SHIVA VERMA · 3Y AGO · 25,015 VIEWS

29 Copy & Edit 38

Input and Output shape in LSTM (Keras)

Python · [Private Datasource]

Run
11.2s

Version 2 of 2

When I started working with the LSTM networks, I was very confused about input and output shape This Kernel will help you to understand the Input and Output shape of the LSTM network. I assume that you already know about the LSTM theoretically. I am using the Keras library in this tutorial.

In [1]:

```
import keras
```

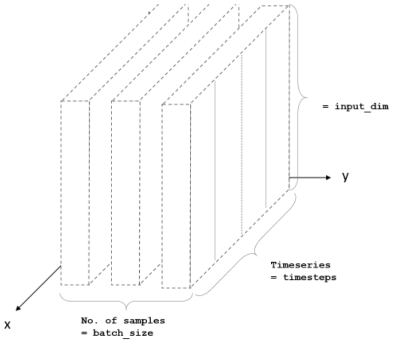
```
Using TensorFlow backend.
```

First, let's understand the Input and its shape in Keras LSTM.

In [2]:

```
from PIL import Image
Image.open('../input/img.png')
```

Out[2]:



- You always have to give a three-dimensional array as an input to your LSTM network (refer to the above image). Where the first dimension represents the batch size, the second dimension represents the number of time-steps you are feeding a sequence. And the third dimension represents the number of units in one input sequence. For example, input shape looks like `(batch_size, time_steps, seq_len)`. Let's look at an example in Keras.

In [3]:

```
model = keras.models.Sequential()

model.add(keras.layers.LSTM(units=3, input_shape=(2,10)))
```

- Let's look at `input_shape` argument. Though it looks like that `input_shape` requires a 2D array, it actually requires a 3D array. In the example above `input_shape` is `(2,10)` which means number of time steps are 2 and number of input units is 10. And you can give any size for a batch. So your input array shape looks like `(batch_size, 2, 10)`.

In [4]:

```
model = keras.models.Sequential()

model.add(keras.layers.LSTM(units=3, batch_input_shape=(8,2,10)))
```

- You can also give an argument called `batch_input_shape` instead of `input_shape`. The difference here is that you have to give a fixed batch size now and your input array shape will look like `(8,2,10)`. Now if you try to give different batch size, you will get an error.

Let's look at the output and its shape in the LSTM network.

In [5]:

```
model = keras.models.Sequential()

model.add(keras.layers.LSTM(units=3, input_shape=(2,10), return_sequences=False))

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_3 (LSTM)                (None, 3)                 168       
=================================================================
Total params: 168
Trainable params: 168
Non-trainable params: 0
_________________________________________________________________
```

- Let's look at the other arguments. `units` is number of the output units in the LSTM which is 3 here. So output shape is `(None, 3)`. First dimension of output is None because we do not know the batch size in advance. So actual output shape will be `(batch_size, 3)` here.

In [6]:

```
model = keras.models.Sequential()

model.add(keras.layers.LSTM(units=3, batch_input_shape=(8,2,10), return_sequences=False))

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_4 (LSTM)                (8, 3)                    168       
=================================================================
Total params: 168
Trainable params: 168
Non-trainable params: 0
_________________________________________________________________
```

- Here you can see that I defined `batch_size` in advance and the output shape is `(8,3)` which makes sense.

In [7]:

```
model = keras.models.Sequential()

model.add(keras.layers.LSTM(units=3, batch_input_shape=(8,2,10), return_sequences=True))

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_5 (LSTM)                (8, 2, 3)                 168       
=================================================================
Total params: 168
Trainable params: 168
Non-trainable params: 0
_________________________________________________________________
```

- Now, look at the other argument which is `return_sequences`. This argument tells Whether to return the output at each time steps instead of the final time step. Now the output shape is 3D array, not a 2D array. And the shape of the array is `(8,2,3)`. You can see that there is one extra dimension in between which represent number of time steps.

Summary

- The input of the LSTM is always is a 3D array. `(batch_size, time_steps, seq_len)` .
- The output of the LSTM could be a 2D array or 3D array depending upon the return_sequences argument.
- If return_sequence is False, the output is a 2D array. `(batch_size, units)`
- If return_sequence is True, the output is a 3D array. `batch_size, time_steps, units)`

License

Continue exploring

Data
1 input and 0 output →

Logs
11.2 second run - successful →

Comments
6 comments →