

Inside the LSTM: update equations

An LSTM layer, at time step t

- Takes input element $\mathbf{x}_{(t)}$
- Updates long term memory $\mathbf{c}_{(t)}$
- Updates control state $\mathbf{h}_{(t)}$
- Optionally outputs $\mathbf{y}_{(t)}$

according to the equations

$$\begin{aligned}
\mathbf{c}_{(t)} &= \mathbf{remember}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{save}_{(t)} \otimes \mathbf{c}'_{(t)} && \text{Long term memory} \\
\mathbf{h}_{(t)} &= \mathbf{focus}_{(t)} \otimes \tanh(\mathbf{c}_{(t)}) && \text{Short term memory/control} \\
\mathbf{y}_{(t)} &= \mathbf{h}_{(t)} && \text{Output}
\end{aligned}$$

where

remember is a gate that allows elements of \mathbf{c} to be remembered/forgotten
focus is a mask that controls movement from long-term to short-term
save is a gate that controls selective updating of elements of \mathbf{c}

A lot of moving parts !

The important thing to remember is that the layer

- Has a matrix **W** of weights
- That controls the update of each part

Training, as usual, seeks to discover the optimal (i.e., loss function minimizing) values for **W**

Let's try to understand the whole by examining each piece.

Memory/States

$\mathbf{c}_{(t)}$ is the long-term memory.

It is a vector of features that need to be retained throughout the computation

- As each element $\mathbf{x}_{(t)}$ of input sequence \mathbf{x} is processed
- It records the "concepts" that are important to solve the task
- Might have many elements

$\mathbf{h}_{(t)}$ is the short-term or control memory.

Very much like a vanilla RNN, it's job is to guide the transition from state $\mathbf{h}_{(t)}$ to state $\mathbf{h}_{(t+1)}$.

It is a vector of features that need to be retained *for the immediate future*

- It might have many elements
- Same length as $\mathbf{c}_{(t)}$

An analogy might help.

Suppose you are driving a car in an unfamiliar city

- Short term memory is a map of the surrounding blocks
- Long term memory is a map of the city plus rules of the road

Gates/Masks

remember, **save**, **focus** are vectors that affect long term memory $\mathbf{c}_{(t)}$

- Element-wise
- So have same length as $\mathbf{c}_{(t)}$

They will be used

- To selectively modify individual elements of $\mathbf{c}_{(t)}$
- Forget/Reset the value of an element that is no longer relevant
- Decide which individual elements to update

The [classic paper that introduced the LSTM](https://www.bioinf.jku.at/publications/older/2604.pdf) (<https://www.bioinf.jku.at/publications/older/2604.pdf>) gives these gates different names

- **remember**_(t) \mapsto **f**_(t)
 - **f** denotes "Forget" (although it really means "don't forget", i.e, remember !)
- **save**_(t) \mapsto **i**_(t)
 - **i** denotes "Input"
- **focus**_(t) \mapsto **o**_(t)
 - **o** denotes "Output"

Hopefully the names in our presentation add clarity.

Output

$\mathbf{y}_{(t)}$ is the value (if any) output at step t , for

- A one to many function
- Or a many to many function

As written

$$\mathbf{y}_{(t)} = \mathbf{h}_{(t)}$$

so it has the same length as a memory element $\mathbf{h}_{(t)}, c_{(t)}$.

This assumption is purely for simplicity

- You can map $\mathbf{h}_{(t)}$ through another layer
- That transforms $\mathbf{h}_{(t)}$ into the appropriate type/shape for output $\mathbf{y}_{(t)}$

In fact, our equation for the vanilla RNN included this final transformation of \mathbf{h} to \mathbf{y} :

$$\mathbf{y}_{(t)} = \mathbf{W}_{hy}\mathbf{h}_{(t)} + \mathbf{b}_y$$

The update process

Let's examine the update equation for each of the parts.

Update long-term memory

Long-term memory is updated in a two step process

- Produce a "candidate" updated value for each element of the state
- Decide which of the candidate updated values get applied to the long term memory
 - Successful candidates become part of long term memory
 - Unsucessful candiates are dropped

The candidate update value vector $\mathbf{c}'_{(t)}$ is a function of

- The prior short term state $\mathbf{h}_{(t-1)}$
- And the current input $\mathbf{x}_{(t)}$
- Controlled by parts of the weight matrix \mathbf{W}

$$\mathbf{c}'_{(t)} = \tanh(\mathbf{W}_{x,c}\mathbf{x}_{(t)} + \mathbf{W}_{h,c}\mathbf{h}_{(t-1)} + \mathbf{b}_c)$$

This is very much like the RNN state update equation

- Although the RNN equation has \mathbf{h} on both sides of the equation, so is directly recursive in form

We now need to decide which elements of $\mathbf{c}_{(t)}$ to change.

The **remember** mask controls forgetting the current value $\mathbf{c}_{(t-1)}$

- When **remember** $_{(t),j} = 0$
 - $\mathbf{c}_{(t-1),j}$, the j^{th} element of $\mathbf{c}_{(t-1)}$
 - Will be reset to 0 ("forgotten")
- When **remember** $_{(t),j} = 1$
 - $\mathbf{c}_{(t-1),j}$, the j^{th} element of $\mathbf{c}_{(t-1)}$
 - Will contribute to the new value $\mathbf{c}_{(t),j}$

The **save** mask controls whether the candidate value $\mathbf{c}'_{(t)}$ contributes to the new value $\mathbf{c}_{(t)}$:

- When $\mathbf{save}_{(t),j} = 1$
 - Candidate value $\mathbf{c}'_{(t),j}$ will contribute to the new value $\mathbf{c}_{(t),j}$
- When $\mathbf{save}_{(t),j} = 0$
 - Candidate value $\mathbf{c}'_{(t),j}$ will **not** contribute to the new value $\mathbf{c}_{(t),j}$

Here is the update equation for $\mathbf{c}_{(t)}$.

- It combines the remember/forget decision for each element
- With the decision on passing through the candidate value for the element

$$\mathbf{c}_{(t)} = \mathbf{remember}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{save}_{(t)} \otimes \mathbf{c}'_{(t)}$$

The role of \tanh in the candidate value equation

Why modify the candidate value $\mathbf{c}'_{(t)}$ by passing it through \tanh ?

The \tanh has the important property

- That its range is $[-1, +1]$

So updates to $\mathbf{c}_{(t)}$ have the flavor of either

- Incrementing existing value $\mathbf{c}_{(t-1),j}$ by 1
- Or decrementing existing value $\mathbf{c}_{(t-1),j}$ by 1

This makes $\mathbf{c}_{(t)}$ act like a *counter*.

Update short-term memory (control state)

The short-term memory update

- Selectively copies parts of the newly updated long-term memory $\mathbf{c}_{(t)}$
- To short-term memory

$$\mathbf{h}_{(t)} = \mathbf{focus}_{(t)} \otimes \tanh(\mathbf{c}_{(t)}) \quad \text{Short term memory/control}$$

The **focus** mask selects which elements of long-term memory are immediately relevant for control

The \tanh activation function applied to long-term memory $\mathbf{c}_{(t)}$

- Squashes the range

The gate/mask update equations

All of the gates are updated via similar equations, taking

- The prior short term state $\mathbf{h}_{(t-1)}$
- And the current input $\mathbf{x}_{(t)}$
- Controlled by parts of the weight matrix \mathbf{W}

$$\begin{aligned}
\mathbf{remember}_{(t)} &= f_{(t)} = \sigma(\mathbf{W}_{x,f}\mathbf{x}_{(t)} + \mathbf{W}_{h,f}\mathbf{h}_{(t-1)} + \mathbf{b}_f) \\
\mathbf{save}_{(t)} &= i_{(t)} = \sigma(\mathbf{W}_{x,i}\mathbf{x}_{(t)} + \mathbf{W}_{h,i}\mathbf{h}_{(t-1)} + \mathbf{b}_i) \\
\mathbf{focus}_{(t)} &= o_{(t)} = \sigma(\mathbf{W}_{x,o}\mathbf{x}_{(t)} + \mathbf{W}_{h,o}\mathbf{h}_{(t-1)} + \mathbf{b}_o)
\end{aligned}$$

Notice the use of the sigmoid activation for each gate/mask ?

- This restricts the range of each element to $[0, 1]$
- As needed by a gate/mask

Conclusion

That was quite a workout.

There were lots of moving parts, but hopefully you can now understand each.

To conclude, here is the full set of update equations

$\mathbf{c}'_{(t)}$	$= \tanh(\mathbf{W}_{x,c}\mathbf{x}_{(t)} + \mathbf{W}_{h,c}\mathbf{h}_{(t-1)} + \mathbf{b}_c)$	Candidate update
$\mathbf{c}_{(t)}$	$= \mathbf{remember}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{save}_{(t)} \otimes \mathbf{c}'_{(t)}$	Long term memory
$\mathbf{h}_{(t)}$	$= \mathbf{focus}_{(t)} \otimes \tanh(\mathbf{c}_{(t)})$	Short term memory
$\mathbf{y}_{(t)}$	$= \mathbf{h}_{(t)}$	Output

where

$\mathbf{remember}_{(t)}$	$= \sigma(\mathbf{W}_{x,f}\mathbf{x}_{(t)} + \mathbf{W}_{h,f}\mathbf{h}_{(t-1)} + \mathbf{b}_f)$
$\mathbf{save}_{(t)}$	$= \sigma(\mathbf{W}_{x,i}\mathbf{x}_{(t)} + \mathbf{W}_{h,i}\mathbf{h}_{(t-1)} + \mathbf{b}_i)$
$\mathbf{focus}_{(t)}$	$= \sigma(\mathbf{W}_{x,o}\mathbf{x}_{(t)} + \mathbf{W}_{h,o}\mathbf{h}_{(t-1)} + \mathbf{b}_o)$

$$\begin{aligned}
\mathbf{remember}_{(t)} &= f_{(t)} = \sigma(\mathbf{W}_{x,f}\mathbf{x}_{(t)} + \mathbf{W}_{h,f}\mathbf{h}_{(t-1)} + \mathbf{b}_f) \\
\mathbf{save}_{(t)} &= i_{(t)} = \sigma(\mathbf{W}_{x,i}\mathbf{x}_{(t)} + \mathbf{W}_{h,i}\mathbf{h}_{(t-1)} + \mathbf{b}_i) \\
\mathbf{focus}_{(t)} &= o_{(t)} = \sigma(\mathbf{W}_{x,o}\mathbf{x}_{(t)} + \mathbf{W}_{h,o}\mathbf{h}_{(t-1)} + \mathbf{b}_o)
\end{aligned}$$

In [2]: `print("Done")`

Done