Lecture 4 Seq2Seq Learning

Overview

1. Problems abstraction
    a. N-to-1: sentiment analysis,topic classification
    b. N-to-N: pos tagging
    c. N-to-path: parsing
    d. N-to-M: dialog, translation
2. Application
    a. Speech recognition

        — input: speech signal

        — output: text

    b. Movie frame labelling

        — input: video frame

        — output: scene labels

    c. PoS tagging

        — input: text

        — output: part of speech

    d. Arithmetic calculation

        Math expression -> Numbers

    e. Machine Translation

        Eng. Text -> Chinese Text

    f. Sentence Completion

        Partial sentence -> partial sentence

g. Coversational Modelling

Utterance -> utterance

Seq2Seq with DL

Recurrent Neural Network (RNN=Neural Network + Memory)

Memory: *retention of the information over time to influence the future action*

Model Structure

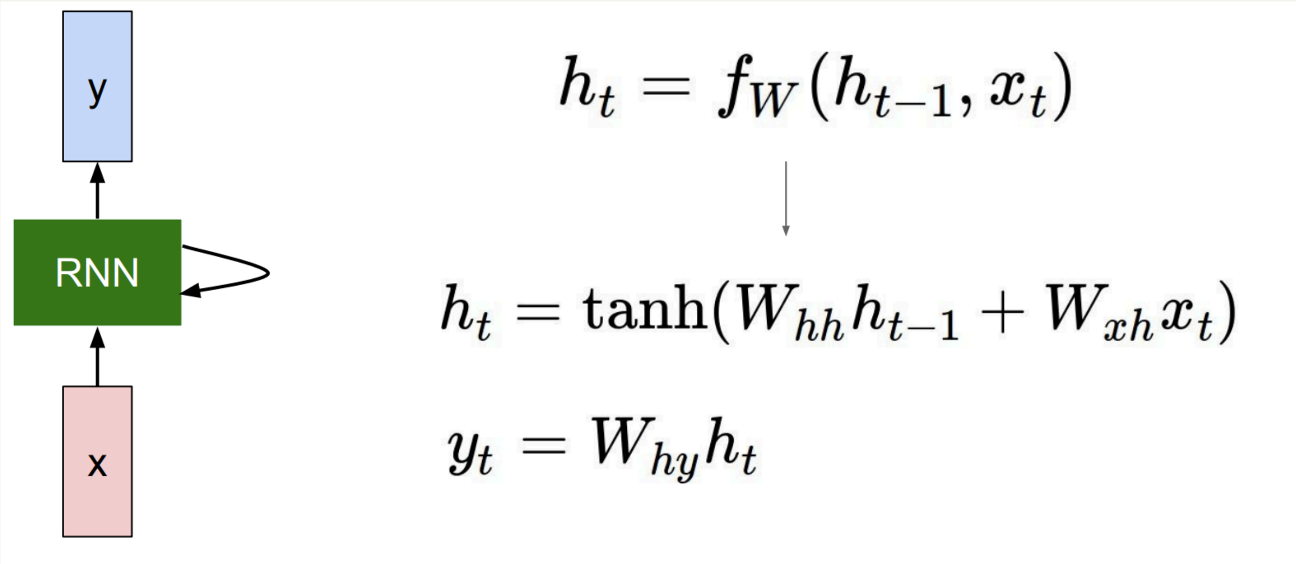
— Solve the limitation of CNN for that input and output has to be the same length

— Process sequence of vectors X by applying *recurrent function* at every time steps

— The RNN state consists of a single hidden vector h

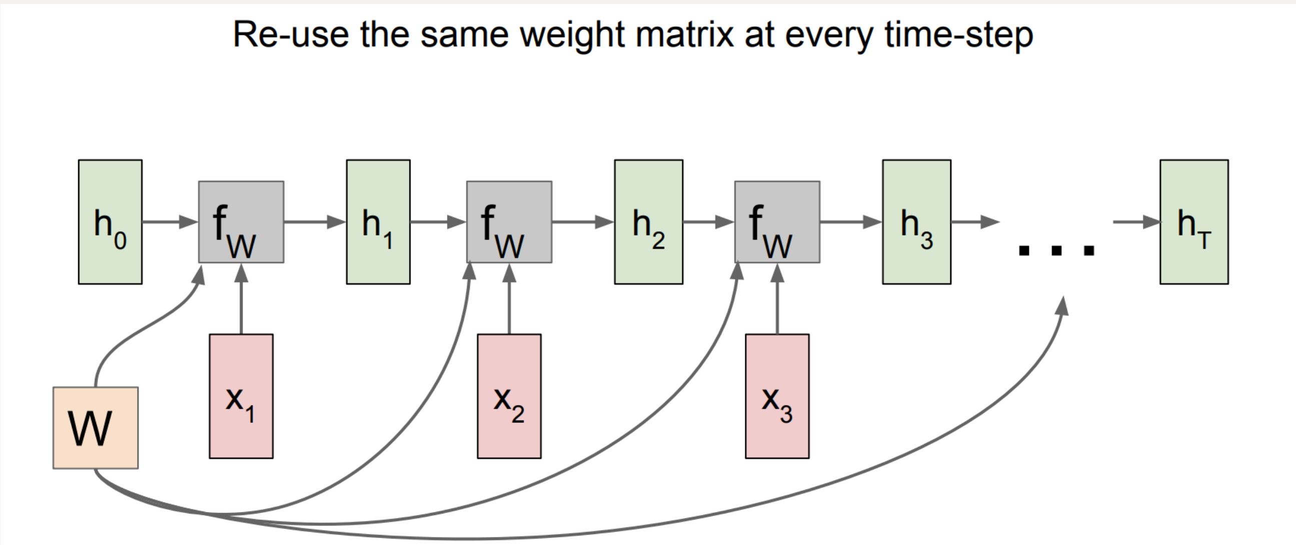
- h_t : current state cal. by the last state and the current input
- h_{t-1} : the state of the last time step
- x_t : the input at the current time step

Note.*every categorical inputs are named as 'time step'*



— The *tanh* function to project the linear output to the non-linear interval $[-1, 1]$

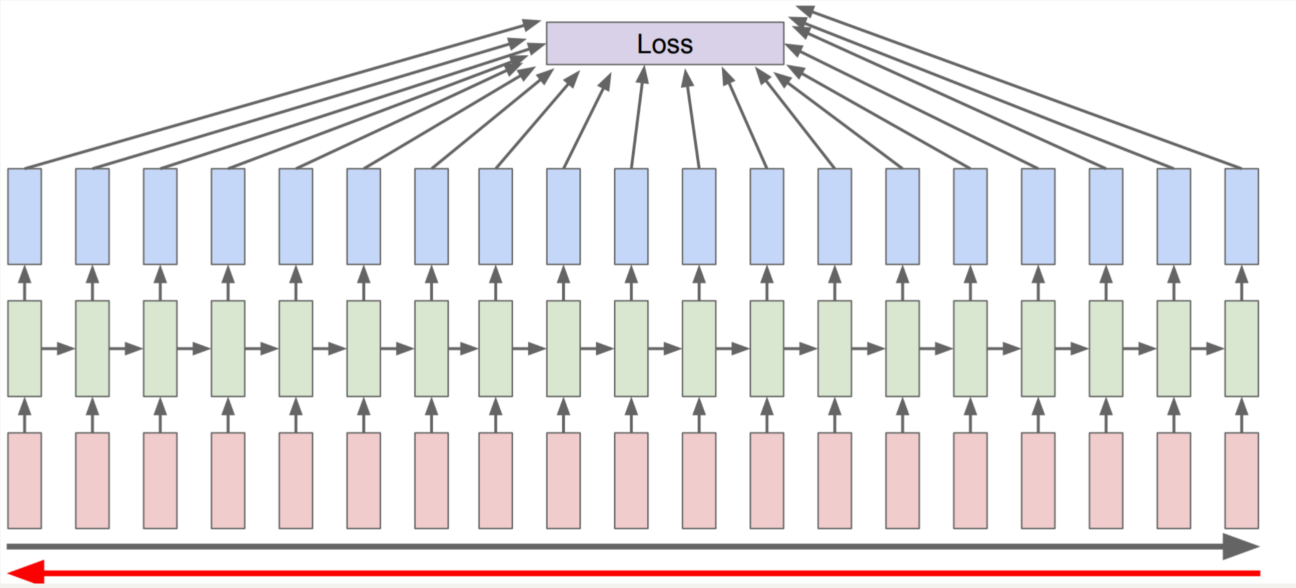
— Share the weights with all inputs



Backward propagation through time (BPTT)

— Forward through entire sequence to compute loss

— Backward through entire sequence to compute gradients

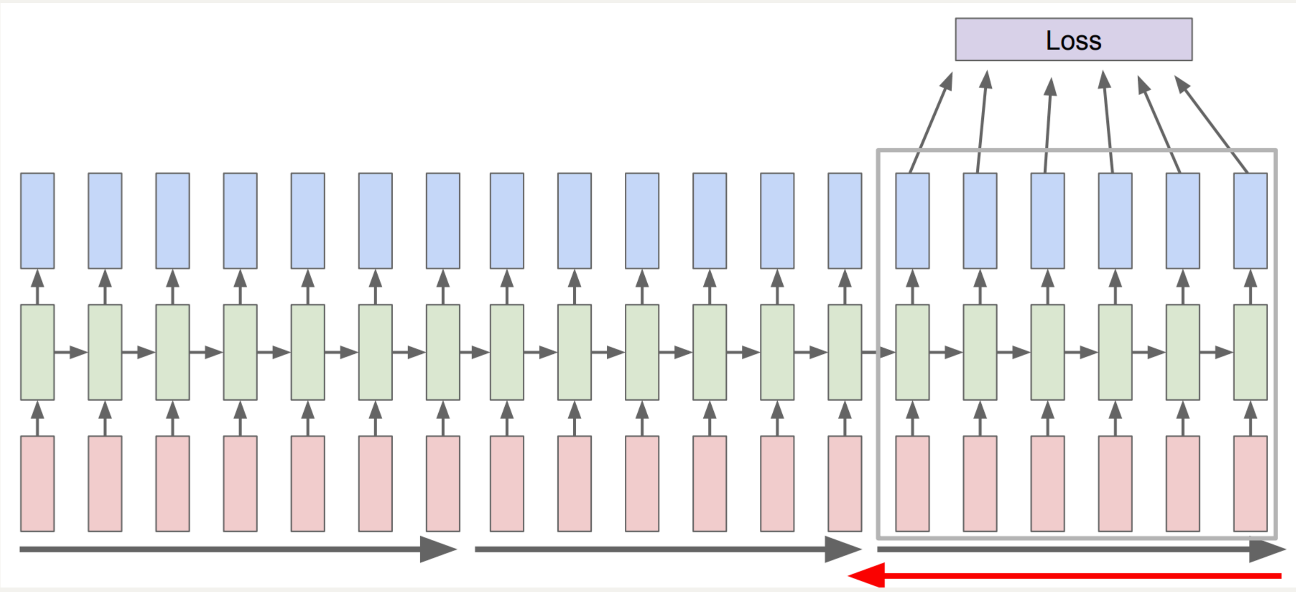


Truncated Backpropagation through time (TBPTT)

— Run forward and backward through chucks of sequences instead of whole sequence

— Carry hidden state in time forever

— Only backpropagate for some smaller number of steps

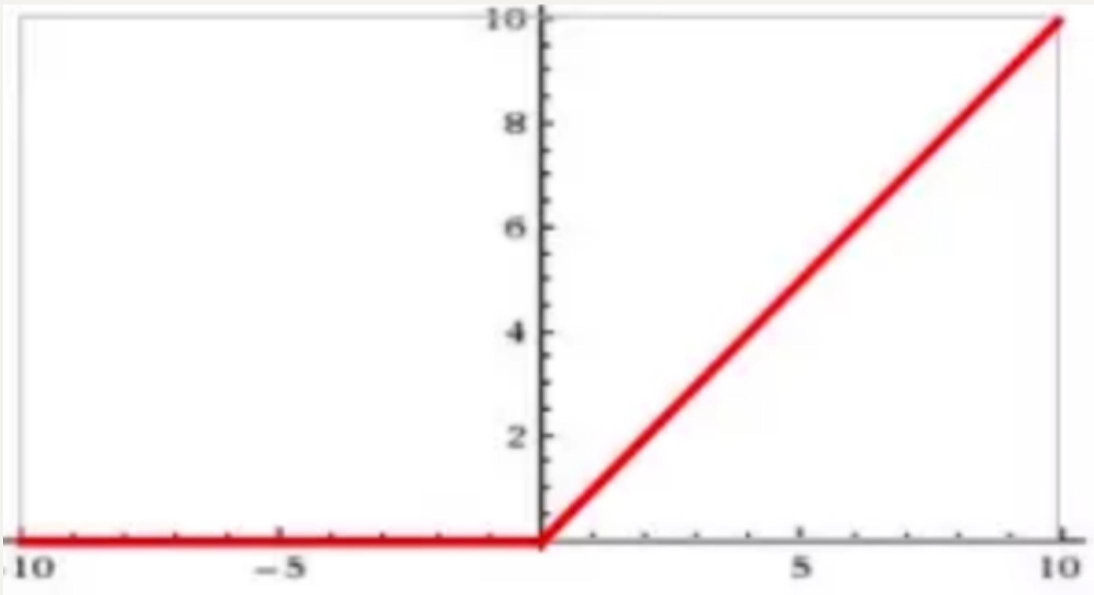


Limitation of vanilla RNN

— Gradients Exploding: if using active function $Relu$

$Relu : \phi(x) = max(0, x)$

$\frac{\alpha\phi(x)}{\alpha x} = 1(x > 0)\ or\ 0(x \leq 0)$

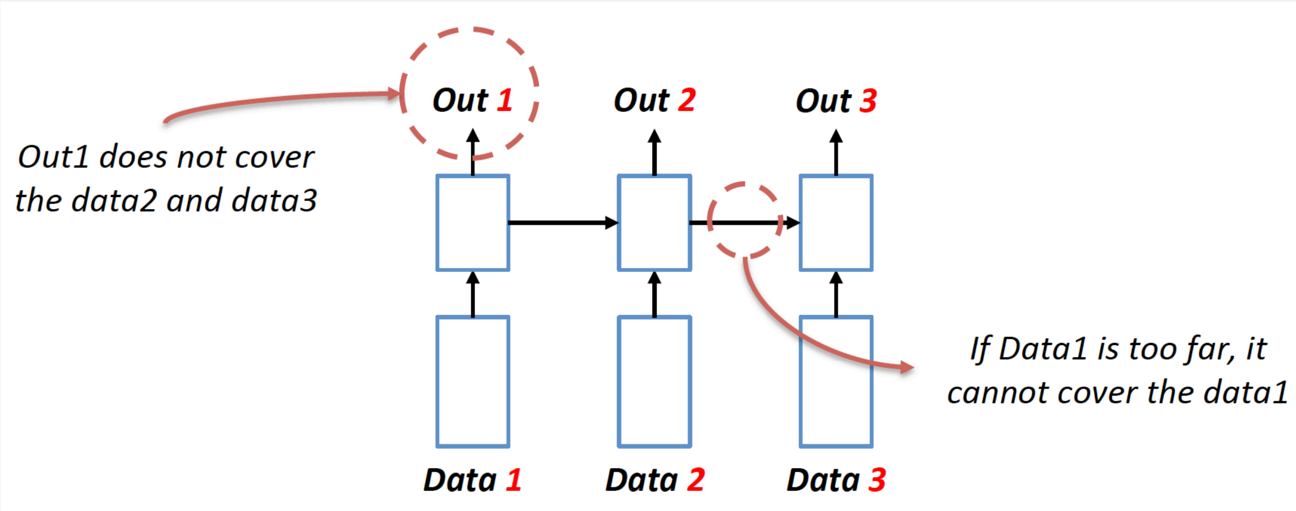


— Vanishing gradients reason: The $taugh$ function output a lot of values under 1, therefore, according to the chain rule, the value of a gradient rate are too small and the model stops learning or takes too long to learn (equation 2)

The $taugh$ function will project all weighted outputs to values between $[-1, 1]$, when using BPTT to calculate the gradients thus minimizing $loss$:

- loss function: cross entropy $E(y, \hat{y}) = \sum_t E(y_t, \hat{y}_t) = -\sum_t y_t log(\hat{y}_t)$
- Cal. Gradients: $\frac{\alpha E}{\alpha W} = \sum_t \frac{\alpha E_t}{\alpha W}$
- $h_t = taugh(w_{hh}h_{t-1} + w_{ht}x_t)$ .... (1)
- $\sum_t \frac{\alpha E_t}{\alpha W} = \sum_t \frac{\alpha E_t}{\alpha \hat{y}_t} \frac{\alpha \hat{y}_t}{\alpha h_t} \frac{\alpha h_t}{\alpha h_{t-1}} \frac{\alpha h_{t-1}}{\alpha h_{t-k}} \ldots \frac{\alpha h_{t-k}}{\alpha W}$ .... (2)

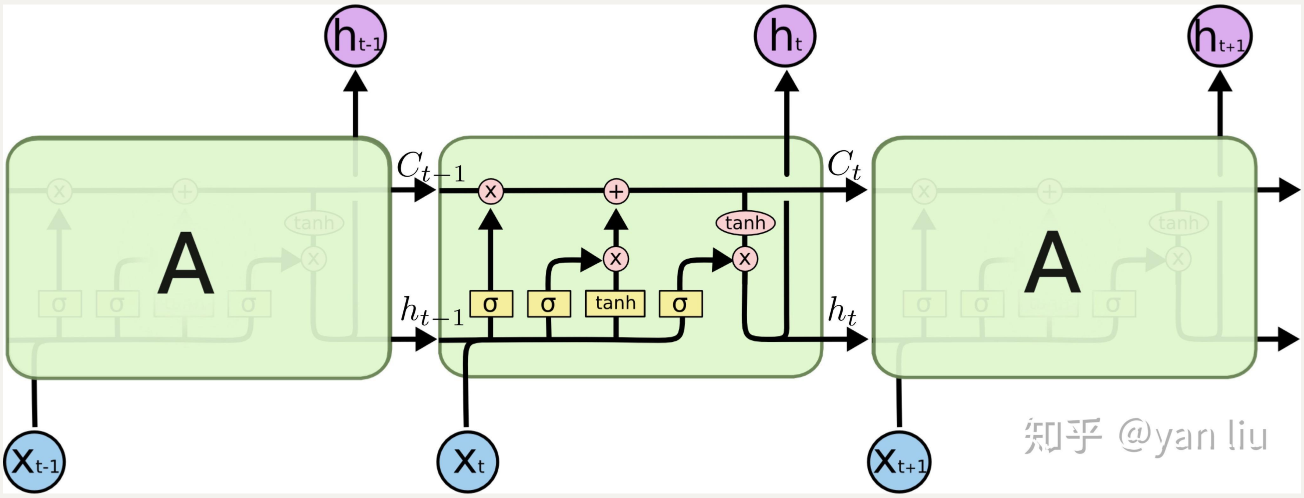
— Long term dependency

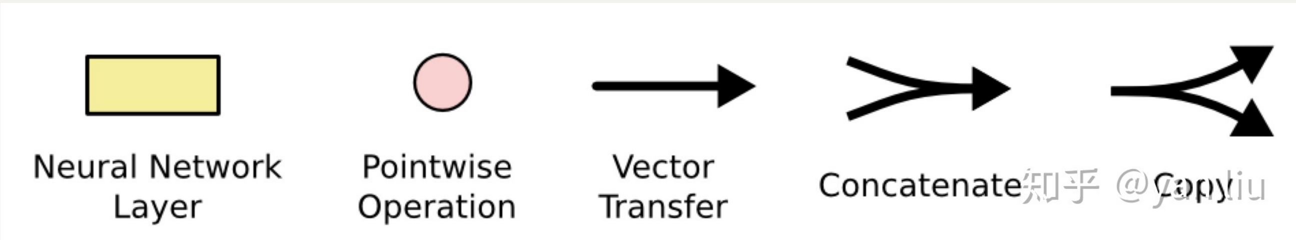


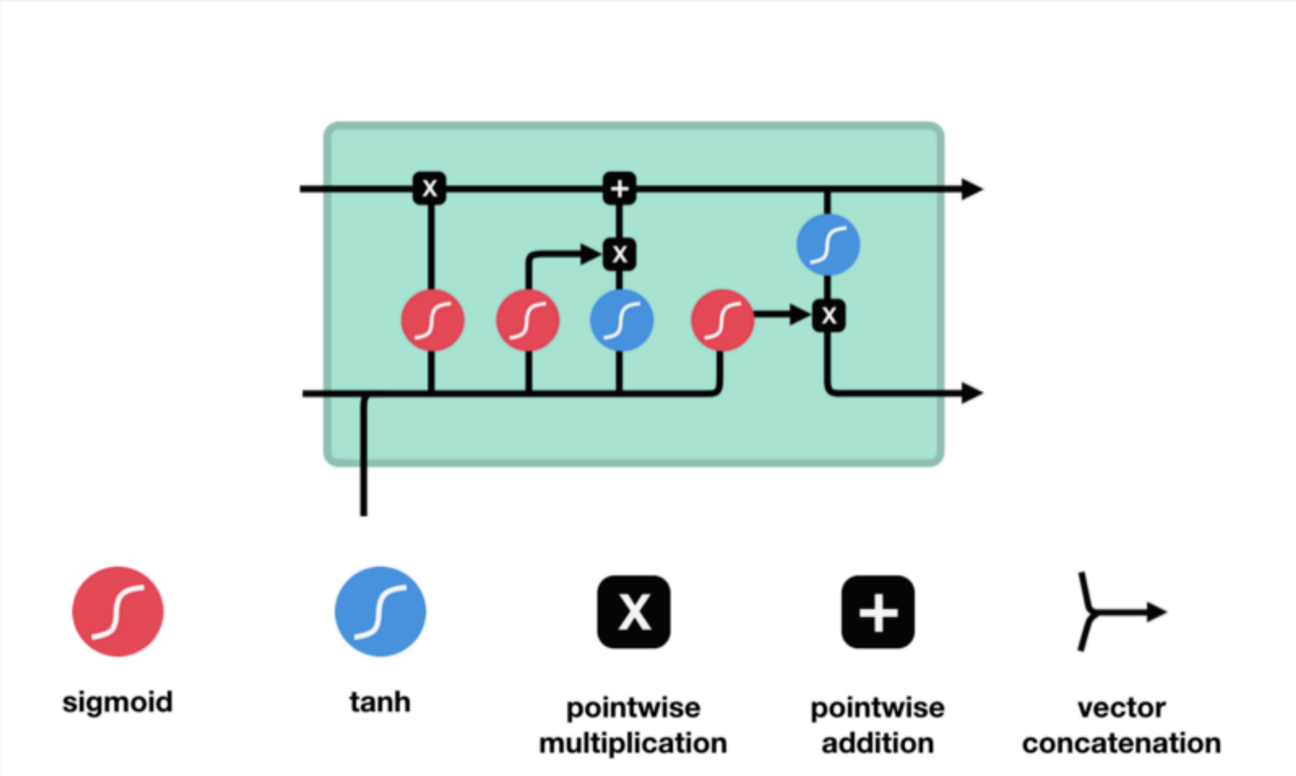
Long Short-term Memory (LSTM)

LSTM cell internal structure

— Finall output state at the current time step: $c_t = f_t * c_{t-1} + i_t * \tilde{c}_t$





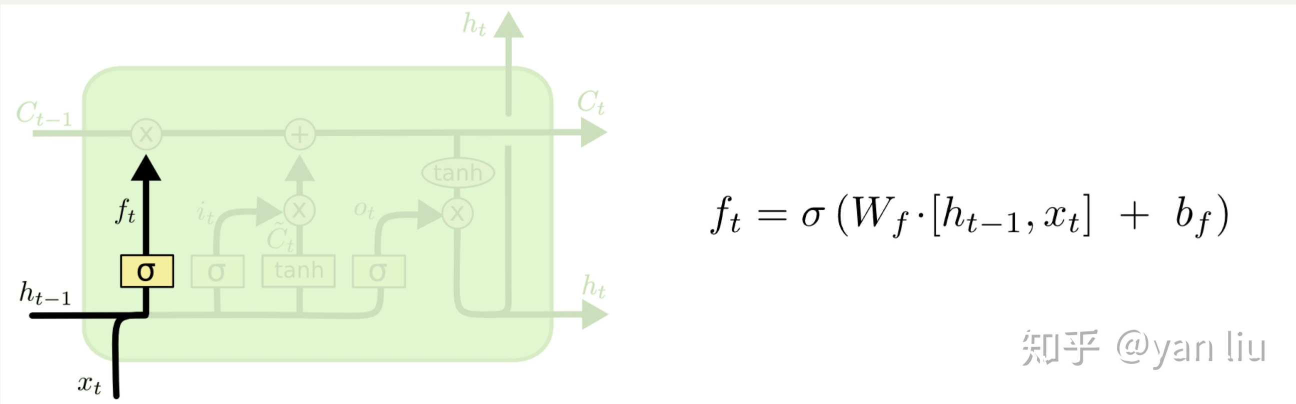


Forget gate

— f_t is the forget gate to decide how much information should be discarded

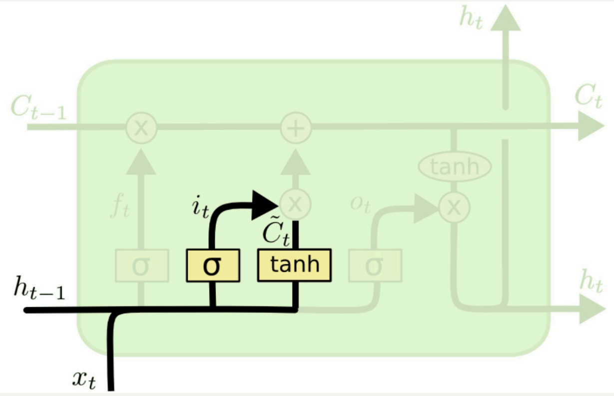
— h_{t-1} and x_t are inputs for f_t

— $sigmoid$ function to covert values in [0,1]: $\alpha(z) = \frac{1}{1+exp(-z)}$



Cell state

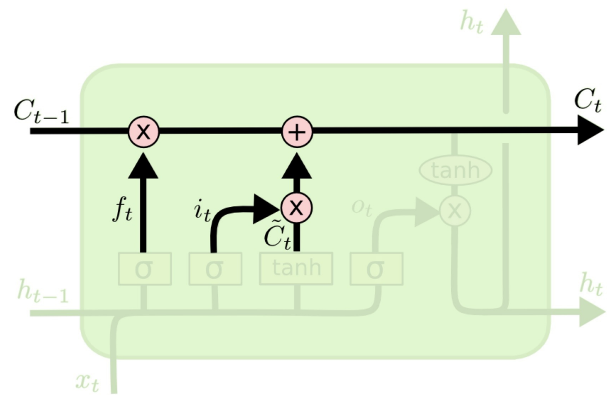
— $\tilde{c}_t$: the update of the cell state



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Input gate

— i_t : input gate to control how much indormation from $\tilde{c}_t$ will be used fo

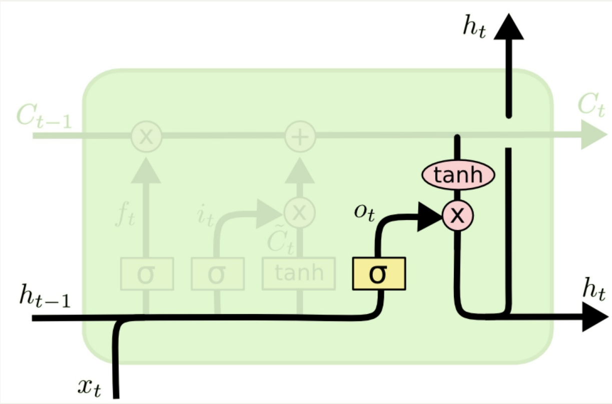


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Output gate

— calculate the output of the hidden state h_t

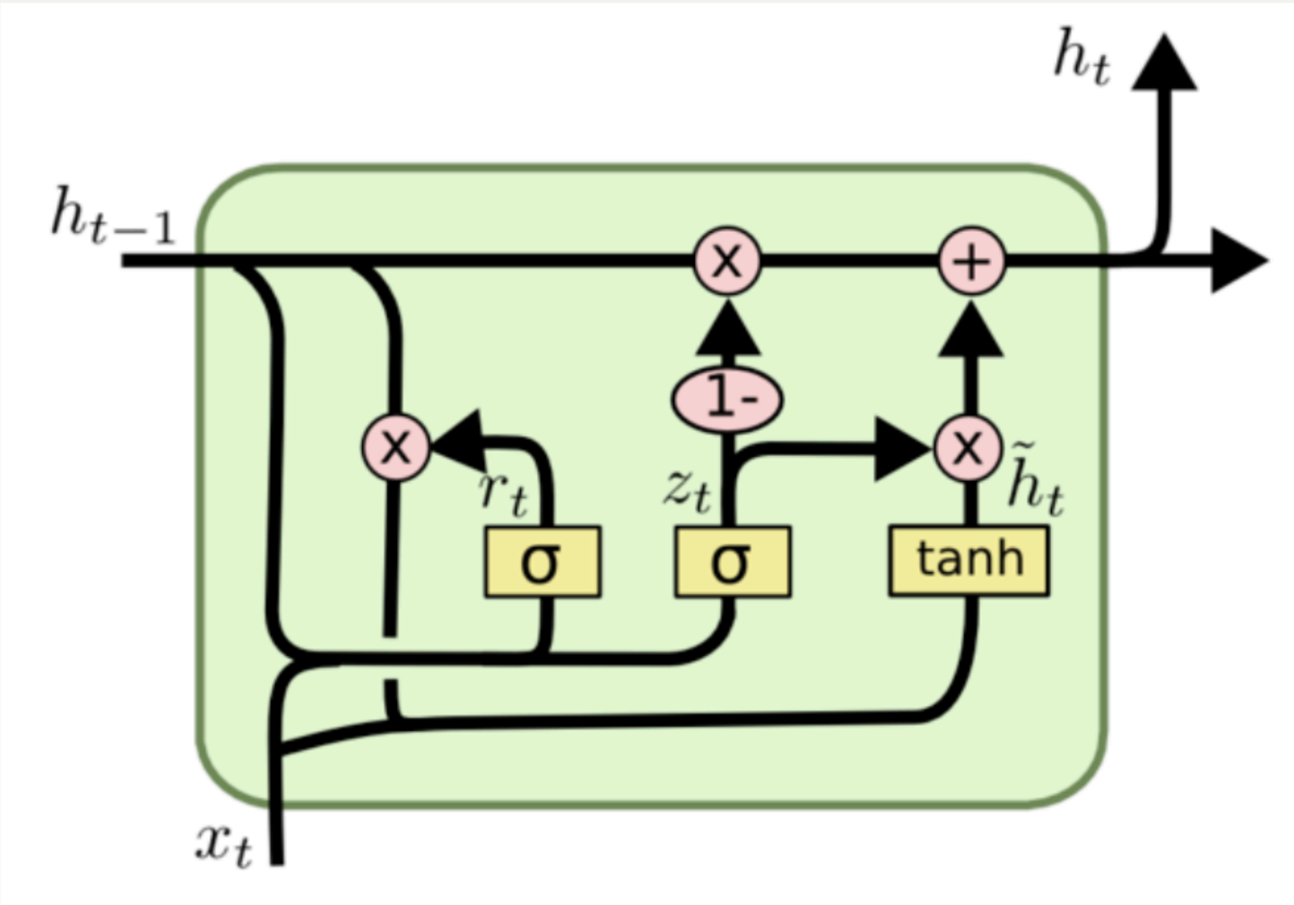
— output the y_t and the input for the next time step c_t



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Gated Recurrent Unit (GRU)

— A simlified LSTM cell,only have two gate: update gate(z_t) and reset gate(r_t)



Update gate:

1. controlling how much information from the last state will be carried

on in this state.
2. the larger the value of z_t is, the more information is carried on
3. useful to capture the long-term memory

Reset gate:

1. combining the forget gate and input gate
2. controlling how much information from the last state will be discard
3. the smaller the value of r_t is, the more information is discarded
4. useful to capture the shor-term memory

GRU Forward feed

- reset gate: $r_t = sigmoid(W_r * [h_{t-1}, x_t] + b_r)$
- update gate: $z_t = sigmoid(W_z * [h_{t-1}, x_t] + b_z)$
- candidate hidden state at current time step: $\tilde{h_t} = tanh(W_{\tilde{h_t}} * [r_t * h_{t-1}, x_t] + b_h)$
- hidden state at current time step: $h_t = z_t * h_{t-1} + (1 - z_t) * \tilde{h_t}$
- $y_t = sigmoid(W_o * h_t)$

GRU BPTT: Learning Parameter

$W_r = W_{rx} + W_{rh}$

$W_z = W_{zx} + W_{zh}$

$W_{\tilde{h}} = W_{\tilde{h}x} + W_{\tilde{h}h}$

Input for the output layer: $y_t^i = W_o h$

Output for the output layer:$y_t^o = sigmoid(y_t^i)$

The MSE loss at the t time step: $E_t = \frac{1}{2}(y_d - y_t^o)^2$, total: $E = \sum_{t=1}^{T} E_t$

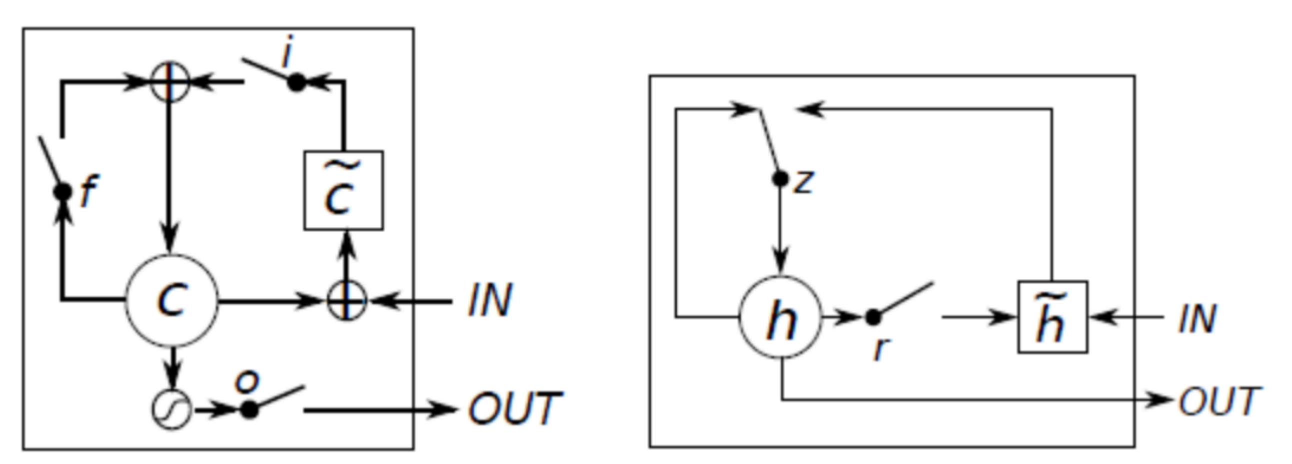
The error δ: $\frac{\alpha E}{\alpha W_o} = \delta_{y_t} h_t$

$W_r : \frac{\alpha E}{\alpha W_{zx}} = \delta_{z_t} x_t$, $\frac{\alpha E}{\alpha W_{zh}} = \delta_{z_t} h_{t-1}$

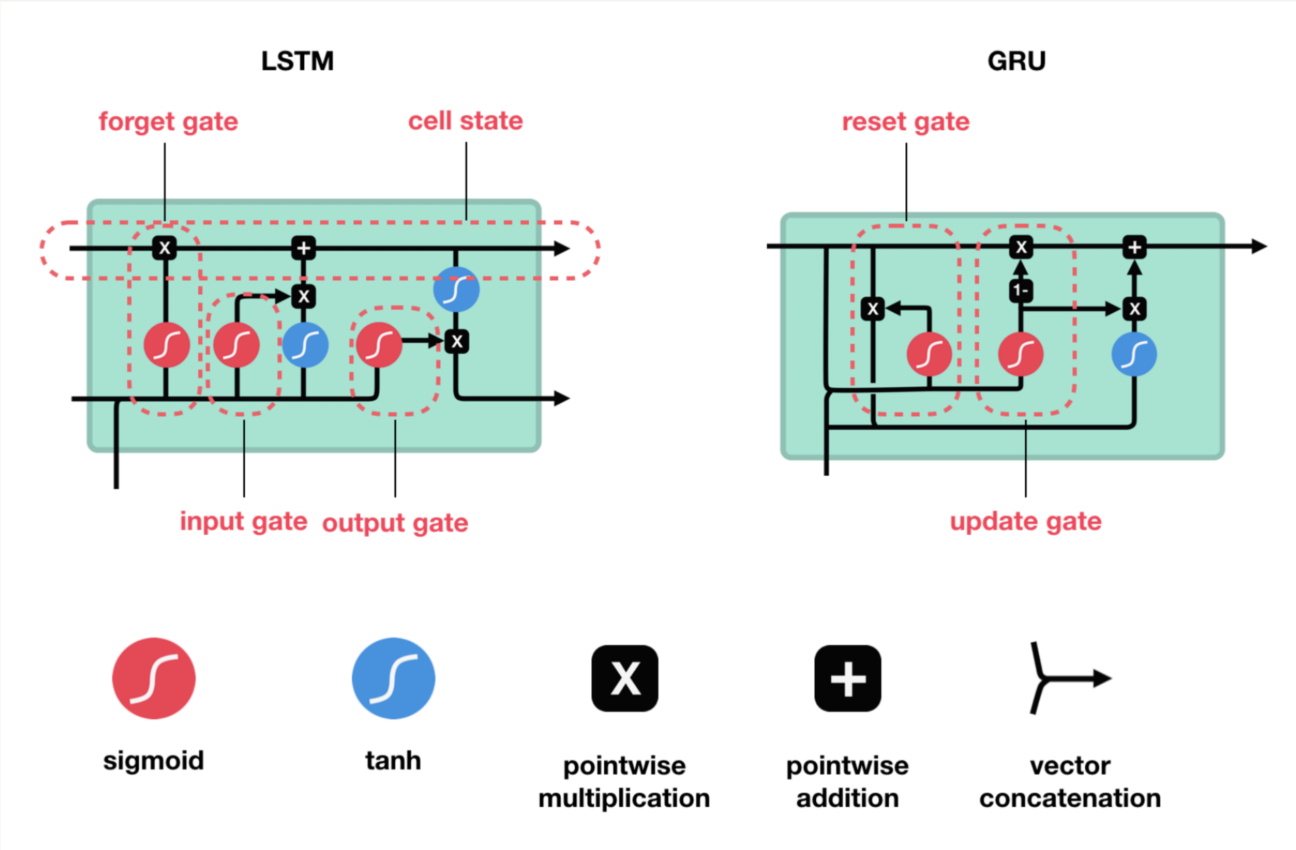
$W_z : \frac{\alpha E}{\alpha W_{\tilde{h}x}} = \delta_t x_t$, $\frac{\alpha E}{\alpha W_{\tilde{h}h}} = \delta_t (r_t, h_{t-1})$

$W_{\tilde{h}} : \frac{\alpha E}{\alpha W_{rx}} = \delta_{r_t} x_t$, $\frac{\alpha E}{\alpha W_{\tilde{h}h}} = \delta_{r_t} h_{t-1}$

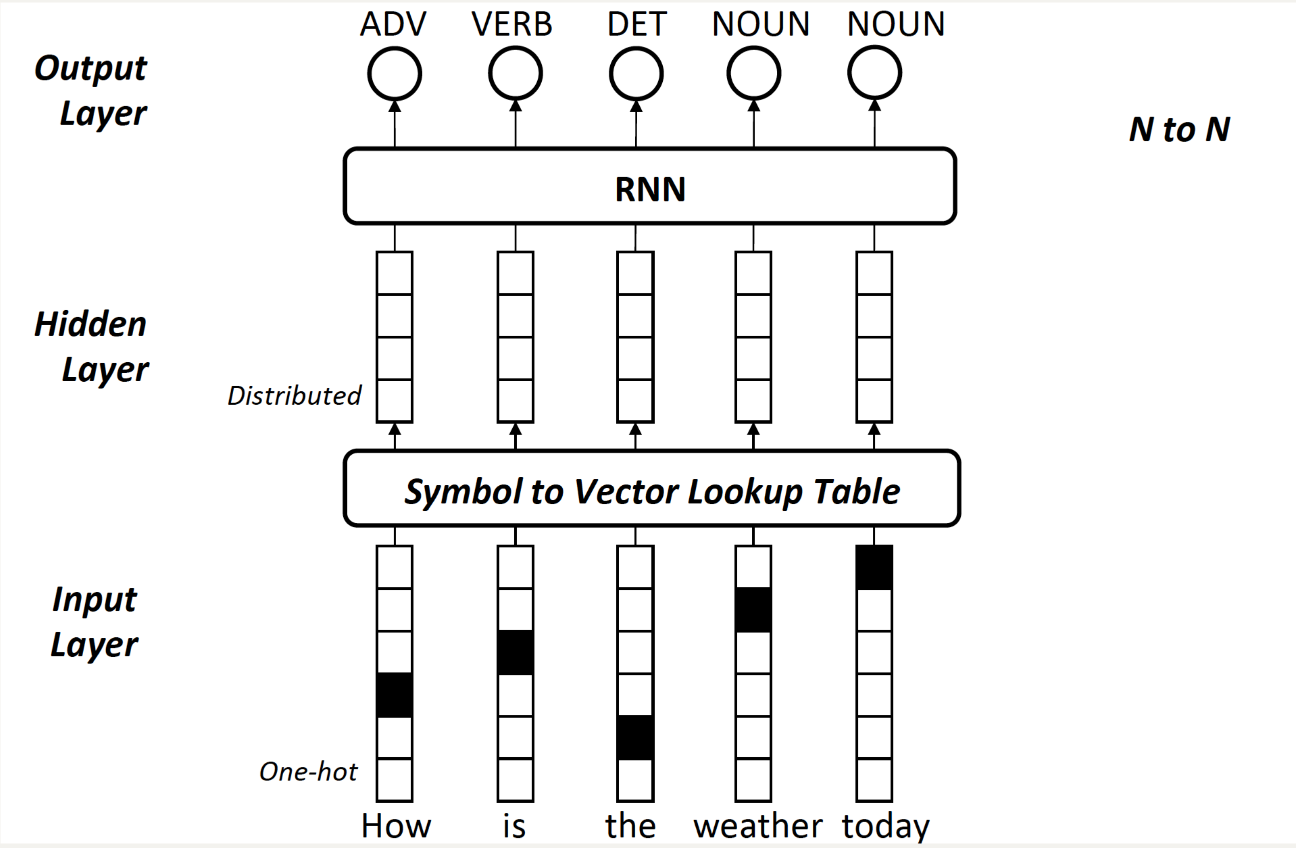
Comparison (LSTM vs GRU)



(a) Long Short-Term Memory (b) Gated Recurrent Unit



Application: POS Tagging



Seq2Seq Encoding and Decoding