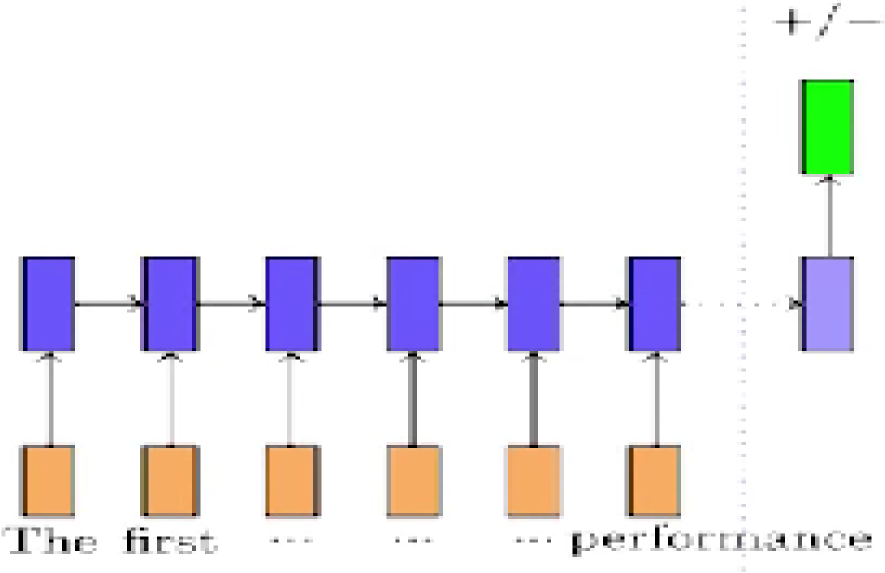
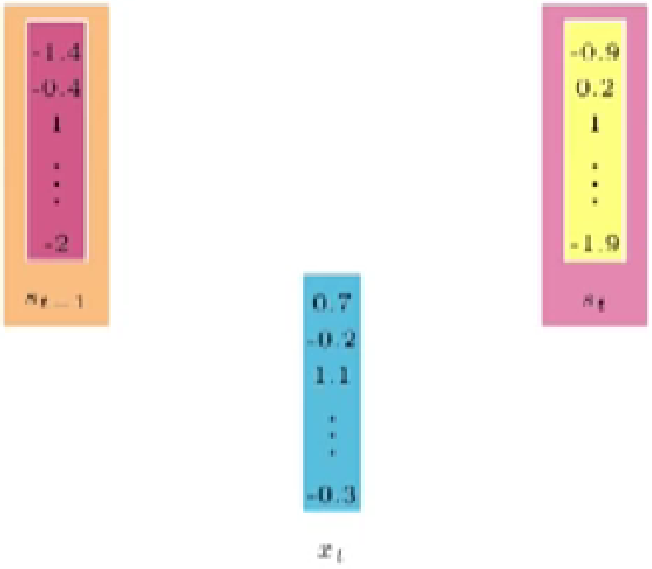
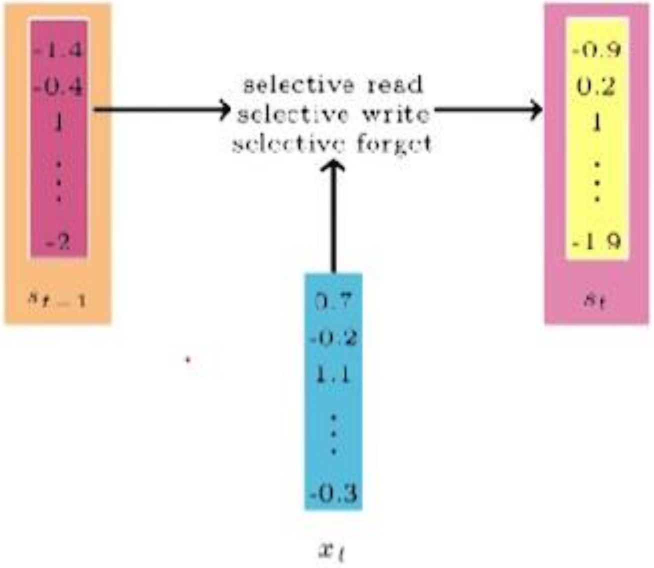
LSTM and GRU

Can we give a concrete example where RNNs also need to selectively read, write and forget?

Consider the task of predicting the sentiment (+/-ve) of a review.



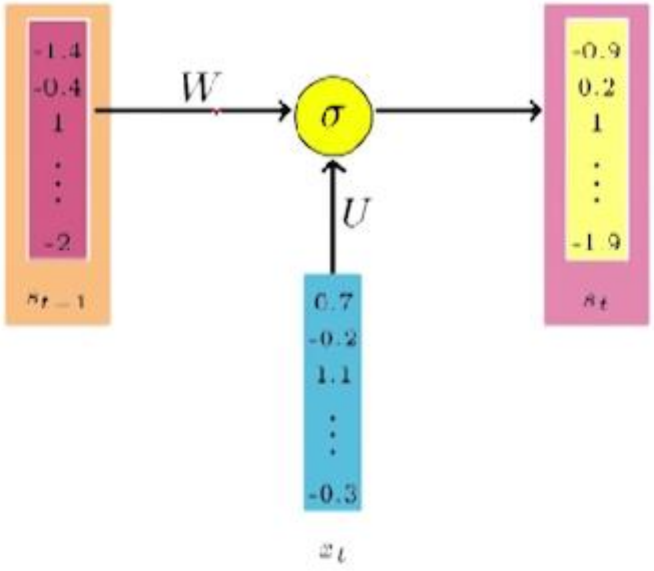




- State S_{t-1} at timestep t-1 and now we want to overload it with new information (x_t) and compute a new state (s_t).
- While doing so we want to make sure that we use :
  - Selective write
  - Selective read
  - Selective forget

  So that only important information is retained in s_t.

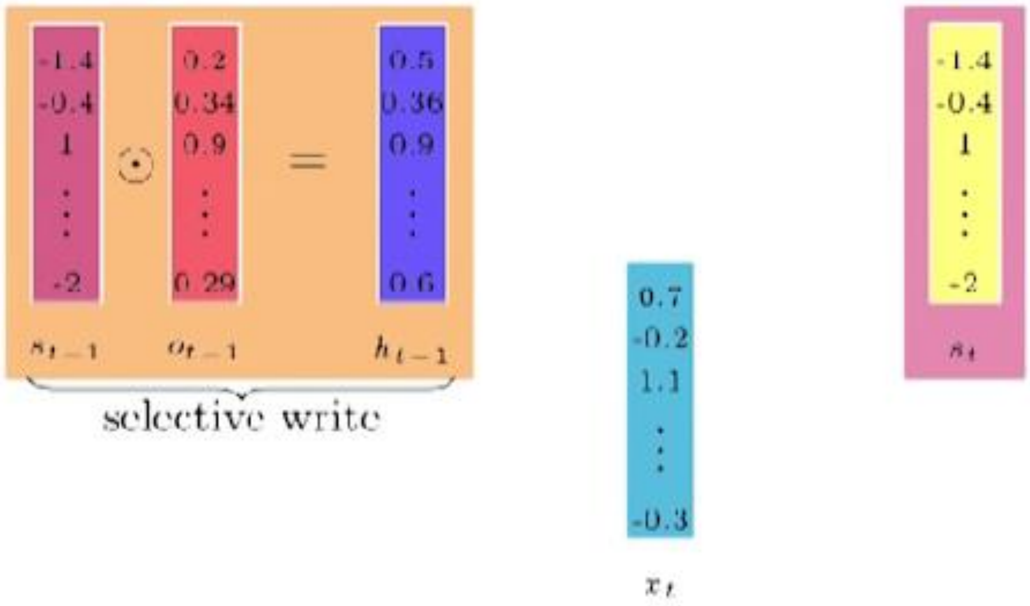
Selective Write



- Recall that in RNN we use S_{t-1} to compute s_t.

$$s_t = \sigma(W s_{t-1} + U x_t)$$

- Instead of passing S_{t-1} as it is, pass only some portions of it to next state.
- We can use binary decision (where we retain some entries and delete the rest of the entries)
- But a more sensible way would be to select a fraction of information of each entry in the current state and pass that info to the next state.



Use a vector O_{t-1} which decides what fraction of each element of S_{t-1} should be passed to the next state.

Each element of O_{t-1} gets multiplied with the corresponding element of S_{t-1}.

Each element of O_{t-1} is restricted to be between 0 and 1.

The RNN has to learn O_{t-1} along with the other parameters (W,U,V).
Compute Ot-1 and ht-1 as

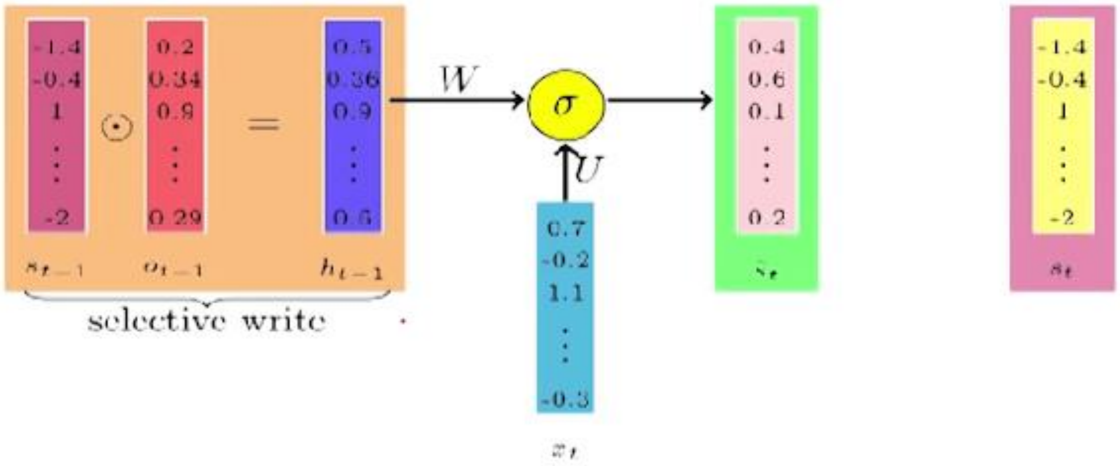
$$o_{t-1} = \sigma(W_o h_{t-2} + U_o x_{t-1} + b_o)$$
$$h_{t-1} = o_{t-1} \odot o_{t-1}$$

The parameter W_0, V_0, b_0 need to be learned along with the existing parameters W,U,V.

The sigmoid function ensures that the values are between 0 and 1.

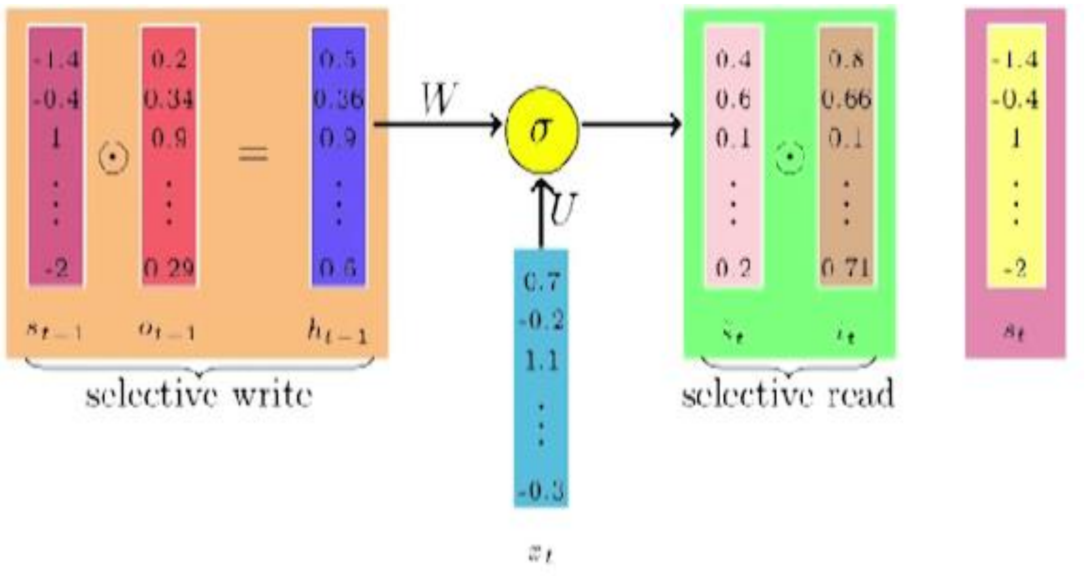
O_t is called the output gate as it decides how much to write to the next time step.

Selective Read



- H_{t-1} is used to compute the new state at the next time step.
- Also use x_t which is the new input at time step t:

$$\tilde{s}_t = \sigma(Wh_{t-1} + Ux_t + b)$$



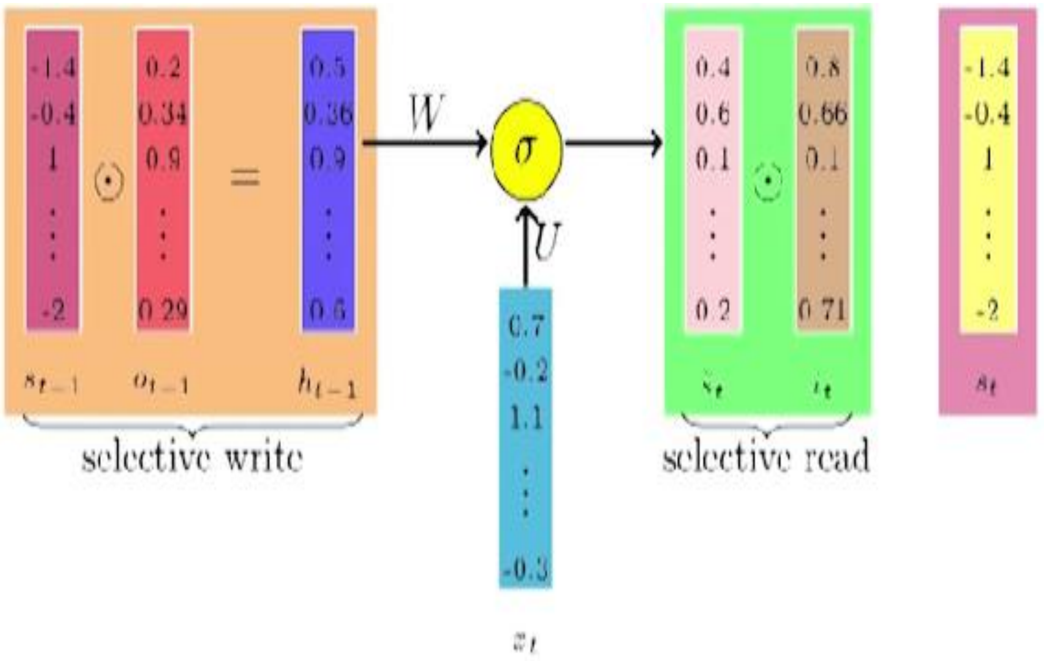
- $S_t^{\sim}$ captures all information from the previous state (H_{t-1}) and the current input x_t.
- However we don't need all this new information and only selectively read from it before constructing the new cell state s_t.

To do this, introduce another gate called the input gate:

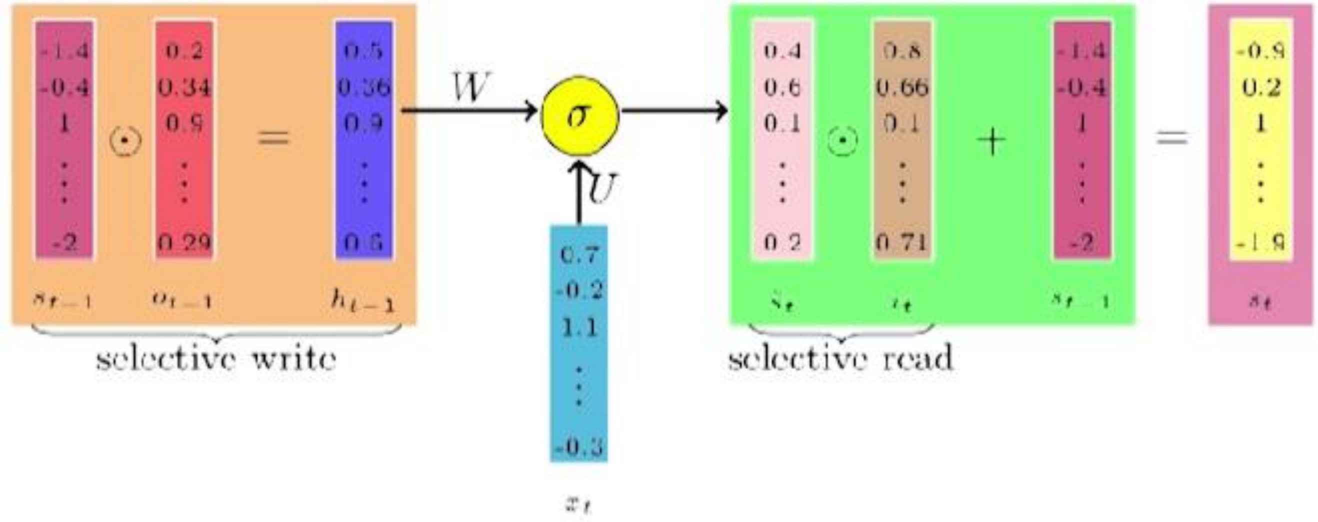
$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$

And use $i_t \odot s_t^{\sim}$ as the selectively read state information.

Selective Forget

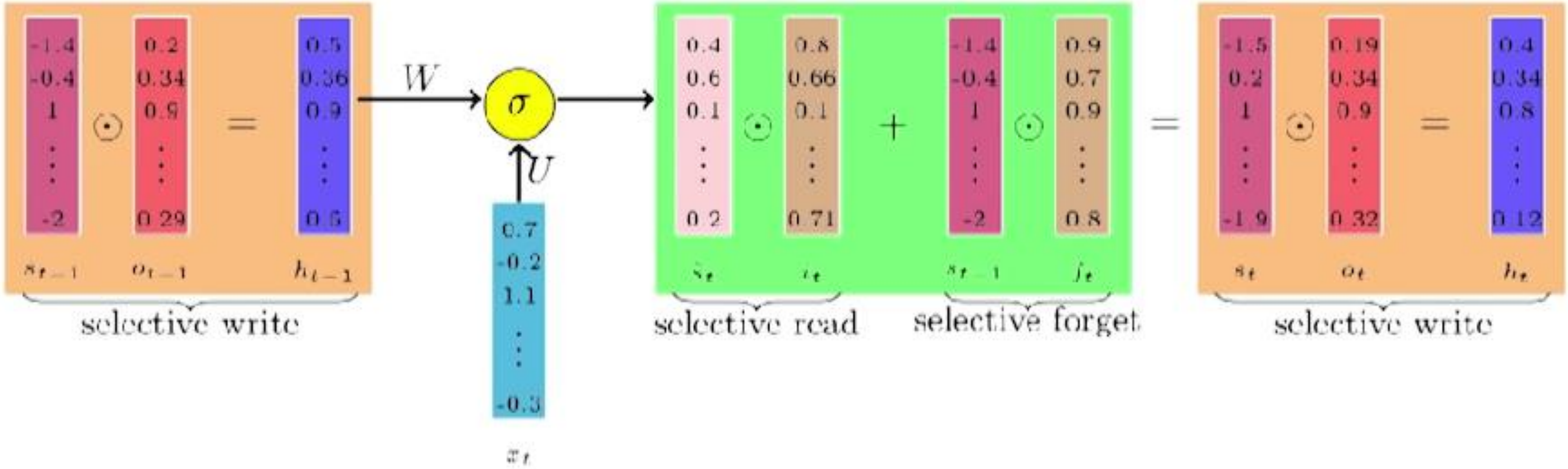


- How do we combine S_{t-1} and $\tilde{S_t}$ to get the new state?



We don’t want to use the whole of st-1 but forget some parts of it.
To do this use forget gate

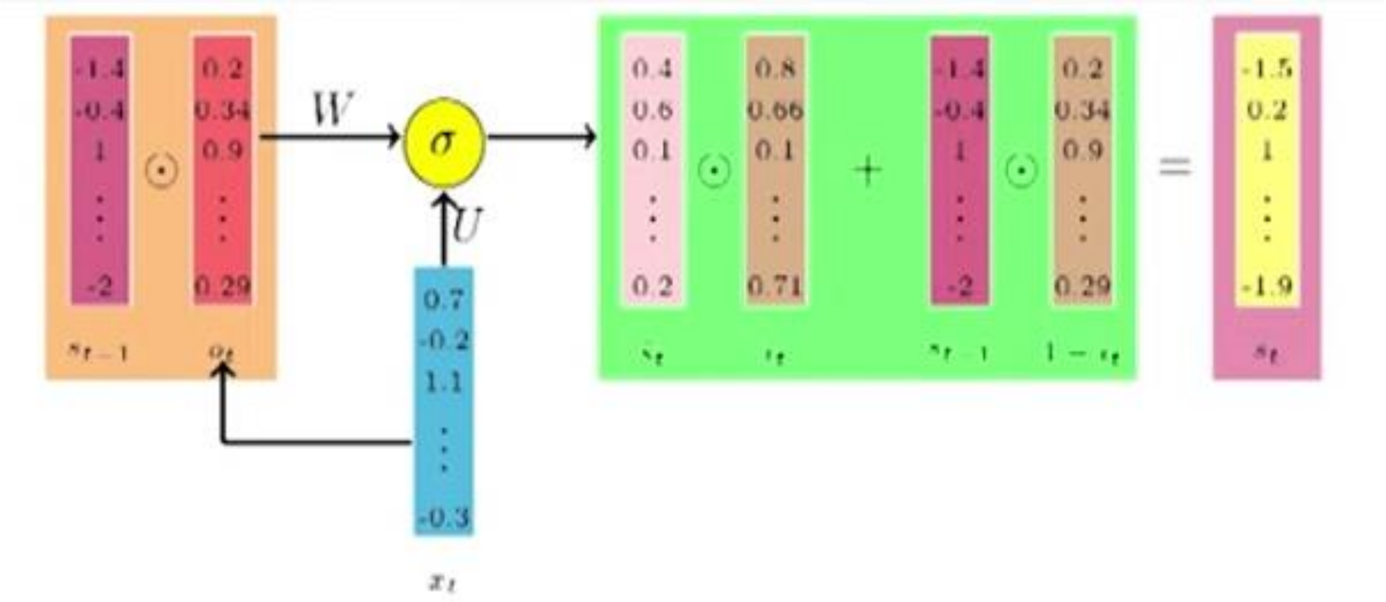
$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$
$$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s}_t$$



$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$
$$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s}_t$$

Note:

- LSTM has many variants which include different number of gates and also different arrangement of gates.
- Another equally popular variant of LSTM is Gated Recurrent Unit.



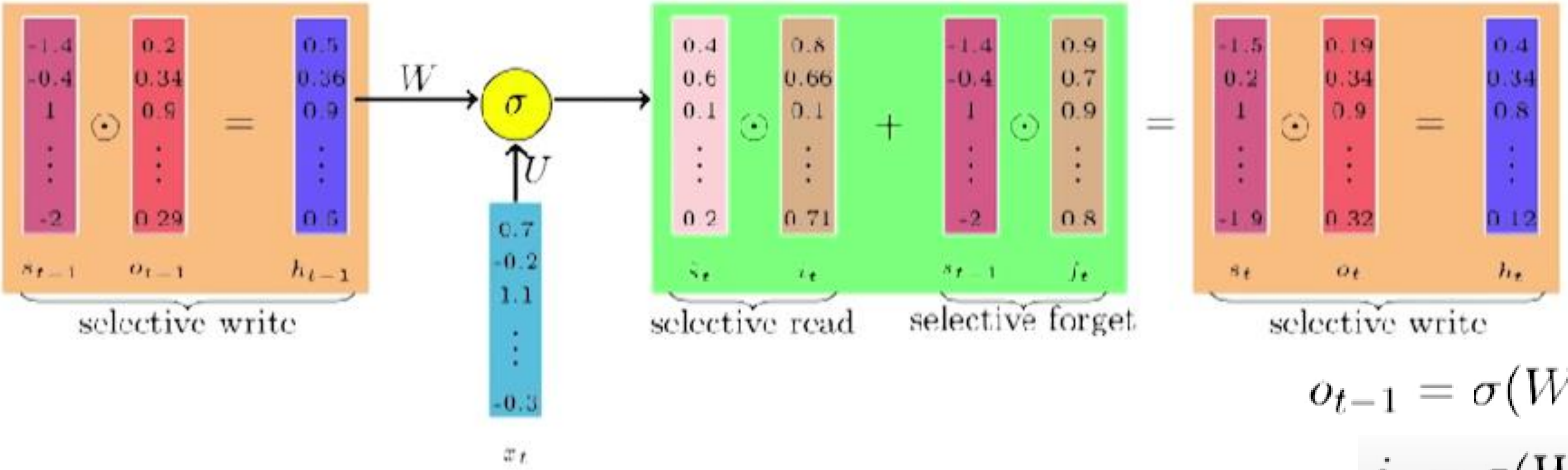
Gates:

$$o_t = \sigma(W_o s_{t-1} + U_o x_t + b_o)$$
$$i_t = \sigma(W_i s_{t-1} + U_i x_t + b_i)$$

States:

$$\tilde{s}_t = \sigma(W(o_t \odot s_{t-1}) + U x_t + b)$$
$$s_t = (1 - i_t) \odot s_{t-1} + i_t \odot \tilde{s}_t$$

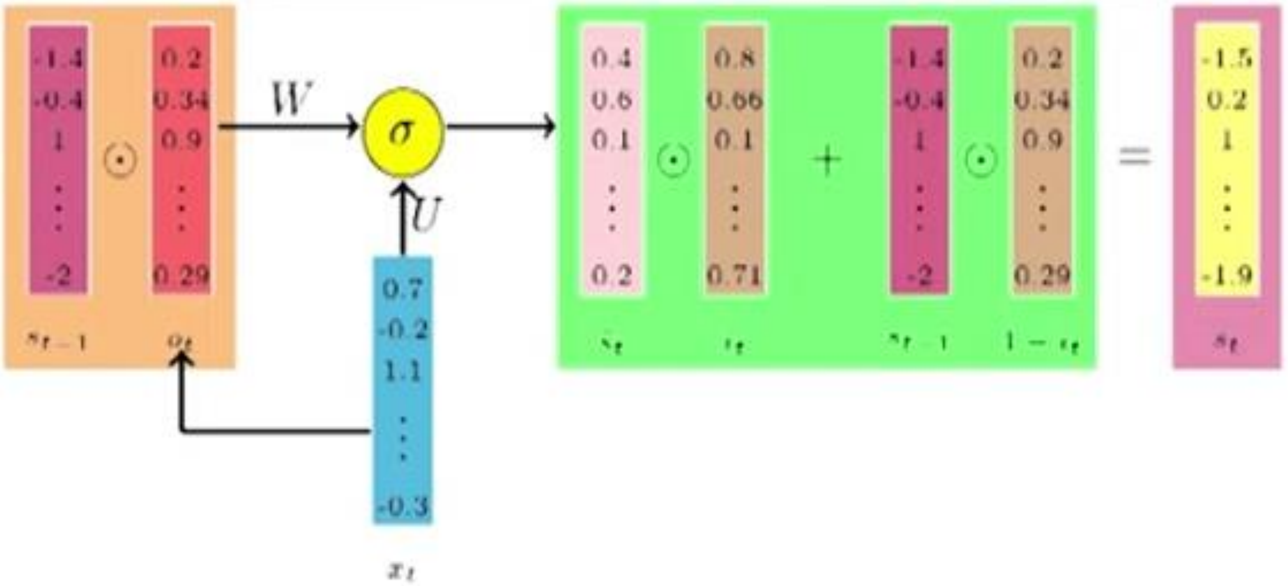
LSTM



$$o_{t-1} = \sigma(W_o h_{t-2} + U_o x_{t-1} + b_o)$$

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$

GRU

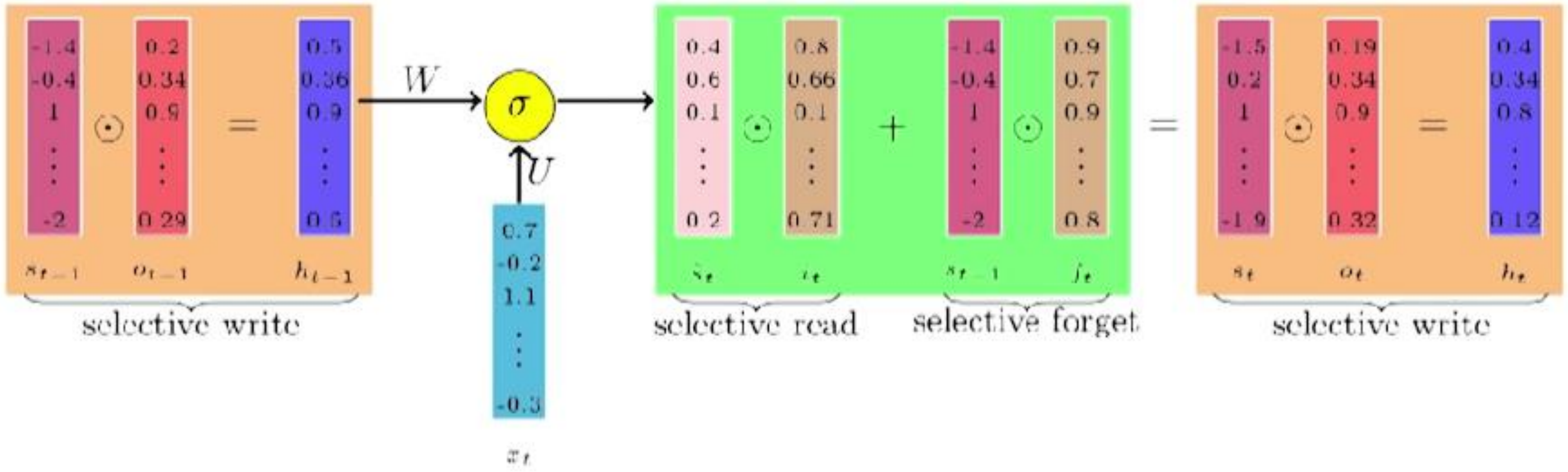


$$o_t = \sigma(W_o s_{t-1} + U_o x_t + b_o)$$

$$i_t = \sigma(W_i s_{t-1} + U_i x_t + b_i)$$

How LSTMs avoid the problem of vanishing gradients?

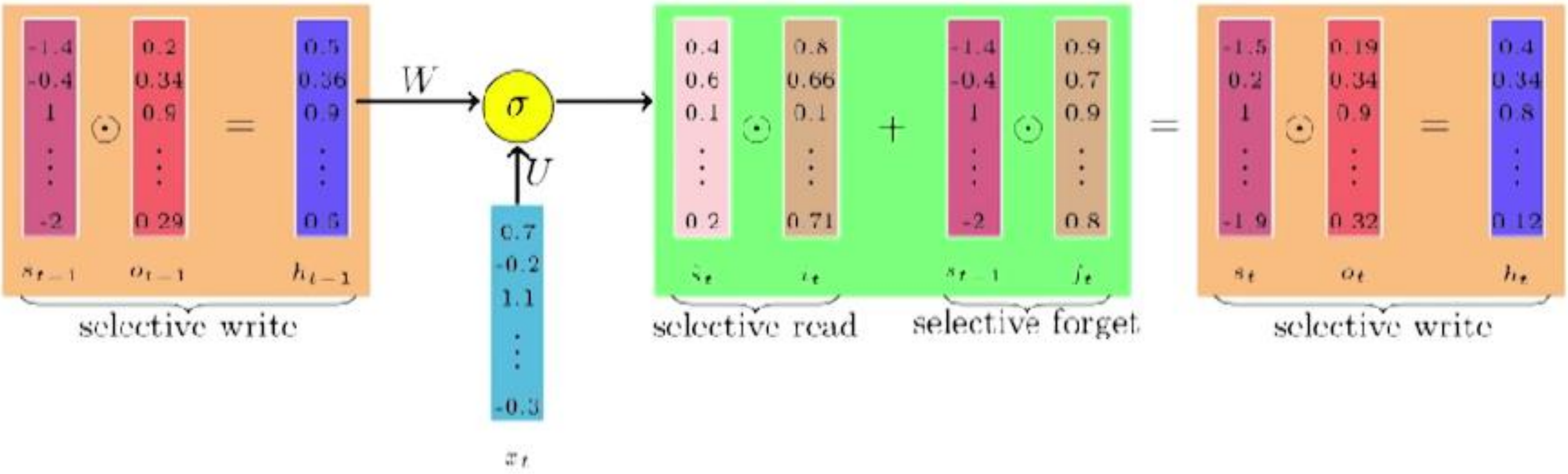
- In RNN, the parameter W will either blow up or vanish.
- Do we have recurrent connection in LSTM? ------Yes
- So, LSTM can have vanishing gradient and exploding gradient problem.
- How does LSTM tackle this problem?
- Exploding gradient can be handled by using gradient clipping,.
- While backpropagating if the norm of the gradient exceeds a certain value, it is scaled to keep its norm within an acceptable threshold.
- But how to handle vanishing gradient? If the gradient vanishes we cant do anything!



Intuition:

- During forward propagation the gates control the flow of information.
- They prevent any irrelevant information from being written to the state.
- Similarly during backward propagation they control the flow of gradients.

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$
$$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s}_t$$



$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$
$$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s}_t$$

- During backward propagation, $\frac{\partial s_t}{\partial s_{t-1}}$ will depend on f_t , in the worst case say the second term vanishes. Even then, f_t contributes in backpropagation.
- During forward propagation, f_t will vanish.

- If the state at time t-1 did not contribute much to the state at time t (i.e, if $\|f_t\| \to 0$, $\|o_{t-1}\| \to 0$) then during backpropagation the gradients flowing into s_{t-1} will vanish.
- But this kind of a vanishing gradient is fine. (since, s_{t-1} did not contribute to s_t).
- The key difference from vanilla RNNs is that the flow of information and gradients is controlled by the gates which ensures that the gradients vanish only when they should.(i.e, when s_{t-1} didn't contribute much to s_t)