Lesson 6 Practice Exercise

Start

- Assignment (https://canvas.uw.edu/courses/1260524/assignments/4656100?module_item_id=9186173)
- LSTM Tutorial (https://adventuresinmachinelearning.com/keras-lstm-tutorial/)
- Understand the Difference Between Return Sequences and Return States for LSTMs in Keras (https://machinelearningmastery.com/return-sequences-and-return-states-for-lstms-in-keras/)
- MIT Deep Learning Book (https://www.deeplearningbook.org/)
- KERAS LSTM (https://keras.io/layers/recurrent/)

In this exercise we will examine the outputs from simple untrained LSTM and GRU networks, in order to better understand the Keras API.

There are four parts, and 12 short questions to answer as you run the code below:

- LSTM: run with return_sequences=False
- LSTM: run with return_sequences=True
- GRU: run with return_sequences=False
- GRU: run with return_sequences=True

In [45]:
```
from keras.models import Model
from keras.layers import Input, LSTM, GRU
import numpy as np

timesteps = 8
num_features = 2
num_nodes = 3

# Create some random data
data = np.random.randn(1, timesteps, num_features)
print(data)
```

```
[[[-1.29732544  0.88370345]
  [ 0.96809361 -1.83162665]
  [ 0.63462653 -0.29527256]
  [-0.15880219  0.47193369]
  [-1.28062212  1.16307752]
  [-0.94114256 -0.97424082]
  [ 0.795397    0.176578  ]
  [-0.77534238  1.60992856]]]
```

Q1: What do the dimensions of the data represent? E.g., what do the 1st dimension, the 2nd, and the 3rd represent here?

`np.random.randn` return a sample from the normal distribution with the parameters 1, timesteps, numfeatures, it return one array of timesteps rows and numfeatures columns This means that we will observe 2 features at each time steps

In [46]:
```
inputs = Input(shape=(timesteps, num_features))
rnn = LSTM(num_nodes, return_state=True)
x = rnn(inputs)

model = Model(inputs=inputs, outputs=x)
output, h, c = model.predict(data)
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_19 (InputLayer)        (None, 8, 2)              0         
_________________________________________________________________
lstm_13 (LSTM)               [(None, 3), (None, 3), (N 72        
=================================================================
Total params: 72
Trainable params: 72
Non-trainable params: 0
_________________________________________________________________
```

In [47]:
```
print("output:", output)
print("h:", h)
print("c:", c)
```

```
output: [[-0.23299168 -0.12314837  0.12369747]]
h: [[-0.23299168 -0.12314837  0.12369747]]
c: [[-0.722294   -0.20794256  0.24609338]]
```

Q2: What do these "predict" return values (output, h, and c) represent? Describe what they mean.

`output` represents the hidden state for the last time step

`h` represents the hidden state for the last time step

`c` represents the cell state for the last time step

Q3: What are the dimensions of output, h, and c?

the dimension of `output` is a vector of the same dimension as the number of nodes

the dimension of `h` is a vector of the same dimension as the number of nodes

the dimension of `c` is a vector of the same dimension as the number of nodes

```
In [48]: output.shape
Out[48]: (1, 3)
```

```
In [49]: h.shape
Out[49]: (1, 3)
```

```
In [50]: c.shape
Out[50]: (1, 3)
```

Q4: Is there any duplication among the values of (output, h, c)? If so, explain.

In this particular case, `output` and `h` are the same

`h` and `output` are equal because we asked to return the last state with output and h return the last state

Q5: Change the LSTM to use "return_state=False" and modify the code accordingly to avoid errors. What does the LSTM return now?

In [62]:
```
inputs = Input(shape=(timesteps, num_features))
rnn = LSTM(num_nodes, return_state=False)
x = rnn(inputs)

model = Model(inputs=inputs, outputs=x)
output = model.predict(data)
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_27 (InputLayer)        (None, 8, 2)              0         
_________________________________________________________________
lstm_18 (LSTM)               (None, 3)                 72        
=================================================================
Total params: 72
Trainable params: 72
Non-trainable params: 0
_________________________________________________________________
```

In [52]:
```
print("output:", output)
```

```
output: [[-0.02160713 -0.0593508   0.10295806]]
```

output is 2D tensor with shape `(batch_size,dimensionality of the output space == num nodes)`

LSTM: run with return_sequences=True

In [67]:
```
inputs = Input(shape=(timesteps, num_features))
rnn = LSTM(num_nodes, return_state=True, return_sequences=True)
x = rnn(inputs)

model = Model(inputs=inputs, outputs=x)
output, h, c = model.predict(data)

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_31 (InputLayer)        (None, 8, 2)              0         
_________________________________________________________________
lstm_21 (LSTM)               [(None, 8, 3), (None, 3), 72        
=================================================================
Total params: 72
Trainable params: 72
Non-trainable params: 0
_________________________________________________________________
```

In [68]:
```
print("output:", output)
print("h:", h)
print("c:", c)
```

```
output: [[[-0.03620154  0.26969242  0.03491046]
  [ 0.08662518  0.00840403 -0.01776307]
  [ 0.0494854  -0.10038266 -0.01998534]
  [ 0.0010259   0.02087294  0.00434066]
  [-0.06275047  0.2853172   0.03018308]
  [ 0.11046072  0.16031958  0.06956935]
  [ 0.00783136  0.05344884 -0.00707584]
  [-0.0901691   0.26428902  0.00217168]]]
h: [[-0.0901691   0.26428902  0.00217168]]
c: [[-0.22954567  0.47477117  0.00665541]]
```

Q6: Now, what are the dimensions of output, h, and c?

In [54]:
```
output.shape
```

Out[54]:
```
(1, 8, 3)
```

In [55]:
```
h.shape
```

Out[55]:
```
(1, 3)
```

In [56]:
```
c.shape
```

Out[56]:
```
(1, 3)
```

Q7: What does the output variable represent now? Explain the dimensions of the returned matrix (e.g., what does the 1st dimension represent? What does the 2nd represent? The 3rd? etc.)

`output` is a 3D tensor with `(batch_size, timesteps, dimensionality of the output space)`

`h` & `c` are state tensors

GRU

In [69]:
```
inputs = Input(shape=(timesteps, num_features))
rnn = GRU(num_nodes, return_state=True)
x = rnn(inputs)
model = Model(inputs=inputs, outputs=x)
output, h = model.predict(data)

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_32 (InputLayer)        (None, 8, 2)              0         
_________________________________________________________________
gru_10 (GRU)                 [(None, 3), (None, 3)]    54        
=================================================================
Total params: 54
Trainable params: 54
Non-trainable params: 0
_________________________________________________________________
```

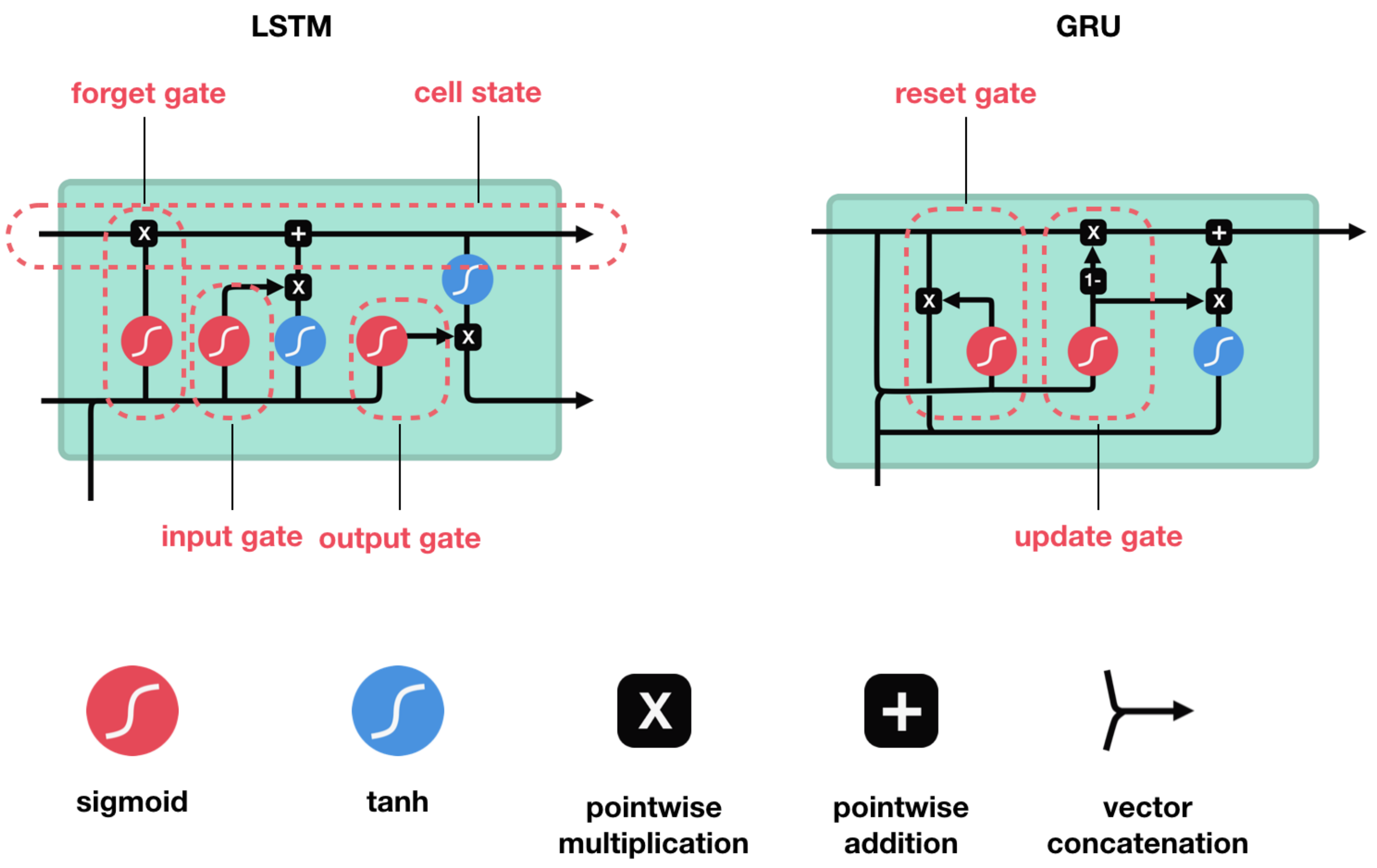
In [70]:
```
print("o:", output)
print("h:", h)
```

```
o: [[0.09062721 0.10693793 0.02225437]]
h: [[0.09062721 0.10693793 0.02225437]]
```

Q8: What happened to the variable "c"? Why does the GRU only return a tuple of 2 values as opposed to 3?

- Illustrated Guide to LSTM’s and GRU’s: A step by step explanation (https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21)

there is only hidden state in GRU cell.



Q9: What are the dimension of output and h?

```
In [71]: output.shape
Out[71]: (1, 3)
```

In [72]:
```
h.shape
```
Out[72]:
```
(1, 3)
```

Q10: Is there any duplication between the values of output and h? If so, explain.

In [73]:
```
np.testing.assert_array_equal(output, h)
```

`h` and `output` are equal because we asked to return the last state with output and h return the last state

In [79]:
```
inputs = Input(shape=(timesteps, num_features))
rnn = GRU(num_nodes, return_state=True, return_sequences=True)
x = rnn(inputs)
model = Model(inputs=inputs, outputs=x)
output, h = model.predict(data)

print("o:", output)
print("h:", h)
```
```
o: [[[-0.3345641  -0.01846615  0.49711123]
  [ 0.5451222  -0.4379254   0.24848235]
  [ 0.5874258  -0.12496339 -0.01109824]
  [ 0.18641673  0.05660372  0.15563084]
  [-0.23902743  0.11323315  0.6097513 ]
  [-0.17114875 -0.3527349   0.06730883]
  [ 0.21735635  0.05933596  0.04711872]
  [-0.24374607  0.2575965   0.6487629 ]]]
h: [[-0.24374607  0.2575965   0.6487629 ]]
```

Q11: Now, what are the dimensions of output and h?

In [80]:
```
output.shape
```
Out[80]:
```
(1, 8, 3)
```

In [81]:
```
h.shape
```
Out[81]:
```
(1, 3)
```

Q12: Is there any duplication within the values of output and h? If so, explain.

`h` and `output` are not equal we return as output return a sequence of state, only the last row of output is equal to `h`

```
In [84]: output[0][-1]

Out[84]: array([-0.24374607,  0.2575965 ,  0.6487629 ], dtype=float32)

In [85]: np.testing.assert_array_equal(output[0][-1], h[0])

In [ ]:
```