GRUS

Each GRU unit, has a C = memory cell, which provides a bit of memory to be later checked:

$$c^{\langle t \rangle} = a^{\langle t \rangle}$$

At every timestep, we consider overwriting the memory cell with a value $\hat{c}^{\langle t \rangle}$.

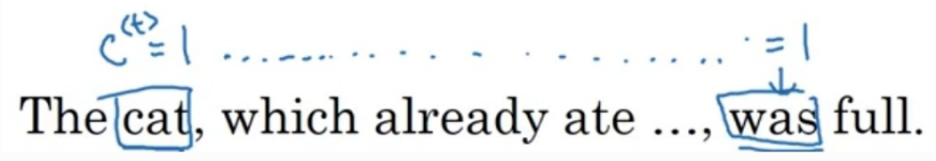
$$\hat{c}^{\langle t \rangle} = \tanh(W_c [c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_c)$$

The important idea is the GATE:

- Update gate $\Rightarrow \Gamma_u = \sigma(W_u [c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u)$

It finally decides wether or not we finally update $\hat{c}^{\langle t \rangle}$.

So, for example, let's say that $c^{\langle t \rangle} = 1$ for cat, as it is singular. Then, was is used because it is stored in the $c^{\langle t \rangle}$ cell.



But, $c^{\langle t \rangle}$ may be required to change its value at another time. That is what Γ_u decides.

Finally, we calculate:

$$c^{\langle t \rangle} = \Gamma_u \cdot \hat{c}^{\langle t \rangle} + (1 - \Gamma_u) \cdot c^{\langle t-1 \rangle}$$

Full GRU

How relevant is $c^{\langle t-1 \rangle}$ to computing the next candidate $c^{\langle t \rangle}$?

$$\tilde{c}^{\langle t \rangle} = \tanh(W_c[\Gamma_r * c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_c)$$

$$\Gamma_u = \sigma(W_u[c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u)$$

$$\Gamma_r = \sigma(W_r[c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_r)$$

$$c^{\langle t \rangle} = \Gamma_u * \tilde{c}^{\langle t \rangle} + (1 - \Gamma_u) * c^{\langle t-1 \rangle}$$

GRU

$$\tilde{c}^{\langle t \rangle} = \tanh(W_c[\Gamma_r * c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_c)$$

$$\Gamma_u = \sigma(W_u[c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u)$$

$$\Gamma_r = \sigma(W_r[c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_r)$$

$$c^{\langle t \rangle} = \Gamma_u * \tilde{c}^{\langle t \rangle} + (1 - \Gamma_u) * c^{\langle t-1 \rangle}$$

$$a^{\langle t \rangle} = c^{\langle t \rangle}$$

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

Update: $\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$

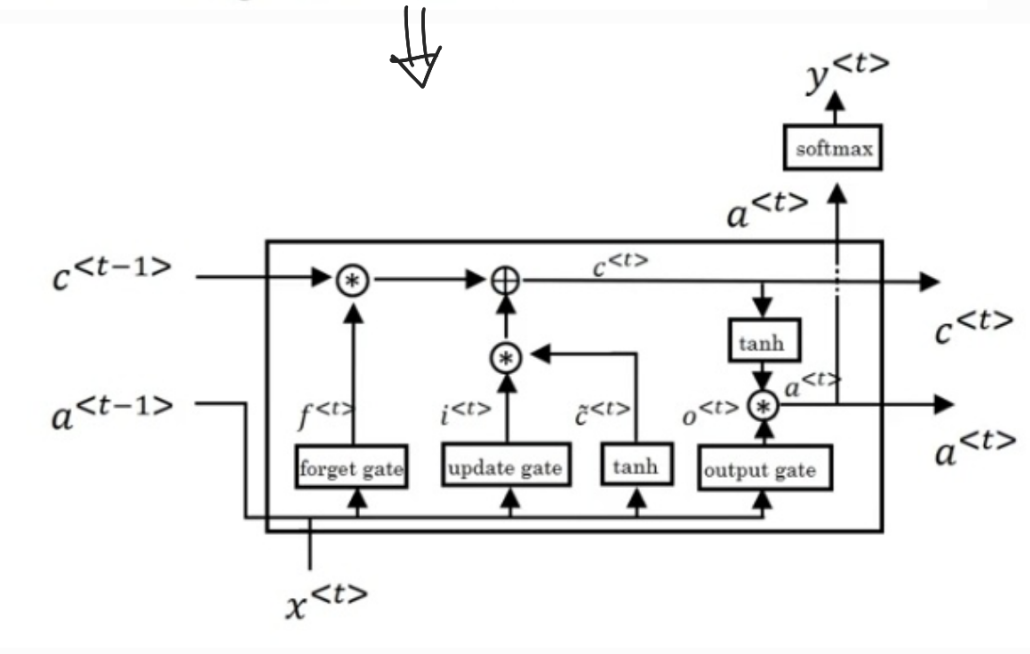
Forget: $\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$

Output: $\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$

} 3 gates

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * \tanh c^{<t>}$$



Variants:

- Peephole connection: for all gates, instead of $x^{<t>}$ use $c^{<t-1>}$, one-to-one.

What you should remember:

- An LSTM is similar to an RNN in that they both use hidden states to pass along information, but an LSTM also uses a cell state, which is like a long-term memory, to help deal with the issue of vanishing gradients
- An LSTM cell consists of a cell state, or long-term memory, a hidden state, or short-term memory, along with 3 gates that constantly update the relevancy of its inputs:
  - A **forget** gate, which decides which input units should be remembered and passed along. It's a tensor with values between 0 and 1.
    - If a unit has a value close to 0, the LSTM will "forget" the stored state in the previous cell state.
    - If it has a value close to 1, the LSTM will mostly remember the corresponding value.
  - An **update** gate, again a tensor containing values between 0 and 1. It decides on what information to throw away, and what new information to add.
    - When a unit in the update gate is close to 1, the value of its candidate is passed on to the hidden state.
    - When a unit in the update gate is close to 0, it's prevented from being passed onto the hidden state.
  - And an **output** gate, which decides what gets sent as the output of the time step

2 - Long Short-Term Memory (LSTM) Network

The following figure shows the operations of an LSTM cell:

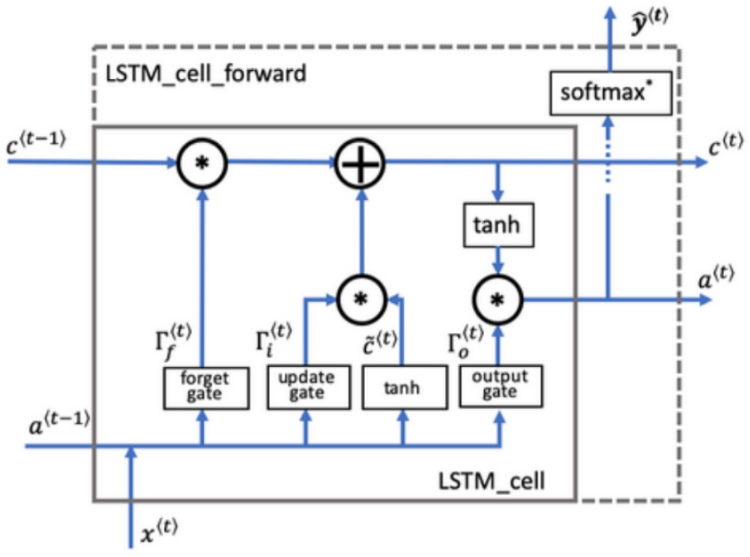


Figure 4: LSTM cell. This tracks and updates a "cell state," or memory variable $c^{\langle t \rangle}$ at every time step, which can be different from $a^{\langle t \rangle}$. Note, the $softmax$ includes a dense layer and softmax.

Similar to the RNN example above, you'll begin by implementing the LSTM cell for a single time step. Then, you'll iteratively call it from inside a "for loop" to have it process an input with T_x time steps.

Overview of gates and states

Forget gate $\mathbf{\Gamma}_f$

- Let's assume you are reading words in a piece of text, and plan to use an LSTM to keep track of grammatical structures, such as whether the subject is singular ("puppy") or plural ("puppies").
- If the subject changes its state (from a singular word to a plural word), the memory of the previous state becomes outdated, so you'll "forget" that outdated state.
- The "forget gate" is a tensor containing values between 0 and 1.
  - If a unit in the forget gate has a value close to 0, the LSTM will "forget" the stored state in the corresponding unit of the previous cell state.
  - If a unit in the forget gate has a value close to 1, the LSTM will mostly remember the corresponding value in the stored state.

Equation

$$\mathbf{\Gamma}_f^{\langle t \rangle} = \sigma(\mathbf{W}_f[\mathbf{a}^{\langle t-1 \rangle}, \mathbf{x}^{\langle t \rangle}] + \mathbf{b}_f)$$

Explanation of the equation:

- $\mathbf{W_f}$ contains weights that govern the forget gate's behavior.
- The previous time step's hidden state [$a^{\langle t-1 \rangle}$ and current time step's input $x^{\langle t \rangle}$] are concatenated together and multiplied by $\mathbf{W_f}$.
- A sigmoid function is used to make each of the gate tensor's values $\mathbf{\Gamma}_f^{\langle t \rangle}$ range from 0 to 1.
- The forget gate $\mathbf{\Gamma}_f^{\langle t \rangle}$ has the same dimensions as the previous cell state $c^{\langle t-1 \rangle}$.
- This means that the two can be multiplied together, element-wise.
- Multiplying the tensors $\mathbf{\Gamma}_f^{\langle t \rangle} * \mathbf{c}^{\langle t-1 \rangle}$ is like applying a mask over the previous cell state.
- If a single value in $\mathbf{\Gamma}_f^{\langle t \rangle}$ is 0 or close to 0, then the product is close to 0.
  - This keeps the information stored in the corresponding unit in $\mathbf{c}^{\langle t-1 \rangle}$ from being remembered for the next time step.
- Similarly, if one value is close to 1, the product is close to the original value in the previous cell state.
  - The LSTM will keep the information from the corresponding unit of $\mathbf{c}^{\langle t-1 \rangle}$, to be used in the next time step.

Variable names in the code

The variable names in the code are similar to the equations, with slight differences.

- `Wf` : forget gate weight $\mathbf{W}_f$
- `bf` : forget gate bias $\mathbf{b}_f$
- `ft` : forget gate $\mathbf{\Gamma}_f^{\langle t \rangle}$

Candidate value $\tilde{\mathbf{c}}^{\langle t \rangle}$

- The candidate value is a tensor containing information from the current time step that **may** be stored in the current cell state $\mathbf{c}^{\langle t \rangle}$.
- The parts of the candidate value that get passed on depend on the update gate.
- The candidate value is a tensor containing values that range from -1 to 1.
- The tilde "~" is used to differentiate the candidate $\tilde{\mathbf{c}}^{\langle t \rangle}$ from the cell state $\mathbf{c}^{\langle t \rangle}$.

Equation

$$\tilde{\mathbf{c}}^{\langle t \rangle} = \tanh\left(\mathbf{W}_c [\mathbf{a}^{\langle t-1 \rangle}, \mathbf{x}^{\langle t \rangle}] + \mathbf{b}_c \right)$$

Explanation of the equation

- The *tanh* function produces values between -1 and 1.

Variable names in the code

- `cct` : candidate value $\tilde{\mathbf{c}}^{\langle t \rangle}$

Update gate $\mathbf{\Gamma}_i$

- You use the update gate to decide what aspects of the candidate $\tilde{\mathbf{c}}^{\langle t \rangle}$ to add to the cell state $c^{\langle t \rangle}$.
- The update gate decides what parts of a "candidate" tensor $\tilde{\mathbf{c}}^{\langle t \rangle}$ are passed onto the cell state $\mathbf{c}^{\langle t \rangle}$.
- The update gate is a tensor containing values between 0 and 1.
  - When a unit in the update gate is close to 1, it allows the value of the candidate $\tilde{\mathbf{c}}^{\langle t \rangle}$ to be passed onto the hidden state $\mathbf{c}^{\langle t \rangle}$
  - When a unit in the update gate is close to 0, it prevents the corresponding value in the candidate from being passed onto the hidden state.
- Notice that the subscript "i" is used and not "u", to follow the convention used in the literature.

Equation

$$\mathbf{\Gamma}_i^{\langle t \rangle} = \sigma(\mathbf{W}_i [a^{\langle t-1 \rangle}, \mathbf{x}^{\langle t \rangle}] + \mathbf{b}_i)$$

Explanation of the equation

- Similar to the forget gate, here $\mathbf{\Gamma}_i^{\langle t \rangle}$, the sigmoid produces values between 0 and 1.
- The update gate is multiplied element-wise with the candidate, and this product ($\mathbf{\Gamma}_i^{\langle t \rangle} * \tilde{c}^{\langle t \rangle}$) is used in determining the cell state $\mathbf{c}^{\langle t \rangle}$.

Variable names in code (Please note that they're different than the equations)

In the code, you'll use the variable names found in the academic literature. These variables don't use "u" to denote "update".

- `Wi` is the update gate weight $\mathbf{W}_i$ (not "Wu")
- `bi` is the update gate bias $\mathbf{b}_i$ (not "bu")
- `it` is the update gate $\mathbf{\Gamma}_i^{\langle t \rangle}$ (not "ut")

Cell state $\mathbf{c}^{\langle t \rangle}$

- The cell state is the "memory" that gets passed onto future time steps.
- The new cell state $\mathbf{c}^{\langle t \rangle}$ is a combination of the previous cell state and the candidate value.

Equation

$$\mathbf{c}^{\langle t \rangle} = \mathbf{\Gamma}_f^{\langle t \rangle} * \mathbf{c}^{\langle t-1 \rangle} + \mathbf{\Gamma}_i^{\langle t \rangle} * \tilde{\mathbf{c}}^{\langle t \rangle}$$

Explanation of equation

- The previous cell state $\mathbf{c}^{\langle t-1 \rangle}$ is adjusted (weighted) by the forget gate $\mathbf{\Gamma}_f^{\langle t \rangle}$
- and the candidate value $\tilde{\mathbf{c}}^{\langle t \rangle}$, adjusted (weighted) by the update gate $\mathbf{\Gamma}_i^{\langle t \rangle}$

Variable names and shapes in the code

- `c` : cell state, including all time steps, $\mathbf{c}$ shape (n_a, m, T_x)
- `c_next` : new (next) cell state, $\mathbf{c}^{\langle t \rangle}$ shape (n_a, m)
- `c_prev` : previous cell state, $\mathbf{c}^{\langle t-1 \rangle}$, shape (n_a, m)

Output gate $\mathbf{\Gamma}_o$

- The output gate decides what gets sent as the prediction (output) of the time step.
- The output gate is like the other gates, in that it contains values that range from 0 to 1.

Equation

$$\mathbf{\Gamma}_o^{\langle t \rangle} = \sigma(\mathbf{W}_o[\mathbf{a}^{\langle t-1 \rangle}, \mathbf{x}^{\langle t \rangle}] + \mathbf{b}_o)$$

Explanation of the equation

- The output gate is determined by the previous hidden state $\mathbf{a}^{\langle t-1 \rangle}$ and the current input $\mathbf{x}^{\langle t \rangle}$
- The sigmoid makes the gate range from 0 to 1.

Variable names in the code

- `Wo` : output gate weight, $\mathbf{W_o}$
- `bo` : output gate bias, $\mathbf{b_o}$
- `ot` : output gate, $\mathbf{\Gamma}_o^{\langle t \rangle}$

Hidden state $\mathbf{a}^{\langle t \rangle}$

- The hidden state gets passed to the LSTM cell's next time step.
- It is used to determine the three gates ($\mathbf{\Gamma}_f, \mathbf{\Gamma}_u, \mathbf{\Gamma}_o$) of the next time step.
- The hidden state is also used for the prediction $y^{\langle t \rangle}$.

Equation

$$\mathbf{a}^{\langle t \rangle} = \mathbf{\Gamma}_o^{\langle t \rangle} * \tanh(\mathbf{c}^{\langle t \rangle})$$

Explanation of equation

- The hidden state $\mathbf{a}^{\langle t \rangle}$ is determined by the cell state $\mathbf{c}^{\langle t \rangle}$ in combination with the output gate $\mathbf{\Gamma}_o$.
- The cell state state is passed through the `tanh` function to rescale values between -1 and 1.
- The output gate acts like a "mask" that either preserves the values of $\tanh(\mathbf{c}^{\langle t \rangle})$ or keeps those values from being included in the hidden state $\mathbf{a}^{\langle t \rangle}$

Variable names and shapes in the code

- `a` : hidden state, including time steps. $\mathbf{a}$ has shape (n_a, m, T_x)
- `a_prev` : hidden state from previous time step. $\mathbf{a}^{\langle t-1 \rangle}$ has shape (n_a, m)
- `a_next` : hidden state for next time step. $\mathbf{a}^{\langle t \rangle}$ has shape (n_a, m)

Prediction $\mathbf{y}_{pred}^{\langle t \rangle}$

- The prediction in this use case is a classification, so you'll use a softmax.

The equation is:

$$\mathbf{y}_{pred}^{\langle t \rangle} = \text{softmax}(\mathbf{W}_y\mathbf{a}^{\langle t \rangle} + \mathbf{b}_y)$$

Variable names and shapes in the code

- `y_pred` : prediction, including all time steps. $\mathbf{y}_{pred}$ has shape (n_y, m, T_x). Note that $(T_y = T_x)$ for this example.
- `yt_pred` : prediction for the current time step t. $\mathbf{y}_{pred}^{\langle t \rangle}$ has shape (n_y, m)

2.2 - Forward Pass for LSTM

Now that you have implemented one step of an LSTM, you can iterate this over it using a for loop to process a sequence of T_x inputs.

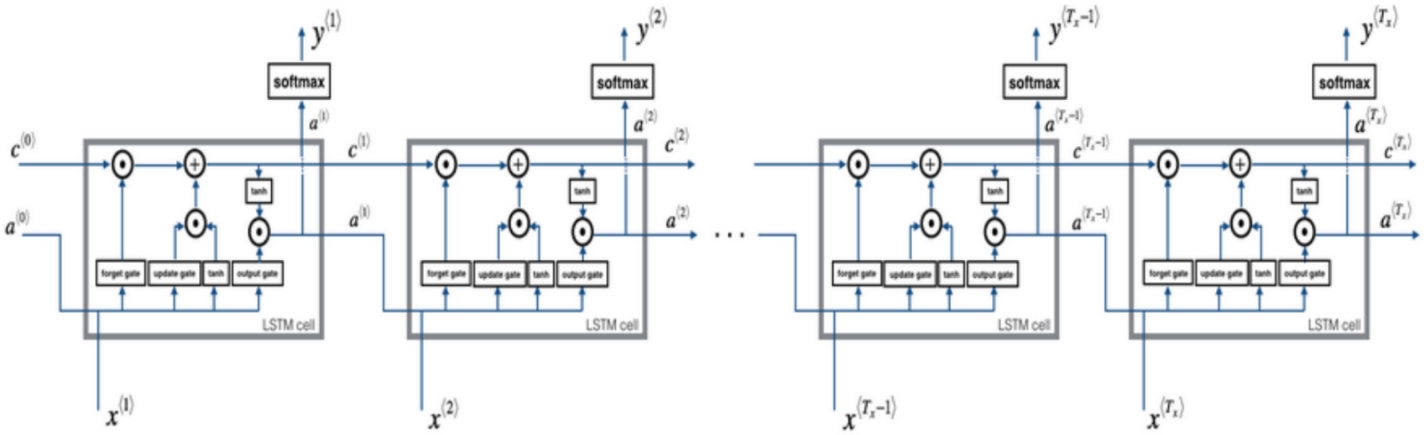


Figure 5: LSTM over multiple time steps.

Exercise 4 - lstm_forward

Implement `lstm_forward()` to run an LSTM over T_x time steps.

Instructions

- Get the dimensions n_x, n_a, n_y, m, T_x from the shape of the variables: `x` and `parameters`
- Initialize the 3D tensors a, c and y
  - a: hidden state, shape (n_a, m, T_x)
  - c: cell state, shape (n_a, m, T_x)
  - y: prediction, shape (n_y, m, T_x) (Note that $T_y = T_x$ in this example)
  - **Note** Setting one variable equal to the other is a "copy by reference". In other words, don't do `c = a', otherwise both these variables point to the same underlying variable.
- Initialize the 2D tensor $a^{\langle t \rangle}$
  - $a^{\langle t \rangle}$ stores the hidden state for time step t. The variable name is `a_next`.
  - $a^{\langle 0 \rangle}$, the initial hidden state at time step 0, is passed in when calling the function. The variable name is `a0`.
  - $a^{\langle t \rangle}$ and $a^{\langle 0 \rangle}$ represent a single time step, so they both have the shape (n_a, m)
  - Initialize $a^{\langle t \rangle}$ by setting it to the initial hidden state ($a^{\langle 0 \rangle}$) that is passed into the function.
- Initialize $c^{\langle t \rangle}$ with zeros.
  - The variable name is `c_next`
  - $c^{\langle t \rangle}$ represents a single time step, so its shape is (n_a, m)
  - **Note**: create `c_next` as its own variable with its own location in memory. Do not initialize it as a slice of the 3D tensor c. In other words, **don't** do