

# Inside the RNN: update equations

An RNN layer, at time step  $t$

- Takes input element  $\mathbf{x}_{(t)}$
- Updates latent state  $\mathbf{h}_{(t)}$
- Optionally outputs  $\mathbf{y}_{(t)}$

according to the equations

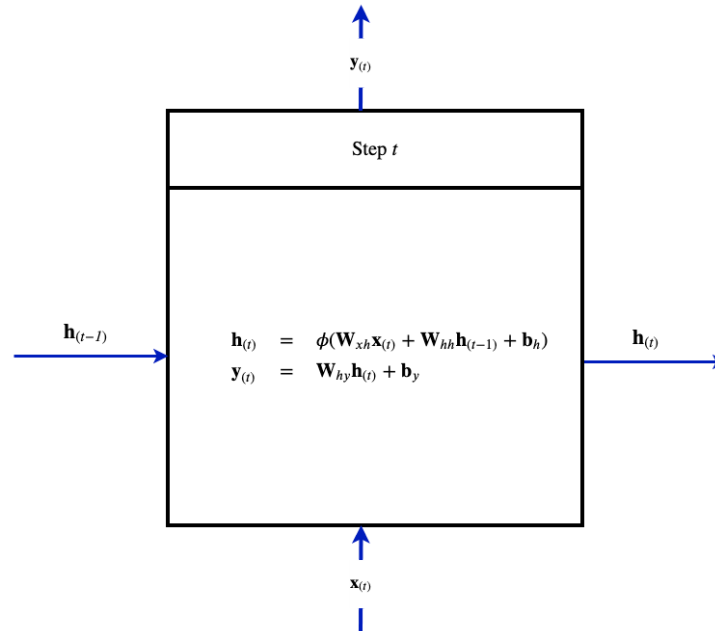
$$\begin{aligned}\mathbf{h}_{(t)} &= \phi(\mathbf{W}_{xh}\mathbf{x}_{(t)} + \mathbf{W}_{hh}\mathbf{h}_{(t-1)} + \mathbf{b}_h) \\ \mathbf{y}_{(t)} &= \mathbf{W}_{hy}\mathbf{h}_{(t)} + \mathbf{b}_y\end{aligned}$$

where

- $\phi$  is an activation function (usually  $\tanh$ )
- $\mathbf{W}$  are the weights of the RNN layer
  - partitioned into  $\mathbf{W}_{xh}, \mathbf{W}_{hh}, \mathbf{W}_{hy}$
  - $\mathbf{W}_{xh}$ : weights that update  $\mathbf{h}_{(t)}$  based on  $\mathbf{x}_{(t)}$
  - $\mathbf{W}_{hh}$ : weights that update  $\mathbf{h}_{(t)}$  based on  $\mathbf{h}_{(t-1)}$
  - $\mathbf{W}_{hy}$ : weights that update  $\mathbf{y}_{(t)}$  based on  $\mathbf{h}_{(t)}$

## RNN

---



## Notes

- The RNN literature uses  $\phi$  rather than  $a_{(l)}$  to denote an activation function
- This is the update equation for a single example  $\mathbf{x}^{(i)}$
- In practice, we can simultaneously update for *multiple examples*
  - The  $m' < m$  examples in a minibatch, as examples are independent
- So if we are counting weights/parameters: it is  $m'$  times bigger

Let's try to understand these equations

$$\mathbf{h}_{(t)} = \phi(\mathbf{W}_{xh}\mathbf{x}_{(t)} + \mathbf{W}_{hh}\mathbf{h}_{(t-1)} + \mathbf{b}_h)$$

$\mathbf{h}_{(t)}$  is the latent state after time step  $t$

- It is a *vector* of length  $||\mathbf{h}||$
- We drop the time subscript as the dimension on each step is the same

$\mathbf{W}_{xh} \mathbf{x}_{(t)}$  must therefore *also be a vector* of length  $||\mathbf{h}||$

- $||\mathbf{W}_{xh}||$  is a matrix of shape  $(||\mathbf{h}|| \times ||\mathbf{x}||)$
- $\mathbf{h}_j$ , the  $j^{th}$  element of latent state  $\mathbf{h}$  is the dot product of row  $j$  of  $\mathbf{W}_{xh}$  and  $\mathbf{x}$
- So  $\mathbf{W}_{xh}^{(j)}$  describes how input  $\mathbf{x}_{(t)}$  influences new state  $h_{(t),j}$

That is: there are separate weights for each  $j$  that describe the interaction of  $\mathbf{h}$  and  $\mathbf{x}$

Similarly,  $\mathbf{W}_{hh} \mathbf{h}_{(t-1)}$  must be a *vector* of length  $\|\mathbf{h}\|$

- $\|\mathbf{W}_{hh}\|$  is a matrix of shape  $(\|\mathbf{h}\| \times \|\mathbf{h}\|)$
- So  $\mathbf{W}_{hh}^{(j)}$  describes how prior state  $\mathbf{h}_{(t-1)}$  influences new state  $\mathbf{h}_{(t),j}$

$\mathbf{b}_h$ , the bias/threshold must also be a vector of length  $||\mathbf{h}||$

- It adjusts the threshold of activation function  $\phi$
- As per our practice: we will usually fold  $\mathbf{b}$  into the weight matrices  $\mathbf{W}_{xh}$ ,  $\mathbf{W}_{hh}$



Finally, activation  $\phi$  maps a vector of length  $||\mathbf{h}||$  to another vector of length  $||\mathbf{h}||$

- The updated state

So updated latent state  $\mathbf{h}_{(t)}$  is influenced

- By the input  $\mathbf{x}_{(t)}$
- The prior latent state  $\mathbf{h}_{(t-1)}$

The second equation

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

is just a "translation" of the latent state  $\mathbf{h}_{(t)}$

- To  $\mathbf{y}_{(t)}$ , the  $t^{th}$  element of the output sequence
- $\|\mathbf{W}_{hy}\|$  is a matrix of shape  $(\|\mathbf{y}\| \times \|\mathbf{h}\|)$ 
  - $\|\mathbf{y}\|$  is the length of each output element and is problem dependent
  - For example: a OHE

It is common to equate  $\mathbf{y}_{(t)} = \mathbf{h}_{(t)}$

- No separate "output"
- Just the latent state
- Particularly when using stacked RNN layers
  - $\mathbf{y}_{(t)}$  becomes the input to the next layer

## Equation in pseudo-matrix form

You will often see a short-hand form of the equation.

Look at  $\mathbf{h}_{(t)}$  as a function of two inputs  $\mathbf{x}, \mathbf{h}_{(t-1)}$ .

We can stack the two inputs into a single matrix.

Stack the two matrices  $\mathbf{W}_{xh}, \mathbf{W}_{hh}$  into a single weight matrix

$$\mathbf{h}_{(t)} = \mathbf{W}\mathbf{I} + \mathbf{b}$$

with

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_{xh} & \mathbf{W}_{hh} \end{bmatrix}$$

$$\mathbf{I} = \begin{bmatrix} \mathbf{x}_{(t)} \\ \mathbf{h}_{(t-1)} \end{bmatrix}$$

## Stacked RNN layers revisited

With the benefit of the RNN update equations, we can clarify how stack RNN layers works.

Let superscript  $[l]$  denote a stacked layer of RNN.

So the RNN update equation for the bottom layer 1 becomes

$$\mathbf{h}_{(t)}^{[1]} = \phi(\mathbf{W}_{xh}\mathbf{x}_{(t)} + \mathbf{W}_{hh}\mathbf{h}_{(t-1)}^{[1]} + \mathbf{b}_h)$$

The RNN update equation for layer  $[l]$  becomes

$$\mathbf{h}_{(t)}^{[l]} = \phi(\mathbf{W}_{xh}\mathbf{h}_{(t)}^{[l-1]} + \mathbf{W}_{hh}\mathbf{h}_{(t-1)}^{[l]} + \mathbf{b}_h)$$

That is: the input to layer  $[l]$  is  $\mathbf{h}_{(t)}^{[l-1]}$  rather than  $\mathbf{x}_{(t)}$

# Loss function

As usual, the objective of training is to find the weights  $\mathbf{W}$  that minimize a loss function

$$\mathcal{L} = L(\hat{\mathbf{y}}, \mathbf{y}; \mathbf{W})$$

which is the average of per example losses  $\mathcal{L}^{(i)}$

$$\mathcal{L} = \frac{1}{m} \sum_{i=1}^m \mathcal{L}^{(i)}$$

When the output is a sequence

- It's important to recognize that the *target* is a sequence too !
- So the per example loss has an added temporal dimension
- Loss per example *per time step*
- Comparing the *predicted*  $t^{th}$  output  $\hat{\mathbf{y}}_{(t)}^{(i)}$  to the  $t^{th}$  target  $\mathbf{y}_{(t)}^{(i)}$

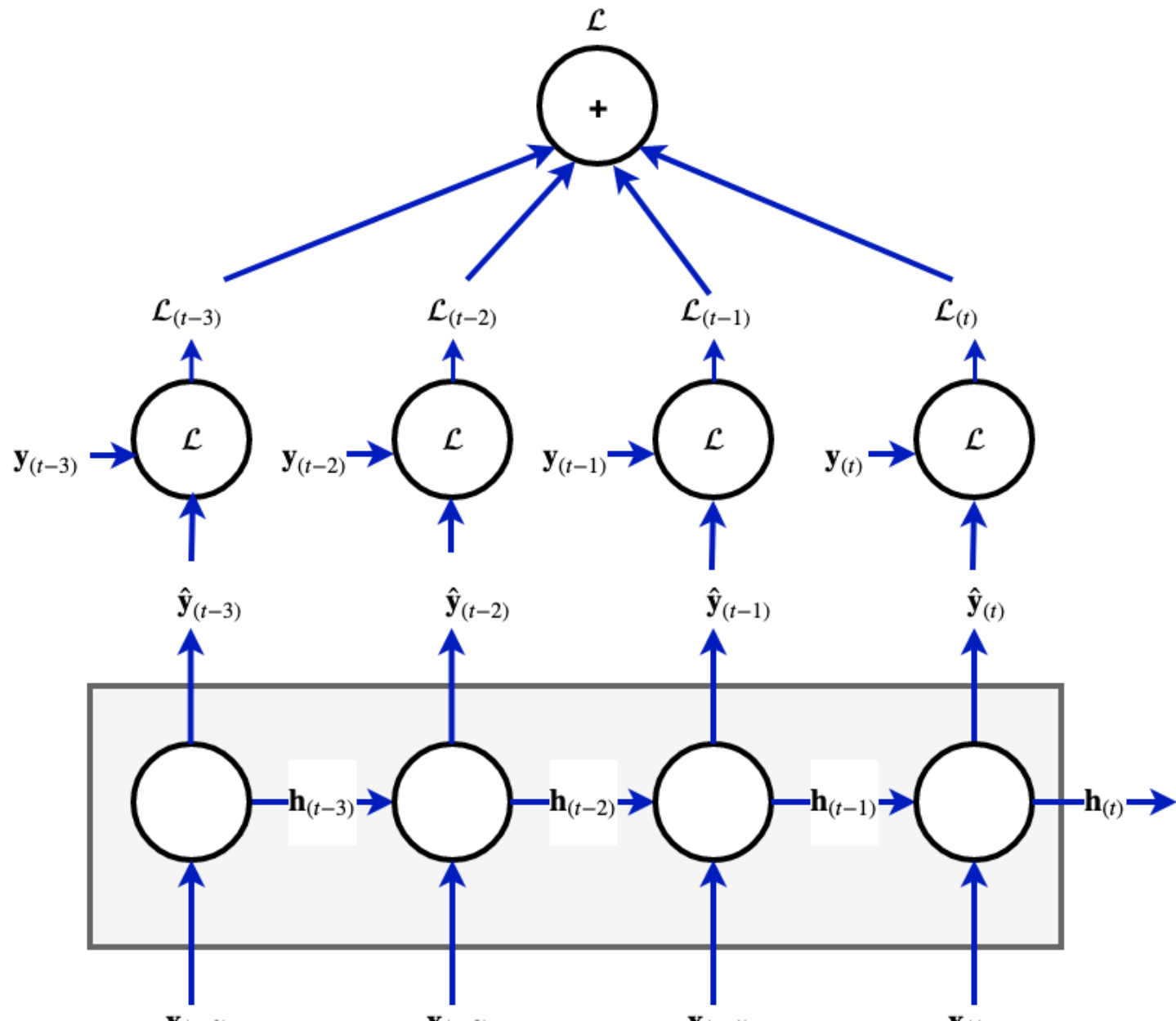
In the case that the API outputs sequences

- $\mathcal{L}^{(\mathbf{i})} = \sum_{t=1}^T \mathcal{L}_{(t)}^{(\mathbf{i})}$

In the case that the API outputs a single value

- $\mathcal{L}^{(\mathbf{i})} = \mathcal{L}_{(T)}$







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

Done