**Overview of gates and states**

**- Forget gate**ΓfΓf[**¶**](https://xihoouzwsengvudiwzedub.coursera-apps.org/notebooks/Week%201/Building%20a%20Recurrent%20Neural%20Network%20-%20Step%20by%20Step/Building_a_Recurrent_Neural_Network_Step_by_Step_v3b.ipynb#--Forget-gate-$\mathbf{\Gamma}_{f}$)

* Let's assume we 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* If the subject changes its state (from a singular word to a plural word), the memory of the previous state becomes outdated, so we "forget" that outdated state.
* The "forget gate" is a tensor containing values that are 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***

Γ⟨t⟩f=σ(Wf[a⟨t−1⟩,x⟨t⟩]+bf)(1)(1)Γf⟨t⟩=σ(Wf[a⟨t−1⟩,x⟨t⟩]+bf)

***Explanation of the equation:***

* WfWf contains weights that govern the forget gate's behavior.
* The previous time step's hidden state [a⟨t−1⟩[a⟨t−1⟩ and current time step's input x⟨t⟩]x⟨t⟩] are concatenated together and multiplied by WfWf.
* A sigmoid function is used to make each of the gate tensor's values Γ⟨t⟩fΓf⟨t⟩ range from 0 to 1.
* The forget gate Γ⟨t⟩fΓf⟨t⟩ has the same dimensions as the previous cell state c⟨t−1⟩c⟨t−1⟩.
* This means that the two can be multiplied together, element-wise.
* Multiplying the tensors Γ⟨t⟩f∗c⟨t−1⟩Γf⟨t⟩∗c⟨t−1⟩ is like applying a mask over the previous cell state.
* If a single value in Γ⟨t⟩fΓf⟨t⟩ is 0 or close to 0, then the product is close to 0.
  + This keeps the information stored in the corresponding unit in c⟨t−1⟩c⟨t−1⟩ 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 c⟨t−1⟩c⟨t−1⟩, 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 WfWf
* bf: forget gate bias bfbf
* ft: forget gate Γ⟨t⟩fΓf⟨t⟩

#### Candidate value c̃ ⟨t⟩c~⟨t⟩

* The candidate value is a tensor containing information from the current time step that **may** be stored in the current cell state c⟨t⟩c⟨t⟩.
* Which parts of the candidate value get passed on depends 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 c̃ ⟨t⟩c~⟨t⟩ from the cell state c⟨t⟩c⟨t⟩.

##### *Equation*

c̃ ⟨t⟩=tanh(Wc[a⟨t−1⟩,x⟨t⟩]+bc)(3)(3)c~⟨t⟩=tanh⁡(Wc[a⟨t−1⟩,x⟨t⟩]+bc)

##### *Explanation of the equation*

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

##### *Variable names in the code*

* cct: candidate value c̃ ⟨t⟩c~⟨t⟩

#### - Update gate ΓiΓi

* We use the update gate to decide what aspects of the candidate c̃ ⟨t⟩c~⟨t⟩ to add to the cell state c⟨t⟩c⟨t⟩.
* The update gate decides what parts of a "candidate" tensor c̃ ⟨t⟩c~⟨t⟩ are passed onto the cell state c⟨t⟩c⟨t⟩.
* 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 c̃ ⟨t⟩c~⟨t⟩ to be passed onto the hidden state c⟨t⟩c⟨t⟩
  + 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 we use the subscript "i" and not "u", to follow the convention used in the literature.

##### *Equation*

Γ⟨t⟩i=σ(Wi[a⟨t−1⟩,x⟨t⟩]+bi)(2)(2)Γi⟨t⟩=σ(Wi[a⟨t−1⟩,x⟨t⟩]+bi)

##### *Explanation of the equation*

* Similar to the forget gate, here Γ⟨t⟩iΓi⟨t⟩, the sigmoid produces values between 0 and 1.
* The update gate is multiplied element-wise with the candidate, and this product (Γ⟨t⟩i∗c̃ ⟨t⟩Γi⟨t⟩∗c~⟨t⟩) is used in determining the cell state c⟨t⟩c⟨t⟩.

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

In the code, we'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 WiWi (not "Wu")
* bi is the update gate bias bibi (not "bu")
* it is the forget gate Γ⟨t⟩iΓi⟨t⟩ (not "ut")

#### - Cell state c⟨t⟩c⟨t⟩

* The cell state is the "memory" that gets passed onto future time steps.
* The new cell state c⟨t⟩c⟨t⟩ is a combination of the previous cell state and the candidate value.

##### *Equation*

c⟨t⟩=Γ⟨t⟩f∗c⟨t−1⟩+Γ⟨t⟩i∗c̃ ⟨t⟩(4)(4)c⟨t⟩=Γf⟨t⟩∗c⟨t−1⟩+Γi⟨t⟩∗c~⟨t⟩

##### *Explanation of equation*

* The previous cell state c⟨t−1⟩c⟨t−1⟩ is adjusted (weighted) by the forget gate Γ⟨t⟩fΓf⟨t⟩
* and the candidate value c̃ ⟨t⟩c~⟨t⟩, adjusted (weighted) by the update gate Γ⟨t⟩iΓi⟨t⟩

##### *Variable names and shapes in the code*

* c: cell state, including all time steps, cc shape (na,m,T)(na,m,T)
* c\_next: new (next) cell state, c⟨t⟩c⟨t⟩ shape (na,m)(na,m)
* c\_prev: previous cell state, c⟨t−1⟩c⟨t−1⟩, shape (na,m)(na,m)

#### - Output gate ΓoΓo

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

##### *Equation*

Γ⟨t⟩o=σ(Wo[a⟨t−1⟩,x⟨t⟩]+bo)(5)(5)Γo⟨t⟩=σ(Wo[a⟨t−1⟩,x⟨t⟩]+bo)

##### *Explanation of the equation*

* The output gate is determined by the previous hidden state a⟨t−1⟩a⟨t−1⟩ and the current input x⟨t⟩x⟨t⟩
* The sigmoid makes the gate range from 0 to 1.

##### *Variable names in the code*

* Wo: output gate weight, WoWo
* bo: output gate bias, bobo
* ot: output gate, Γ⟨t⟩oΓo⟨t⟩

#### - Hidden state a⟨t⟩a⟨t⟩

* The hidden state gets passed to the LSTM cell's next time step.
* It is used to determine the three gates (Γf,Γu,ΓoΓf,Γu,Γo) of the next time step.
* The hidden state is also used for the prediction y⟨t⟩y⟨t⟩.

##### *Equation*

a⟨t⟩=Γ⟨t⟩o∗tanh(c⟨t⟩)(6)(6)a⟨t⟩=Γo⟨t⟩∗tanh⁡(c⟨t⟩)

##### *Explanation of equation*

* The hidden state a⟨t⟩a⟨t⟩ is determined by the cell state c⟨t⟩c⟨t⟩ in combination with the output gate ΓoΓo.
* 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(c⟨t⟩)tanh⁡(c⟨t⟩) or keeps those values from being included in the hidden state a⟨t⟩a⟨t⟩

##### *Variable names and shapes in the code*

* a: hidden state, including time steps. aa has shape (na,m,Tx)(na,m,Tx)
* 'a\_prev`: hidden state from previous time step. a⟨t−1⟩a⟨t−1⟩ has shape (na,m)(na,m)
* a\_next: hidden state for next time step. a⟨t⟩a⟨t⟩ has shape (na,m)(na,m)

#### - Prediction y⟨t⟩predypred⟨t⟩

* The prediction in this use case is a classification, so we'll use a softmax.

The equation is:

y⟨t⟩pred=softmax(Wya⟨t⟩+by)ypred⟨t⟩=softmax(Wya⟨t⟩+by)

##### *Variable names and shapes in the code*

* y\_pred: prediction, including all time steps. ypredypred has shape (ny,m,Tx)(ny,m,Tx). Note that (Ty=Tx)(Ty=Tx) for this example.
* yt\_pred: prediction for the current time step tt. y⟨t⟩predypred⟨t⟩ has shape (ny,m)

**Exercise:** Implement lstm\_forward() to run an LSTM over TxTx time-steps.

**Instructions**

* Get the dimensions nx,na,ny,m,Txnx,na,ny,m,Tx from the shape of the variables: x and parameters.
* Initialize the 3D tensors aa, cc and yy.
  + aa: hidden state, shape (na,m,Tx)(na,m,Tx)
  + cc: cell state, shape (na,m,Tx)(na,m,Tx)
  + yy: prediction, shape (ny,m,Tx)(ny,m,Tx) (Note that Ty=TxTy=Tx 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⟨t⟩a⟨t⟩
  + a⟨t⟩a⟨t⟩ stores the hidden state for time step tt. The variable name is a\_next.
  + a⟨0⟩a⟨0⟩, the initial hidden state at time step 0, is passed in when calling the function. The variable name is a0.
  + a⟨t⟩a⟨t⟩ and a⟨0⟩a⟨0⟩ represent a single time step, so they both have the shape (na,m)(na,m)
  + Initialize a⟨t⟩a⟨t⟩ by setting it to the initial hidden state (a⟨0⟩a⟨0⟩) that is passed into the function.
* Initialize c⟨t⟩c⟨t⟩ with zeros.
  + The variable name is c\_next.
  + c⟨t⟩c⟨t⟩ represents a single time step, so its shape is (na,m)(na,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 cc. In other words, **don't** do c\_next = c[:,:,0].
* For each time step, do the following:
  + From the 3D tensor xx, get a 2D slice x⟨t⟩x⟨t⟩ at time step tt.
  + Call the lstm\_cell\_forward function that you defined previously, to get the hidden state, cell state, prediction, and cache.
  + Store the hidden state, cell state and prediction (the 2D tensors) inside the 3D tensors.
  + Also append the cache to the list of caches.