深度學習基礎概論

0401

目錄

- Embedding
- RNN and LSTM
- Code : 標題產生器

Embedding

將單詞轉換為數字

「A dog barked at a cat.
(有一隻狗對著一隻貓吠叫。)」

字典

1. A (一個)
2. AN (一個)
3. AND (和)
4. AT (對著)
5. ATE (吃)
6. BARK (吠叫)
7. BARKED (吠叫)
8. CAT (貓)
9. CATS (貓)
10. DOG (狗)
11. DOGS (狗)
12. EAT (吃)

One-hot Encoding

「A dog barked at a cat.
(有一隻狗對著一隻貓吠叫。)」

a	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
dog	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
barked	[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
at	[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
a	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
cat	[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]

字典

1. A (一個)
2. AN (一個)
3. AND (和)
4. AT (對著)
5. ATE (吃)
6. BARK (吠叫)
7. BARKED (吠叫)
8. CAT (貓)
9. CATS (貓)
10. DOG (狗)
11. DOGS (狗)
12. EAT (吃)

one hot encoding 的問題

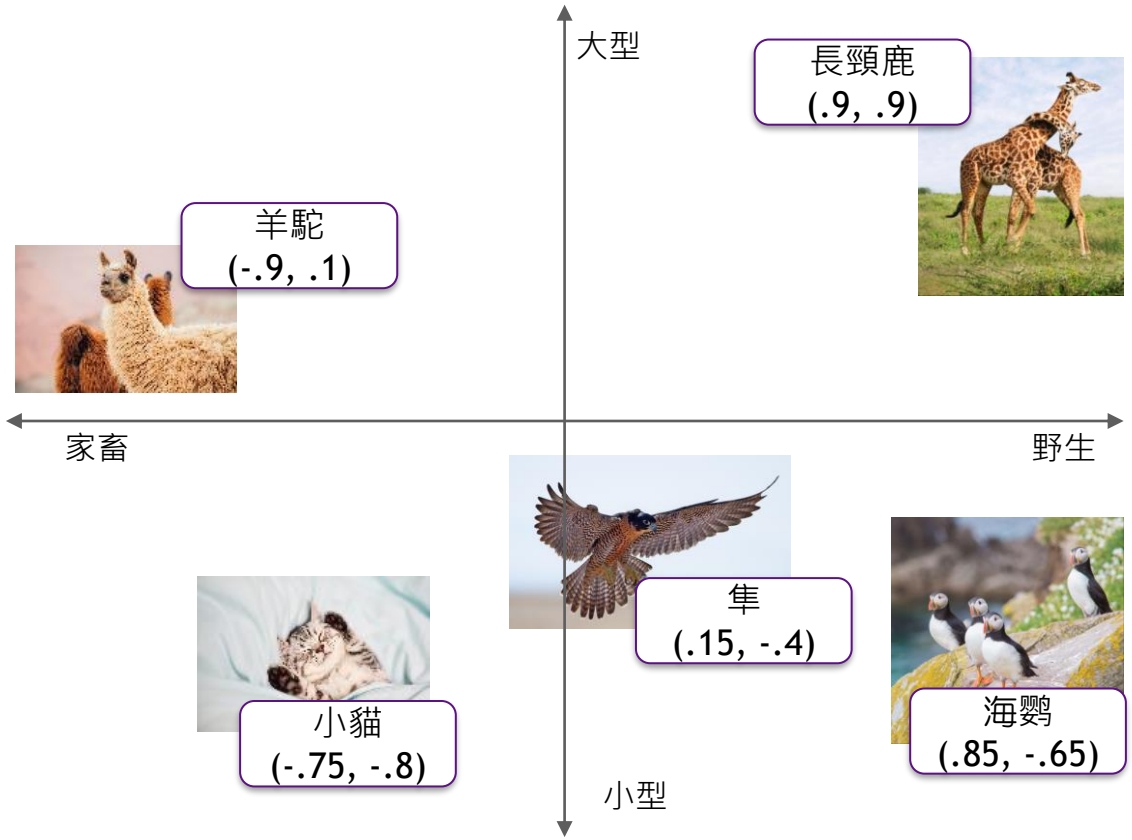
- One hot vector 維度會隨著字典大小的增加而增加
- 無法表達字跟詞之間的關係，也就是說 vector 的資訊密度很低，兩者內積為 0

Word Embedding

- 希望將原本資訊密度低、維度高的向量，改成資訊密度高、維度低的向量
- 這個低維度的向量有個特性，當詞與詞的意思越接近，在向量空間中也會越接近，兩者的夾角也會越小
- 每個維度可以代表一個特徵

Word Embedding

以二維的 word embedding 為例



$$[1,0,0,0,0]_{1\times5} \times \underbrace{\begin{bmatrix} & \\ & \\ & \\ & \\ & \end{bmatrix}}_{5\times2} = \underbrace{[x, y]}_{1\times2}$$

裡面的 weights 也是可以 train 的

RNN model

RNN 感覺很像 time series 中的 AR model，這期跟前一期存在 autoregressive，以 AR(1) model 為例

$$r_t = \phi_0 + \phi_1 r_{t-1} + a_t$$

$$r_{t-1} = \phi_0 + \phi_1 r_{t-2} + a_{t-1}$$

$$r_t - \mu = \sum_{i=0} \phi_1^i a_{t-1}$$

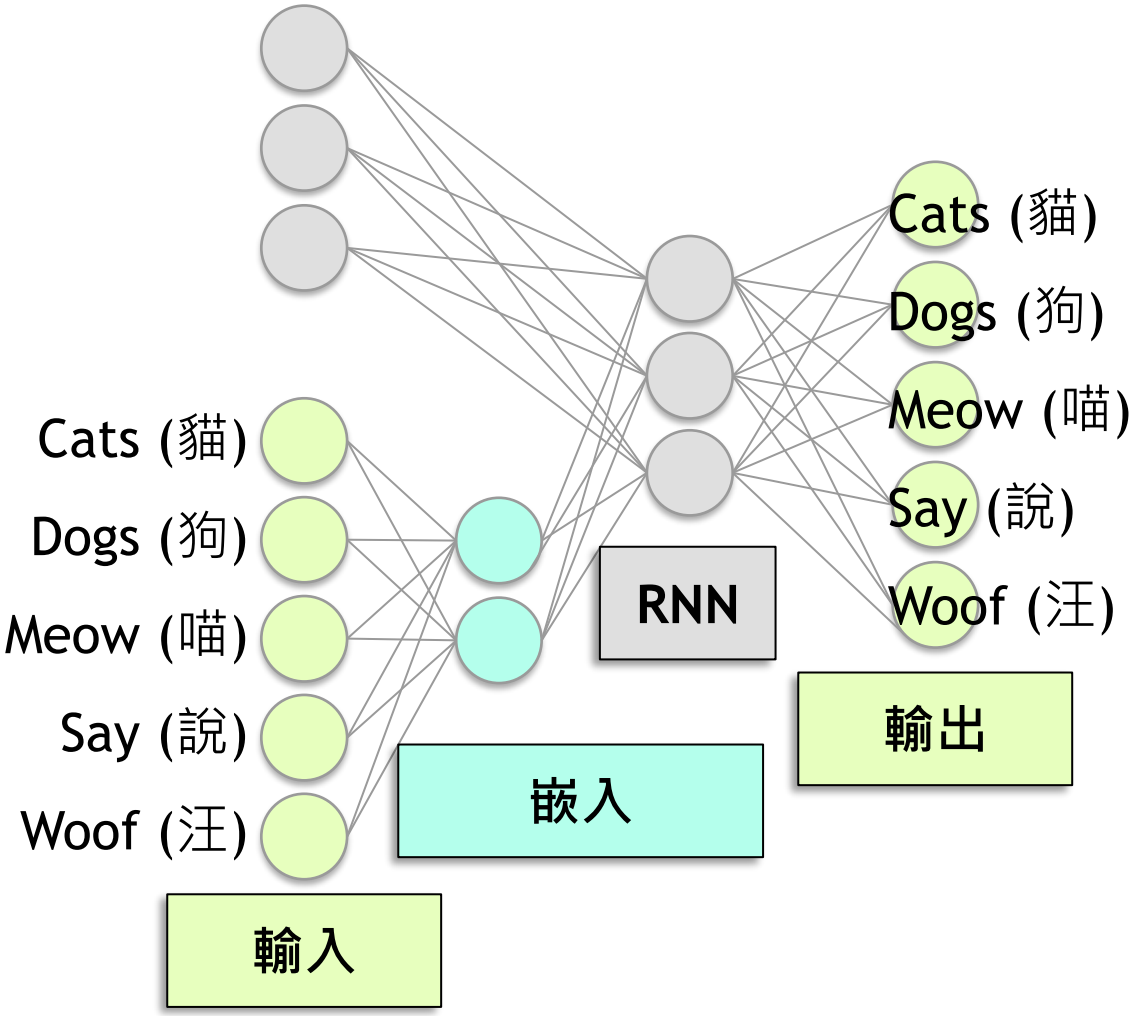
RNN

- 我們嘗試去預測句子中第三個單字，但這些句子的低一個單字會對預測結果造成重大的影響
- 我們希望在預測第三個字時也要考慮到前面兩個字

「Cats say ____.
(貓說 ____。)」

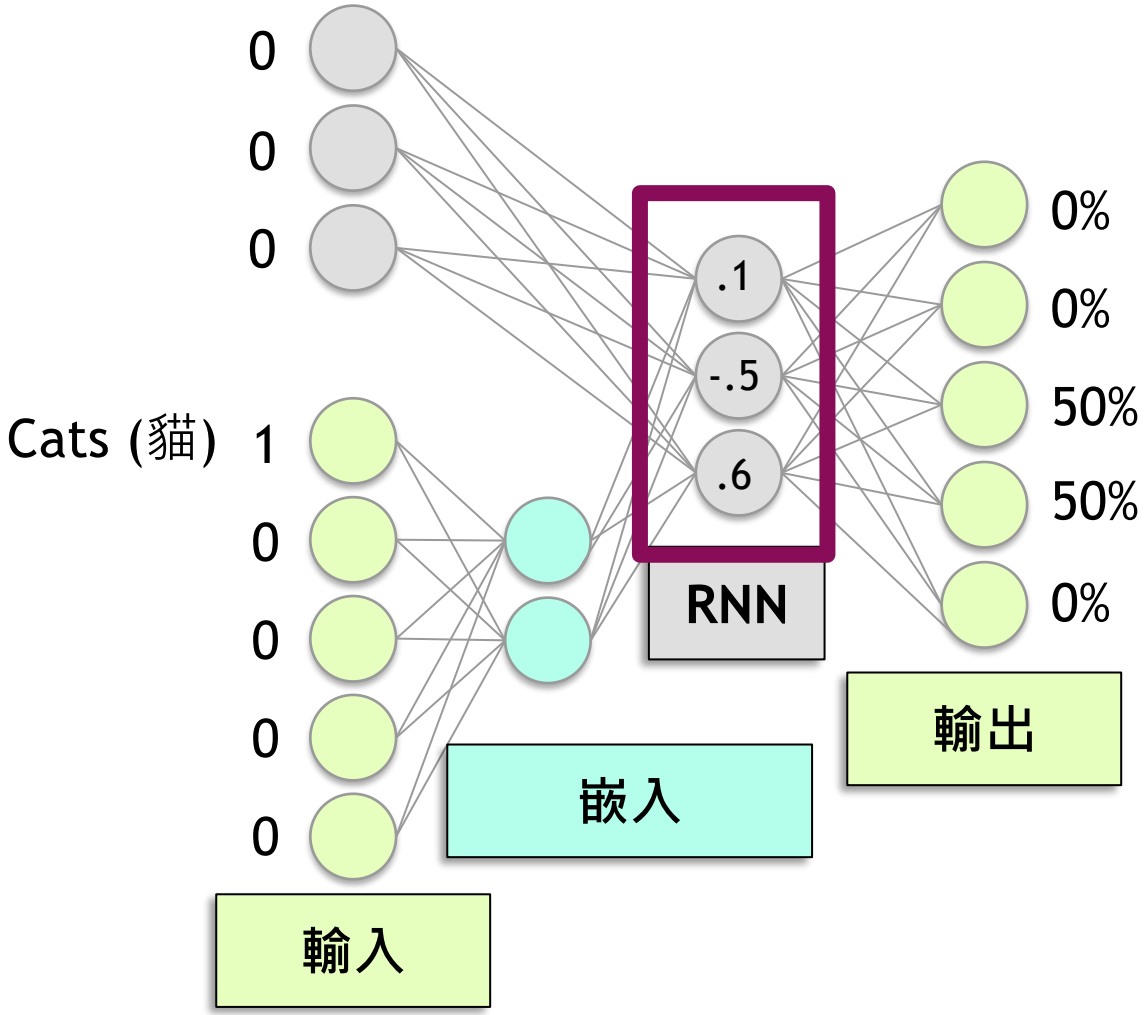
「Dogs say ____.
(狗說 ____。)」

RNN



「Cats say ____.
(貓說 ____。)」

RNN

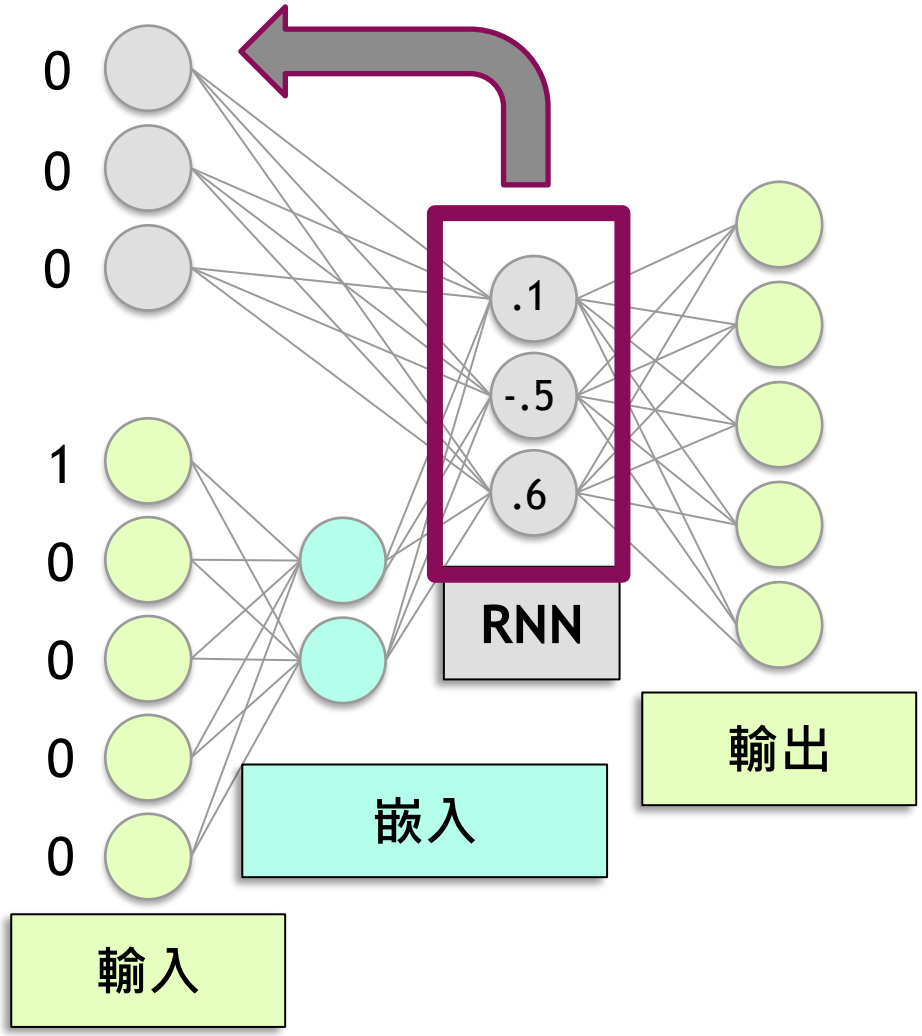


「Cats say ____.
(貓說 ____。)」

字典

1. CATS (貓)
2. DOGS (狗)
3. MEOW (喵)
4. SAY (說)
5. WOOF (汪)

RNN

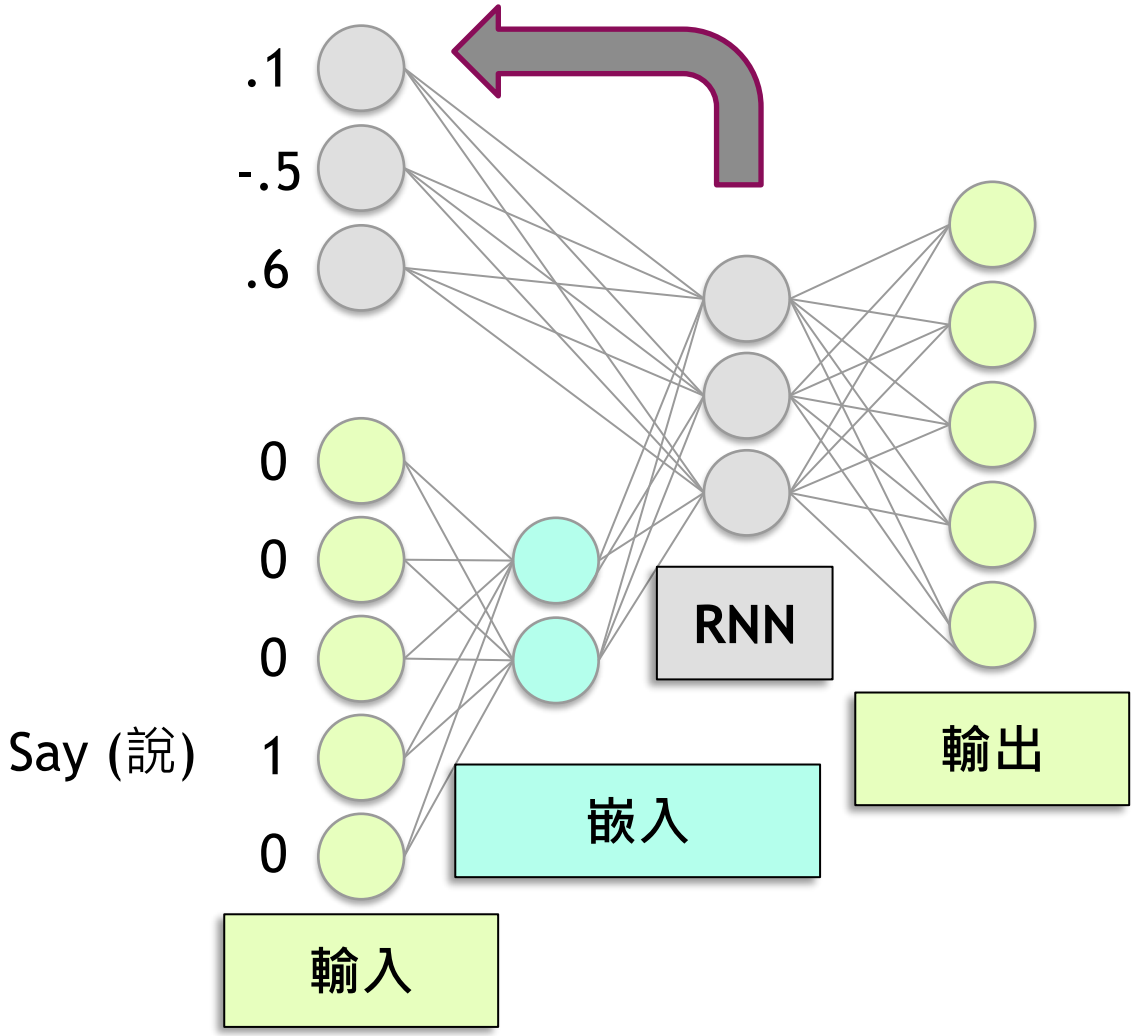


「Cats say ____.
(貓說 ____。)」

字典

1. CATS (貓)
2. DOGS (狗)
3. MEOW (喵)
4. SAY (說)
5. WOOF (汪)

遞歸神經網路

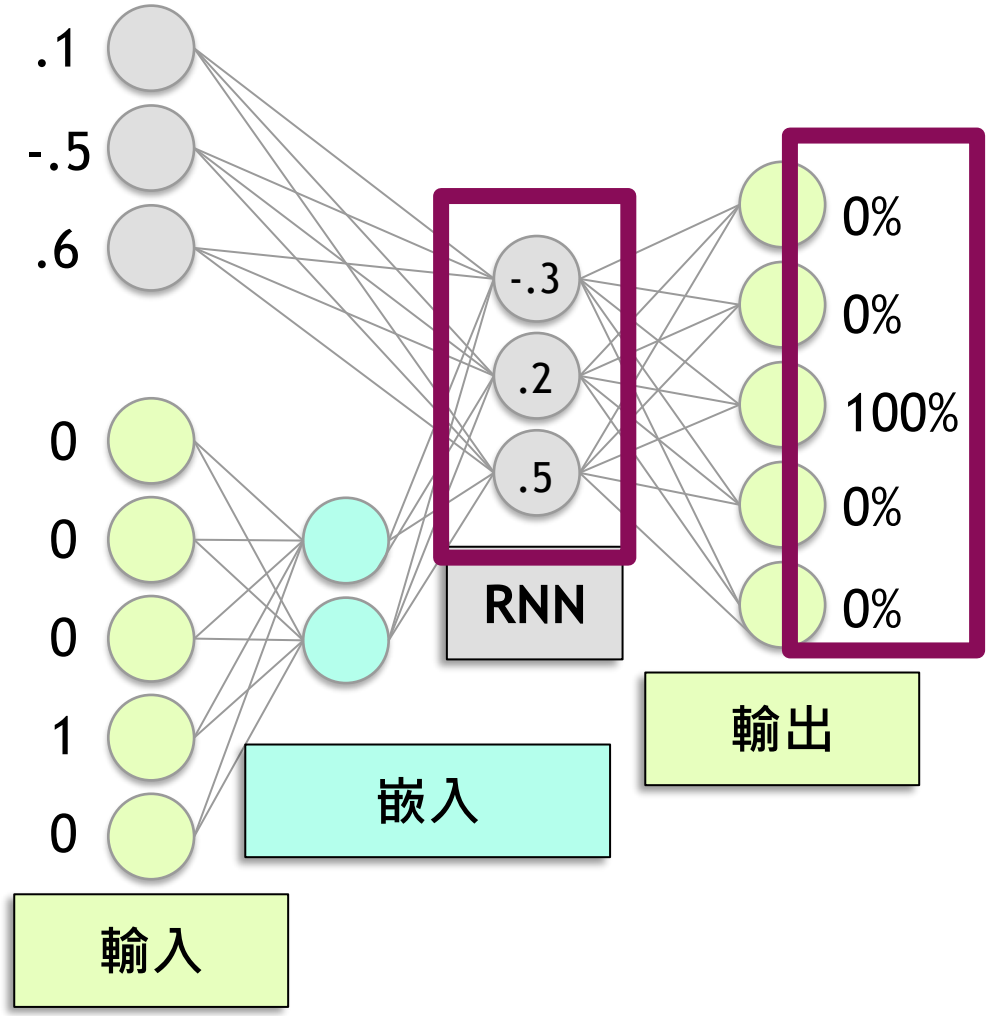


「Cats say ____.
(貓說 ____。)」

字典

1. CATS (貓)
2. DOGS (狗)
3. MEOW (喵)
4. SAY (說)
5. WOOF (汪)

RNN

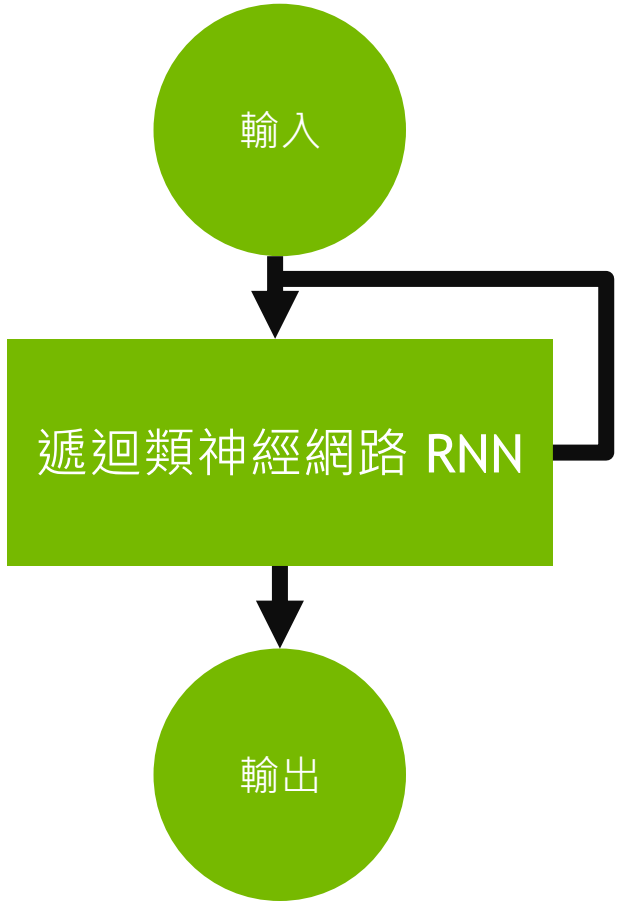


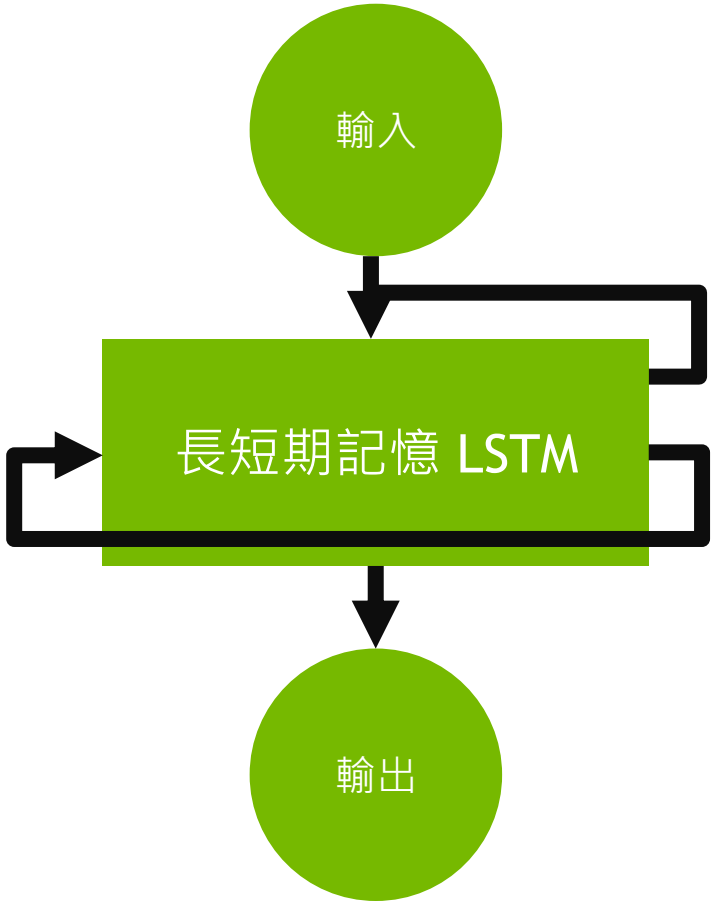
「Cats say ____.
(貓說 ____。)」

字典

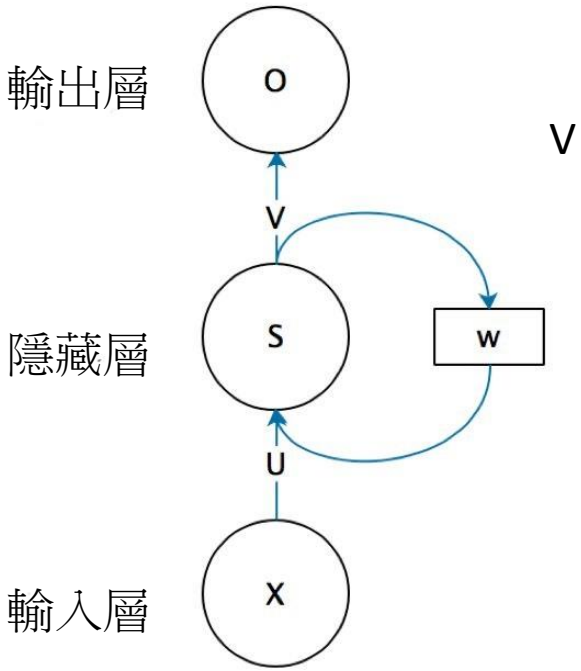
1. CATS (貓)
2. DOGS (狗)
3. MEOW (喵)
4. SAY (說)
5. WOOF (汪)

RNN

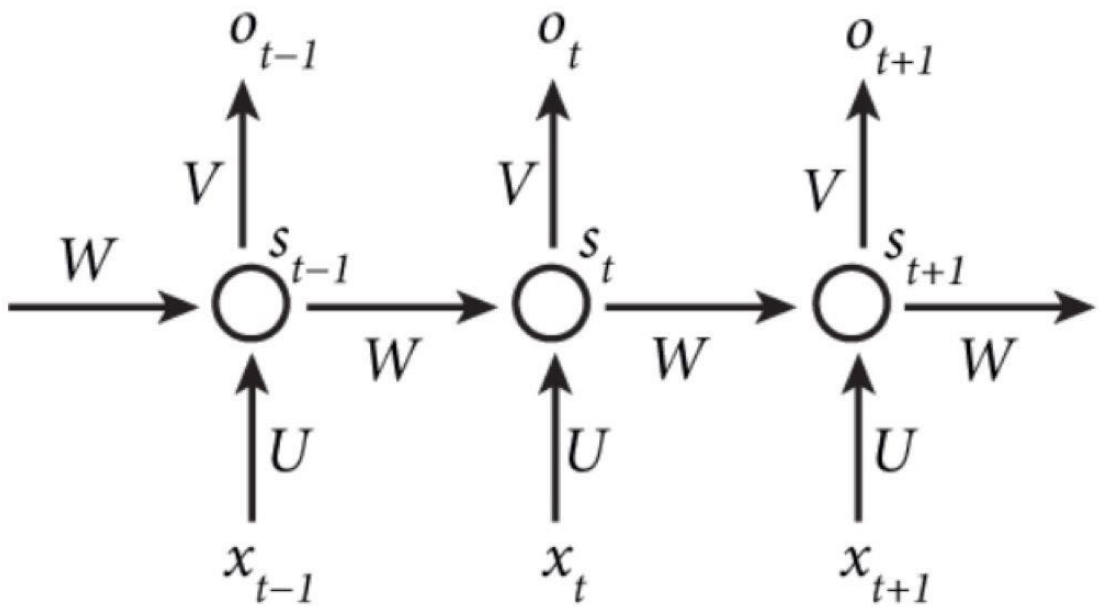




RNN



V、W、U 都是權重



LSTM

LSTM 是 RNN 的改良，為了解決梯度爆炸和梯度消失的問題

vanishing gradient and exploding gradient

f 是 activation function

$$f_{i+1} = f(f_i * w_{i+1} + b_{i+1})$$

$$\Delta w_2 = \frac{\partial Loss}{\partial w_2} = \frac{\partial Loss}{\partial f_4}\frac{\partial f_4}{\partial f_3}\frac{\partial f_3}{\partial f_2}\frac{\partial f_2}{\partial w_2} \qquad \frac{\partial f_2}{\partial w_2} = \frac{\partial f}{\partial (f_1 * w_2)} f_1$$

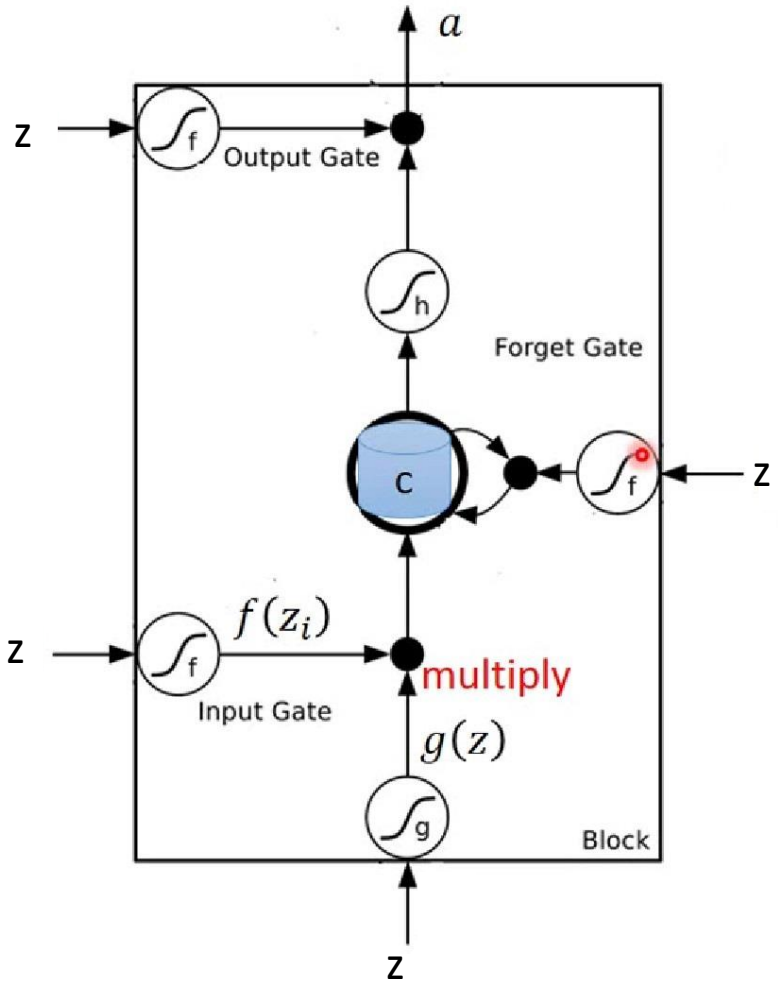
也就是當$\frac{\partial f_4}{\partial f_3} * w_4 > 1$，當層數很多時會發生 exploding gradient

也就是當$\frac{\partial f_4}{\partial f_3} * w_4 < 1$，當層數很多時會發生vanishing gradient

LSTM

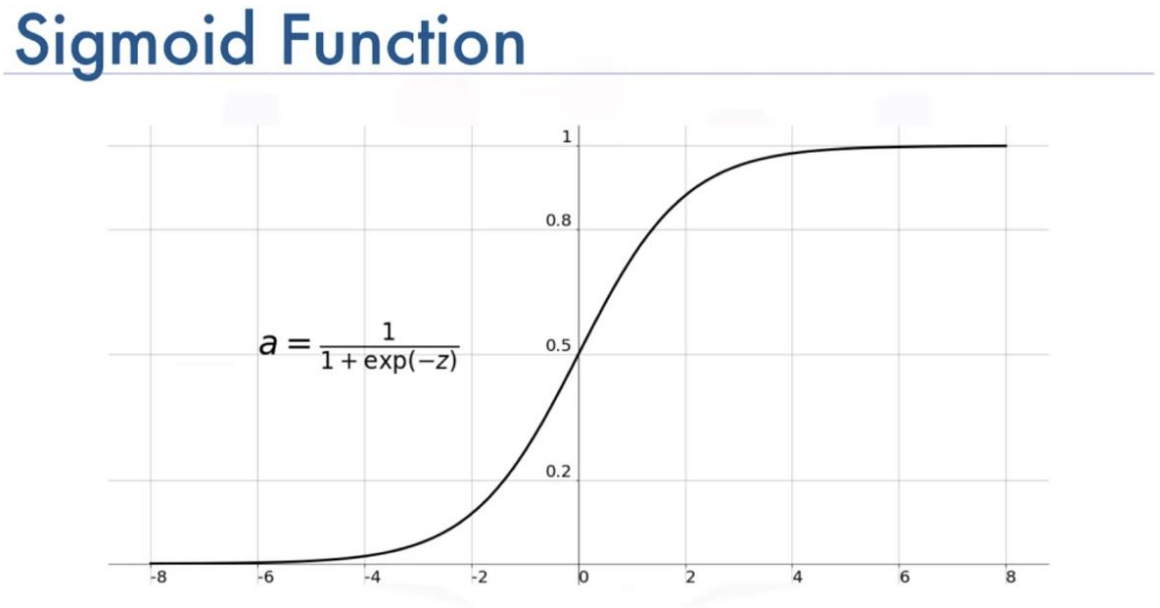
LSTM 由 3 個gate 組成:

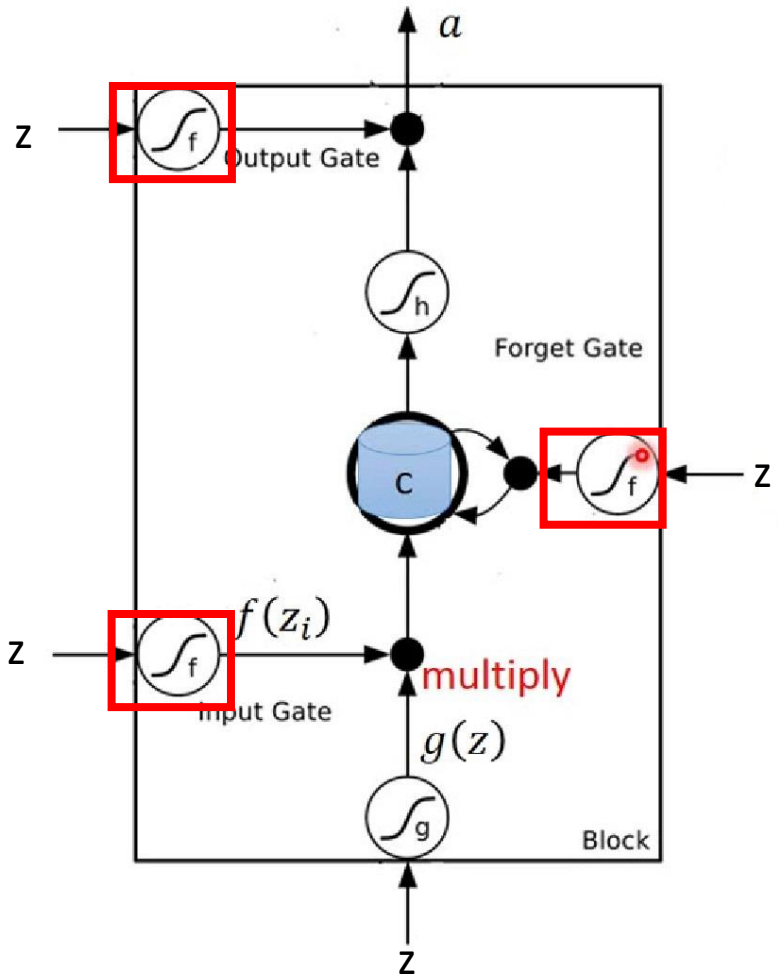
- Input Gate : 決定這次 input 要不要加入運算
- Output Gate : 決定是否將運算結果輸出，否則輸出 0
- Forget Gate : 是否遺忘而原本 memory cell內的值
- 4 的 inputs，1 個 output



LSTM

紅色圈起來的 activation function 是sigmoid function 其值域界在 0 ~ 1 之間





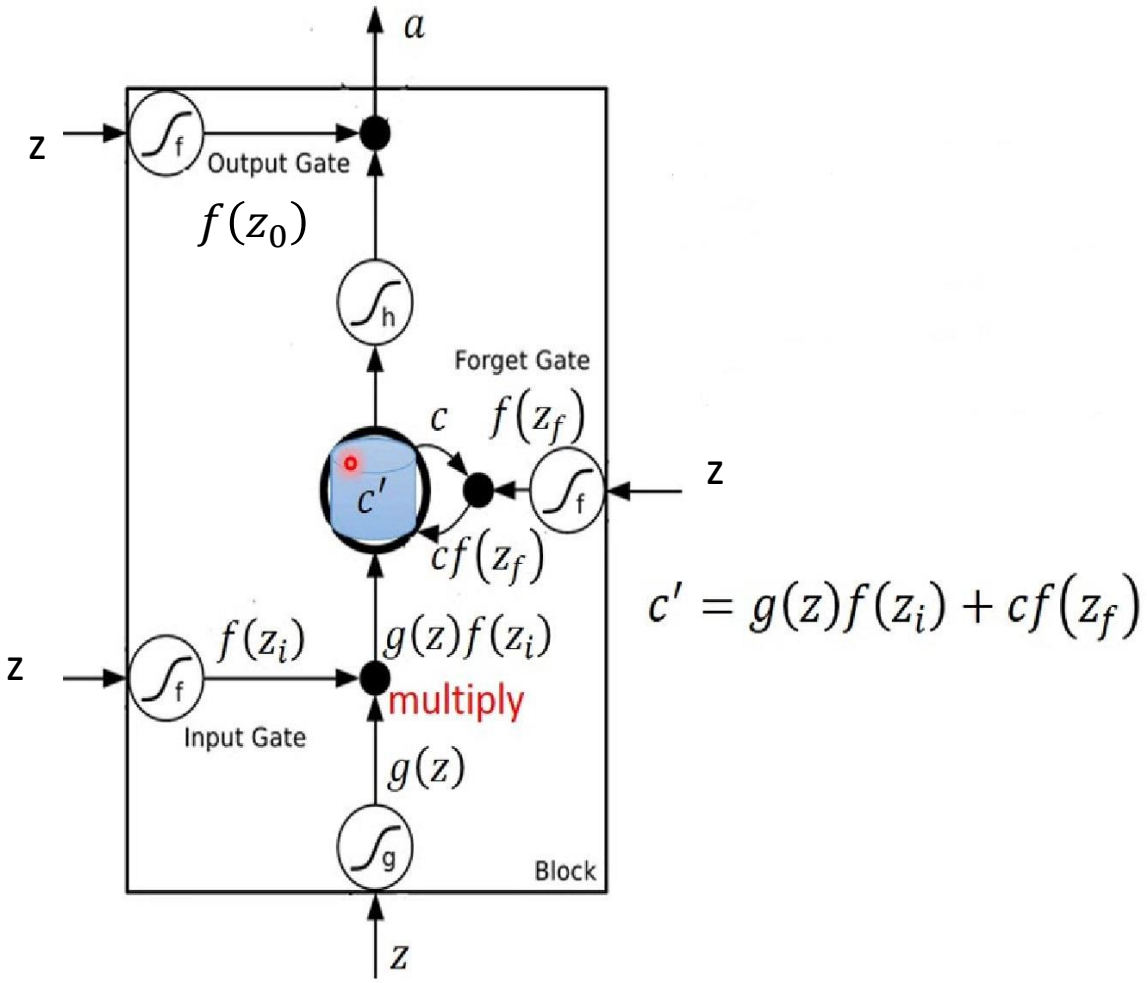
LSTM

Forget Gate: 1 是加入本次input 進 memory cell，
0 是 不將 input 加入 memory cell

c 是原本 memory cell 裡的值

$$c' = g(z)f(z_i) + cf(z_f)$$

$$a = h(c')f(z_0)$$



Code