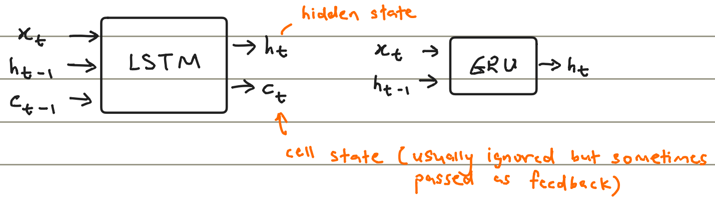
LSTM

- more complex than GRU
- more performant than GRU in recent research
- ML is experimentation, not theoreticizing
- LSTMs = GRU with more state vectors & gates



hidden state

cell state (usually ignored but sometimes passed as feedback)

* Need 2 initial states h_0 & c_0

LSTM Equations

$f_t = \sigma(W_{xf}{}^T x_t + W_{hf}{}^T h_{t-1} + \beta_f)$: forget gate

$i_t = \sigma(W_{xi}{}^T x_t + W_{hi}{}^T h_{t-1} + \beta_i)$: input / update gate

$o_t = \sigma(W_{xo}{}^T x_t + W_{ho}{}^T h_{t-1} + \beta_o)$: output gate

} Acts as binary classifier — Remember / forget

$c_t = \underline{f_t \odot c_{t-1}} + \underline{i_t} \odot \underline{f_c} \underline{(W_{xc}{}^T x_t + W_{hc}{}^T h_{t-1} + \beta_c)}$: cell state, similar concept to ht in GRU

$h_t = \underline{o_t} \odot f_h(c_t)$

Hidden state

how much of c_{t-1} to forget

Activate f_c

Simple RNN

how much of Simple RNN to remember / forget

Activation (typically tanh)

controls which values of cell state to pass through

(squashed output of cell state)

* In TensorFlow, cannot customize f_c & f_h Independently.

LSTM TensorFlow

input = Input (shape = (T, D))

x = LSTM (M) (input) , M = No. of hidden units

x = Dense (K) (x) , K = No. of output units

model = Model (input, x)

x = LSTM (M, return_sequences = True) (input) ← get $h_1, h_2 \ldots h_T$ to calculate $y_1, y_2 \ldots y_T$ eg: for language translation

output will be N x T x M

All h_t

No. of hidden units

x = Dense (K) (x)

output will be N x T x K

No. of output nodes

Options for RNN units

SimpleRNN (M, return_state = True)
GRU (M, return_state = True)
} output o & hidden state h are same thing. h_T returned

LSTM (M, return_state = True) Get both h_T & c_T

Cell State