Sequence Model.

sequence data: Speech recognition

DNA sequence Analysis

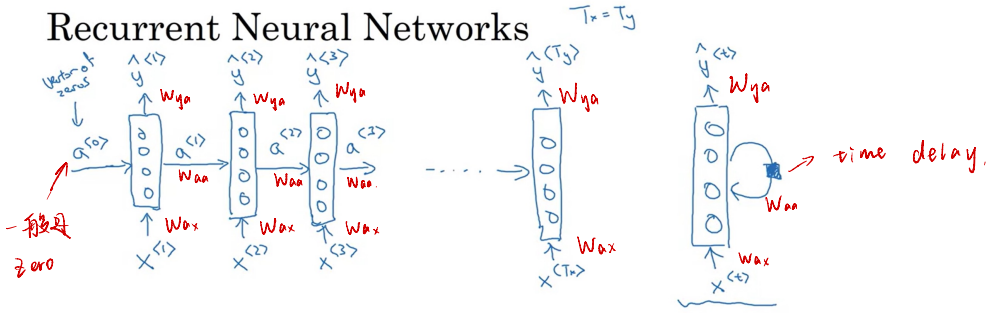
Video activity recognition.

- Recurrent Neural Network.

Why not just use standard Network?

Doesn't share feature learned across different position of text.

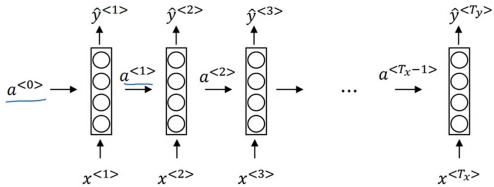
Recurrent Neural Networks

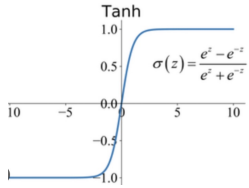


RNN 的缺点是仅使用序列中较早信息来进行预测.

这可以用 Bi directional RNN 解决.

Forward Propagation





activation function

↙ tanh / Relu

常用

$a^{<0>} = \vec{0}$

$a^{<1>} = g(W_{aa}\, a^{<0>} + W_{ax}\, x^{<1>} + b_a)$

$\hat{y}^{<1>} = g(W_{ya}\, a^{<1>} + b_y)$ ← 二分 sigmoid, K类.

$a^{<t>} = g(W_{aa}\, a^{<t-1>} + W_{ax}\, x^{<t>} + b_a)$

$\hat{y}^{<t>} = g(W_{ya}\, a^{<t>} + b_y)$

$$a^{\langle t \rangle} = g\left(W_{aa}\, a^{\langle t-1 \rangle} + W_{ax}\, x^{\langle t \rangle} + b_a\right)$$

(100, 100) (100, 10000) 10,000 100

化简

$$a^{\langle t \rangle} = g\left(W_a \left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_a\right)$$

$$\begin{matrix}100\\\updownarrow\end{matrix}\left[\underset{\longleftarrow\ 100\ \longrightarrow}{W_{aa}} \mid \underset{\longleftarrow\ 10{,}000\ \longrightarrow}{W_{ax}}\right] = W_a \quad (100,\ 10100)$$

$$\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] = \begin{bmatrix} a^{\langle t-1 \rangle} \\ \hline x^{\langle t \rangle} \end{bmatrix} \begin{matrix}\updownarrow 100\\ \updownarrow 10{,}000\end{matrix} \quad \updownarrow 10{,}100$$

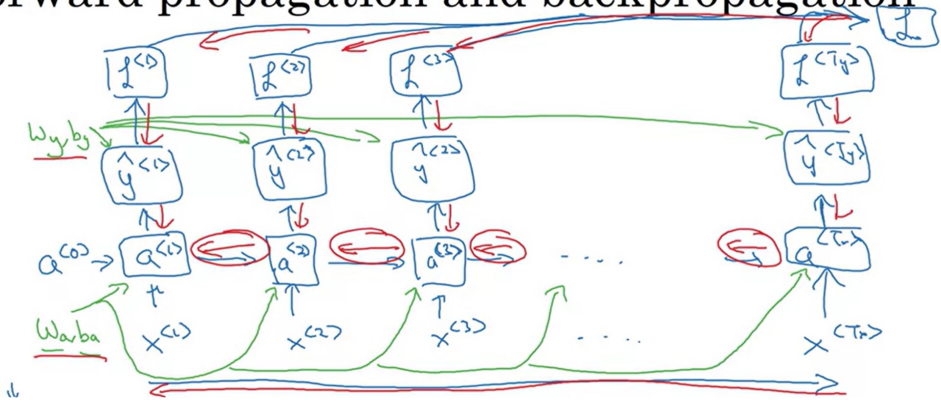
$$\left[W_{aa} \mid W_{ax}\right] \begin{bmatrix} a^{\langle t-1 \rangle} \\ \hline x^{\langle t \rangle} \end{bmatrix} = W_{aa}\, a^{\langle t-1 \rangle} + W_{ax} x^{\langle t \rangle}$$

△ Back propagation.

$$\mathcal{L}^{\langle t \rangle}(\hat{y}^{\langle t \rangle}, y^{\langle t \rangle}) = -y^{\langle t \rangle} \log \hat{y}^{\langle t \rangle} - (1 - y^{\langle t \rangle}) \log (1 - \hat{y}^{\langle t \rangle})$$

$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_y} \mathcal{L}^{\langle t \rangle}(\hat{y}^{\langle t \rangle}, y^{\langle t \rangle})$$

Forward propagation and backpropagation



可以看作从最后一层逐步调整权重参数.

3.1 - Basic RNN Backward Pass

Begin by computing the backward pass for the basic RNN cell. Then, in the following sections, iterate through the cells.

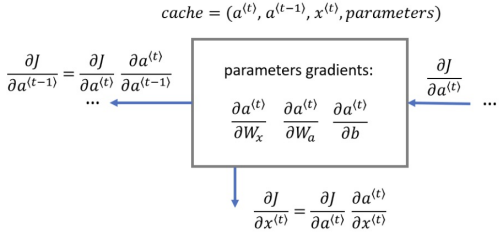


Figure 6: The RNN cell's backward pass. Just like in a fully-connected neural network, the derivative of the cost function J backpropagates through the time steps of the RNN by following the chain rule from calculus. Internal to the cell, the chain rule is also used to calculate $(\frac{\partial J}{\partial W_{ax}}, \frac{\partial J}{\partial W_{aa}}, \frac{\partial J}{\partial b})$ to update the parameters (W_{ax}, W_{aa}, b_a). The operation can utilize the cached results from the forward path.

$$a^{\langle t \rangle} = \tanh\left(W_{ax} x^{\langle t \rangle} + W_{aa} a^{\langle t-1 \rangle} + b_a\right)$$

$$\frac{\partial \tanh(x)}{\partial x} = 1 - \tanh^2(x)$$

element-wise multiplication.

$$d\tanh = da_{next} \times \left(1 - \tanh^2\left(W_{ax} x^{\langle t \rangle} + W_{aa} a^{\langle t-1 \rangle} + b_a\right)\right)$$

$$dW_{ax} = d\tanh \cdot x^{\langle t \rangle T}$$

$$dW_{aa} = d\tanh \cdot a^{\langle t-1 \rangle T}$$

$$db_a = \sum_{batch} d\tanh$$

$f(x) = \tanh(x)$

$f'(x) = 1 - \tanh^2(x)$

指定最后一个

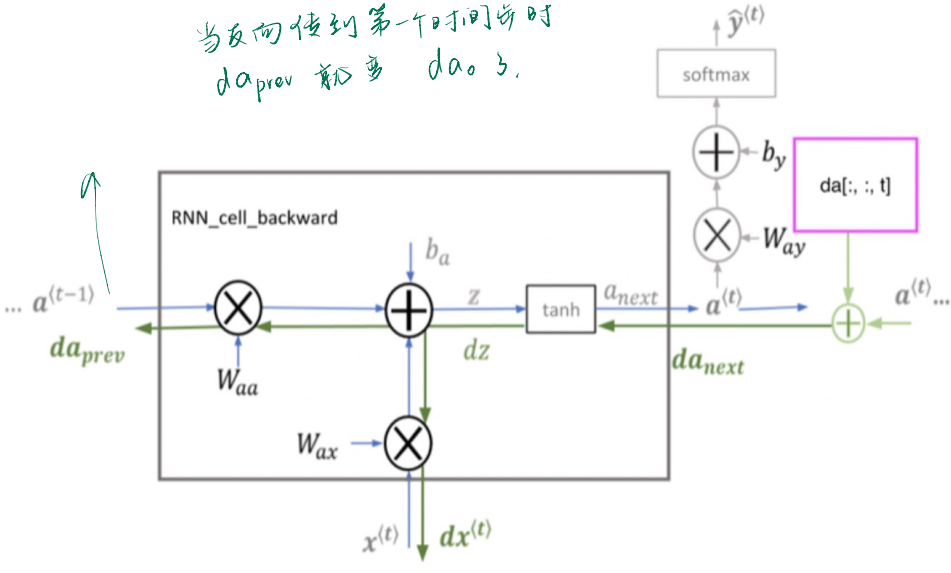
→ np.sum(dtanh, axis=-1, keepdim=True)

保证 db_a 和 dtanh 相同维度数

$$dx^{\langle t \rangle} = W_{ax}^T \cdot d\tanh$$

$$da_{prev} = W_{aa}^T \cdot d\tanh$$

当反向传到第一个时间步时 da_{prev} 就变 da_0 了.



✳ Vector Derivatives

Ex1: $\vec{y}$: length C

W: C row, D column

$\vec{x}$: length D.

$$\vec{y} = W\vec{x}$$

$$\vec{y}_3 = \sum_{j=1}^{D} W_{3,j}\vec{x}_j$$

$$= W_{3,1}\vec{x}_1 + W_{3,2}\vec{x}_2 + \cdots W_{3,D}\vec{x}_D$$

我们要计算 $\frac{\partial \vec{y}_i}{\partial \vec{x}_j}$

$$\frac{\partial \vec{y}_3}{\partial \vec{x}_7} = \frac{\partial}{\partial \vec{x}_7}(\cdots\cdots + W_{3,7}\vec{x}_7 + \cdots)$$

$$= 0 + 0 + \cdots + \frac{\partial}{\partial \vec{x}_7}(W_{3,7}\vec{x}_7) + \cdots + 0$$

$$= W_{3,7}$$

也就是我们得 $\frac{\partial \vec{y}_i}{\partial \vec{x}_j} = W_{ij}$

所以 $\frac{\partial \vec{y}}{\partial \vec{x}} = W$

Ex2: $\vec{y} = \vec{x}W$ (反过来)

$$\vec{y}_3 = \sum_{j=1}^{D} \vec{x}_j W_{j,3}$$

$$\frac{\partial \vec{y}_3}{\partial \vec{x}_7} = W_{7,3}$$

$$\frac{\partial \vec{y}}{\partial \vec{x}} = W.$$

Ex3: $\frac{d\vec{y}}{dW}$?

$$\vec{y}_3 = \vec{x}_1 W_{1,3} + \cdots + \vec{x}_D W_{D,3}$$

$$\frac{\partial \vec{y}_3}{\partial W_{7,8}} = 0$$ 不在 W 的列

$$\frac{\partial \vec{y}_3}{\partial W_{2,3}} = \vec{x}_2$$

$$\therefore \frac{\partial \vec{y}_j}{\partial W_{i,j}} = \vec{x}_i$$

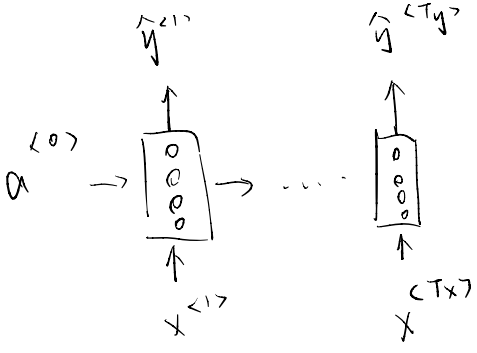
if it is 3-D. $F_{i,j,k} = \frac{\partial \vec{y}_i}{\partial W_{j,k}}$

then. $F_{i,j,k} = \vec{x}_j$

for more detail it's in vecDerivs.Pdf.

△ Architecture

- Many to Many



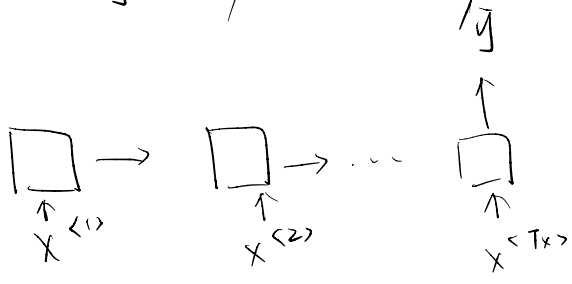
$T_x = T_y$

- Many to one

Ex: Sentimant classification

x = text

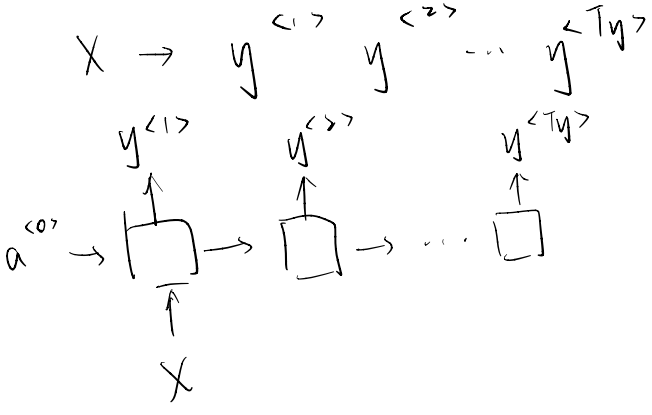
$y = 0 / 1$; 1 ··· 5 打分.



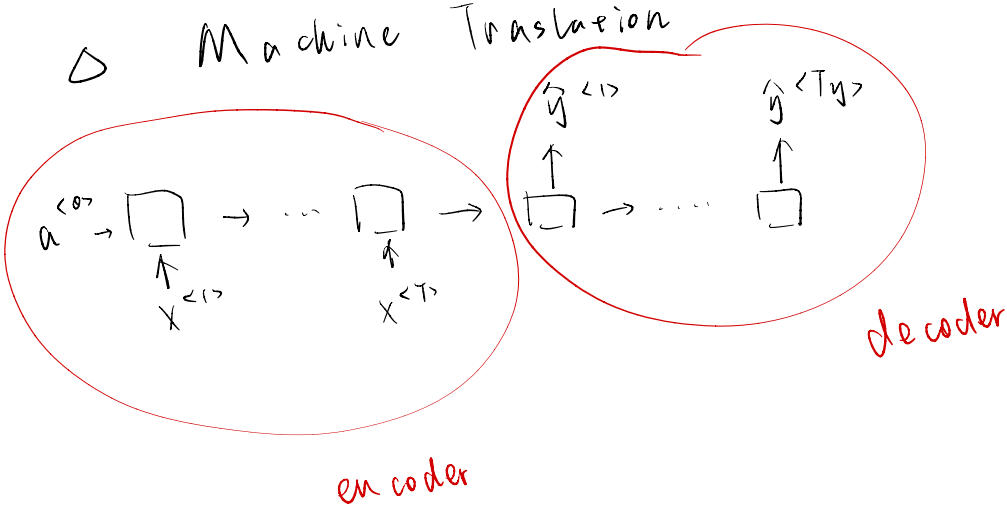
- One to One

- One to Many

Music Generation.



- Machine Translation



- Language modelling with RNN.

Training set: large corpus of english text.

Tokenize

Cats average 15 hours of sleep a day. <EOS>

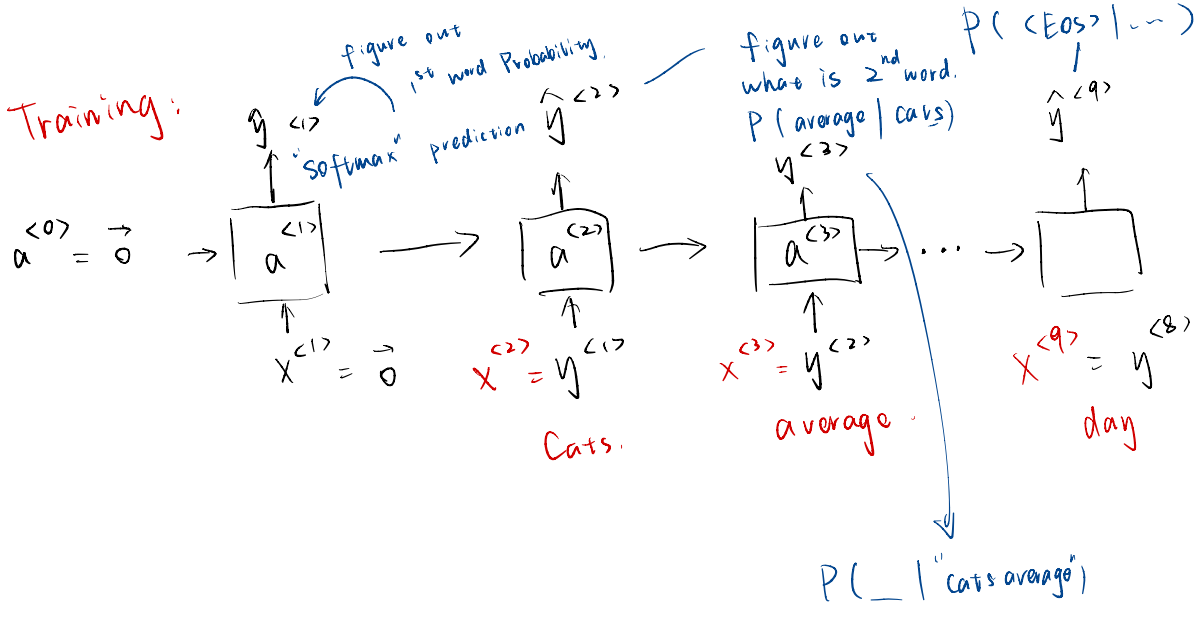
$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ ... $y^{<8>}$ $y^{<9>}$

$x^{<t>} = y^{<t-1>}$

The Egyptian ~~Mau~~ is a bread of cat. <EOS>

10,000 <UNK>

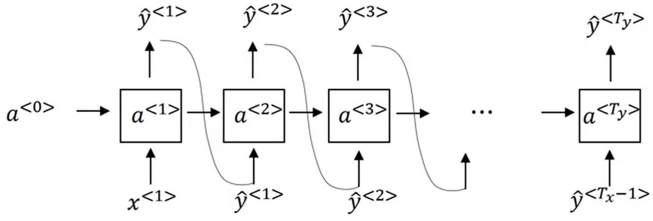
Ex: Cats average 15 hours of sleep a day. <EOS>.



$$P(y^{<1>}, y^{<2>}, y^{<3>})$$

$$= P(y^{<1>}) \cdot P(y^{<2>} | y^{<1>}) \cdot P(y^{<3>} | y^{<1>}, y^{<2>})$$

Sampling:



<EOS>

= 0.

First word randomly sample according to softmax distribution.

np.random.choice to sample according to distribution defined by vector probability.

- **Step 1**: Input the "dummy" vector of zeros $x^{\langle 1 \rangle} = \vec{0}$.
  - This is the default input before you've generated any characters. You also set $a^{\langle 0 \rangle} = \vec{0}$
- **Step 2**: Run one step of forward propagation to get $a^{\langle 1 \rangle}$ and $\hat{y}^{\langle 1 \rangle}$. Here are the equations:

hidden state:

$$a^{\langle t+1 \rangle} = \tanh(W_{ax} x^{\langle t+1 \rangle} + W_{aa} a^{\langle t \rangle} + b) \tag{1}$$

activation:

$$z^{\langle t+1 \rangle} = W_{ya} a^{\langle t+1 \rangle} + b_y \tag{2}$$

prediction:

$$\hat{y}^{\langle t+1 \rangle} = softmax(z^{\langle t+1 \rangle}) \tag{3}$$

- Details about $\hat{y}^{\langle t+1 \rangle}$:
  - Note that $\hat{y}^{\langle t+1 \rangle}$ is a (softmax) probability vector (its entries are between 0 and 1 and sum to 1).
  - $\hat{y}_i^{\langle t+1 \rangle}$ represents the probability that the character indexed by "i" is the next character.
  - A `softmax()` function is provided for you to use.

- Vanishing gradients with RNNs

The cat, which ate fish, ... , was full

The cats, which ate fish, ... , were full

Language 非常长的依赖性，较早的措辞会影响后面的内容.

RNN 不擅于此，因为训练 very deep neural network

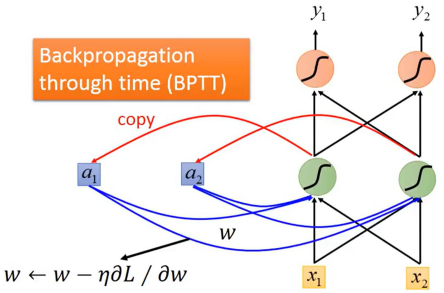
会出现 vanishing gradients problem (梯度消失)

输出y的梯度将很难传播回来影响早期层的权重.

⇒ $\hat{y}^{<Ty>}$ 只受最近几个 $x^{<Tx>}$ 影响

Learning

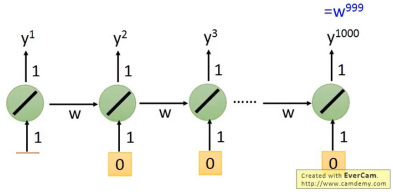
Exploding gradients. NaN.



Why?

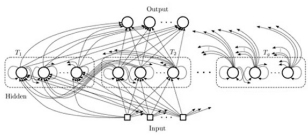
$w = 1$ ⇒ $y^{1000} = 1$	Large $\partial L/\partial w$	Small Learning rate?
$w = 1.01$ ⇒ $y^{1000} \approx 20000$		
$w = 0.99$ ⇒ $y^{1000} \approx 0$	small $\partial L/\partial w$	Large Learning rate?
$w = 0.01$ ⇒ $y^{1000} \approx 0$		

Toy Example



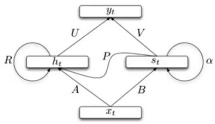
Helpful Techniques

Clockwise RNN



[Jan Koutnik, JMLR'14]

Structurally Constrained Recurrent Network (SCRN)



[Tomas Mikolov, ICLR'15]

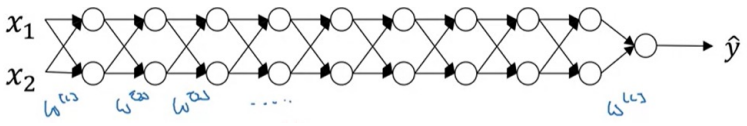
Vanilla RNN Initialized with Identity matrix + ReLU activation function [Quoc V. Le, arXiv'15]

- Outperform or be comparable with LSTM in 4 different tasks

补：

Vanishing / Exploding Gradients

以一个多的 Nerual Network 为例：



假设我们使用 $g(z) = z$ $b^{[l]} = 0$.

$y = w^{[L]} w^{[L-1]} \cdots w^{[3]} w^{[2]} w^{[1]} x$

$a^{[1]} = z^{[1]} = w^{[1]} x$

$a^{[2]} = g(z^{[2]}) = g(w^{[2]} a^{[1]})$

exp 爆炸

假设 $w^{[l]} = \begin{bmatrix} 0.9 & 0 \\ 0 & 0.9 \end{bmatrix}$

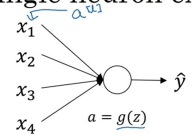
$\hat{y} = w^{[L]} \begin{bmatrix} 0.9 & 0 \\ 0 & 0.9 \end{bmatrix}^{L-1} x$, 0.9^L

$w^{[l]} > I$ explode big

$w^{[l]} < I$ small.

Weight Initialization for Deep Network

Single neuron example



$z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$

large n → smaller w_i

$Var(w_i) = \frac{1}{n}$

matrix

$w^{[l]} = np.random.randn(shape) * np.sqrt(\frac{1}{n^{[l-1]}})$

if use ReLu $Var = \frac{2}{n}$ $\frac{2}{n^{[l-1]}}$

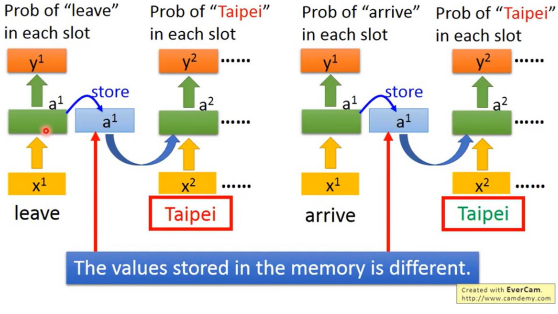
if use tanh

或者 $\sqrt{\frac{1}{n^{[l-1]}}}$ Xavier Initialization

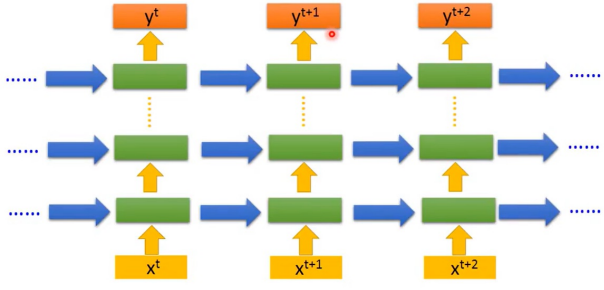
$\sqrt{\frac{2}{n^{[l-1]} \times n^{[l]}}}$ Yoshua Bengio.

常见基础 RNN.

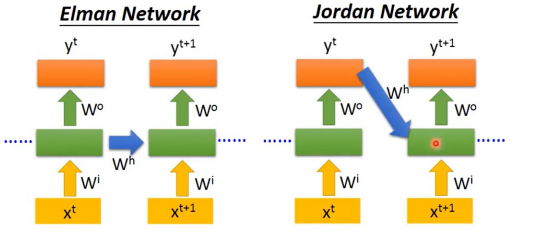
RNN



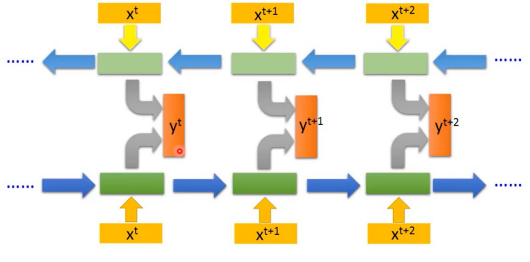
Of course it can be deep ...



Elman Network & Jordan Network

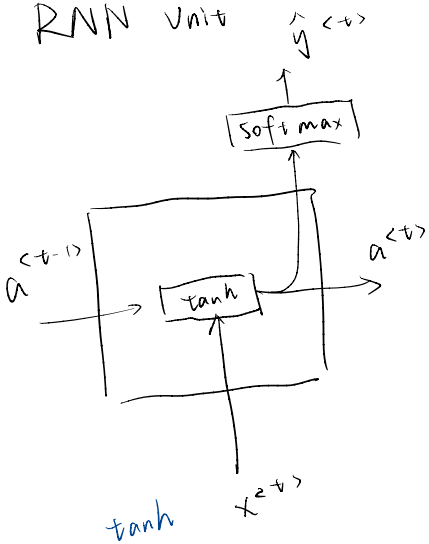


Bidirectional RNN



- GRU (Gated Recurrent Unit)

RNN unit



tanh

$$a^{<t>} = g(W_a[a^{<t-1>}, x^{<t>}] + b_a)$$

GRU (Simplify)

多一个新变量 "c" = Memory cell 用来记住类 cat 还是 cats.

$$c^{<t>} = a^{<t>}$$

$$\tilde{c}^{<t>} = \tanh(W_c[c^{<t-1>}, x^{<t>}] + b_c)$$

Gate: $\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u)$

Γ_u: update

↓ sigmoid function

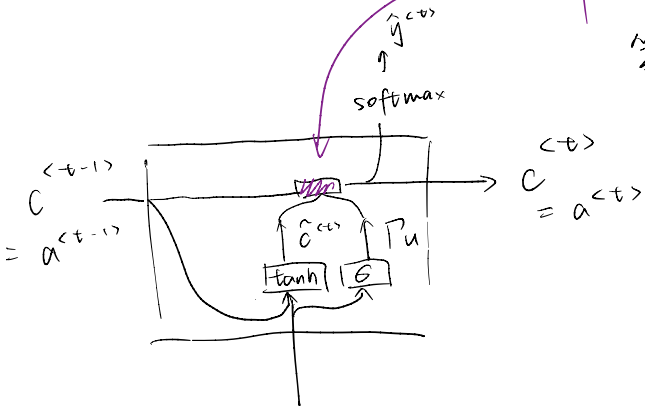
↓

Γ_u 是 0 或 1.

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1-\Gamma_u) * c^{<t-1>}$$

等于1更新，等于0保留旧值.

element wise



Full GRU

$$\tilde{c}^{\langle t \rangle} = \tanh(W_c[\Gamma_r * c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_c)$$

两个都有的相关

$$\Gamma_u = \sigma(W_u[c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u)$$

$$\Gamma_r = \sigma(W_r[c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_r)$$

$$c^{\langle t \rangle} = \Gamma_u * \tilde{c}^{\langle t \rangle} + (1 - \Gamma_u) * c^{\langle t-1 \rangle}$$

$$a^{\langle t \rangle} = c^{\langle t \rangle}$$

LSTM

$$\tilde{c}^{\langle t \rangle} = \tanh(W_c[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_c)$$

update $\Gamma_u = \sigma(W_u[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u)$

forget $\Gamma_f = \sigma(W_f[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_f)$

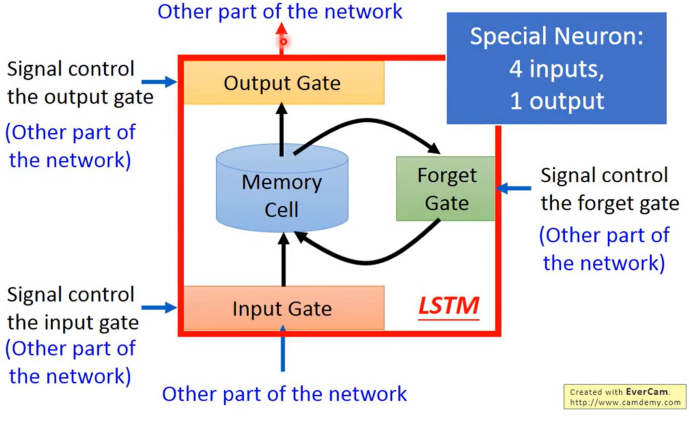
output $\Gamma_o = \sigma(W_o[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_o)$

$$c^{\langle t \rangle} = \Gamma_u \times \tilde{c}^{\langle t \rangle} + \Gamma_f \times c^{\langle t-1 \rangle}$$

$$a^{\langle t \rangle} = \Gamma_o \times \tanh c^{\langle t \rangle}$$

多体现 $c^{\langle t-1 \rangle}$

peek hole connection.



Flexible.

LSTM 老，GRU 新

两个 Gate 运行更快

但两者各有优点.

LSTM Back Prop

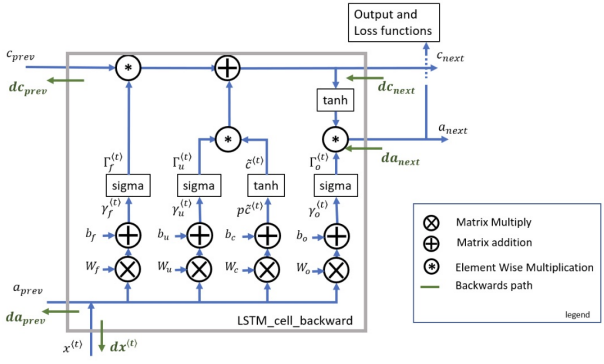


Figure 8: LSTM Cell Backward. Note the output functions, while part of the `lstm_cell_forward`, are not included in `lstm_cell_backward`

The equations for the LSTM backward pass are provided below. (If you enjoy calculus exercises feel free to try deriving these from scratch yourself.)

2. Gate Derivatives

Note the location of the gate derivatives ($\gamma..$) between the dense layer and the activation function (see graphic above). This is convenient for computing parameter derivatives in the next step.

$$d\gamma_o^{\langle t \rangle} = da_{next} * \tanh(c_{next}) * \Gamma_o^{\langle t \rangle} * \left(1 - \Gamma_o^{\langle t \rangle}\right) \tag{7}$$

$$dp\tilde{c}^{\langle t \rangle} = \left(dc_{next} * \Gamma_u^{\langle t \rangle} + \Gamma_o^{\langle t \rangle} * (1 - \tanh^2(c_{next})) * \Gamma_u^{\langle t \rangle} * da_{next}\right) * \left(1 - \left(\tilde{c}^{\langle t \rangle}\right)^2\right) \tag{8}$$

$$d\gamma_u^{\langle t \rangle} = \left(dc_{next} * \tilde{c}^{\langle t \rangle} + \Gamma_o^{\langle t \rangle} * (1 - \tanh^2(c_{next})) * \tilde{c}^{\langle t \rangle} * da_{next}\right) * \Gamma_u^{\langle t \rangle} * \left(1 - \Gamma_u^{\langle t \rangle}\right) \tag{9}$$

$$d\gamma_f^{\langle t \rangle} = \left(dc_{next} * c_{prev} + \Gamma_o^{\langle t \rangle} * (1 - \tanh^2(c_{next})) * c_{prev} * da_{next}\right) * \Gamma_f^{\langle t \rangle} * \left(1 - \Gamma_f^{\langle t \rangle}\right) \tag{10}$$

3. Parameter Derivatives.

$$dW_f = d\gamma_f^{\langle t \rangle} \begin{bmatrix} a_{prev} \\ x_t \end{bmatrix}^T \qquad db_f = \sum_{batch} d\gamma_f^{\langle t \rangle}$$

$$dW_u = d\gamma_u^{\langle t \rangle} \begin{bmatrix} a_{prev} \\ x_t \end{bmatrix}^T \qquad db_u = \sum_{batch} d\gamma_u^{\langle t \rangle}$$

$$dW_c = dp\tilde{c}^{\langle t \rangle} \begin{bmatrix} a_{prev} \\ x_t \end{bmatrix}^T \qquad db_c = \sum_{batch} dp\tilde{c}^{\langle t \rangle}$$

$$dW_o = d\gamma_o^{\langle t \rangle} \begin{bmatrix} a_{prev} \\ x_t \end{bmatrix}^T \qquad db_o = \sum_{batch} d\gamma_o^{\langle t \rangle}$$

Finally, you will compute the derivative with respect to the previous hidden state, previous memory state, and input.

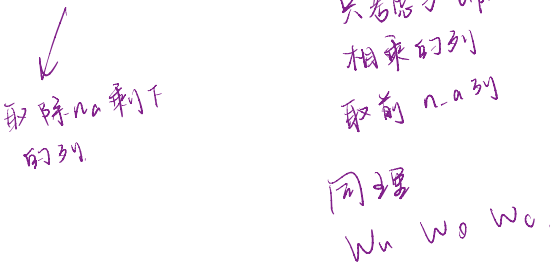
$$da_{prev} = W_f^T d\gamma_f^{\langle t \rangle} + W_u^T d\gamma_u^{\langle t \rangle} + W_c^T d\tilde{pc}^{\langle t \rangle} + W_o^T d\gamma_o^{\langle t \rangle} \tag{19}$$

Here, to account for concatenation, the weights for equations 19 are the first n_a, (i.e. $W_f = W_f[:, :n_a]$ etc...)

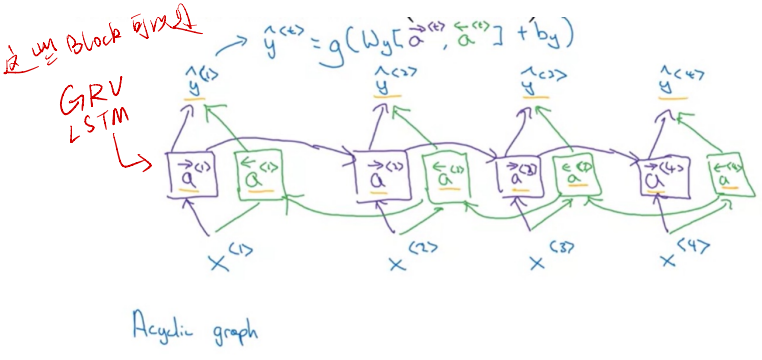
$$dc_{prev} = dc_{next} * \Gamma_f^{\langle t \rangle} + \Gamma_o^{\langle t \rangle} * (1 - \tanh^2(c_{next})) * \Gamma_f^{\langle t \rangle} * da_{next} \tag{20}$$

$$dx^{\langle t \rangle} = W_f^T d\gamma_f^{\langle t \rangle} + W_u^T d\gamma_u^{\langle t \rangle} + W_c^T d\tilde{pc}^{\langle t \rangle} + W_o^T d\gamma_o^{\langle t \rangle} \tag{21}$$

where the weights for equation 21 are from n_a to the end, (i.e. $W_f = W_f[:, n_a :]$ etc...)

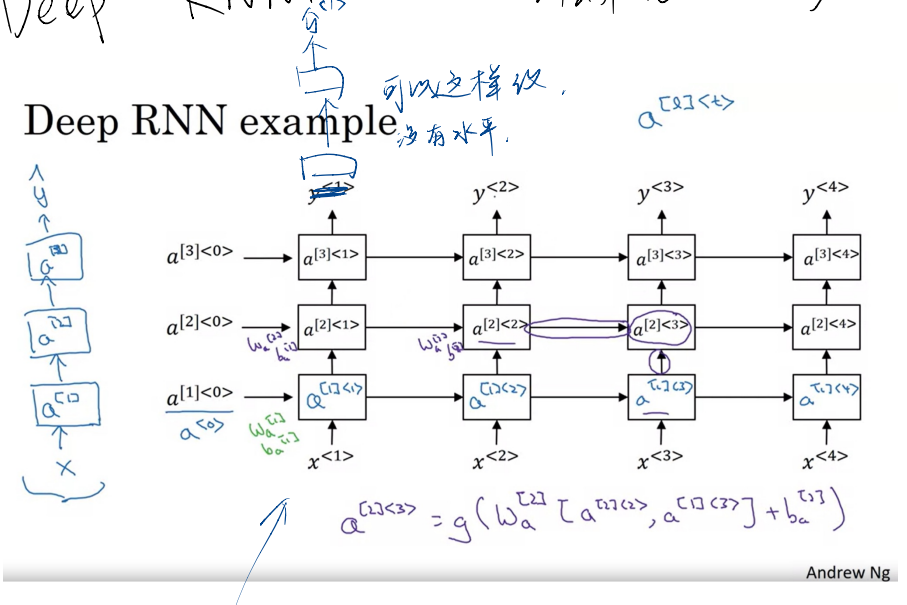


- Bidirectional RNN (BRNN)



- Deep RNN

从层来说以往很多



RNN
GRU
LSTM.

ConvLSTM.

ConvLSTM is a type of recurrent neural network for spatio-temporal prediction that has convolutional structures in both the input-to-state and state-to-state transitions. The ConvLSTM determines the future state of a certain cell in the grid by the inputs and past states of its local neighbors. This can easily be achieved by using a convolution operator in the state-to-state and input-to-state transitions (see Figure). The key equations of ConvLSTM are shown below, where $*$ denotes the convolution operator and $\odot$ the Hadamard product:

$$i_t = \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \odot \mathcal{C}_{t-1} + b_i)$$
$$f_t = \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \odot \mathcal{C}_{t-1} + b_f)$$
$$\mathcal{C}_t = f_t \odot \mathcal{C}_{t-1} + i_t \odot \tanh(W_{xc} * X_t + W_{hc} * \mathcal{H}_{t-1} + b_c)$$
$$o_t = \sigma(W_{xo} * X_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \odot \mathcal{C}_t + b_o)$$
$$\mathcal{H}_t = o_t \odot \tanh(C_t)$$

A ⊙ B

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \odot \begin{bmatrix} h & i \\ j & k \end{bmatrix}$$

$$= \begin{bmatrix} a \times h & b \times i \\ c \times j & d \times k \end{bmatrix}$$

Next-Frame Video Prediction with convLSTM.

https://keras.io/examples/vision/conv_lstm/

A * B convolution operator (fft convolve)

$$(1, 2, 3) * (4, 5, 6) = (4, 13, 28, 27, 18)$$

$$(1 + 2x + 3x^2)\,(4 + 5x + 6x^2) = 4 + 13x + 28x^2 + 27x^3 + 18x^4$$

	1	$2x$	$3x^2$
4	4	8	12
5	5	10	15
6	6	12	18

5 + 8 = 13.