LONG SHORT-TERM MEMORY NETWORKS II

Prof. Alexander Huth
4.21.2020

RECAP

* Recurrent Neural Networks (and other very deep neural networks) suffer from **vanishing & exploding gradients**

  * & severity grows with the length (depth) of the network

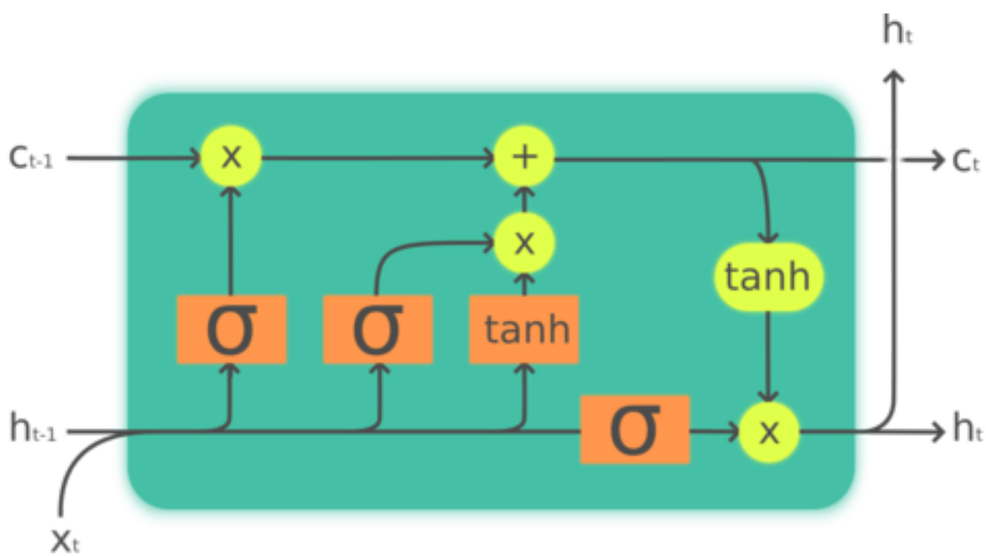
* This makes it difficult to learn very long time dependencies between input and output (e.g. how does x[t] affect y[t+n] when n is big?)

RECAP

* One solution is the **long short-term memory (LSTM)** network

* LSTMs avoid vanishing & exploding gradients by keeping a **cell state**, propagated across time without passing through nonlinearities or being multiplied by a weight matrix

RECAP



* LSTMs control whether information is **added, removed,** or **output** from the cell state using gates
    * **Input gate** controls how much info is added
    * **Forget gate** controls how much info is removed
    * **Output gate** controls how much info is output

TODAY

* Re-interpretation (and derivation) of LSTMs through the lens of “time warping”

* Follows “Can recurrent neural networks warp time?” by Tallec & Ollivier, ICLR 2018

TIME SCALING

* Suppose we train an RNN to do a very simple task: repeat the last input, e.g.
  x = [h e l l o t h e r e g]
  y = [_ h e l l o t h e r e]

* This is super easy! Even a simple RNN is able to solve this!

TIME SCALING

* Now suppose we trivially modify the task by **repeating** each input, e.g.

  x = [h h e e l l l l o o t t h h]

  y = [_ _ h h e e l l l l o o t t]

* This *should* also be super easy. It’s the same task!

* But if we keep increasing the number of repeats, it gets *REALLY HARD* for a simple RNN to solve this task

TIME SCALING

* What's the difference between the simple task ([h e l l o]) and the difficult task ([h h e e …])?
  * ***Time scaling!***
  * They are the **same input**, but with time passing at different rates

TIME SCALING

* We can define a **time warping function**:
  c(t) = floor(a * t), with 0 < a <= 1

* Now if a = 0.5,
  x[t] = [h e l l o], then
  x[c(t)] = [h h e e l l o o]

* Since the only difference between these inputs is the time scale, & simple RNNs can learn one but not the other, this means that **simple RNNs are not invariant to time scaling**

TIME SCALE INVARIANCE

* Now suppose we want to build an RNN that *__is__* invariant to time scaling

* To do this, we're going to rewrite our original RNN equation and switch from discrete time to continuous time

$$h_{t+1} = g(W_x x_t + W_h h_t + b)$$

--> $$\frac{dh(t)}{dt} = g(W_x x(t) + W_h h(t) + b) - h(t)$$

TIME SCALE INVARIANCE

* To account for time scaling, replace *t* with *at* and expand, giving an equivalent model:

$$\frac{dh(t)}{dt} = \boxed{a}g(W_x x(t) + W_h h(t) + b) - \boxed{a}h(t)$$

the derivative is scaled by a

TIME SCALE INVARIANCE

* Now if we convert this derivative back to a recurrence relation:

$$\frac{dh(t)}{dt} = ag(W_x x(t) + W_h h(t) + b) - ah(t)$$

--> $h_{t+1} = ag(W_x x_t + W_h h_t + b) + (1-a)h_t$

TIME SCALE INVARIANCE

$$h_{t+1} = a\,g(W_x x_t + W_h h_t + b) + (1-a)h_t$$

* This now what’s called a **leaky RNN**, where the new hidden state is a convex combination of the **normal RNN state update** and the **last hidden state**

* Here the parameter *a* controls how slowly or quickly time passes for the RNN

  * We can use this leaky RNN to solve our earlier task, as long as we know *a* (or can learn it)

VARIABLE TIME SCALING (AKA TIME WARPING)

* Let’s make our original task more complicated by repeating each element a **random** number of times, e.g.

  x = [h h e e e l l l o t t t t]

  y = [_ _ h h h e e l l o o o o]

* We can still describe this using a **time warping function** c(t), just a more complicated one than before

VARIABLE TIME SCALING (AKA TIME WARPING)

* We can generalize our earlier equation for the time derivative of h, replacing *a* with dc(t)/dt:

$$\frac{dh(t)}{dt} = ag(W_x x(t) + W_h h(t) + b) - ah(t)$$

$$\frac{dh(t)}{dt} = \frac{dc(t)}{dt} g(W_x x(t) + W_h h(t) + b) - \frac{dc(t)}{dt} h(t)$$

VARIABLE TIME SCALING (AKA TIME WARPING)

* So how do we fit this kind of thing?

* Let's replace the derivative dc(t)/dt with a learnable function r(t) aka r_t

$$\frac{dh(t)}{dt} = r(t)g(W_x x(t) + W_h h(t) + b) - r(t)h(t)$$

$$h_{t+1} = r_t g(W_x x_t + W_h h_t + b) + (1 - r_t)h_t$$

VARIABLE TIME SCALING (AKA TIME WARPING)

* This is now a simple gated recurrent network; recall the LSTM cell state eq.

$$c_t = \boxed{f_t} \circ \boxed{c_{t-1}} + i_t \circ \tilde{c}_t$$

prev. state

forget gate

$$h_{t+1} = r_t g(W_x x_t + W_h h_t + b) + \boxed{(1 - r_t)}\boxed{h_t}$$

VARIABLE TIME SCALING (AKA TIME WARPING)

* This is now a simple gated recurrent network; recall the LSTM cell state eq.

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$

input gate

proposed state

$$h_{t+1} = r_t g(W_x x_t + W_h h_t + b) + (1 - r_t) h_t$$

VARIABLE TIME SCALING (AKA TIME WARPING)

* What form should the learnable function r(t) take? One nice option would be to make it an RNN itself,

$$r_t = \sigma(W_{rx}x_t + W_{rh}h_t^r + b_r)$$

* We use a sigmoid here to ensure that $0 \le r_t \le 1$

* We can also set $h^r_t = h_t$, making r dependent upon the main hidden state

VARIABLE TIME SCALING (AKA TIME WARPING)

- Now we have an update equation that looks a *lot* like the LSTM cell state (albeit with tied input & forget gates)

  - & the “input gate” r_t looks a *lot* like the LSTM input gate

- Thus, with some margin of error, we have re-derived the LSTM (or gated RNN) from scratch

- **LSTMs are RNNs that have learned that time can be warped**

BIAS CONTROLS THE TIMESCALE

* r_t is the rate at which time is passing at time t (similar to our *a* from before)

* We can interpret $1/r_t$ as the **forgetting time** or **time constant** of the network: how many time steps does it take until the equivalent of 1 un-warped time step has passed?

BIAS CONTROLS THE TIMESCALE

* Going back to an earlier example of repeating each element a **random** number of times, e.g.
  x = [h h e e e l l l o t t t t]
  y = [_ _ h h h e e l l o o o o]

* Here $1/r_t$ should be roughly the number of times each element is repeated

BIAS CONTROLS THE TIMESCALE

* x = [h h e e e l l l o t t t t]
  y = [_ _ h h h e e l l o o o o]

* Suppose we know (or can reasonably guess) that each element is repeated ~50 times

* How can we tell the network this information?

BIAS CONTROLS THE TIMESCALE

* We want $1/r_t \approx 50$, where

$$r_t = \sigma(W_{rx}x_t + W_{rh}h_t^r + b_r)$$

BIAS CONTROLS THE TIMESCALE

* We want $1/r_t \approx 50$, where

$$r_t = \sigma(W_{rx}x_t + W_{rh}h_t^r + \boxed{b_r})$$

* One way that we can tell the network this information is by adjusting the **bias value** b_r for our “input gate” r_t

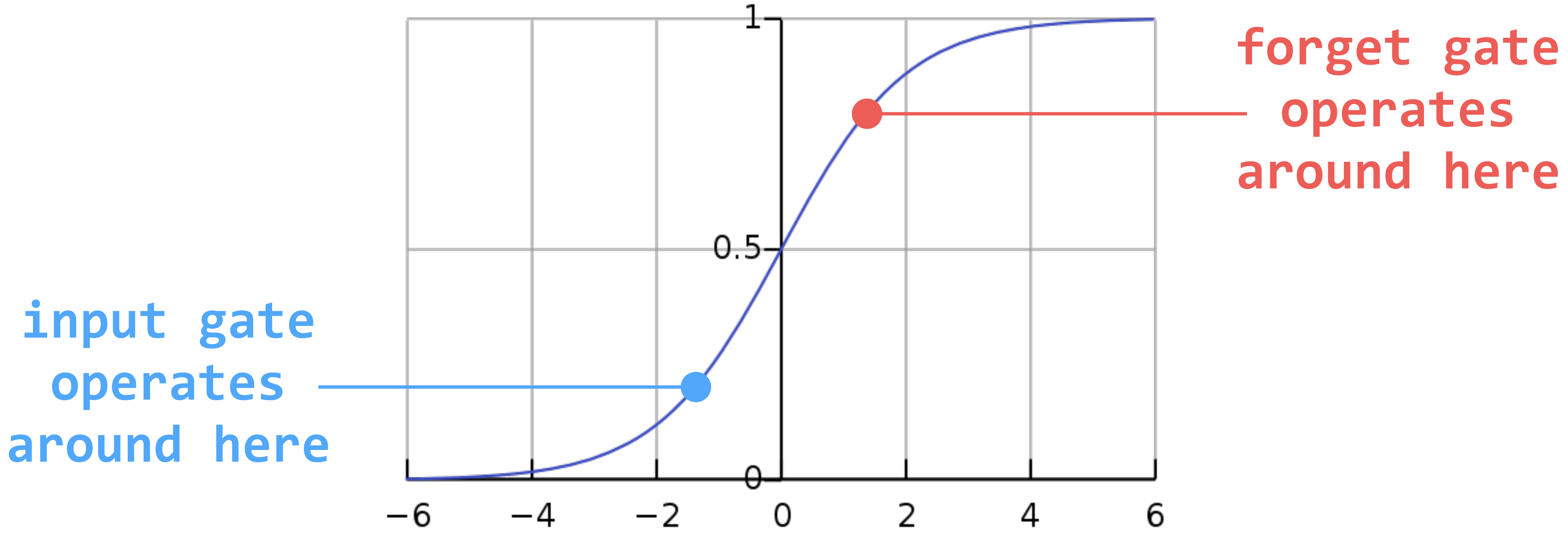
BIAS CONTROLS THE TIMESCALE

* So if we want $r_t \approx 0.02$ for the average input & hidden state (i.e. x = h = 0), then:

$$
\begin{aligned}
r_t &= \sigma(b_r) \\
b_r &= \sigma^{-1}(r_t) \\
b_r &= -\log(r_t^{-1} - 1) \approx -1.69
\end{aligned}
$$

BIAS CONTROLS THE TIMESCALE

* This gives us a good value for the bias on the input gate. By similar arguments the bias on the **forget gate** is simply the negative, $b_f = 1.69$



BIAS CONTROLS THE TIMESCALE

* We can use this mechanism to either **initialize** or **fix** the input & forget gate biases so that the network learns input-output relationships at specific timescales!

BIAS CONTROLS THE TIMESCALE

* For example, in a **language model** that is trained to predict the next word from context, it is important to consider information at many different timescales

* *When she tried to print her tickets, she found that the printer was out of toner. She went to the stationery store to buy more toner. It was very overpriced. After installing the toner into the printer, she finally printed her ________*

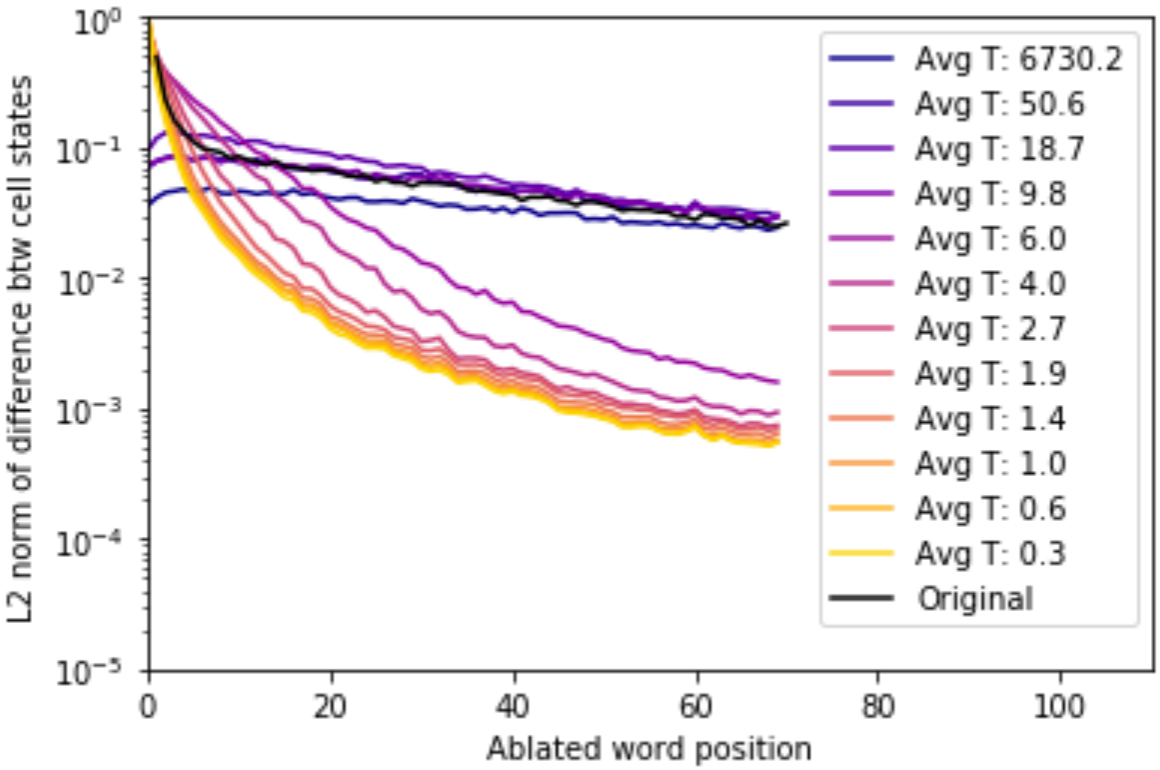
BIAS CONTROLS THE TIMESCALE

* Suppose we set input & forget biases for different units in the network so they correspond to different time scales

* Then measure how much the cell state of each unit at time t is affected by the input at times [t-1, t-2, …, t-n]

  * This is done by replacing one of the input words by a null token (“ablating” an input word) and then comparing states

BIAS CONTROLS THE TIMESCALE



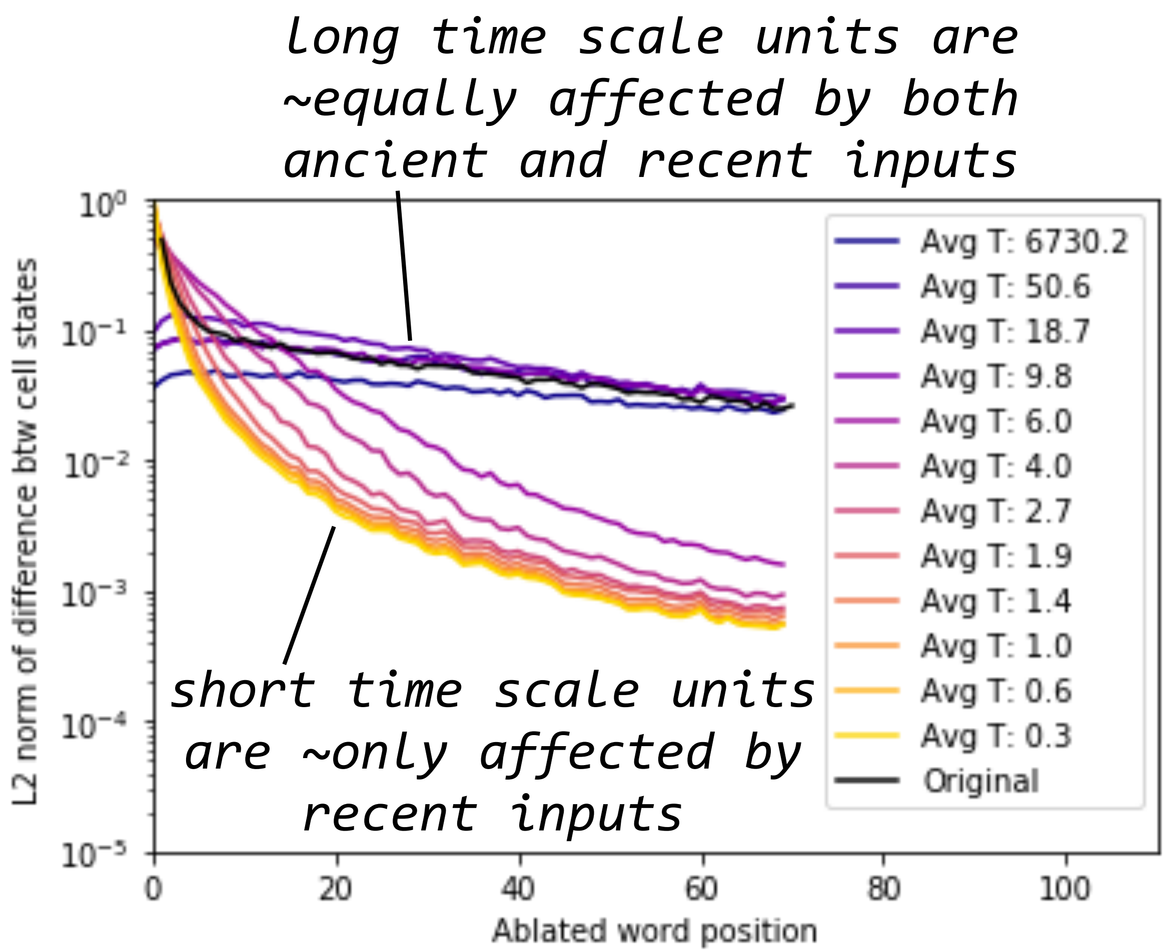
long

different time scales

short

h/t Shivangi Mahto

BIAS CONTROLS THE TIMESCALE



long

different time scales

short

h/t Shivangi Mahto

NEXT TIME

* Interpreting artificial neural network models!