Long-Short Term Memory and Other Gated RNNs

Sargur Srihari
srihari@buffalo.edu

Topics in Sequence Modeling

- Recurrent Neural Networks

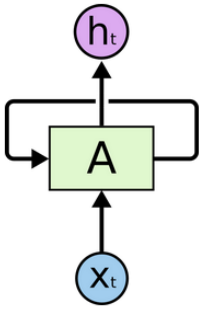
1. Unfolding Computational Graphs
2. Recurrent Neural Networks
3. Bidirectional RNNs
4. Encoder-Decoder Sequence-to-Sequence Architectures
5. Deep Recurrent Networks
6. Recursive Neural Networks
7. The Challenge of Long-Term Dependencies
8. Echo-State Networks
9. Leaky Units and Other Strategies for Multiple Time Scales
10. LSTM and Other Gated RNNs
11. Optimization for Long-Term Dependencies
12. Explicit Memory

Topics in LSTM and Other Gated RNNs

- From RNN to LSTM
- Problem of Long-Term Dependency
- Core idea behind LSTM
- Step-by-Step LSTM Walk-through
- Description of LSTM Cell
- Equations for: forget gate, state update, output gate
- Gated RNNs

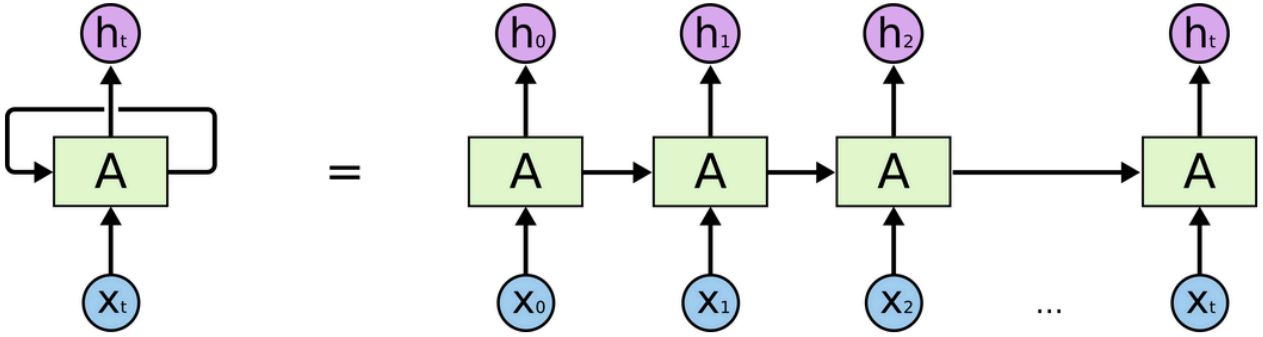
From RNN to LSTM

1. RNNs have loops



A chunk of neural network A looks at some input x_t and outputs a value h_t. A loop allows information to be passed from one step of the network to the next

2. An unrolled RNN

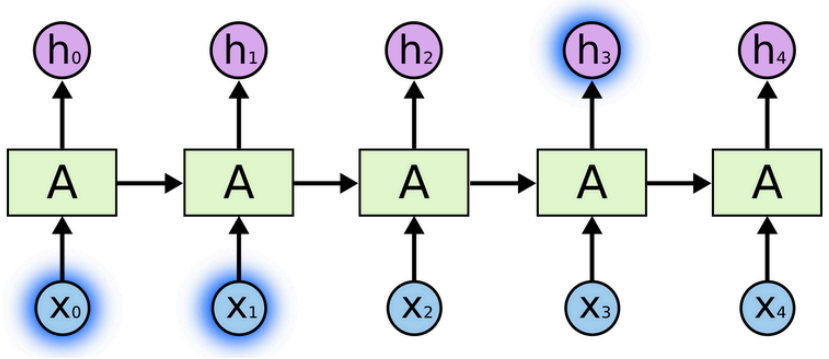


Chain-like structure reveals that RNNs are intimately related to sequences and lists

Source: colah.github

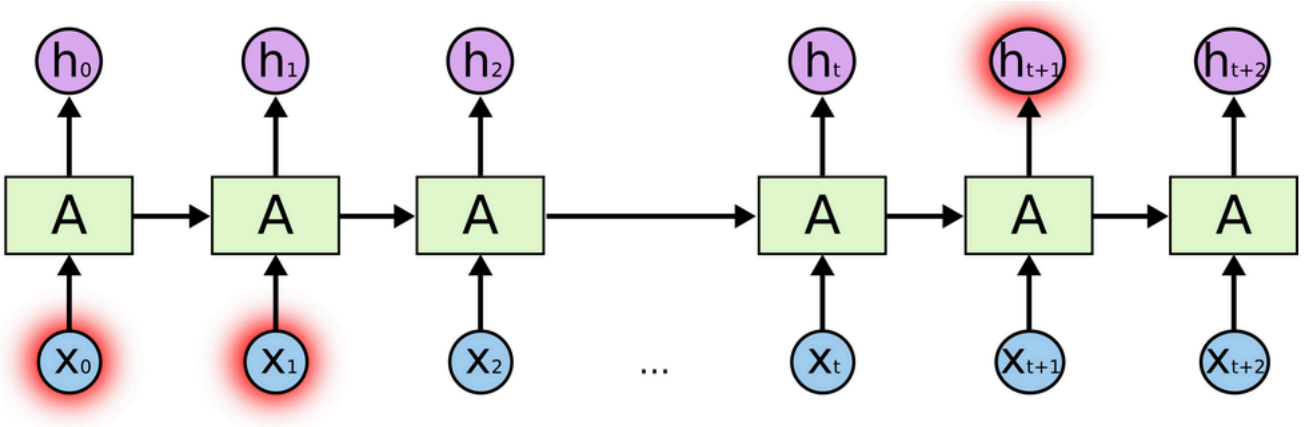
Prediction with only recent previous information

- RNNs connect previous information to the present task
  - Previous video frames may help understand present frame
- Sometimes we need only look at recent previous information to predict
  - To predict the last word of "The clouds are in the *sky*" we don't need any further context. It is obvious that the word is "sky"



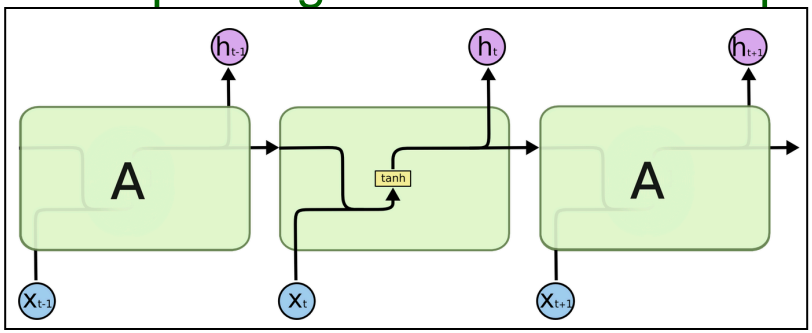
Problem of Long-term dependency

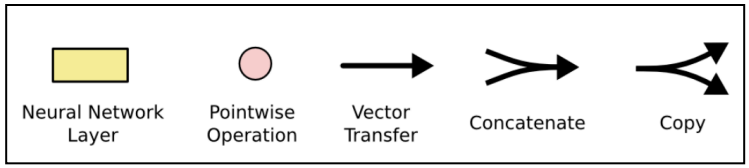
- There are cases where we need more context
- To predict the last word in the sentence "I grew up in France⋯.I speak fluent *French*"
- Using only recent information suggests that the last word is the name of a language. But more distant past indicates that it is French
- It is possible that the gap between the relevant information and where it is needed is very large



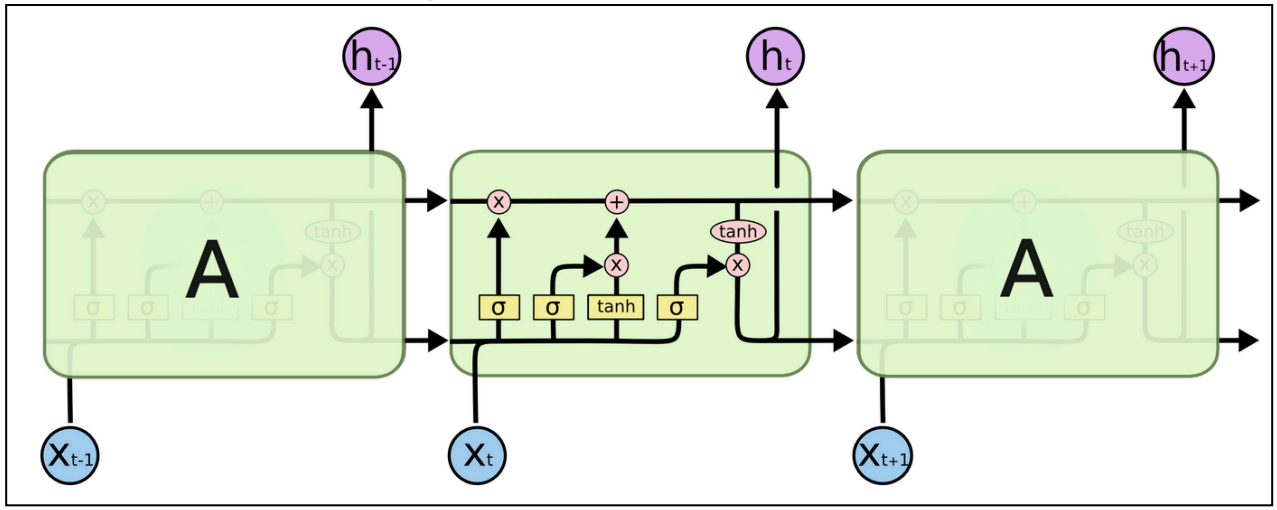
Long Short Term Memory

- Explicitly designed to avoid the long-term dependency problem
- RNNs have the form of a repeating chain structure
  - The repeating module has a simple structure such as tanh



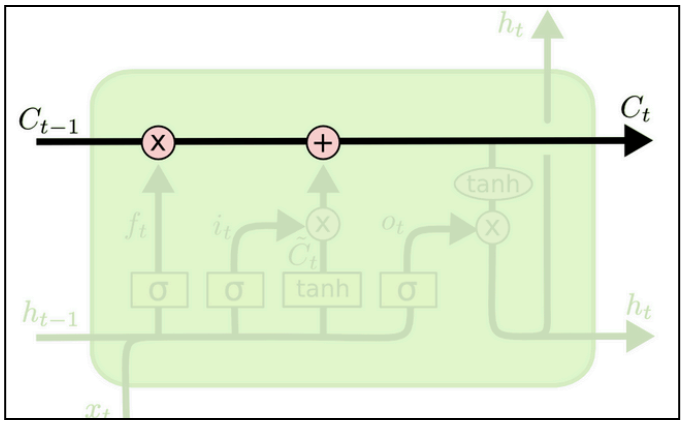


- LSTMs also have a chain structure
  - but the repeating module has a different structure

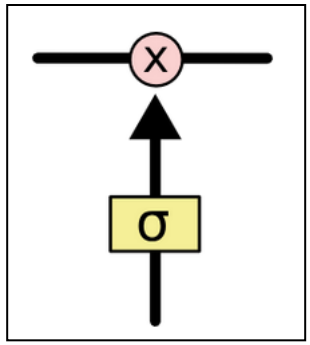


Core idea behind LSTM

- The key to LSTM is the cell state, C_t, the horizontal line running through the top of the diagram
- Like a conveyor belt
  - Runs through entire chain with minor interactions
  - LSTM does have the ability to remove/add information to cell state regulated by structures called gates



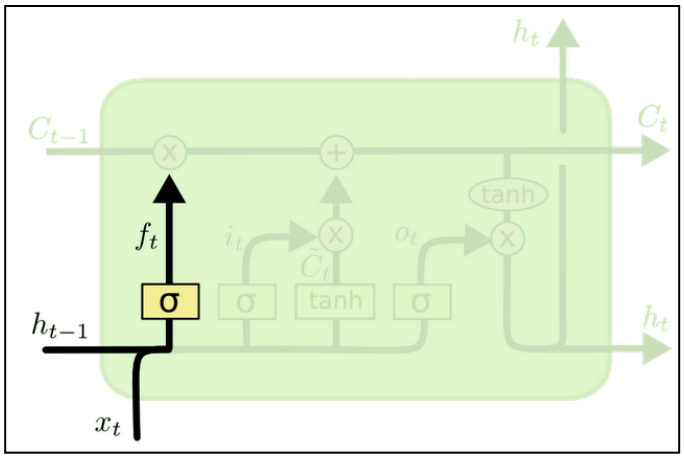
- Gates are an optional way to let information through
- Consist of a sigmoid and a multiplication operation
- Sigmoid outputs a value between 0 and 1
  - 0 means let nothing through
  - 1 means let eveything through



- LSTM has three of these gates, to protect and control cell state

Step-by-step LSM walk through

- Example of language model: predict next word based on previous ones
  - Cell state may include the gender of the present subject
- First step: information to throw away from cell state



Called *forget gate layer*

It looks at $\boldsymbol{h}_{t-1}$ and $\boldsymbol{x}_t$ and outputs a number between 0 and 1 for each member of $\boldsymbol{C}_{t-1}$ for whether to forget

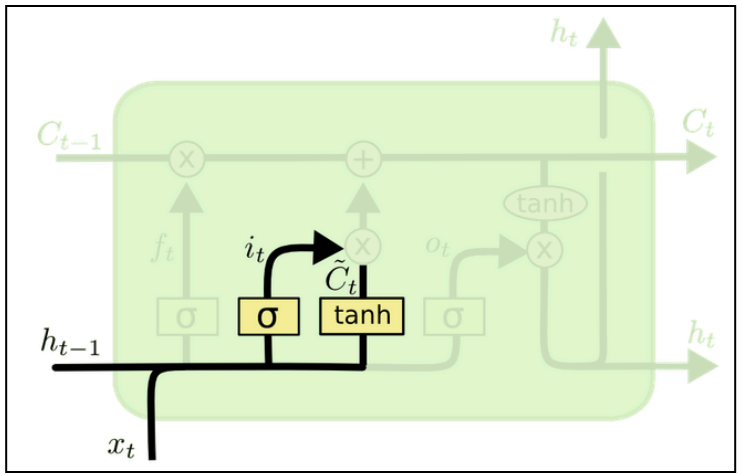
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

In language model

consider trying to predict the next word based on all previous ones
The cell state may include the gender of the present subject
so that the proper pronouns can be used
When we see a new subject we want to forget old subject

LSM walk through: Second step

- Next step is to decide as to what new information we're going to store in the cell state



This has two parts:
first a sigmoid layer called *Input gate layer*: decides which values we will update
Next a tanh layer creates a vector of new candidate values $\tilde{C}_t$ that could be added to the state.
In the third step we will combine these two to create an update to the state

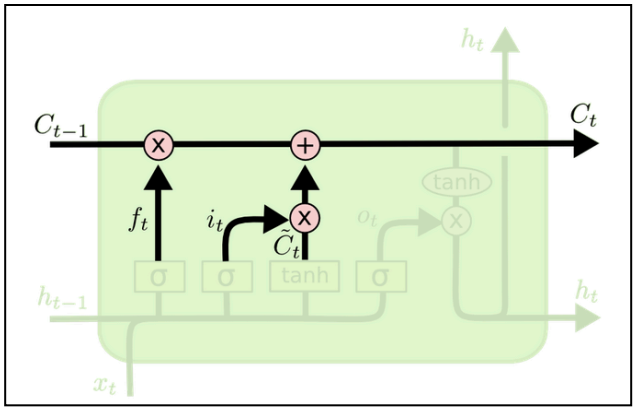
In the Language model,
we'd want to add the gender of the new subject to the cell state, to replace the old One we are forgetting

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

LSTM walk-through: Third Step

- It's now time to update old cell state C_{t-1} into new cell state C_t
  - The previous step decided what we need to do
  - We just need to do it



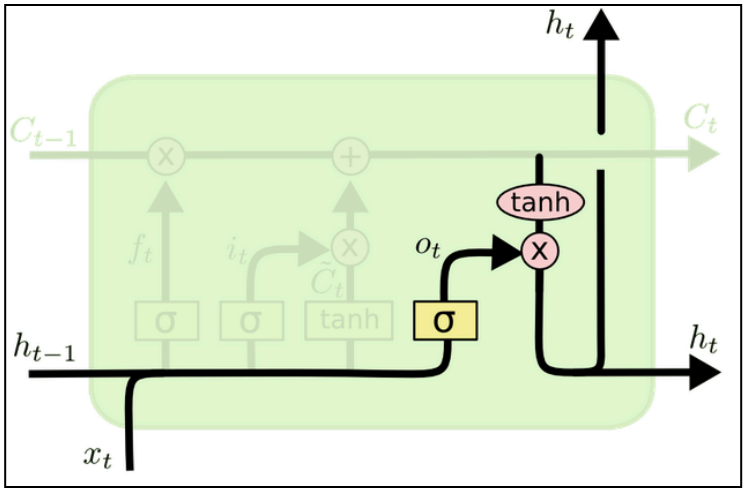
We multiply the old state by f_t, forgetting the things we decided to forget earlier.
Then we add $i_t * \tilde{C}_t$
This is the new candidate values, scaled by how much we decided to update each state value

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

In the Language model,
this is where we'd actually drop the information about the old subject's gender and add the new information,
as we decided in previous steps

LSTM walk-through: Fourth Step

- Finally we decide what we are going to output



This output will be based on our cell state, but will be a filtered version.
First we run a sigmoid layer which decides what parts of cell state we're going to output. Then we put the cell state through $\tanh$ (to push values to be between -1 and 1) and multiply it by the output of the sigmoid gate, so that we output only the parts we decided to

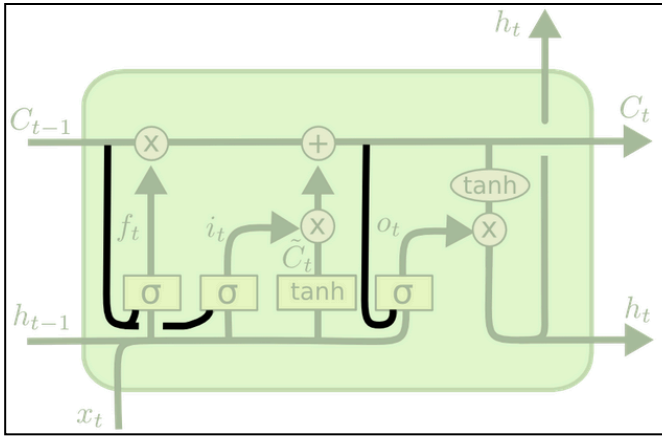
For the Language model,
Since it just saw a subject it might want to output information relevant to a verb, in case that is what is coming next, e.g., it might output whether the subject is singular or plural so that we know what form a verb should be conjugated into if that's what follows next.

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

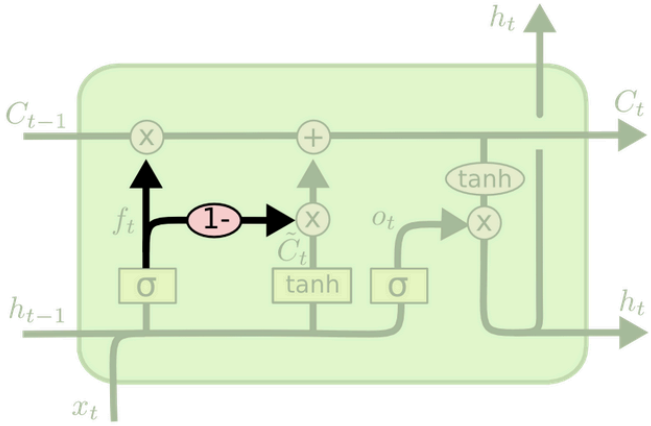
Variants of LSTM

- There are several minor variants of LSTM
  - LSTM with "peephole" connections



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

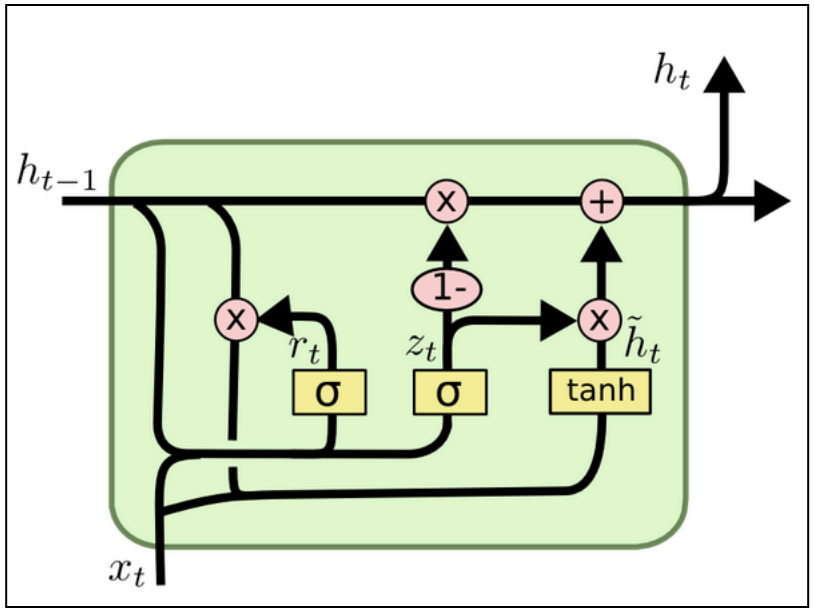
  - Coupled forget and input gates



$$C_t = f_t * C_{t-1} + (\mathbf{1} - \boldsymbol{f_t}) * \tilde{C}_t$$

Gated Recurrent Unit (GRU)

- A dramatic variant of LSTM
  - It combines the forget and input gates into a single update gate
  - It also merges the cell state and hidden state, and makes some other changes
  - The resulting model is simpler than LSTM models
  - Has become increasingly popular



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Effective Sequential Models

- Most effective sequence models in practical applications are called gated RNNs
- They include
  - Long short-term memory
  - Networks based on the gated recurrent unit

Idea of Gated RNNs

- Like leaky units, gated RNNs are based on the idea of creating paths through time that have derivatives that neither vanish nor explode
- Leaky units do this with connection weights that are manually chosen or were parameters α

  generalizing the concept of discrete skipped connections
- Gated RNNs generalize this to connection weights that may change with each time step

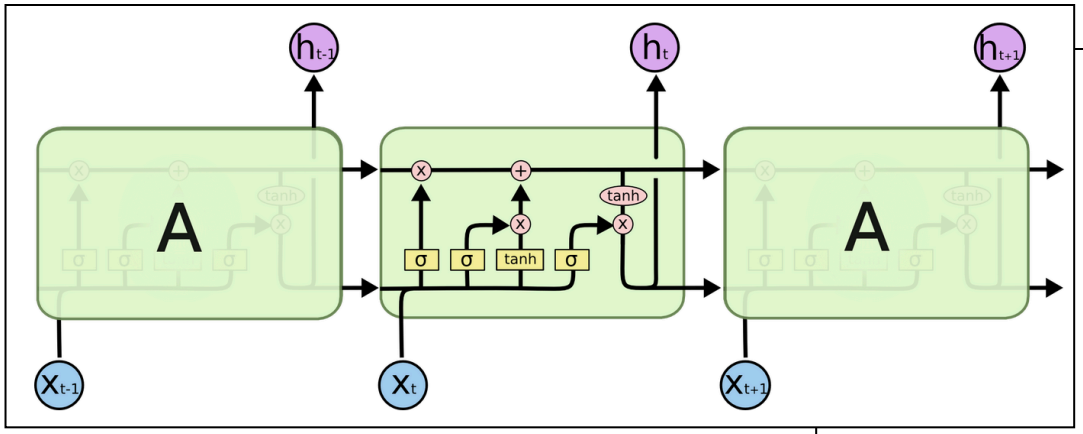
Accumulating Information over Longer Duration

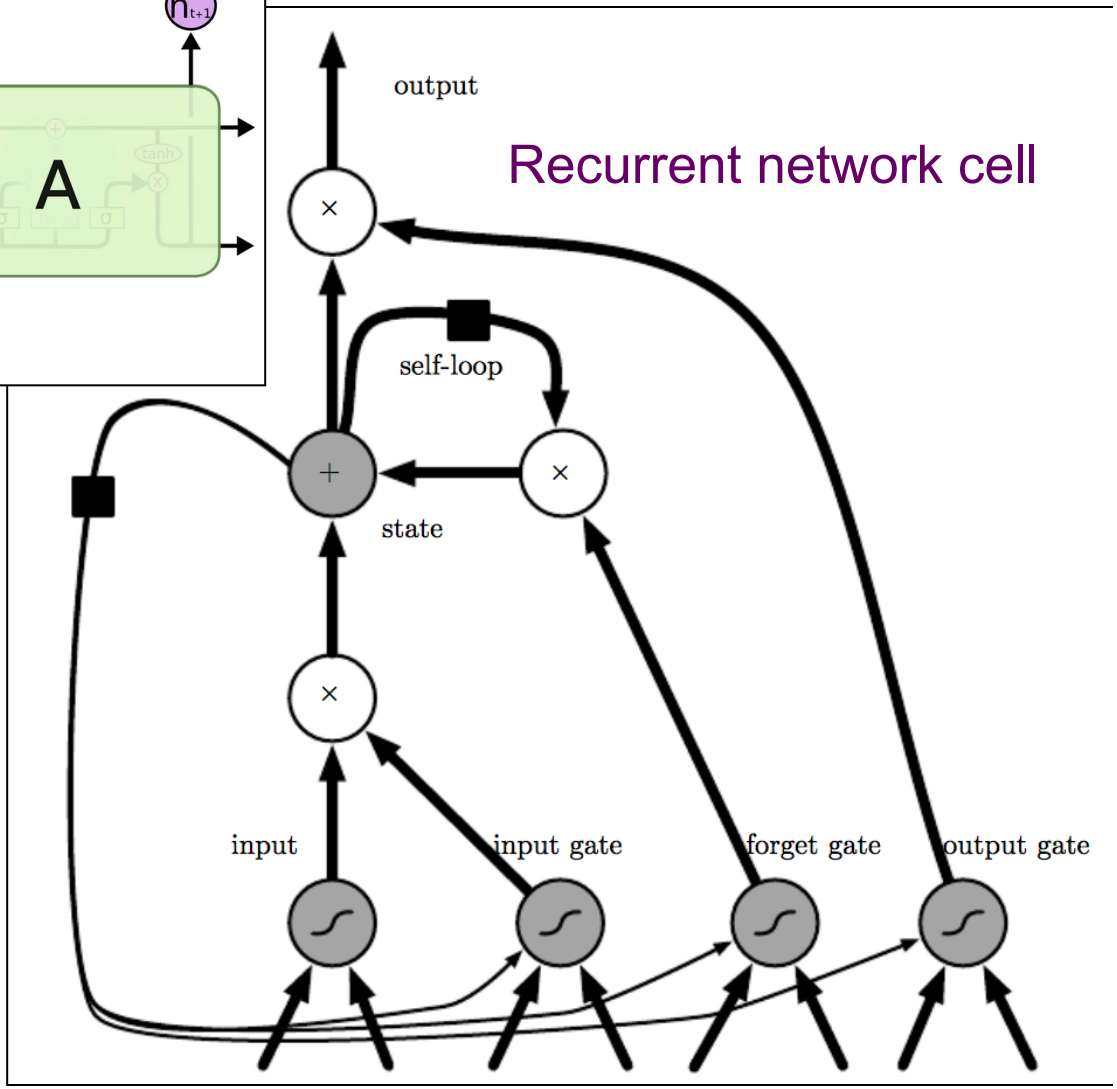
- Leaky Units Allow the network to accumulate information
  - Such as evidence for a particular feature or category
  - Over a long duration
- However, once the information has been used, it might be useful to forget the old state
  - E.g., if a sequence is made of sub-sequences and we want a leaky unit to accumulate evidence inside each sub-sequence, we need a mechanism to forget the old state by setting it to zero
- Gated RNN:
  - Instead of manually deciding when to clear the state, we want the neural network to learn to decide when to do it

LSTM

- Contribution of LSTMs:
  - clever idea of introducing self-loops to produce paths where the gradient can flow for long durations
- A crucial addition: make weight on this self-loop conditioned on the context, rather than fixed
  - By making weight of this self-loop gated (controlled by another hidden unit), time-scale can be changed dynamically
  - Even for an LSTM with fixed parameters, time scale of integration can change based on the input sequence
    - Because time constants are output by the model itself
- LSTM found extremely successful in:
    - Unconstrained handwriting recognition, Speech recognition
    - Handwriting generation, Machine Translation
    - Image Captioning, Parsing

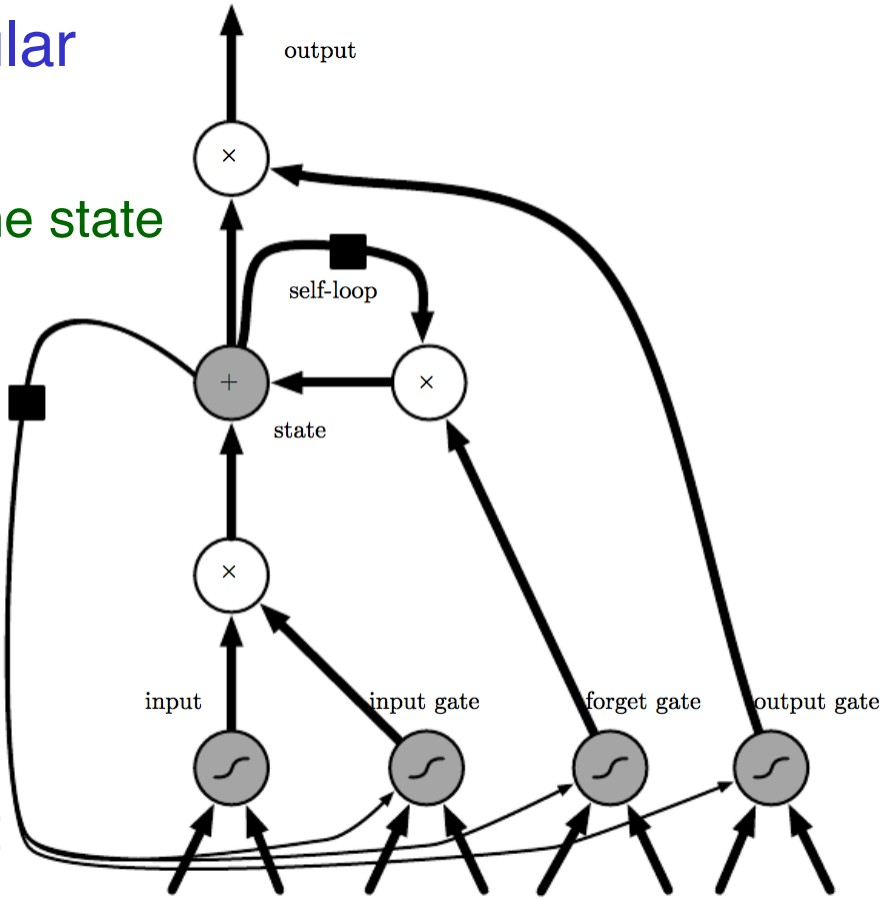
LSTM Block Diagram





LSTM Recurrent Network Cell

- Cells are connected recurrently to each other
  - Replacing hidden units of ordinary recurrent networks
- Input feature computed with regular artificial neuron unit
  - Its value can be accumulated into the state
- State unit has linear self-loop
  - Weight controlled by forget gate
- Output of cell can be shut off by output gate
- All units: sigmoid nonlinearity
  - Input gate can be any squashing
- State unit can also be extra input to gating units
- Black square indicates delay of single time step



Description of the LSTM cell

- Forward propagation equations for the LSTM cell are given in the next slide
  - In the case of a shallow recurrent network architecture
  - Deeper architectures have also been successively used
- Instead of a unit that simply applies an elementwise nonlinearity to the affine transformation of inputs and recurrent units LSTM recurrent units have *LSTM cells* that have an internal recurrence (a self-loop) in addition to the outer recurrence of the RNN
- Each cell has the same inputs and outputs as an ordinary RNN
  - But has more parameters and a system of gating units that controls the flow of information

Equation for LSTM forget gate

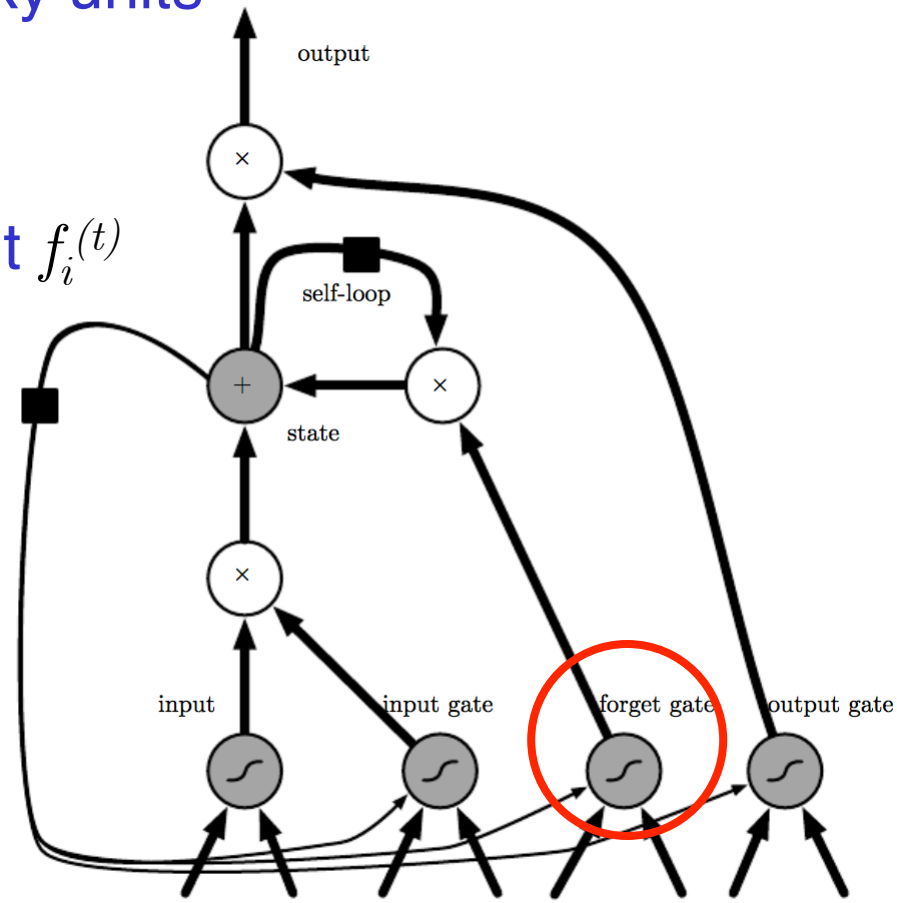
- The most important component is the state unit $s_i^{(t)}$ that has a linear self-loop similar to the leaky units
- However the self-loop weight

  (or the associated time constant)

  is controlled by a forget gate unit $f_i^{(t)}$

  (for time step t and cell i)

  that sets this weight to a value between 0 and 1 via a sigmoid unit



$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}\right)$$

where $\boldsymbol{x}^{(t)}$ is the current input vector and $\boldsymbol{h}^{(t)}$ is the current hidden layer vector containing the outputs of all the LSTM cells, and $\boldsymbol{b}^f$, $\boldsymbol{u}^f$ and $\boldsymbol{W}^f$ are respectively biases, input weights and recurrent weights for forget gates

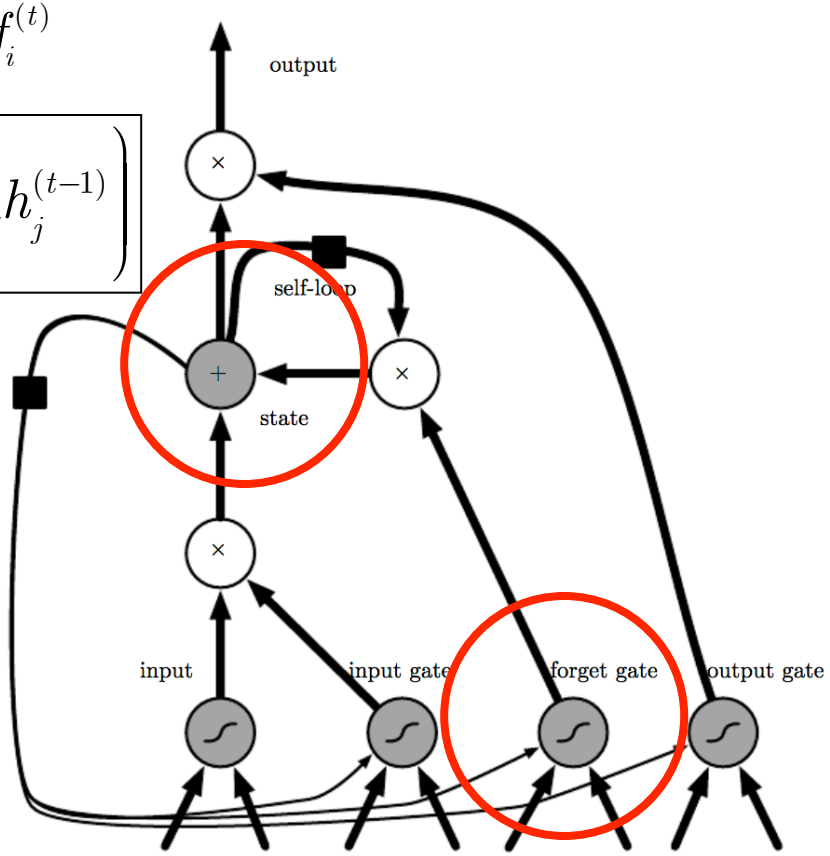
Equation for LSTM internal state update

- The LSTM cell internal state is updated as follows
  - But with conditional self-loop weight $f_i^{(t)}$

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)} \right)$$

  - where $\boldsymbol{b}$, $\boldsymbol{u}$ and $\boldsymbol{W}$ respectively denote the biases, input weights and recurrent weights into the LSTM cell
- External input gate unit $g_i^{(t)}$ is
  - computed similar to forget gate
  - With a sigmoid unit to obtain a gating value between 0 and 1 but with its own parameters

$$g_i^{(t)} = \sigma\left(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)} \right)$$



Equation for output of LSTM cell

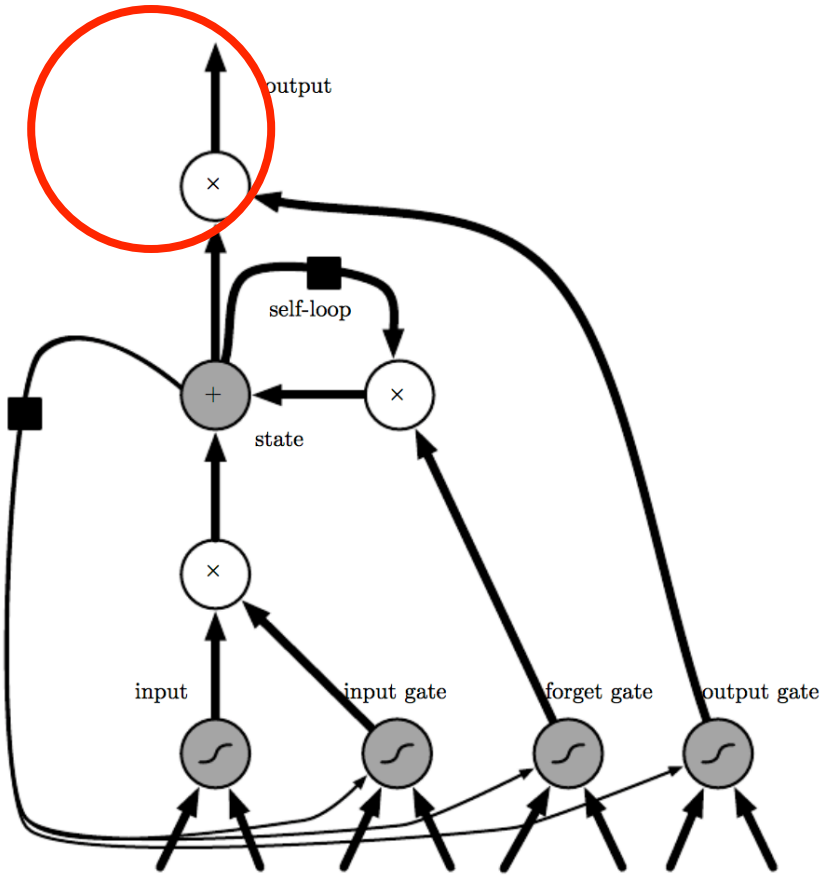
- Output $h_i^{(t)}$ of the LSTM cell can be shut off via the output gate $q_i^{(t)}$ which also uses a sigmoid unit for gating

$$h_i^{(t)} = \tanh\left(s_i^{(t)}\right) q_i^{(t)}$$

$$q_i^{(t)} = \sigma\left(b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)}\right)$$

$\boldsymbol{b}^0$, $\boldsymbol{U}^0$ and $\boldsymbol{W}^0$ are biases, input weights and recurrent weights

- Among the variants one can choose to use the cell state $s_i^{(t)}$ as an extra input (with its weight) into the three gates of the i-th unit
  This would require three additional parameters



Power of LSTMs

- LSTM networks have been shown to learn long-term dependencies more easily than simple recurrent architectures
  - First on artificial data sets
  - Then on challenging sequential tasks
- Variants and alternatives to LSTM have been studied and used and are discussed next

Other gated RNNs

- What pieces of the LSTM architecture are actually necessary?
- What other successful architectures could be designed that allow the network to dynamically control the time scale and forgetting behavior of different units?
- Some answers to these questions are given with the recent work on gated RNNs, whose units are also known as gated recurrent units or GRUs
- Main difference with LSTM
  - Single gating unit simultaneously controls the forgetting factor and decision to update state unit

Update equations for Gated RNNs

- Update equations are

$$h_i^{(t)} = u_i^{(t)} h_i^{(t-1)} + (1 - u_i^{(t-1)}) \sigma \left(b_i + \sum_j U_{i,j} x_j^{(t-1)} + \sum_j W_{i,j} r_j^{(t-1)} h_j^{(t-1)} \right)$$

  - Where $\boldsymbol{u}$ stands for the update gate and $\boldsymbol{r}$ for reset gate. Their value is defined as usual:

$$u_i^{(t)} = \sigma \left(b_i^u + \sum_j U_{i,j}^u x_j^{(t)} + \sum_j W_{i,j}^u h_j^{(t)} \right) \quad \text{and} \quad r_i^{(t)} = \sigma \left(b_i^r + \sum_j U_{i,j}^r x_j^{(t)} + \sum_j W_{i,j}^r h_j^{(t)} \right)$$

  - Reset and update gates can individually ignore parts of the state vector
    - Update gates act conditional leaky integrators that can linearly gate any dimension thus choosing to copy it (at one extreme of sigmoid) or completely ignore it (at the other extreme) by replacing it by the new target state value (towards which the leaky integrator converges
    - Reset gates control which parts of the state get used to compute next target state, introducing an additional nonlinear effect in the relationship between past state and future state