Dealing with Vanishing Gradients in RNN

Tirtharaj Dash

Dept. of CS & IS and APPCAIR
BITS Pilani, Goa Campus

November 6, 2021

Where we are:

→ Forward computation in RNN

→ Backprop equations (BPTT)

Issues:

1. Exploding gradients → Easy.
   - Truncated BPTT
   - Gradient clipping.
     - by value
     - by Norm

2. Vanishing gradients

Vanishing gradient problem: (Hard)

→ Deal with it using

(Architectural change to RNN).

GRU

LSTM

RNN Unit : or RNN cell :

$$h_t = f(x_t, h_{t-1} ; W_h)$$

" $\hat{y}_t$ " $\quad o_t = g(h_t ; W_o)$

$$\downarrow$$

(tanh)

vectors $\qquad$ vector

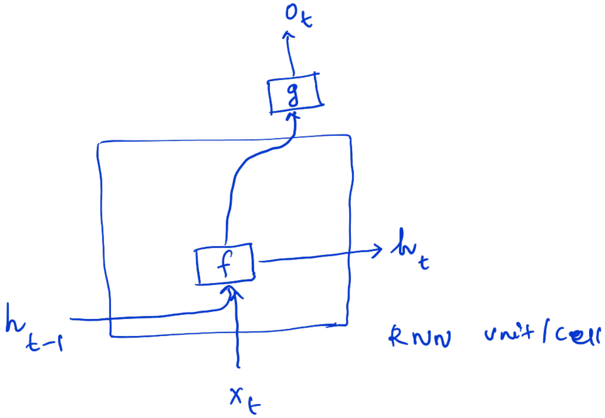
$$h_t = f\left(W_h [x_t, h_{t-1}] + b_h\right)$$

vector $\qquad$ act. fn.

$$o_t = g\left(W_o h_t + b_o\right)$$

vector $\qquad$ vector

softmax. $\quad$ matrices


o_t
g
f
h_t
h_{t-1}
x_t
RNN unit/cell


Vanishing grad.

Long-term dependencies (Long-range dependency)

"That $\boxed{\text{guy}}$, who we met at the airport ...., $\boxed{\text{was}}$ one of my students"

Singular.

guy — was

The model has to "remember"

Memory.

In RNN:

hidden state is updated.

hidden state overwritten.

"guy" — "was"

$t = 2$ $t = 35$

(GRU: Gated Recurrent Unit.)

GRU: (Basic) uses a memory cell (addition variable)

- relevance (reset) gates
- update gates.

gate

[1, 0] open / close

→ candidate memory

$$\tilde{c}_t = \tanh\left(W_c\,[c_{t-1}, x_t] + b_c\right)$$

→ Update gate

$$z_t = \sigma\left(W_u\,[c_{t-1}, x_t] + b_u\right)$$

→ Memory cell

$$c_t = z_t * \tilde{c}_t + (1 - z_t)\, c_{t-1}$$

hidden state

$$h_t = c_t$$

$$C_t = \underbrace{Z_t}_{\text{push}} * \underbrace{\tilde{C}_t}_{\text{new info}} + \underbrace{(1 - Z_t)}_{\text{forget relevance}} \underbrace{C_{t-1}}_{\text{old memory}}$$

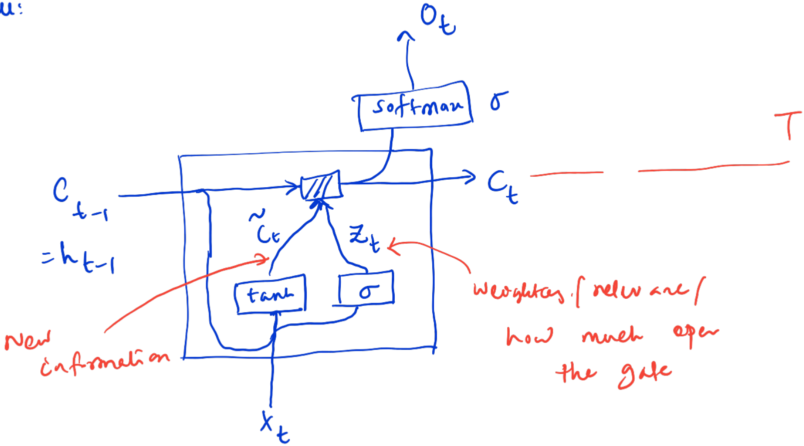
$\cancel{0.8}$ 0.99 $\qquad$ $\cancel{0.2}$ 0.01

$Z_t = \cancel{0.8}$ 0.99 $\quad Z_w = 0$

$C_t = 1 \quad C_t = \dots \qquad C_t =$

That $\boxed{\text{guy}}$, ... , $\boxed{\text{was}}$ one of . . .

GRU cell:



O_t
softmax
σ
T
C_{t-1}
$= h_{t-1}$
C_t
$\tilde{C}_t$
z_t
tanh
σ
x_t
New information
weightage / relevance / how much open the gate

The original GRU:

How relevant is c_{t-1} in computing c_t

$$\tilde{c}_t = \tanh(W_c [Z_r * c_{t-1}, x_t] + b_c)$$

Update $$Z_t = \sigma(W_u [x_t, h_{t-1}] + b_u)$$

relevance (reset) $$Z_r = \sigma(W_r [x_t, h_{t-1}] + b_r)$$

$$c_t = Z_t \tilde{c}_t + (1 - Z_t) c_{t-1}$$

$$h_t = c_t$$

LSTM: Long Short-Term Memory. (1991)

candidate memory $\tilde{C}_t = \tanh(W_c[x_t, h_{t-1}] + b_c)$

update $Z_u = \sigma(W_u[x_t, h_{t-1}, c_{t-1}] + b_u)$

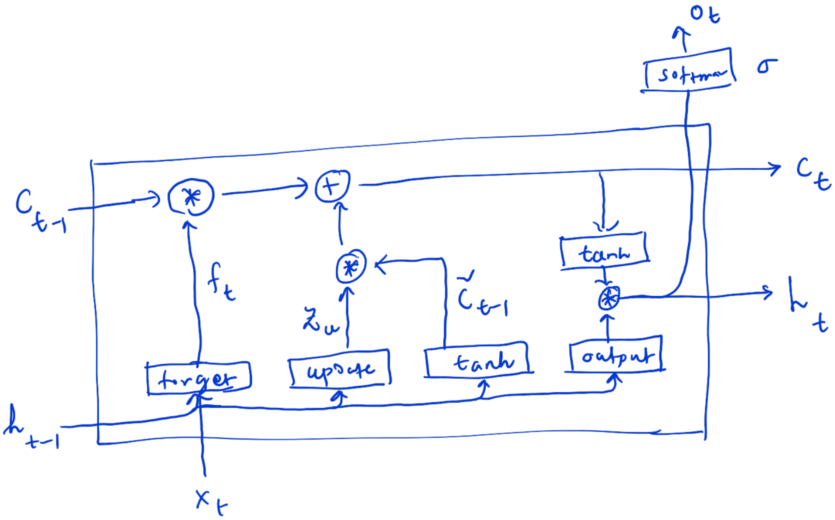
forget $Z_f = \sigma(W_f[x_t, h_{t-1}, c_{t-1}] + b_f)$

output $Z_o = \sigma(W_o[x_t, h_{t-1}, c_{t-1}] + b_o)$

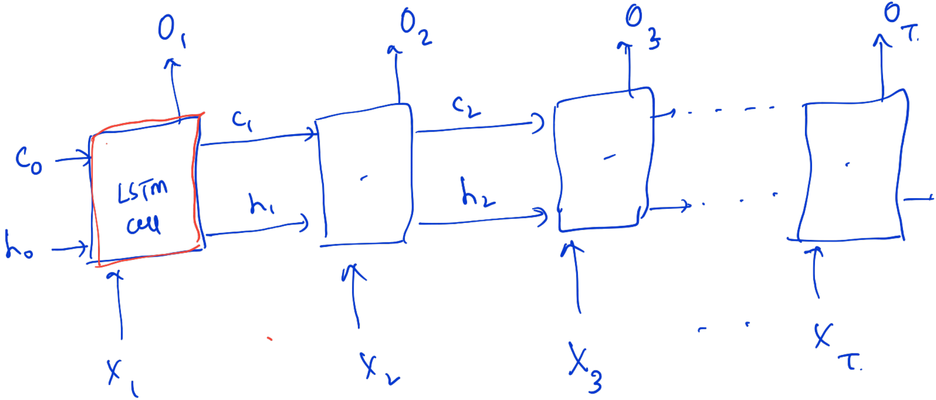
new memory $c_t = Z_u * \tilde{C}_t + Z_f * C_{t-1}$

hidden state $h_t = Z_o * \tanh(c_t)$

Peephole.



o_t
softmax
σ
C_{t-1}
C_t
f_t
z_u
$\tilde{C}_{t-1}$
tanh
h_t
forget
update
tanh
output
h_{t-1}
x_t



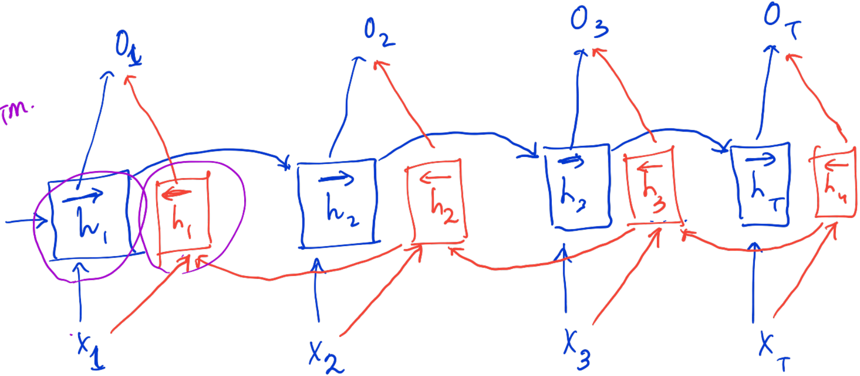
* GRU
* RNN

Read about:
Peephole connection.

Bidirectional RNNs

$$O_t = g(W_o[\overrightarrow{h}_t, \overleftarrow{h}_t] + b_o)$$

* RNN
* GRU
* LSTM.



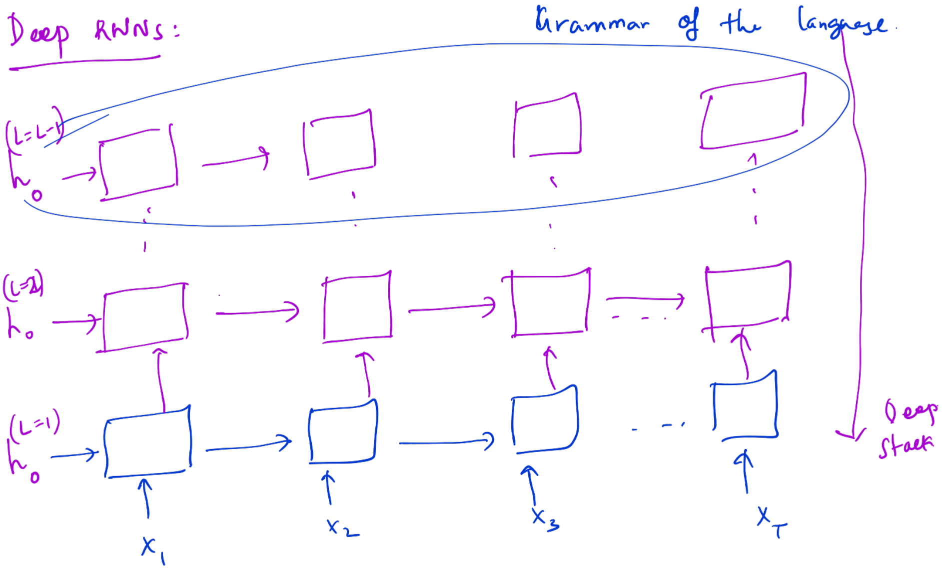
I think, "Deep learning is fun."

I think, "Deep sea is dangerous"

$\longrightarrow$ L to R.

$\longleftarrow$ R to

Deep RNNs:



Grammar of the language
(L=L-1)
h_0
(L=2)
h_0
(L=1)
h_0
x_1
x_2
x_3
x_T
Deep stack