I still don't understand the output reshape part

DLNDF Project 3: Generate TV scripts
Intro to Recurrent Neural Networks

Kai4b08b83b91e682c3a 2017-05-11 18:40:36 UTC #1

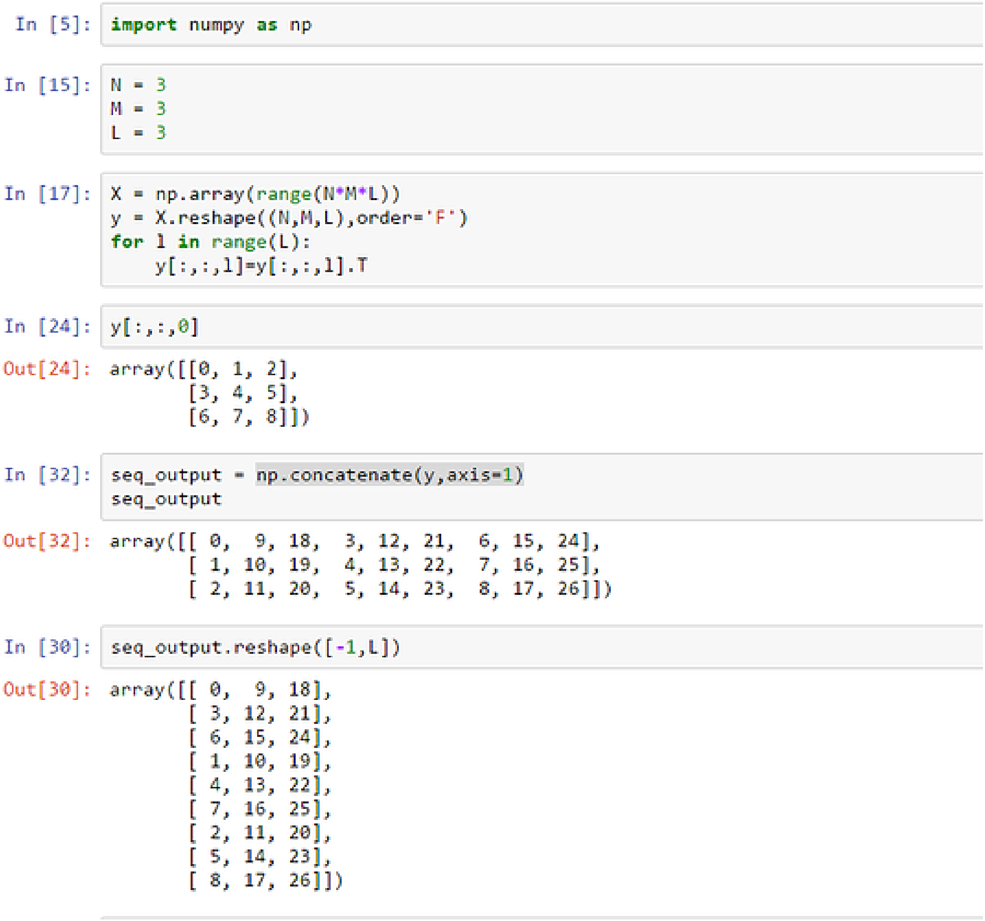
Hi have already read this post: **https://discussions.udacity.com/t/reshaping-the-hidden-layer-outputs/233296**
But i am still super confused about the reshaping lstm output part.

in Anna KaRNNa exercise, we did something like this:

```
seq_output = tf.concat(lstm_output, axis=1)
x = tf.reshape(seq_output, [-1, in_size])
```

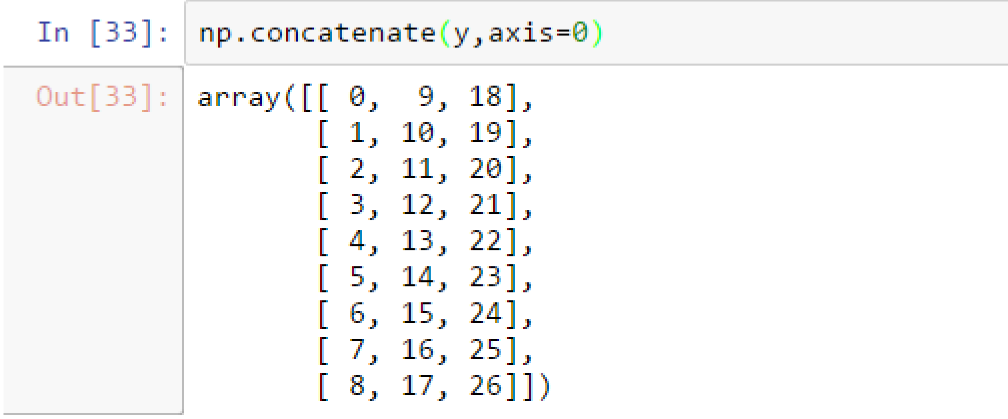
so the lstm_output is N by M by L matrix (N is the batch size, M is the number of steps and L is the hidden layer size). and the ideas is that we are tying to make it into (M*N) by L matrix for performing a fully connected layer. I understand all those fine. But i don't feel like the sample code is doing what we want to do.

i did follow exercise:



```
In [5]:  import numpy as np

In [15]: N = 3
         M = 3
         L = 3

In [17]: X = np.array(range(N*M*L))
         y = X.reshape((N,M,L),order='F')
         for l in range(L):
             y[:,:,l]=y[:,:,l].T

In [24]: y[:,:,0]

Out[24]: array([[0, 1, 2],
                [3, 4, 5],
                [6, 7, 8]])

In [32]: seq_output = np.concatenate(y,axis=1)
         seq_output

Out[32]: array([[ 0,  9, 18,  3, 12, 21,  6, 15, 24],
                [ 1, 10, 19,  4, 13, 22,  7, 16, 25],
                [ 2, 11, 20,  5, 14, 23,  8, 17, 26]])

In [30]: seq_output.reshape([-1,L])

Out[30]: array([[ 0,  9, 18],
                [ 3, 12, 21],
                [ 6, 15, 24],
                [ 1, 10, 19],
                [ 4, 13, 22],
                [ 7, 16, 25],
                [ 2, 11, 20],
                [ 5, 14, 23],
                [ 8, 17, 26]])
```

i thought we want some thing like this:



```
In [33]: np.concatenate(y,axis=0)

Out[33]: array([[ 0,  9, 18],
                [ 1, 10, 19],
                [ 2, 11, 20],
                [ 3, 12, 21],
                [ 4, 13, 22],
                [ 5, 14, 23],
                [ 6, 15, 24],
                [ 7, 16, 25],
                [ 8, 17, 26]])
```

Since it is an RNN, so I assume the order dose matter here, Please let me know where i get it wrong.

Thanks

Reshaping the hidden layer outputs

RNN Output: How the array is 3d

Question Regarding RNN output

I don't understand def build_output.

Kai4b08b83b91e682c3a 2017-05-14 14:00:43 UTC #2

Sorry to mention you directly. **@rahul_ahuja** . but no one is answering this post.

rahul_ahuja 2017-05-14 20:44:02 UTC #3

Hi **@Kai4b08b83b91e682c3a**

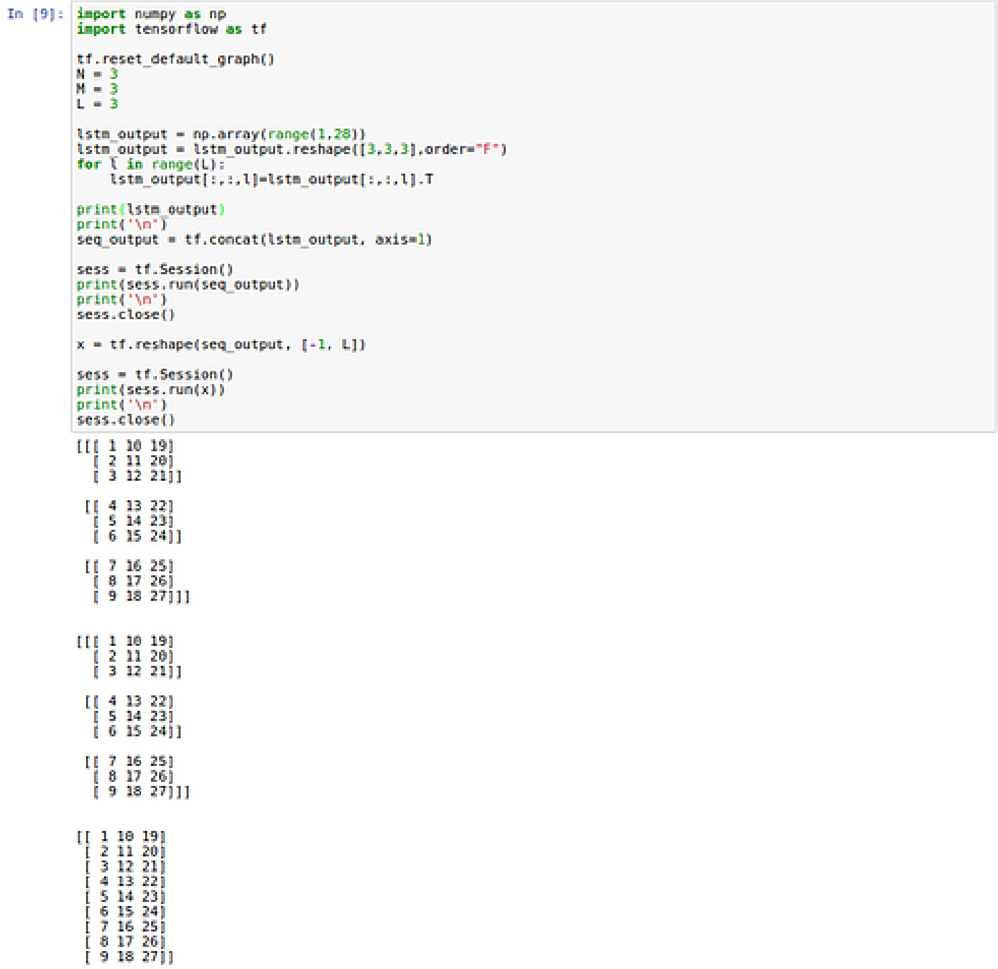
You can also initialize the LSTM output matrix with the 3D matrix as described in the notebook and reshape it the way it has been done in the notebook.

```
import numpy as np
import tensorflow as tf

N = 3
M = 3
L = 3

lstm_output = np.random.rand(N,M,L)
print(lstm_output)

seq_output = tf.concat(lstm_output, axis=1)
```

```
sess = tf.Session()
print(sess.run(seq_output))
sess.close()

x = tf.reshape(seq_output, [-1, L])

sess = tf.Session()
print(sess.run(x))
sess.close()
```

Kai4b08b83b91e682c3a 2017-05-14 21:39:26 UTC #4

Thanks! I tried your code. So apparently np and tf are handling this differently.



In [9]:

```
import numpy as np
import tensorflow as tf

tf.reset_default_graph()
N = 3
M = 3
L = 3

lstm_output = np.array(range(1,28))
lstm_output = lstm_output.reshape([3,3,3],order="F")
for l in range(L):
    lstm_output[:,:,l]=lstm_output[:,:,l].T

print(lstm_output)
print('\n')
seq_output = tf.concat(lstm_output, axis=1)

sess = tf.Session()
print(sess.run(seq_output))
print('\n')
sess.close()

x = tf.reshape(seq_output, [-1, L])

sess = tf.Session()
print(sess.run(x))
print('\n')
sess.close()
```

```
[[[ 1 10 19]
  [ 2 11 20]
  [ 3 12 21]]

 [[ 4 13 22]
  [ 5 14 23]
  [ 6 15 24]]

 [[ 7 16 25]
  [ 8 17 26]
  [ 9 18 27]]]


[[[ 1 10 19]
  [ 2 11 20]
  [ 3 12 21]]

 [[ 4 13 22]
  [ 5 14 23]
  [ 6 15 24]]

 [[ 7 16 25]
  [ 8 17 26]
  [ 9 18 27]]]


[[ 1 10 19]
 [ 2 11 20]
 [ 3 12 21]
 [ 4 13 22]
 [ 5 14 23]
 [ 6 15 24]
 [ 7 16 25]
 [ 8 17 26]
 [ 9 18 27]]
```

It seems to put everything in the right order. Though, tf.concat doesn't seems to do anything.

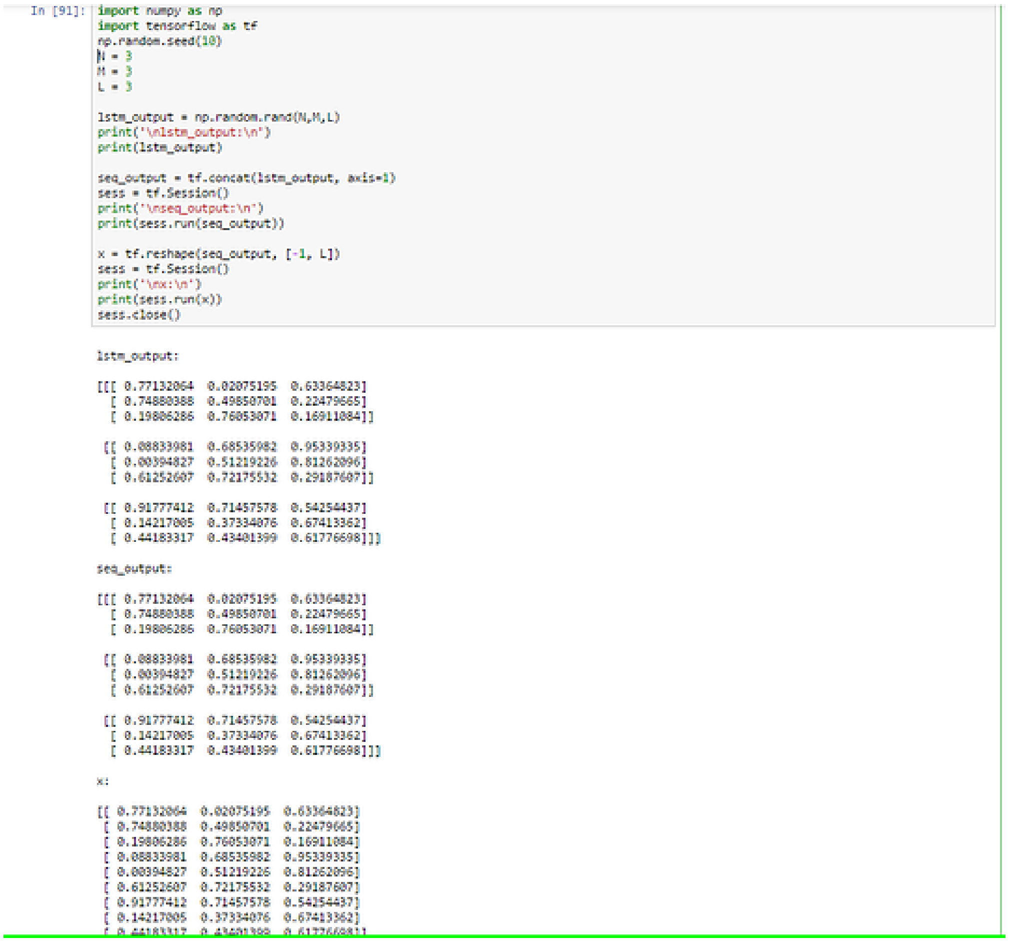
rahul_ahuja 2017-05-14 21:46:09 UTC #5

Yes, even I realized when I just worked on it.
There might be a reason to use tf.concat but I can't think of it right now.
Atleast you're good with it now and know how to print tensorflow arrays. 🙂

alaylien11 2017-06-07 07:04:37 UTC #6

If tf.concat doesn't seem to do anything, then what if we remove that code? Because the reshaping does work the same way without tf.concat command, as shown below:

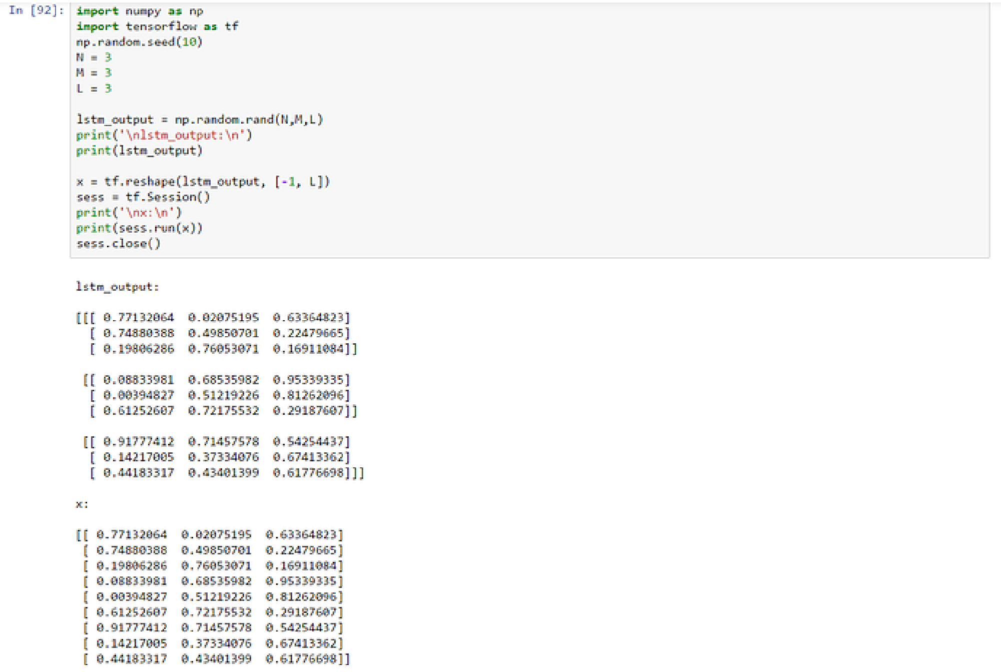
1) With tf.concat command:



```
In [91]: import numpy as np
         import tensorflow as tf
         np.random.seed(10)
         N = 3
         M = 3
         L = 3

         lstm_output = np.random.rand(N,M,L)
         print('\nlstm_output:\n')
         print(lstm_output)

         seq_output = tf.concat(lstm_output, axis=1)
         sess = tf.Session()
         print('\nseq_output:\n')
         print(sess.run(seq_output))

         x = tf.reshape(seq_output, [-1, L])
         sess = tf.Session()
         print('\nx:\n')
         print(sess.run(x))
         sess.close()
```

```
lstm_output:

[[[ 0.77132064  0.02075195  0.63364823]
  [ 0.74880388  0.49850701  0.22479665]
  [ 0.19806286  0.76053071  0.16911084]]

 [[ 0.08833981  0.68535982  0.95339335]
  [ 0.00394827  0.51219226  0.81262096]
  [ 0.61252607  0.72175532  0.29187607]]

 [[ 0.91777412  0.71457578  0.54254437]
  [ 0.14217005  0.37334076  0.67413362]
  [ 0.44183317  0.43401399  0.61776698]]]

seq_output:

[[[ 0.77132064  0.02075195  0.63364823]
  [ 0.74880388  0.49850701  0.22479665]
  [ 0.19806286  0.76053071  0.16911084]]

 [[ 0.08833981  0.68535982  0.95339335]
  [ 0.00394827  0.51219226  0.81262096]
  [ 0.61252607  0.72175532  0.29187607]]

 [[ 0.91777412  0.71457578  0.54254437]
  [ 0.14217005  0.37334076  0.67413362]
  [ 0.44183317  0.43401399  0.61776698]]]

x:

[[ 0.77132064  0.02075195  0.63364823]
 [ 0.74880388  0.49850701  0.22479665]
 [ 0.19806286  0.76053071  0.16911084]
 [ 0.08833981  0.68535982  0.95339335]
 [ 0.00394827  0.51219226  0.81262096]
 [ 0.61252607  0.72175532  0.29187607]
 [ 0.91777412  0.71457578  0.54254437]
 [ 0.14217005  0.37334076  0.67413362]
 [ 0.44183317  0.43401399  0.61776698]]
```

2) Without tf.concat command:



```
In [92]: import numpy as np
         import tensorflow as tf
         np.random.seed(10)
         N = 3
         M = 3
         L = 3

         lstm_output = np.random.rand(N,M,L)
         print('\nlstm_output:\n')
         print(lstm_output)

         x = tf.reshape(lstm_output, [-1, L])
         sess = tf.Session()
         print('\nx:\n')
         print(sess.run(x))
         sess.close()
```

```
lstm_output:

[[[ 0.77132064  0.02075195  0.63364823]
  [ 0.74880388  0.49850701  0.22479665]
  [ 0.19806286  0.76053071  0.16911084]]

 [[ 0.08833981  0.68535982  0.95339335]
  [ 0.00394827  0.51219226  0.81262096]
  [ 0.61252607  0.72175532  0.29187607]]

 [[ 0.91777412  0.71457578  0.54254437]
  [ 0.14217005  0.37334076  0.67413362]
  [ 0.44183317  0.43401399  0.61776698]]]

x:

[[ 0.77132064  0.02075195  0.63364823]
 [ 0.74880388  0.49850701  0.22479665]
 [ 0.19806286  0.76053071  0.16911084]
 [ 0.08833981  0.68535982  0.95339335]
 [ 0.00394827  0.51219226  0.81262096]
 [ 0.61252607  0.72175532  0.29187607]
 [ 0.91777412  0.71457578  0.54254437]
 [ 0.14217005  0.37334076  0.67413362]
 [ 0.44183317  0.43401399  0.61776698]]
```

rahul_ahuja 2017-06-07 11:41:09 UTC #7

Yes, it clearly seems to be the same.

> what if we remove that code?

Can you confirm if you've been getting the same training results by removing `tf.concat` in the notebook?

---

alaylien11 2017-06-07 14:08:11 UTC #8

Sure, I am still working on the code.
I will confirm the results once I'm done training for both instances!

---

zhi_2902798234802459 2017-06-09 13:22:08 UTC #9

Hey, I am also confused with that the tf.concat() function is used here. I scrutinized the code and found that the output of the lstm layer is actually a 4D tensor (batch_size, num_steps, number_classes, lstm_size). It is not a 3D tensor because the input is one_hot transformed in prior. So, I do not know why they use a tf.concat() function for the 1 axis here.

---

rahul_ahuja 2017-06-09 20:06:58 UTC #10

Hi **@zhi_2902798234802459**

The only way to confirm the use of `tf.concat()` is to run and see the difference in results when using and without using `tf.concat()` function.

LSTM output is 3D tensor. Here's a way to find out;

Put the below print statement into `build_output` function and then when printing the results, you will also see the LSTM output shape.
`print("LSTM OUTPUT SHAPE: ", lstm_output.get_shape())`

---

zhi_2902798234802459 2017-06-10 01:30:19 UTC #11

Yes,you are right. I made a big mistake yesterday.

Home **Categories** **FAQ/Guidelines** **Terms of Service** **Privacy Policy**