

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.

② Vanishing gradients

Truncated
BPTT

Gradient
clipping.

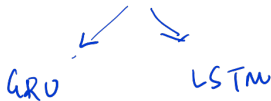
by value

by norm

Vanishing gradient problem: (Hard)

→ Deal with it using

(Architectural change to RNN)-

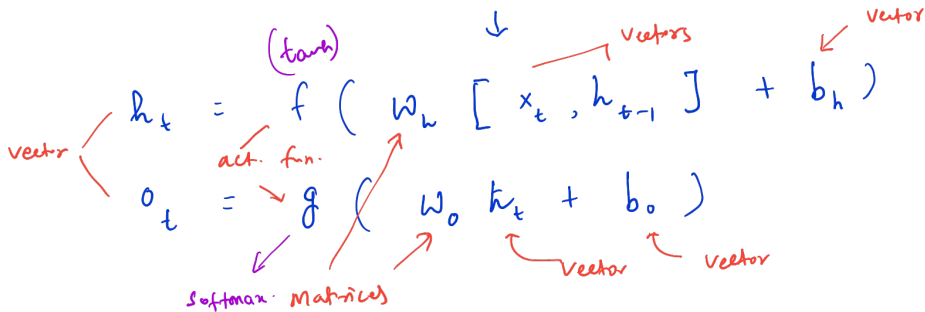


RNN Unit: or RNN cell:

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

" \hat{y}_t "

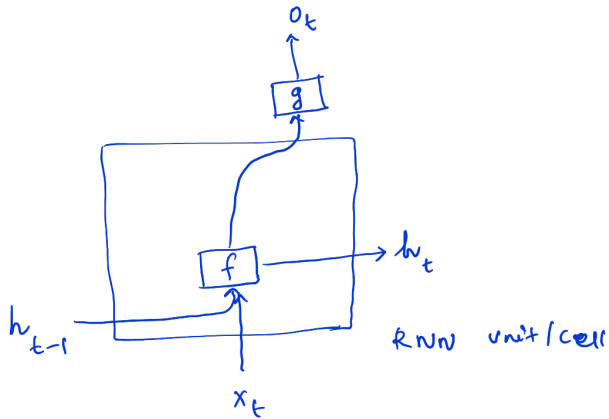
$$o_t = g(h_t; W_o)$$



The diagram shows the equations for an RNN unit with various annotations in red and purple:

- $h_t = f(x_t, h_{t-1}; W_h)$: The function f is labeled "(tanh)". The input x_t is labeled "vector". The weight matrix W_h is labeled "Softmax Matrices". The bias b_h is labeled "vector".
- $o_t = g(h_t; W_o)$: The function g is labeled "act. fun.". The weight matrix W_o is labeled "Softmax Matrices". The hidden state h_t is labeled "vector". The bias b_o is labeled "vector".

Red arrows point from the labels to the corresponding terms in the equations. A red arrow also points from the word "vectors" to the inputs x_t and h_{t-1} in the first equation.



RNN unit/cell



Long-term dependencies (Long-range dependency)

That guy, who we met at the airport ---, was one of my students "

Singular.

guy — was

The model has to "remember"
↓
memory.

In RNN:

hidden state is updated.

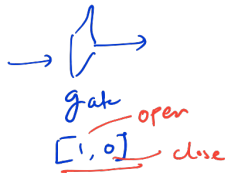
hidden state overwritten.
"guy" "was"
 $t=2$ $t=35$

(GRU: Gated Recurrent Unit)

GRU: uses a memory cell. (addition variable)

(Basic)

relevance (reset) gates
update gates.



→ candidate memory

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

→ update gate

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

→ Memory cell

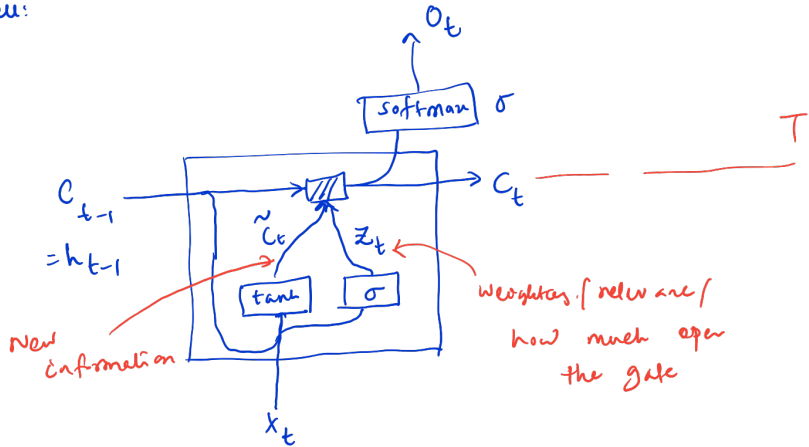
$$c_t = z_t * \tilde{c}_t + (1 - z_t) c_{t-1} \quad \left| \begin{array}{l} \text{hidden state} \\ h_t = c_t \end{array} \right.$$

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

$\begin{matrix} \nearrow 0.8 \\ \underline{\underline{0.8}} \end{matrix} \quad 0.99$
 $\nwarrow 0.99$
 $Z_t = 0.8 \quad \tilde{z}_w = 0$
 $C_t = 1 \quad C_t = \dots \quad C_t =$

That guy, was one of . . .

GRU cell:



The original GRU =

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

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

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

relevance (reset) $z_r = \sigma \left(w_r \left[x_t, h_{t-1} \right] + b_r \right)$

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

$$h_t = c_t$$

LSTM: Long Short-Term Memory. (1991)

candidate
memory

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

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

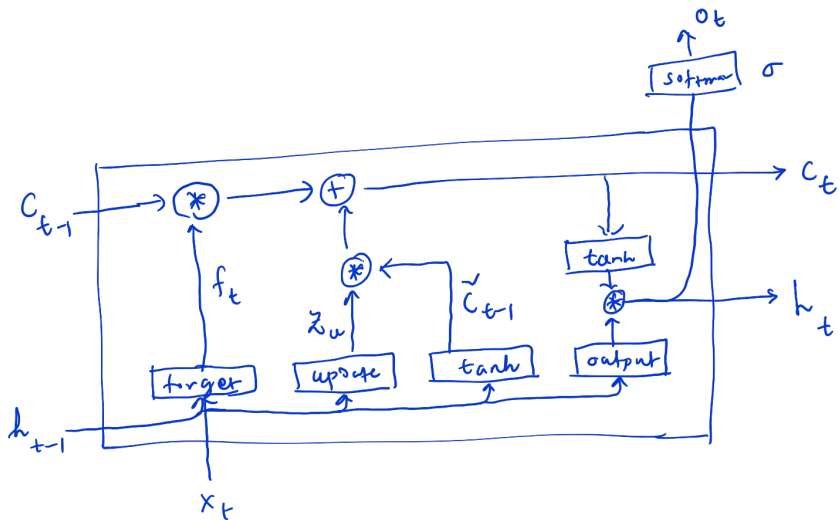
forget $Z_f = \sigma(w_f [x_t, h_{t-1}, c_{t-1}] + b_f)$

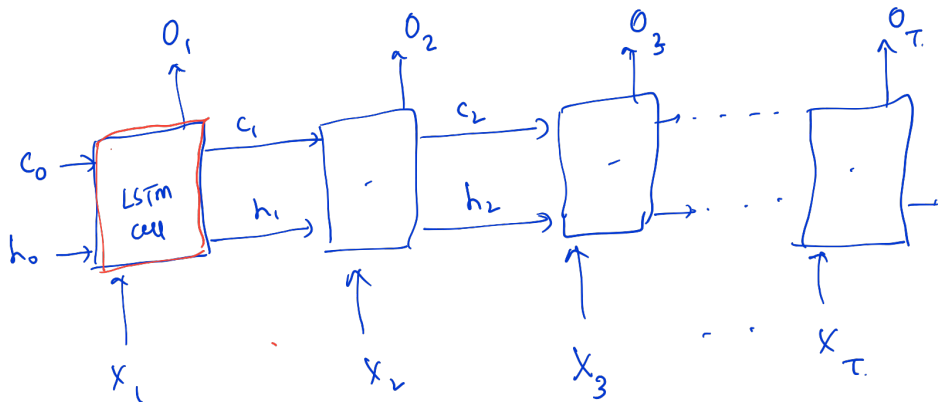
output $Z_o = \sigma(w_o [x_t, h_{t-1}, c_{t-1}] + b_o)$

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

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

perforator





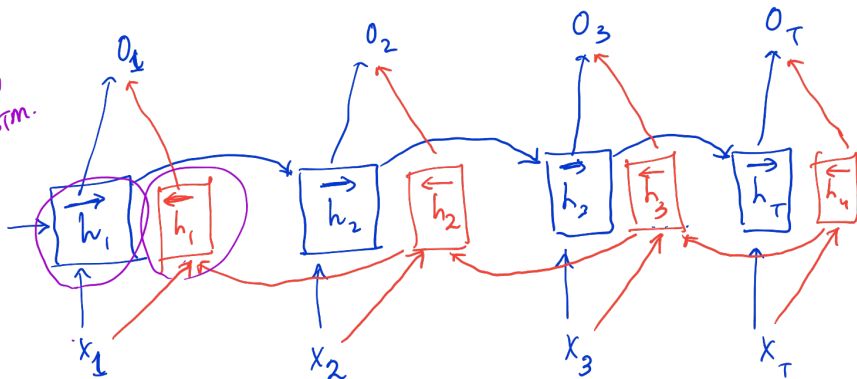
* GRU
* RNN

Read about:
peephole connection.

Bidirectional RNNs

$$o_t = g(w_o [\vec{h}_t, \overleftarrow{h}_t] + b_o)$$

* RNN
* GRU
* LSTM.



I think, "Deep learning is fun."

I think, "Deep sea is dangerous"

→ L to R.

← R to L.

Deep RNNs:

Grammar of the language.

