Recurrent Neural Networks (RNNs)

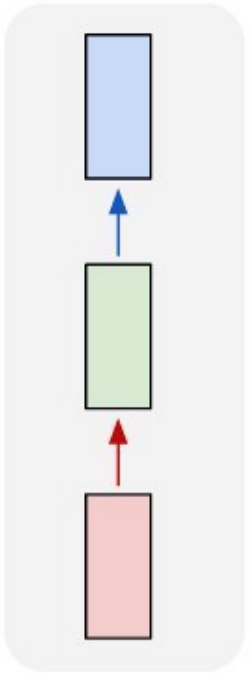
小规模用RNN
大规模用Transforer

Shusen Wang

How to model sequential data?

时序数据的建模

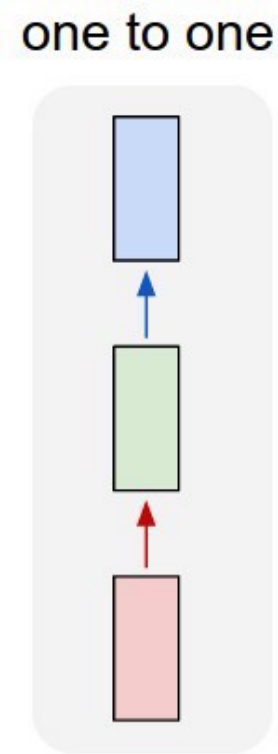
one to one



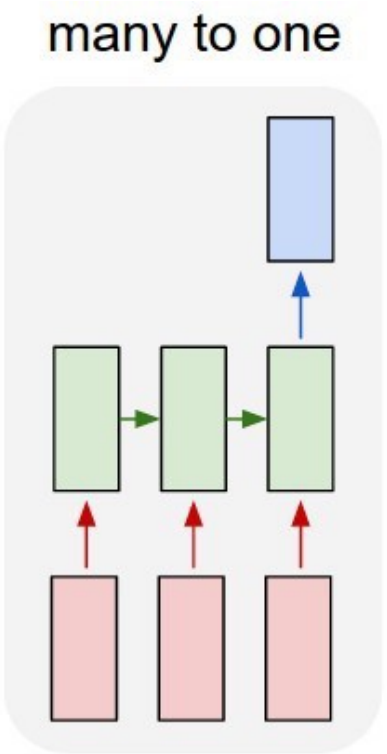
适合图像输入
输入大小 和 输出大小 一致

- Limitations of FC Nets and ConvNets:
  - Process a paragraph as a whole.
  - Fixed-size input (e.g., image).
  - Fixed-size output (e.g., predicted probabilities).

How to model sequential data?



时序数据更适合这种模型

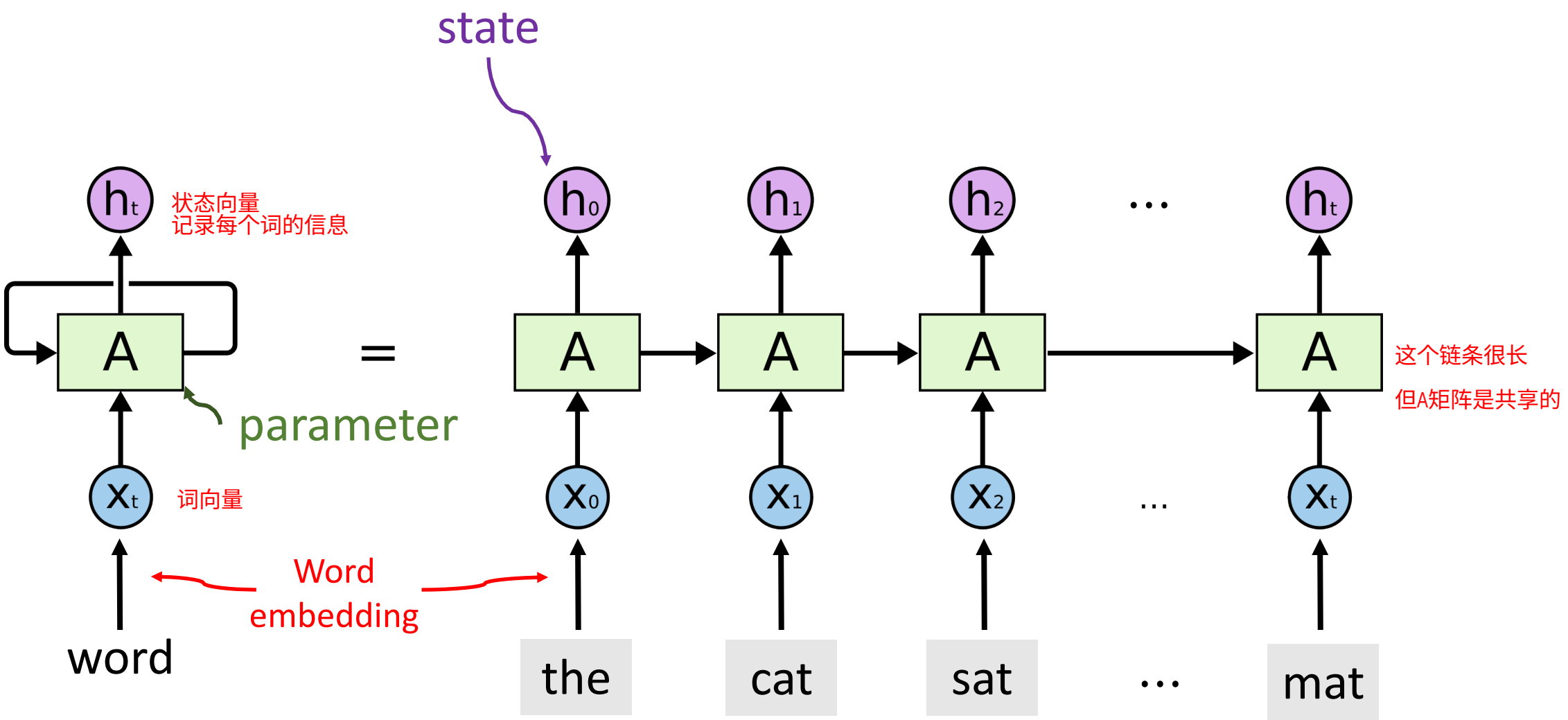


输入和输出的数量都是不固定的

- Limitations of FC Nets and ConvNets:
  - Process a paragraph as a whole.
  - Fixed-size input (e.g., image).
  - Fixed-size output (e.g., predicted probabilities).
- RNNs are better ways to model the sequential data (e.g., text, speech, and time series).

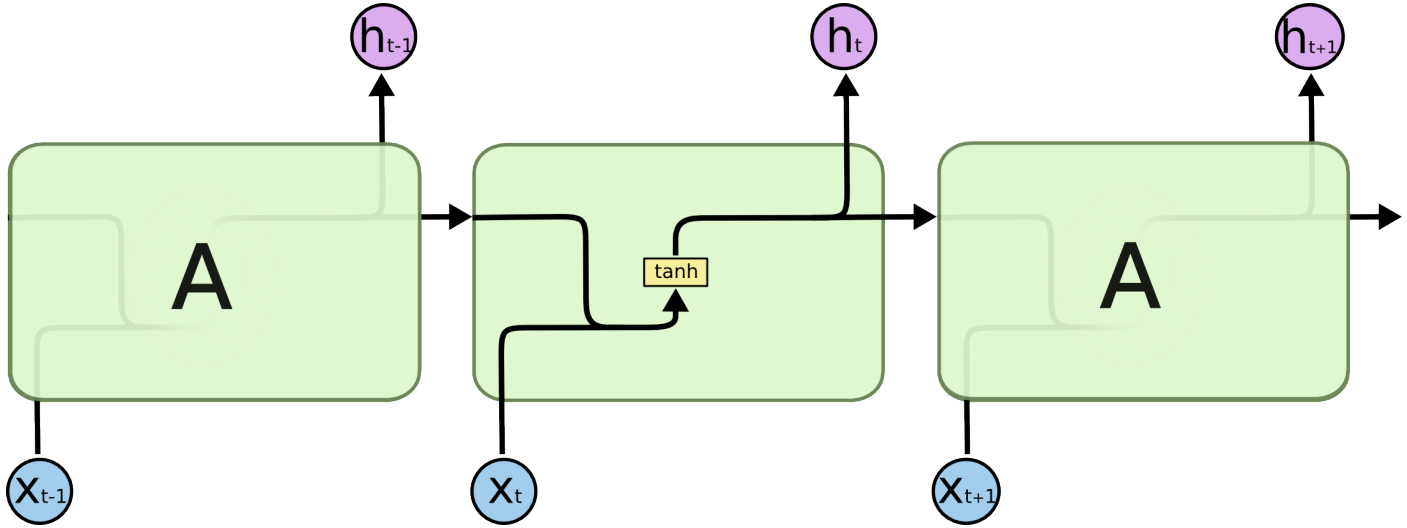
rnn是对序列数据（例如，文本、语音和时间序列）进行建模的更好方法。

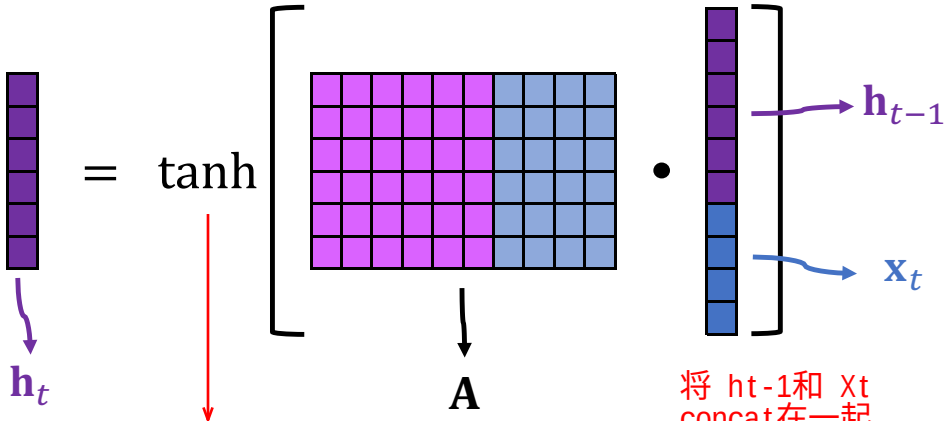
Recurrent Neural Networks (RNNs)

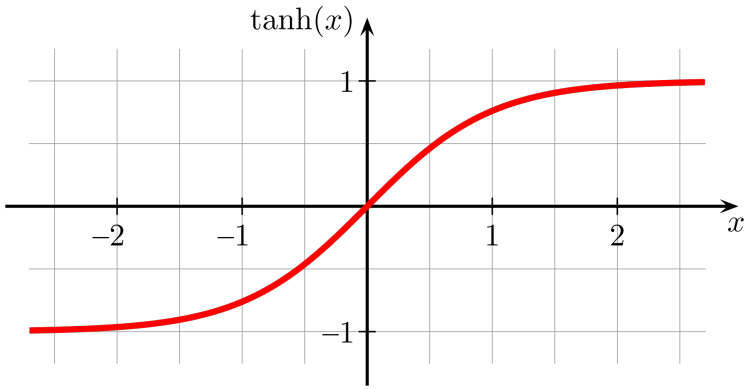


Simple RNN Model

Simple RNN





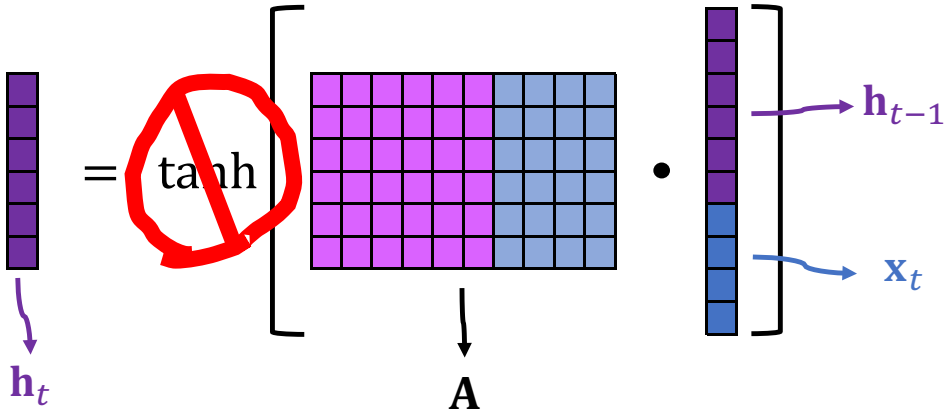


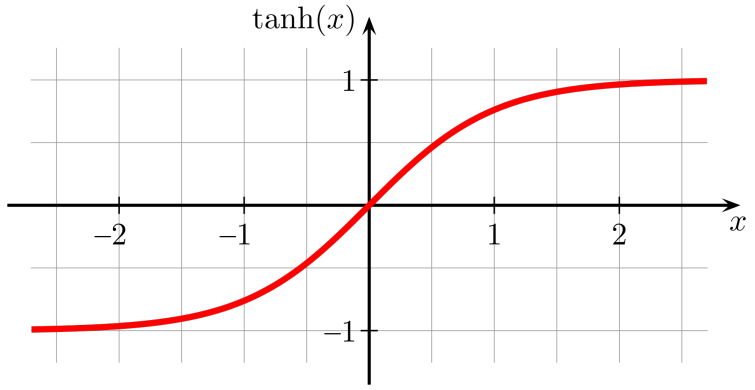
hyperbolic tangent function

双曲正切函数

Simple RNN

Question: Why do we need the tanh function?





hyperbolic tangent function

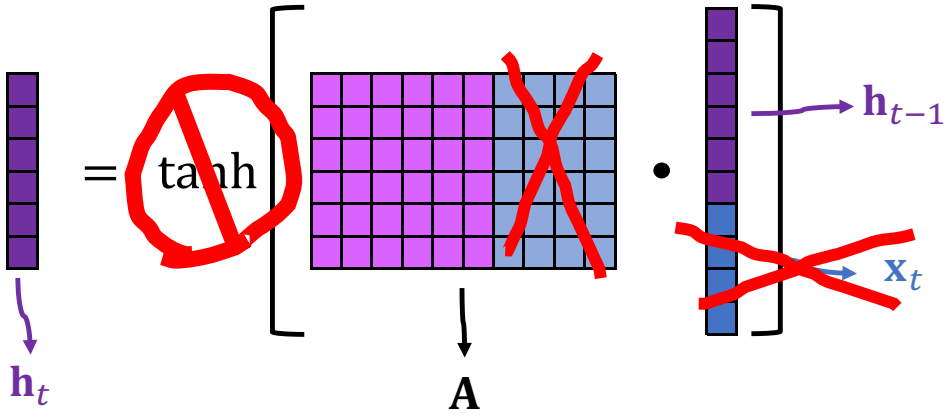
Simple RNN

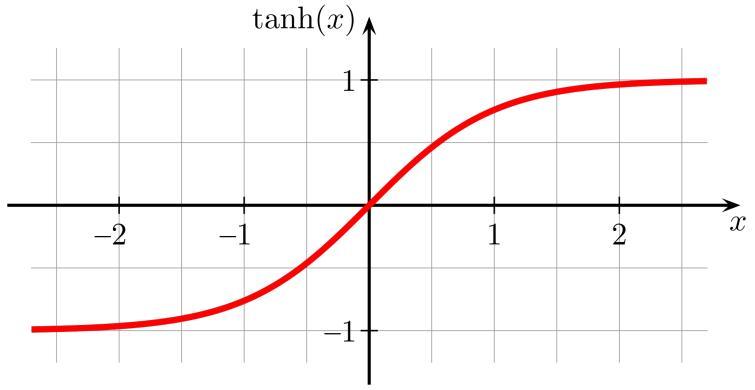
Question: Why do we need the tanh function?

- Suppose $\mathbf{x}_0 = \cdots = \mathbf{x}_{100} = 0$.

为什么要使用双曲正切函数作为激活函数？

假设 所有的输入都是 0





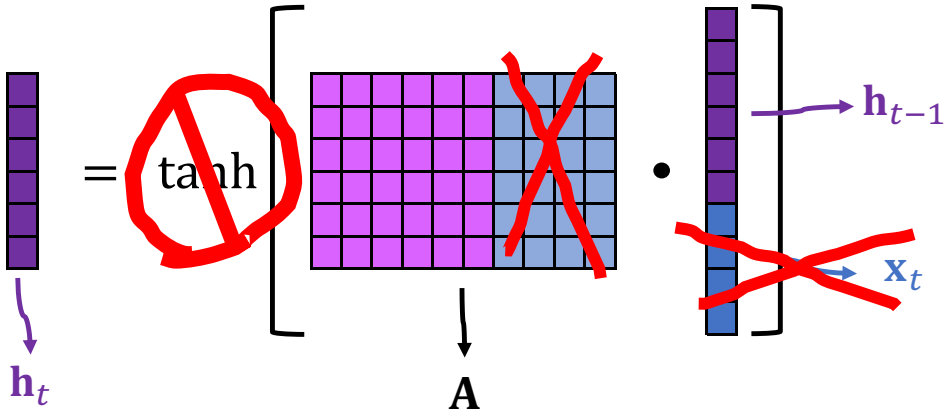
hyperbolic tangent function

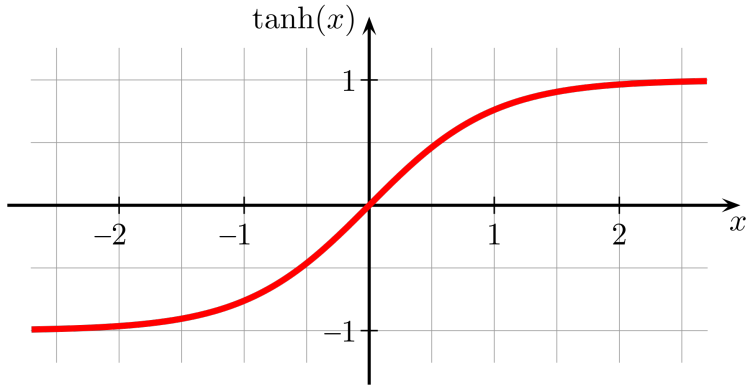
Simple RNN

Question: Why do we need the tanh function?

- Suppose $\mathbf{x}_0 = \cdots = \mathbf{x}_{100} = 0$.
- $\mathbf{h}_{100} = \mathbf{A}\mathbf{h}_{99} = \mathbf{A}^2\mathbf{h}_{98} = \cdots = \mathbf{A}^{100}\mathbf{h}_0$.
- What will happen if $\lambda_{\max}(\mathbf{A}) = 0.9$?
- What will happen if $\lambda_{\max}(\mathbf{A}) = 1.2$?

矩阵的特征值





hyperbolic tangent function

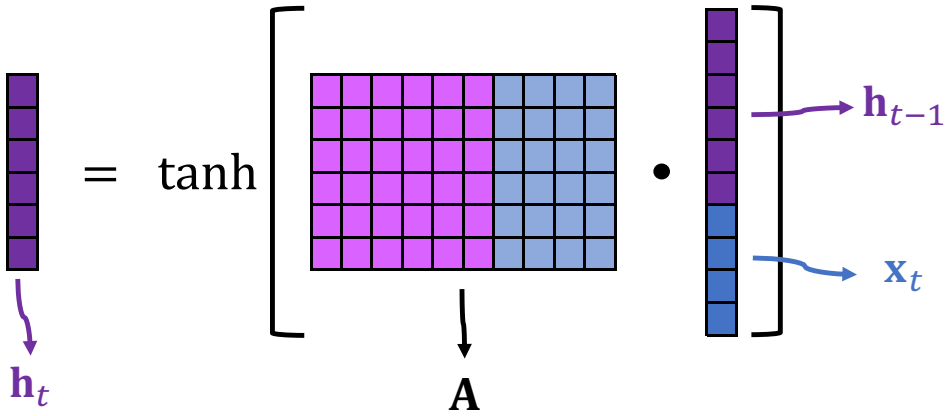
Simple RNN

Trainable parameters: matrix **A**

- #rows of **A**: shape(**h**)
- #cols of **A**: shape(**h**)+shape(**x**)

- Total #parameter: shape(**h**)× [shape(**h**)+shape(**x**)]

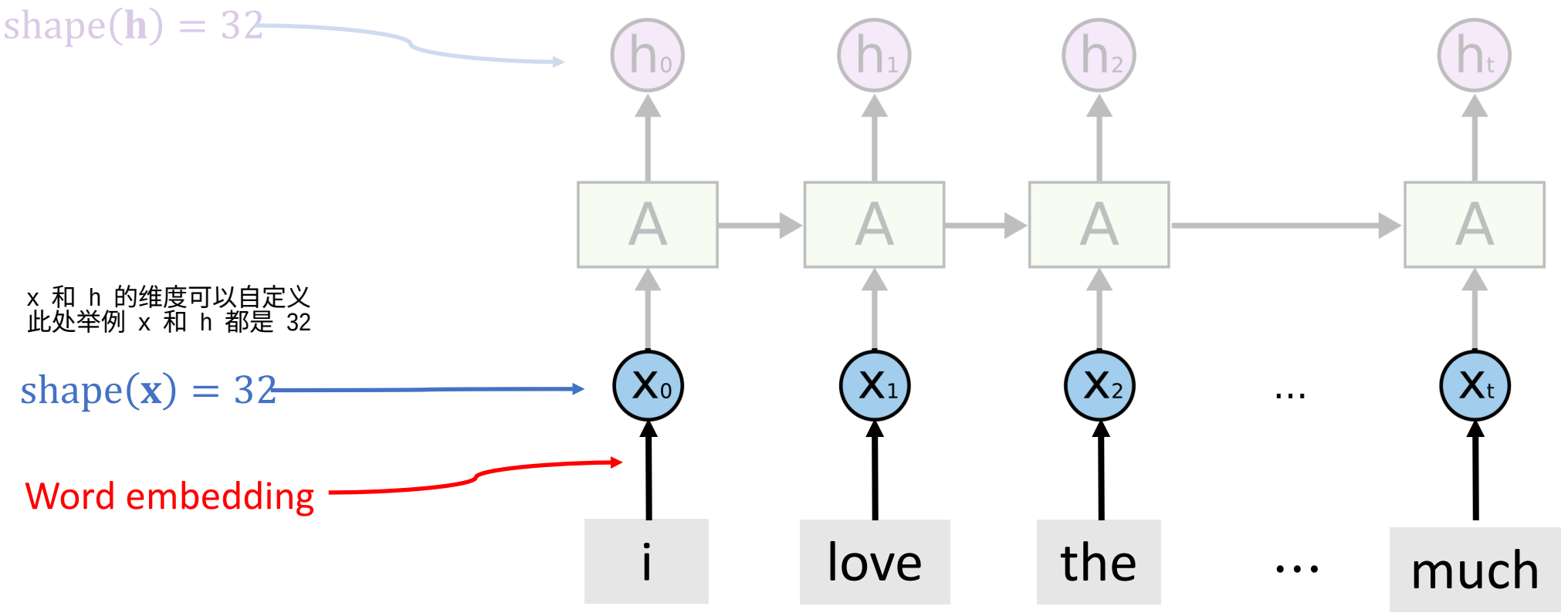
参数量 重要



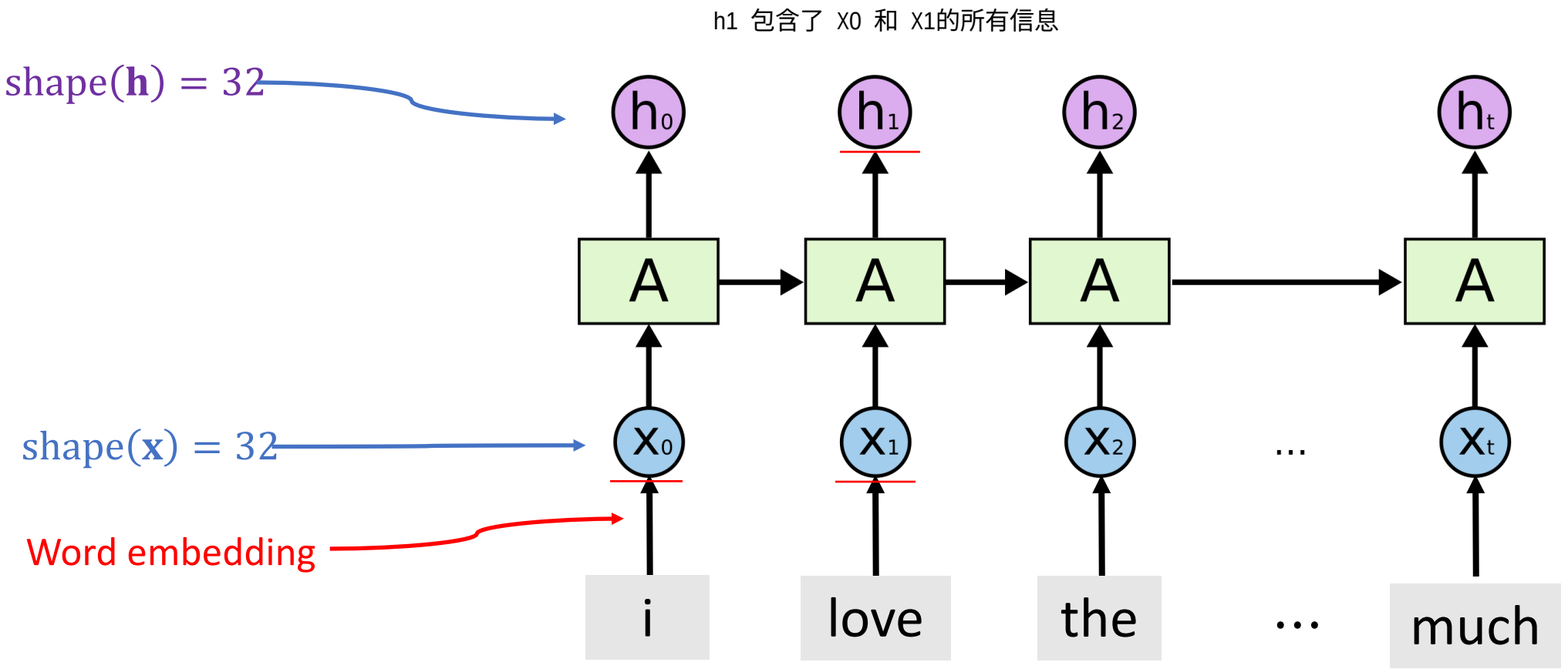
Simple RNN for Movie Review Analysis

电影评论分析 ： 二分类问题

Simple RNN for IMDB Review



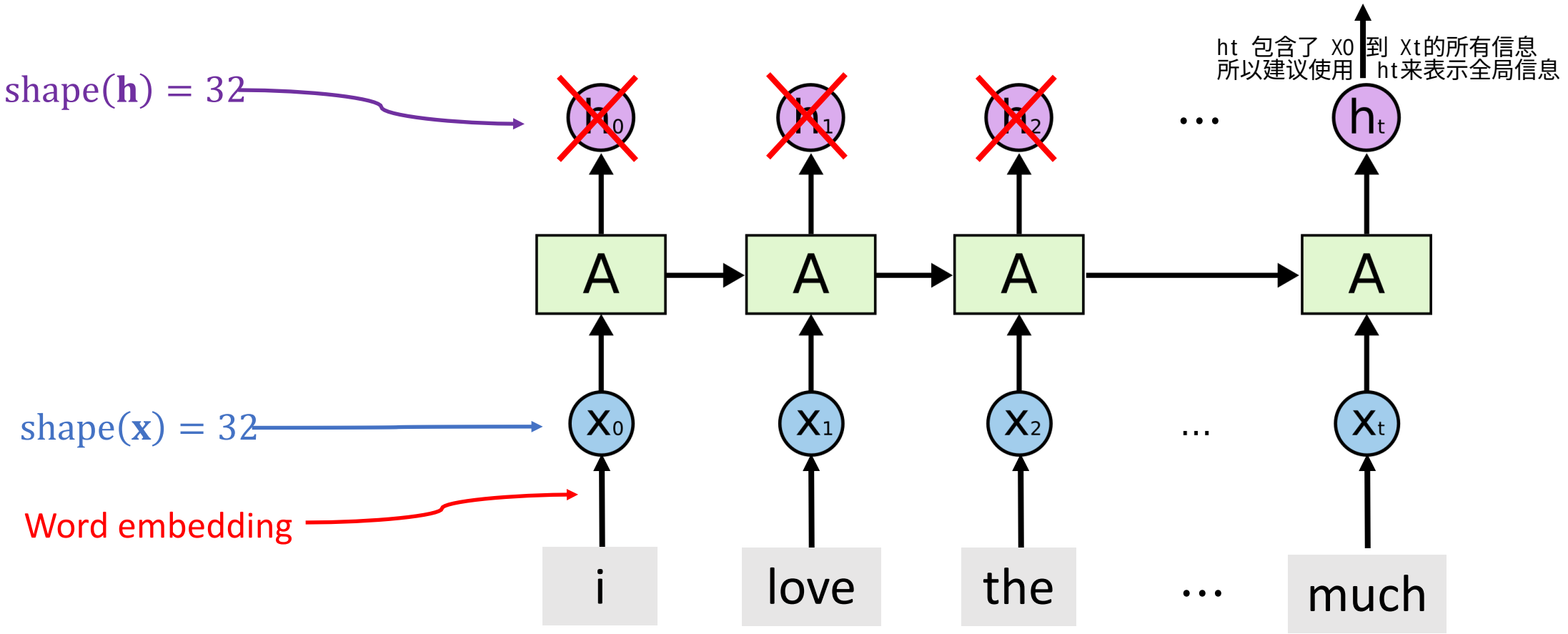
Simple RNN for IMDB Review



Simple RNN for IMDB Review

电影评论分析 ： 二分类问题

$$sigmoid(\mathbf{v}^T\ \mathbf{h}_t)$$



Simple RNN for IMDB Review

```
from keras.models import Sequential
from keras.layers import SimpleRNN, Embedding, Dense

vocabulary = 10000    #unique words in the dictionary
embedding_dim = 32    shape(x) = 32
word_num = 500        sequence length
state_dim = 32        shape(h) = 32

model = Sequential()
model.add(Embedding(vocabulary, embedding_dim, input_length=word_num))
model.add(SimpleRNN(state_dim, return_sequences=False))
model.add(Dense(1, activation='sigmoid'))

model.summary()
```

Only return the last state h_t.

Simple RNN for IMDB Review

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 500, 32)	320000
simple_rnn_1 (SimpleRNN)	(None, 32)	2080
dense_1 (Dense)	(None, 1)	33

Total params: 322,113
Trainable params: 322,113
Non-trainable params: 0

#parameters in RNN:

$$2080 = 32 \times (32 + 32) + 32$$

偏移量

$\text{shape}(\mathbf{h}) = 32$

$\text{shape}(\mathbf{x})$

Simple RNN for IMDB Review

```
from keras import optimizers

epochs = 3

model.compile(optimizer=optimizers.RMSprop(lr=0.001),
              loss='binary_crossentropy', metrics=['acc'])
history = model.fit(x_train, y_train, epochs=epochs,
                    batch_size=32, validation_data=(x_valid, y_valid))
```

提前停止，缓解了过拟合

Early stopping alleviates overfitting

```
Train on 20000 samples, validate on 5000 samples
Epoch 1/3
20000/20000 [==============================] - 65s 3ms/step - loss: 0.5514 - acc: 0.6959 - val_loss: 0.4095 - val_acc: 0.8176
Epoch 2/3
20000/20000 [==============================] - 66s 3ms/step - loss: 0.3336 - acc: 0.8620 - val_loss: 0.3296 - val_acc: 0.8658
Epoch 3/3
20000/20000 [==============================] - 65s 3ms/step - loss: 0.2774 - acc: 0.8918 - val_loss: 0.3569 - val_acc: 0.8428
```

Simple RNN for IMDB Review

```
loss_and_acc = model.evaluate(x_test, labels_test)
print('loss = ' + str(loss_and_acc[0]))
print('acc = ' + str(loss_and_acc[1]))
```

```
25000/25000 [==============================] - 21s 829us/step
loss = 0.36507524153709414
acc = 0.84364
```

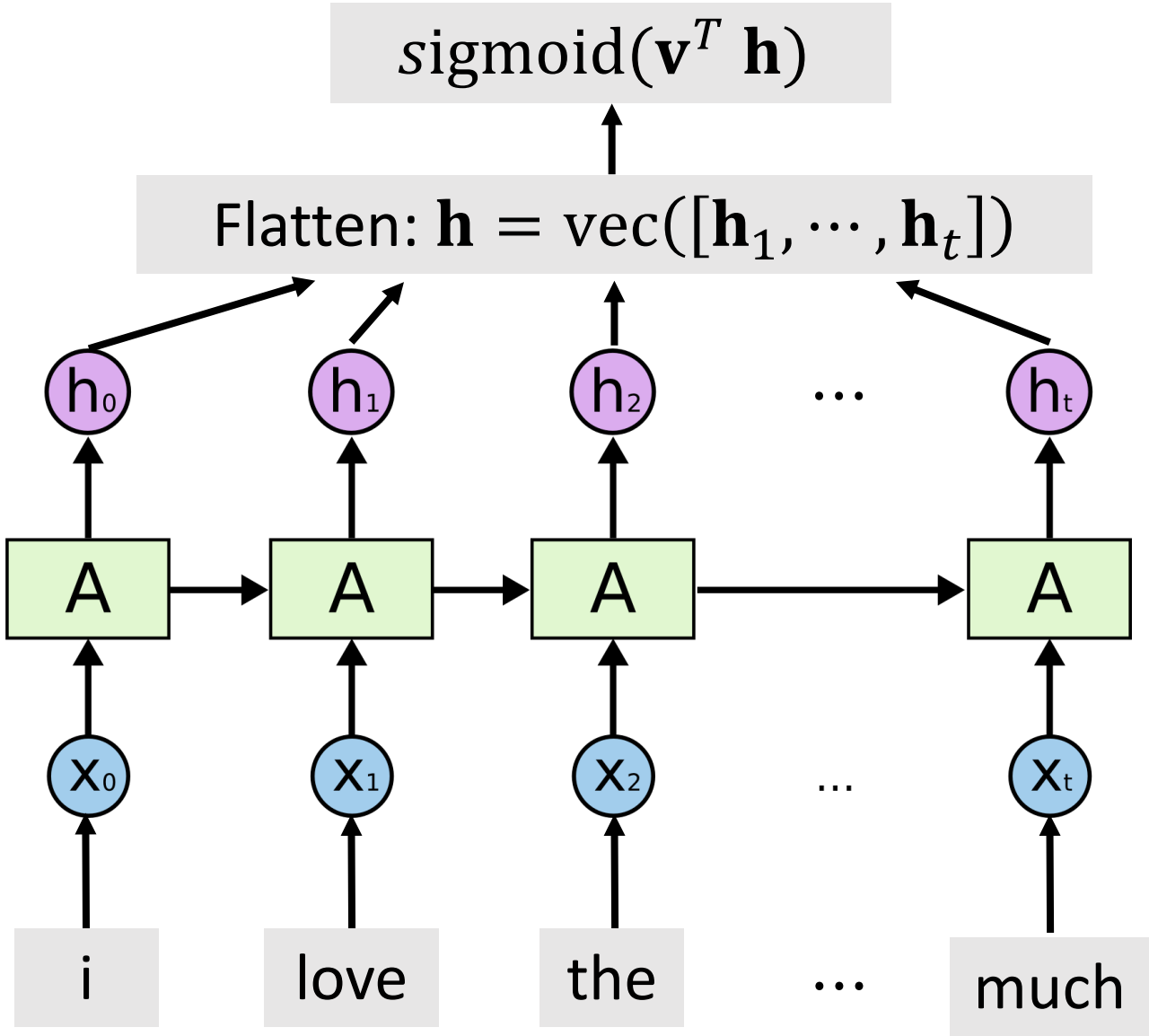
Higher than a naïve shallow model (whose test accuracy is about 75%).

Simple RNN for IMDB Review

- Training Accuracy: 89.2%
- Validation Accuracy: 84.3%
- Test Accuracy: 84.4%

Higher than a naïve shallow model (whose test accuracy is about 75%).

Simple RNN for IMDB Review



Simple RNN for IMDB Review

```
from keras.models import Sequential
from keras.layers import SimpleRNN, Embedding, Dense

vocabulary = 10000
embedding_dim = 32
word_num = 500
state_dim = 32

model = Sequential()
model.add(Embedding(vocabulary, embedding_dim, input_length=word_num))
model.add(SimpleRNN(state_dim, return_sequences=True))
model.add(Flatten())
model.add(Dense(1, activation='sigmoid'))

model.summary()
```

Return all the states $h_1, \cdots, h_t$.

Simple RNN for IMDB Review

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 500, 32)           320000
_________________________________________________________________
simple_rnn_2 (SimpleRNN)     (None, 500, 32)           2080
_________________________________________________________________
flatten_2 (Flatten)          (None, 16000)             0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 16001
=================================================================
Total params: 338,081
Trainable params: 338,081
Non-trainable params: 0
```

Simple RNN for IMDB Review

- Training Accuracy: 96.3%
- Validation Accuracy: 85.4%
- Test Accuracy: 84.7%

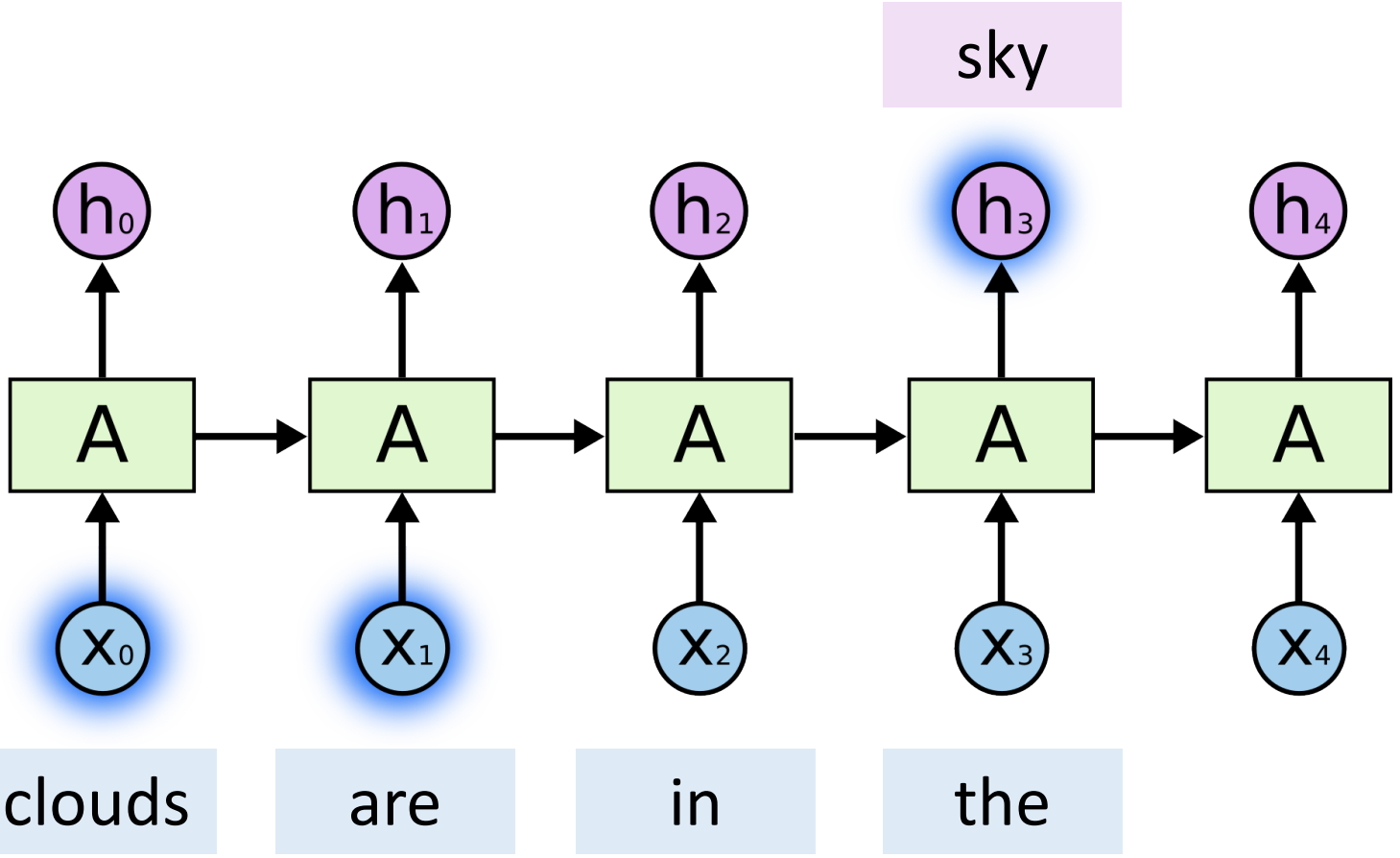
Not really better than using only the final state (whose accuracy is 84.4%).

Shortcomings of SimpleRNN

SimpleRNN的缺点

SimpleRNN is good at short-term dependence.

Predicted next words:



Input text:

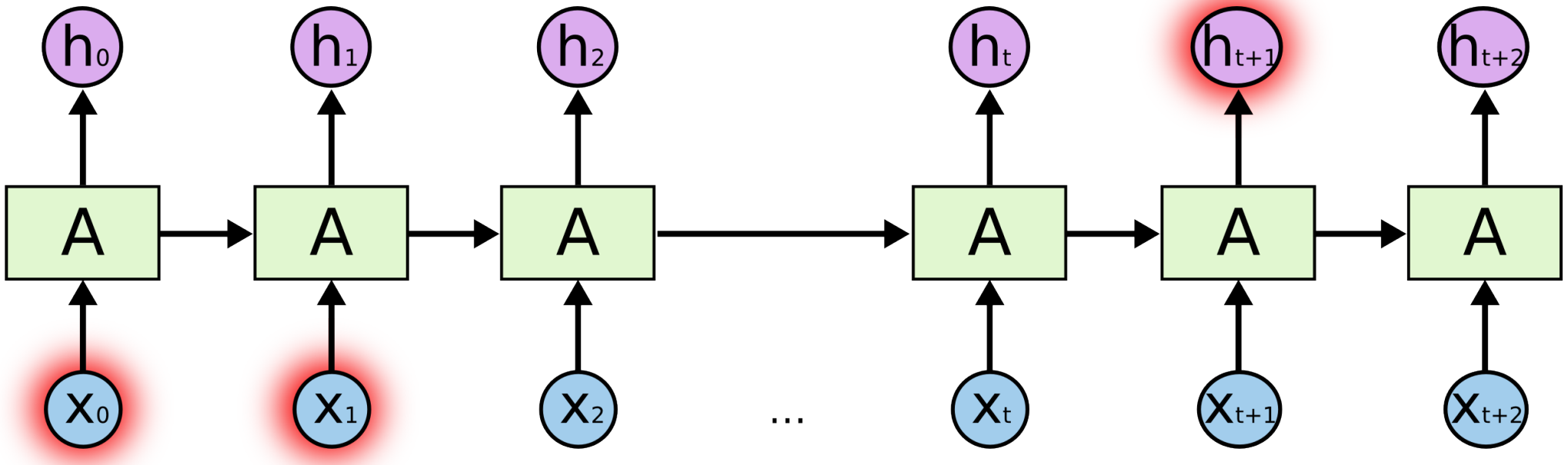
clouds | are | in | the

看近期的词 就可以预测 后一个词，无需全文

Figures are from Christopher Olah's blog: Understanding LSTM Networks.

SimpleRNN is bad at long-term dependence.

simpleRNN 的缺点是 不适合解决 较长的依赖关系



$\mathbf{h}_{100}$ is almost irrelevant to $\mathbf{x}_1$: $\frac{\partial \mathbf{h}_{100}}{\partial \mathbf{x}_1}$ is near zero.

h100 和 X1 的 相关性 约等于 0

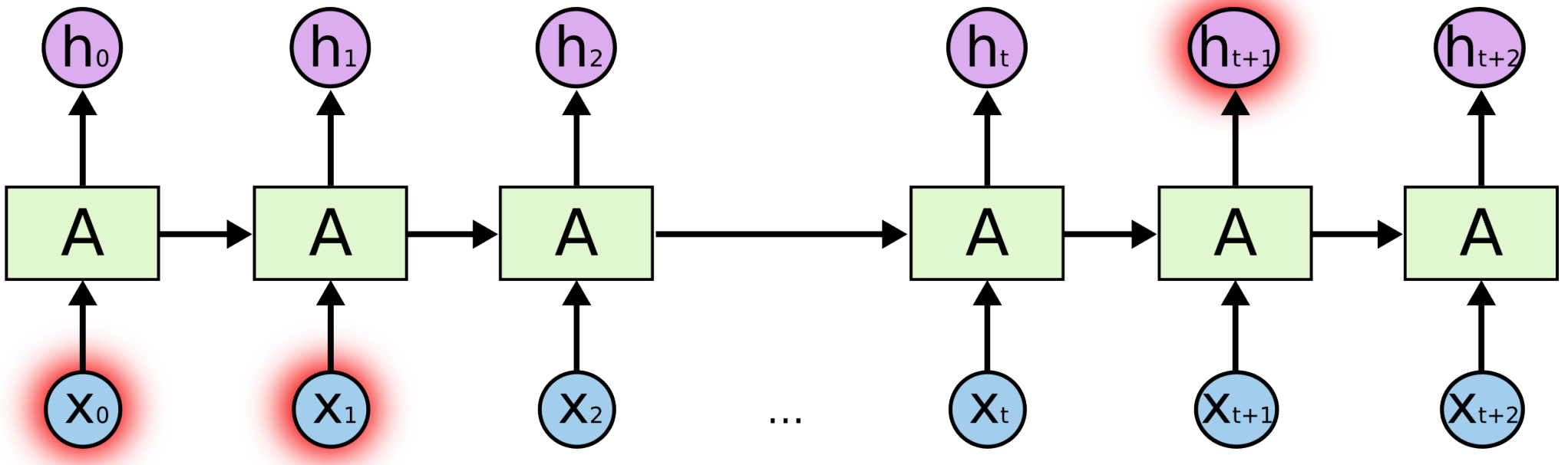
SimPleRNN 有遗忘性（梯度消失）

Figures are from Christopher Olah's blog: Understanding LSTM Networks.

SimpleRNN is bad at long-term dependence.

Predicted next words:

Chinese



Input text:

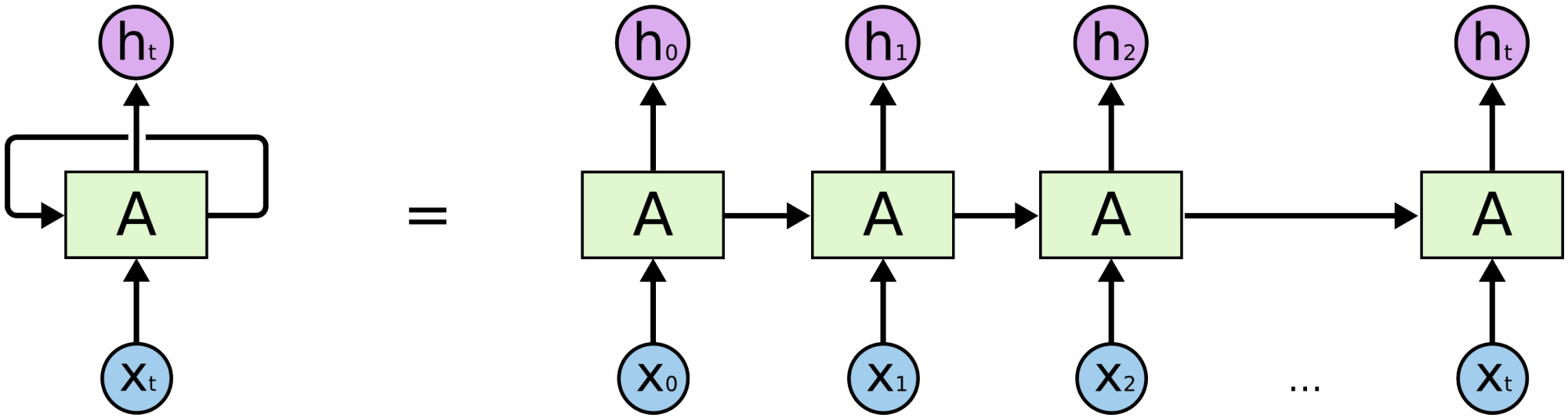
in | China | speak | fluent

Figures are from Christopher Olah's blog: Understanding LSTM Networks.

Summary

Summary

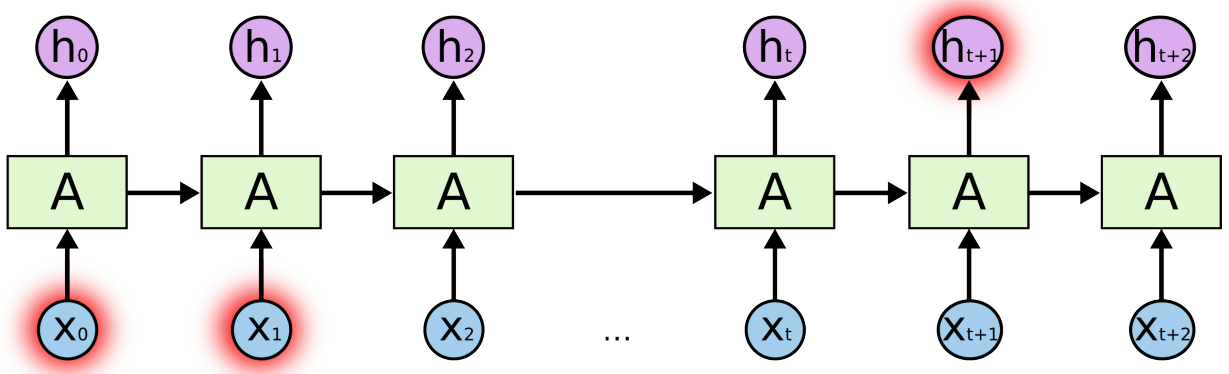
- RNN for text, speech, and time series data.
- Hidden state $\mathbf{h}_t$ aggregates information in the inputs $\mathbf{x}_0, \cdots, \mathbf{x}_t$.



Summary

- RNN for text, speech, and time series data.
- Hidden state $\mathbf{h}_t$ aggregates information in the inputs $\mathbf{x}_0, \cdots, \mathbf{x}_t$.
- RNNs can forget early inputs.
  - It forgets what it has seen early on.
  - If t is large, $\mathbf{h}_t$ is almost irrelevant to $\mathbf{x}_0$.

RNN 是短记忆的



Number of Parameters

- SimpleRNN has a parameter matrix (and perhaps an intercept vector).
- Shape of the parameter matrix is

$$\texttt{shape(h)} \times \texttt{[shape(h)+shape(x)]}.$$

Number of Parameters

- SimpleRNN has a parameter matrix (and perhaps an intercept vector).
- Shape of the parameter matrix is

$$\texttt{shape}(\mathbf{h}) \times [\texttt{shape}(\mathbf{h}) + \texttt{shape}(\mathbf{x})].$$

- Only one such parameter matrix, no matter how long the sequence is.

Thank You!