Applied Deep Learning

Chapter 6: Recurrent NNs

Ali Bereyhi

ali.bereyhi@utoronto.ca

Department of Electrical and Computer Engineering

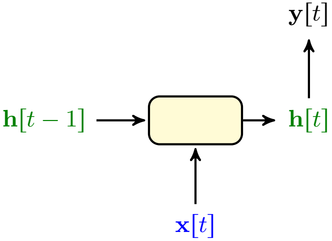
University of Toronto

Fall 2025

Bidirectional RNNs

We have up to now considered *unidirectional RNNs*

we start from beginning of the sequence and move in one direction



But, can't we learn from future input as well?

Bidirectional RNNs

Future entries can have information about past: *say our RNN wants to fill the empty field*

... the color □ *that many people assume is the color of sun ...*

Obviously, future input in the sequence is helping in this example!

+ *But, how can we get information from future?*
- Well, we have the whole sequence: *we could move once from left to right and once from right to left*

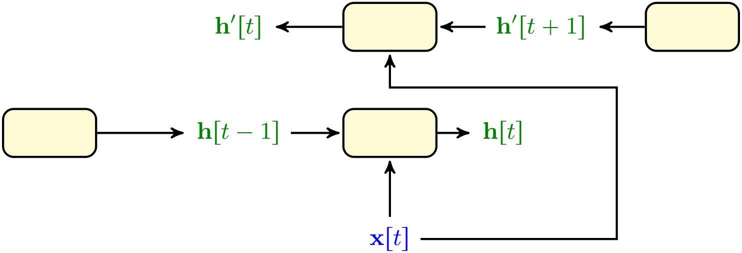
Bidirectional RNNs

Bidirectional RNNs

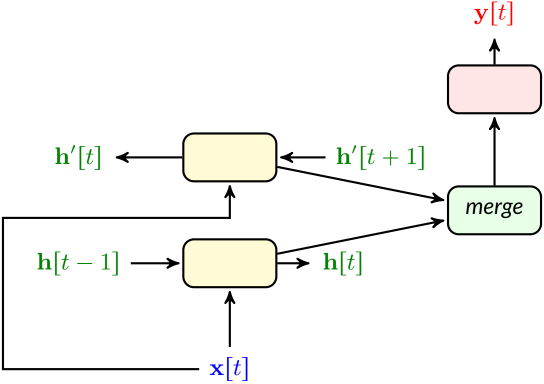
A bidirectional RNN (BRNN) consists of two RNNs

- *one that starts with an initial hidden state at* $t = 0$ *and computes* $\mathbf{h}[t]$ *from* $\mathbf{h}[t-1]$ *and* $\mathbf{x}[t]$
- *another that starts with an initial hidden state at* $t = T + 1$ *and computes* $\mathbf{h}'[t]$ *from* $\mathbf{h}'[t+1]$ *and* $\mathbf{x}[t]$

Output at time t *is determined from merged version of the two hidden states*



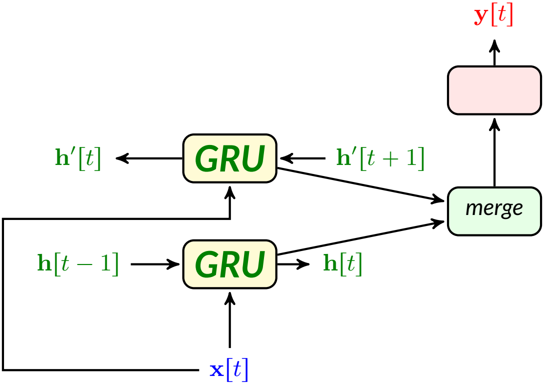
Bidirectional RNNs



+ *What exactly is this merge block?*

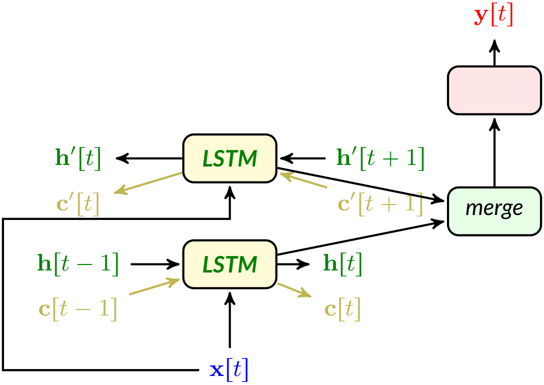
– It gets the two states and *returns a vector that matches output layer*

Bidirectional GRU



- + *Should we use any RNN here?*
- – Sure! *We may use GRU*

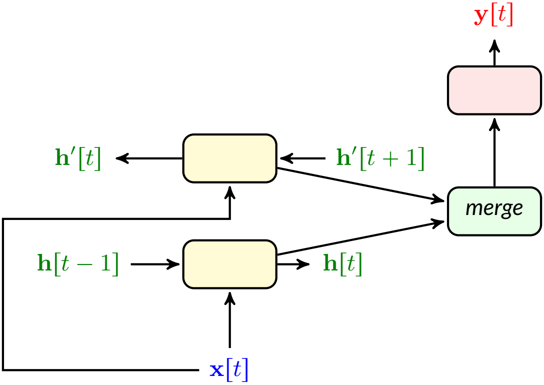
Bidirectional LSTM



+ *Should we use any RNN here?*

– Sure! *We may use LSTM*

RNNs in PyTorch



+ *Any suggestion for merging the hidden states?*
– Sure! Let's see some code

RNNs in PyTorch

In PyTorch, we can *access a basic RNN in* `torch.nn` *module*

```
torch.nn.RNN()
```

We can make it deep by simply choosing `num_layers` *more than one and bidirectional by setting* `bidirectional` *to* `True`. *Same with GRU and LSTM*

```
torch.nn.LSTM()

torch.nn.GRU()
```

In bidirectional case, we get access to both states. To merge them, we could

- *add the two states*
- *average them*
- *concatenate them, i.e.,* $\mathbf{h}_{\mathrm{c}}[t] = (\mathbf{h}[t], \mathbf{h}'[t])$

or do any other operation that we find useful