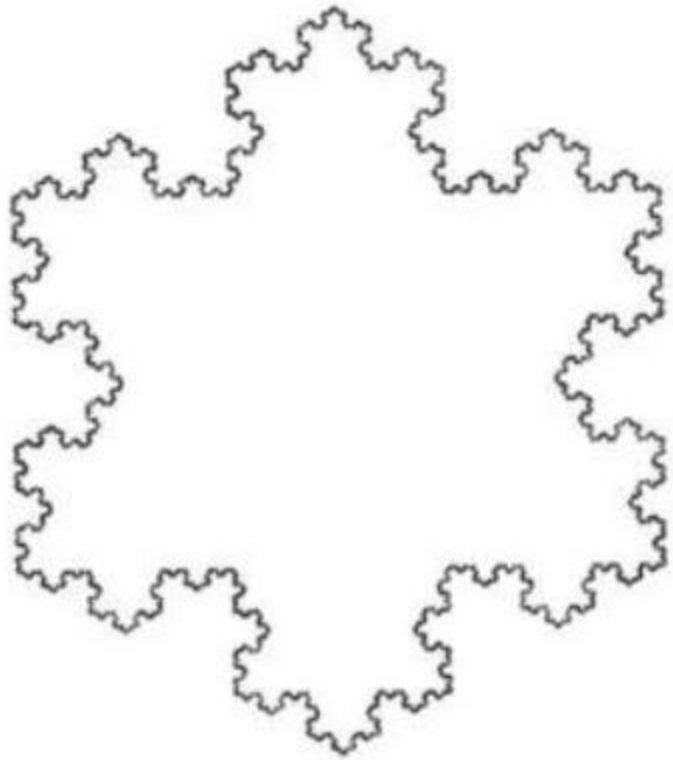
——莱布尼茨（德）

时间序列预测

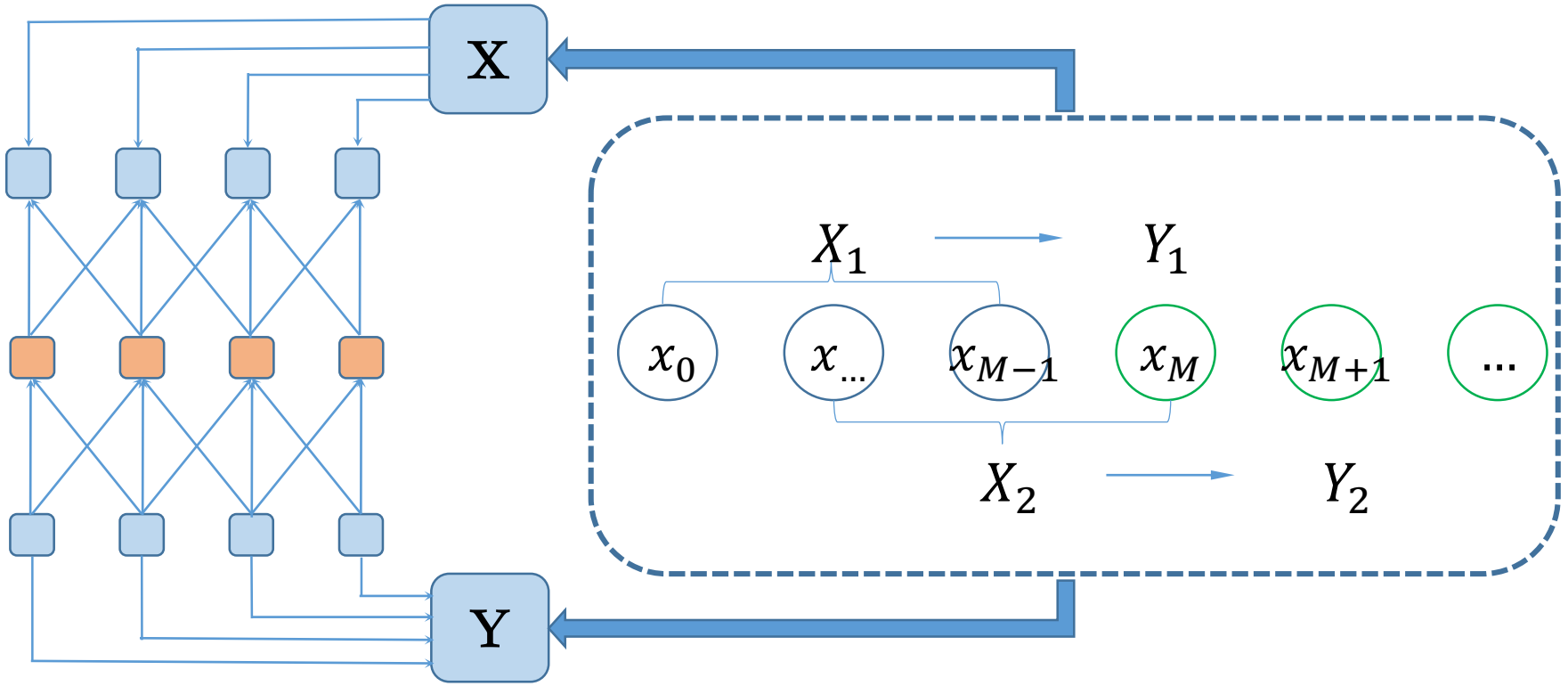


$$\begin{cases} \frac{dx}{dt} = \sigma(y - x) \\ \frac{dy}{dt} = x(\rho - z) - y \\ \frac{dz}{dt} = xy - \beta z \end{cases}$$

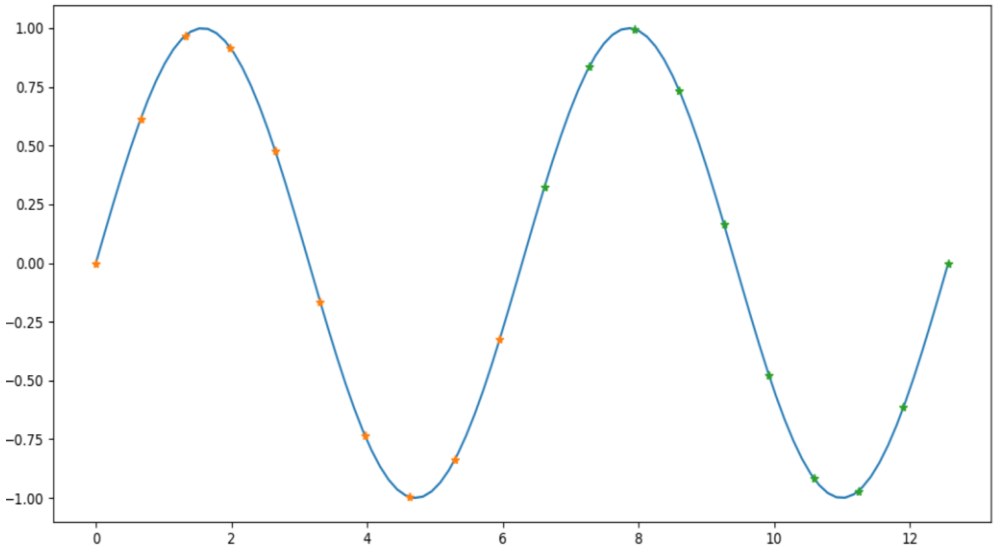




时间序列预测



时间序列预测




```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout
import matplotlib.pyplot as plt
x = np.linspace(0,10*np.pi,100)
dataset = (np.cos(x)+1)/2.
def create_dataset(dataset, look_back=10):
    dataX,dataY = [],[]
    for i in range(len(dataset)-look_back):
        dataX.append(dataset[i:(i+look_back)])
        dataY.append(dataset[i + look_back])
    return np.array(dataX),np.array(dataY)
look_back = 10
train_size = test_size = len(dataset)//2
train, test = dataset[:train_size], dataset[train_size:]
trainX,trainY = create_dataset(train, look_back)
testX,testY = create_dataset(test, look_back)
trainX = np.reshape(trainX, (trainX.shape[0], trainX.shape[1], 1))
testX = np.reshape(testX, (testX.shape[0], testX.shape[1], 1))
model = Sequential()
model.add(LSTM(32, input_shape=(look_back, 1)))
model.add(Dropout(0.2))
model.add(Dense(1))
model.compile(loss = 'mse', optimizer = 'adam')
model.fit(trainX,trainY,batch_size = 128,epochs=50, verbose=2)
yp = model.predict(testX)
plt.plot(np.arange(len(yp)),testY,np.arange(len(yp)),yp)
plt.show()
```

Thank you!

Questions?