Intro to Neural Nets

Session 6: Recurrent Neural Networks (RNNs)

Session Agenda

Sequence Data

- Recurrent Neural Networks (RNNs):
    - SimpleRNN
    - LSTM (Long Short-Term Memory)
    - GRU (Gated Recurrent Unit)

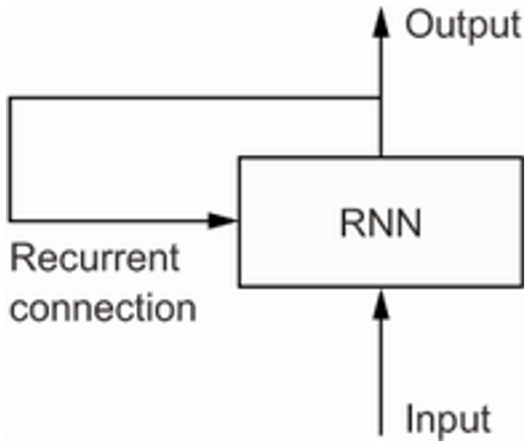
Other Stuff

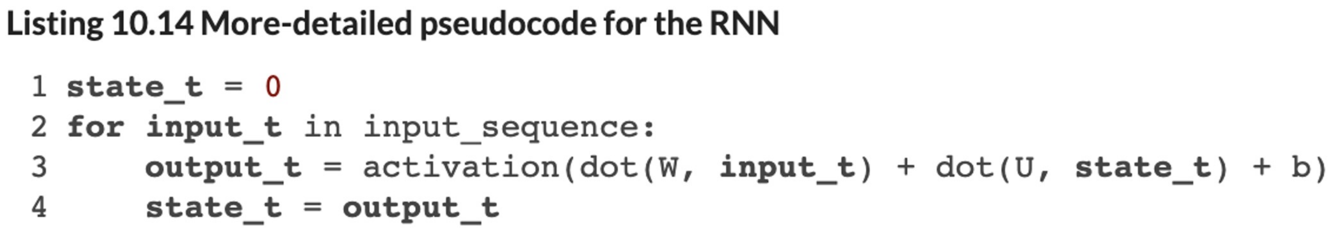
- Bidirectional RNN
- Advanced Forecasting

RNN: Processing /w Memory

Incorporating Memory into an NN

- We include a feedback loop, where output feeds back into the same layer alongside the next input in the sequence (each gets its own separate set of weights, and they are able to interact with one another.



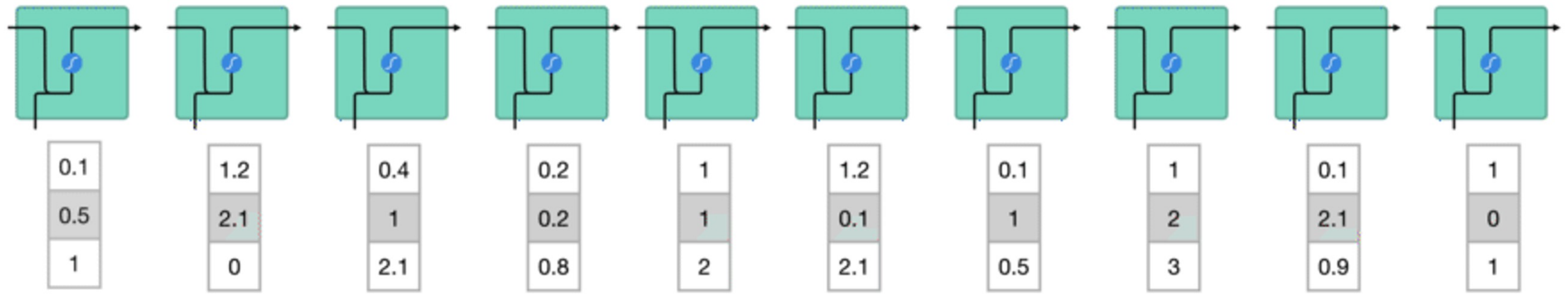

Listing 10.14 More-detailed pseudocode for the RNN

```
state_t = 0
for input_t in input_sequence:
    output_t = activation(dot(W, input_t) + dot(U, state_t) + b)
    state_t = output_t
```

Keras RNN Layers

SimpleRNN

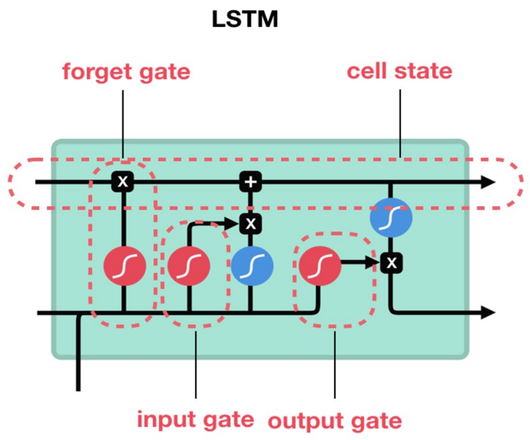
- We can, in practice, unroll an RNN layer into a series as follows.
- This is just a large Dense network with many inputs and many outputs. The inputs are arranged to interact with each other based on their temporal sequencing in the data.

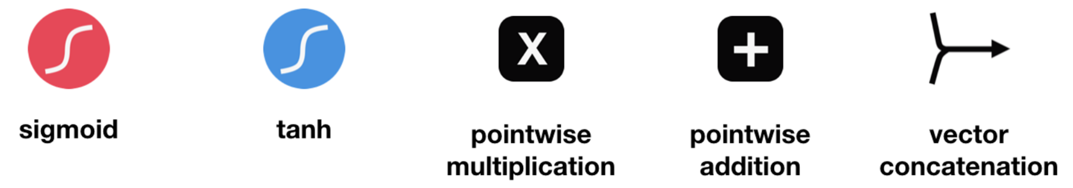


Keras RNN Layers

Long Short-Term Memory (LSTM)

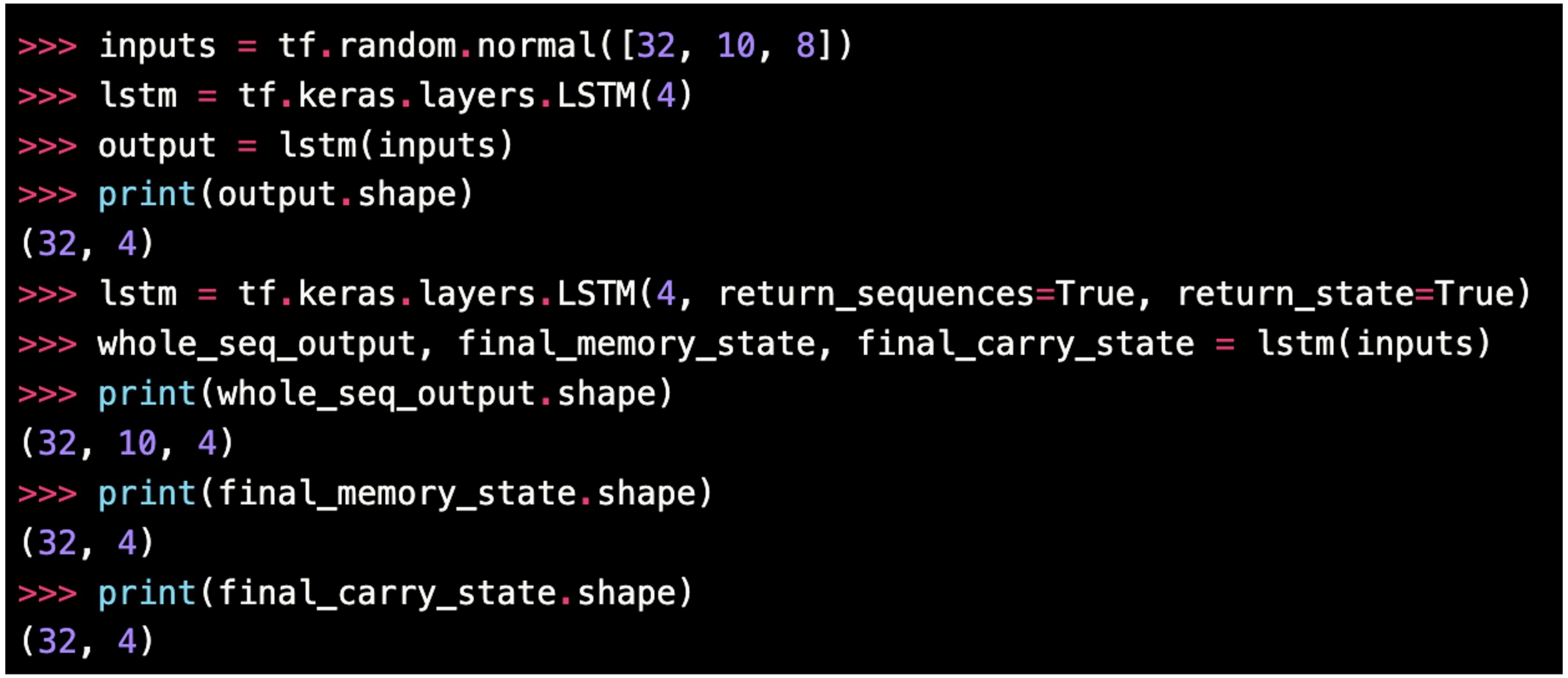
- We cannot use SimpleRNNs to achieve meaningful memory (vanishing gradients arise quickly). We add what's called a "carry track" – an additional connection that combines output at step t, inputs at step t, and the last carry track's output. The topology means your network can 'learn' to use these connections as passthroughs for old info, or it can learn to 'block' that information in favor of more recent information (whatever is useful for accurate prediction).
- These ideas inspired the design of LSTM units, but nothing guarantees that these gates serve these functions. Also, it turns out there's a better architecture for learning what to attend to…





Keras RNN Layers

Long Short-Term Memory (LSTM)

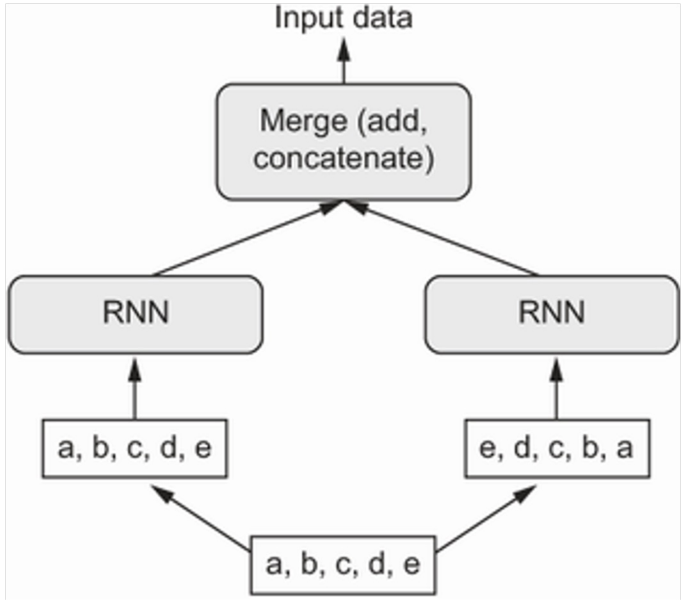


```
>>> inputs = tf.random.normal([32, 10, 8])
>>> lstm = tf.keras.layers.LSTM(4)
>>> output = lstm(inputs)
>>> print(output.shape)
(32, 4)
>>> lstm = tf.keras.layers.LSTM(4, return_sequences=True, return_state=True)
>>> whole_seq_output, final_memory_state, final_carry_state = lstm(inputs)
>>> print(whole_seq_output.shape)
(32, 10, 4)
>>> print(final_memory_state.shape)
(32, 4)
>>> print(final_carry_state.shape)
(32, 4)
```

Bidirectional RNNs

Some Sequences Yield Information in Both Directions

- Consider that, in language, words that come later in a sentence can be predictive of what came before.
- Bidirectional RNNs implement a standard RNN, but they also incorporate a parallel layer implementation that takes the sequence ordered in reverse.



Questions?