exam 개인학습

Lec 25

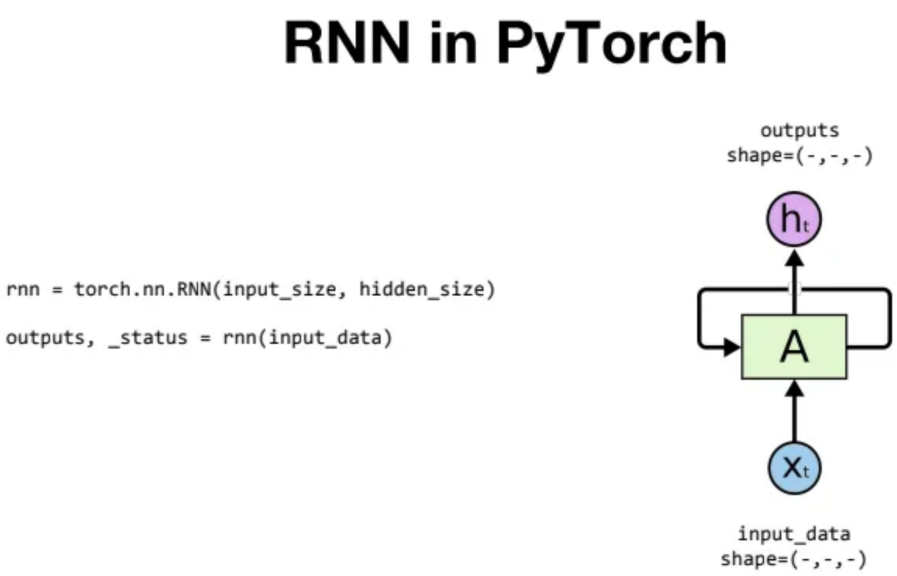
Type something...

01) RNN Basics

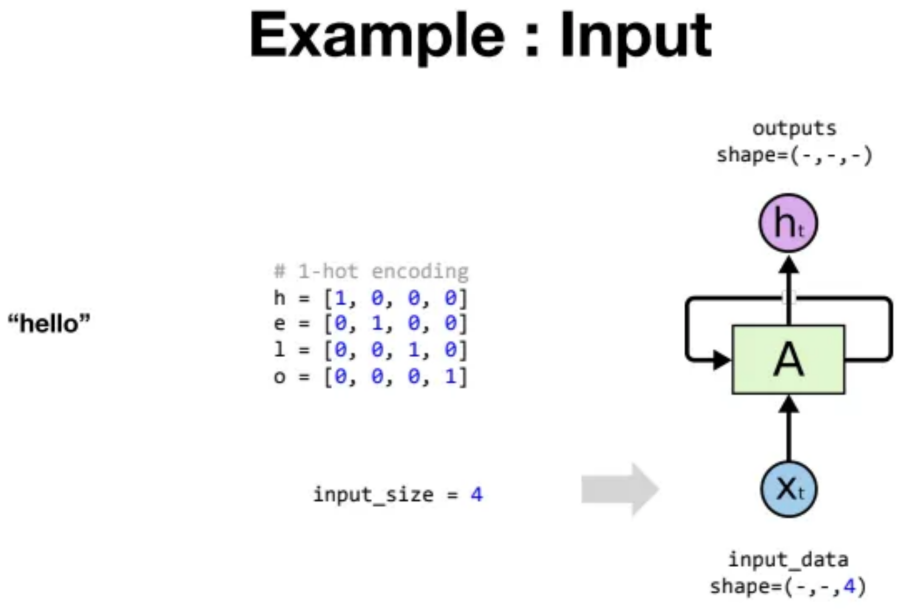
01_RNN in PyTorch

01) RNN Basics

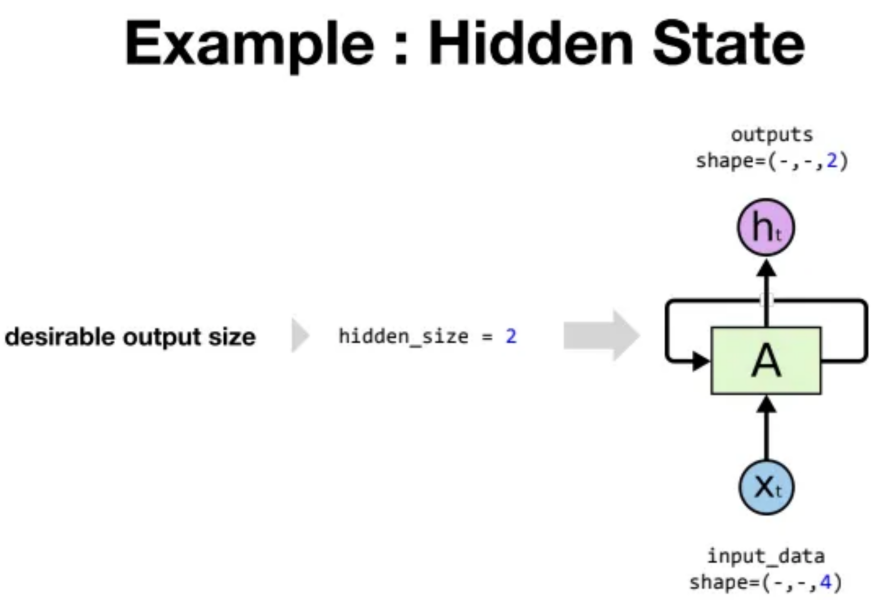
01_RNN in PyTorch



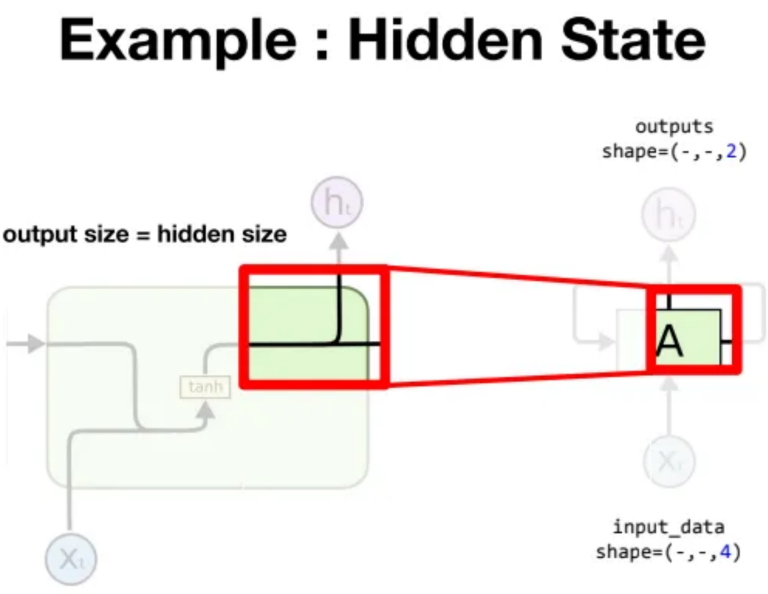
- RNN은 input_size와 hidden_size만 있으면 정의가 가능하다.
- 각각의 shape가 어떻게 구성되어 있는지 알아볼 것이다.
- Input data shape = `(batch_size, sequence_length, input_size)`
- output data shape = `(batch_size, sequence_length, hidden_size)`



- Input_size는 Input_data의 마지막 dim을 담당한다.
- 문자열을 벡터로 받기 위해 `1-hot encdoing` 을 사용하였다.

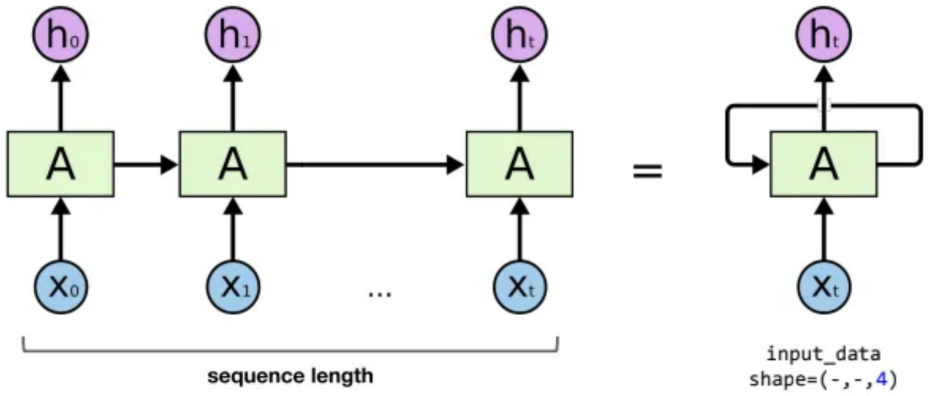


- Hidden_size는 output_size와 동일하다.

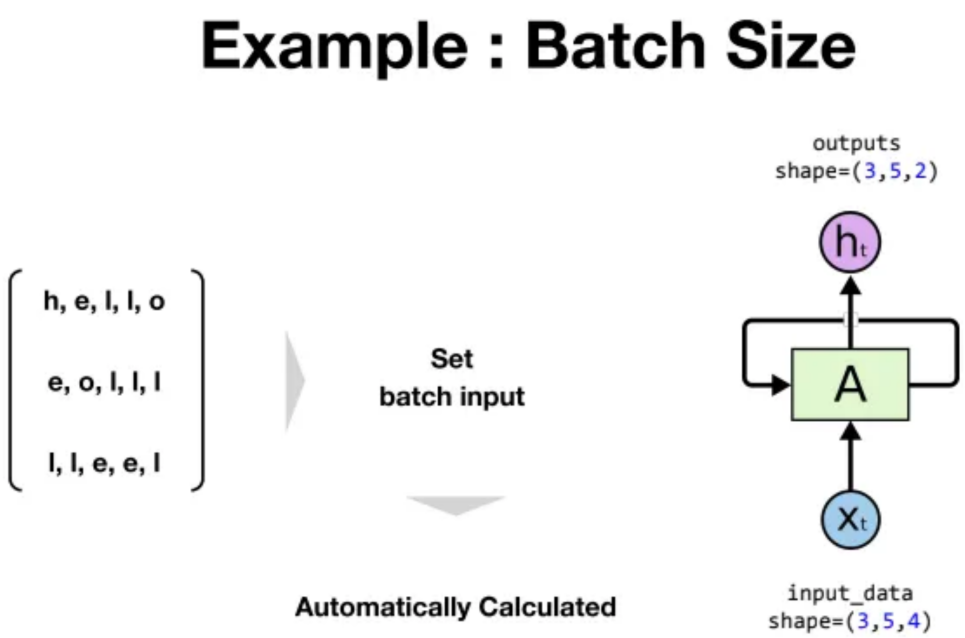


http://colah.github.io/posts/2015-08-Understanding-LSTMs

- 그 이유는 hidden에서 나온 data가 output에 사용되기 때문이다.



- Sequence Length는 input_data의 2번째 dim이다.
- 자동적으로 계산된다.



- 마지막은 CNN에서도 동일하듯이 첫번째 dim은 `batch_size`이다.

```
input_size = 4
hidden_size = 2
```

```
# singleton example
# shape : (1, 1, 4)
# input_data_np = np.array([[[1, 0, 0, 0]]])

# sequential example
# shape : (3, 5, 4)
h = [1, 0, 0, 0]
e = [0, 1, 0, 0]
l = [0, 0, 1, 0]
o = [0, 0, 0, 1]
input_data_np = np.array([[h, e, l, l, o], [e, o, l, l, l], [l, l, e, e, l]], dtype = np.float32)
```

```
input_data = torch.Tensor(input_data_np)
```

```
rnn = torch.nn.RNN(input_size, hidden_size)
```

```
[8]  outputs, _status = rnn(input_data)
```

```
[10] print(outputs)
     print(outputs.size())
```

```
tensor([[[-0.4147,  0.5972],
         [-0.2881,  0.7695],
         [-0.8337,  0.2788],
         [-0.8337,  0.2788],
         [-0.7734,  0.7448]],

        [[-0.2018,  0.3469],
         [-0.7344,  0.2653],
         [-0.7948, -0.4056],
         [-0.7948, -0.4056],
         [-0.7842, -0.6027]],
```

- 코드는 매우 간단하다.

- 다음 시간에는 RNN을 학습시키는 방법에 대해 알아보자.