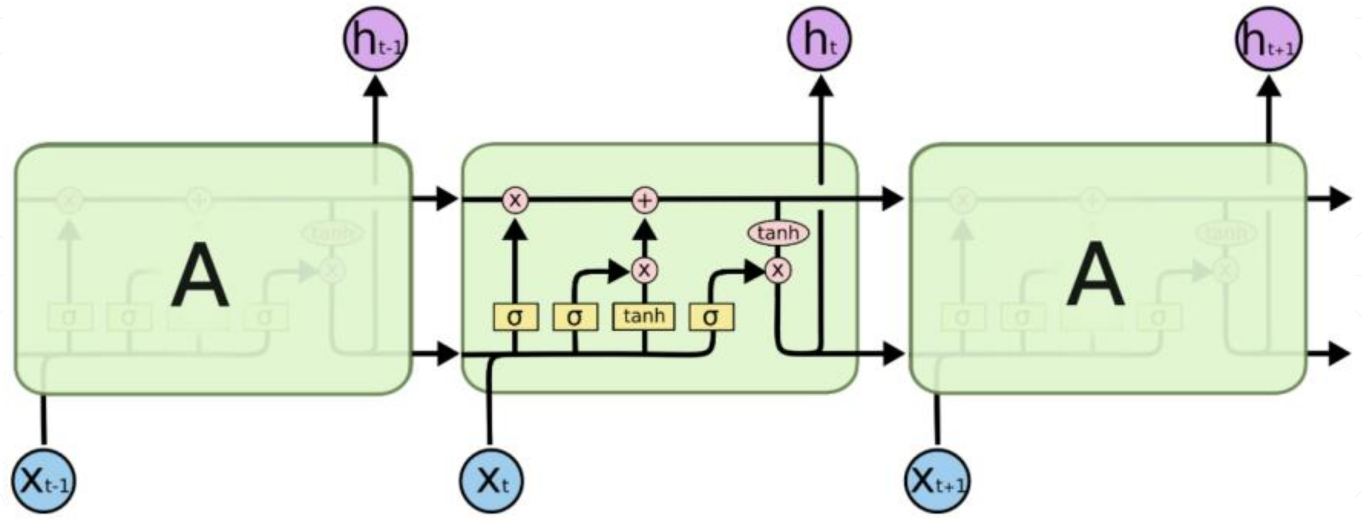
PyTorch

LSTM使用

主讲人：龙良曲



$$\begin{pmatrix} \mathbf{i}^{(t)} \\ \mathbf{f}^{(t)} \\ \mathbf{o}^{(t)} \\ \mathbf{g}^{(t)} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \mathbf{W} \begin{pmatrix} \mathbf{x}^{(t)} \\ \mathbf{h}^{(t-1)} \end{pmatrix} \tag{6}$$

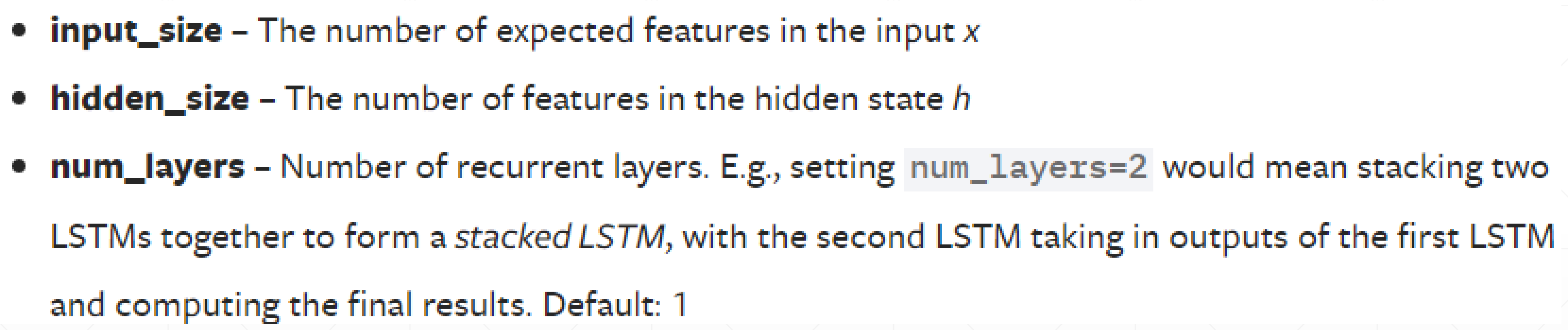
$$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \circ \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \circ \mathbf{g}^{(t)} \tag{7}$$

$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \circ \tanh(\mathbf{c}^{(t)}). \tag{8}$$

http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/readings/L15%20Exploding%20and%20Vanishing%20Gradients.pdf

nn.LSTM

- __init__

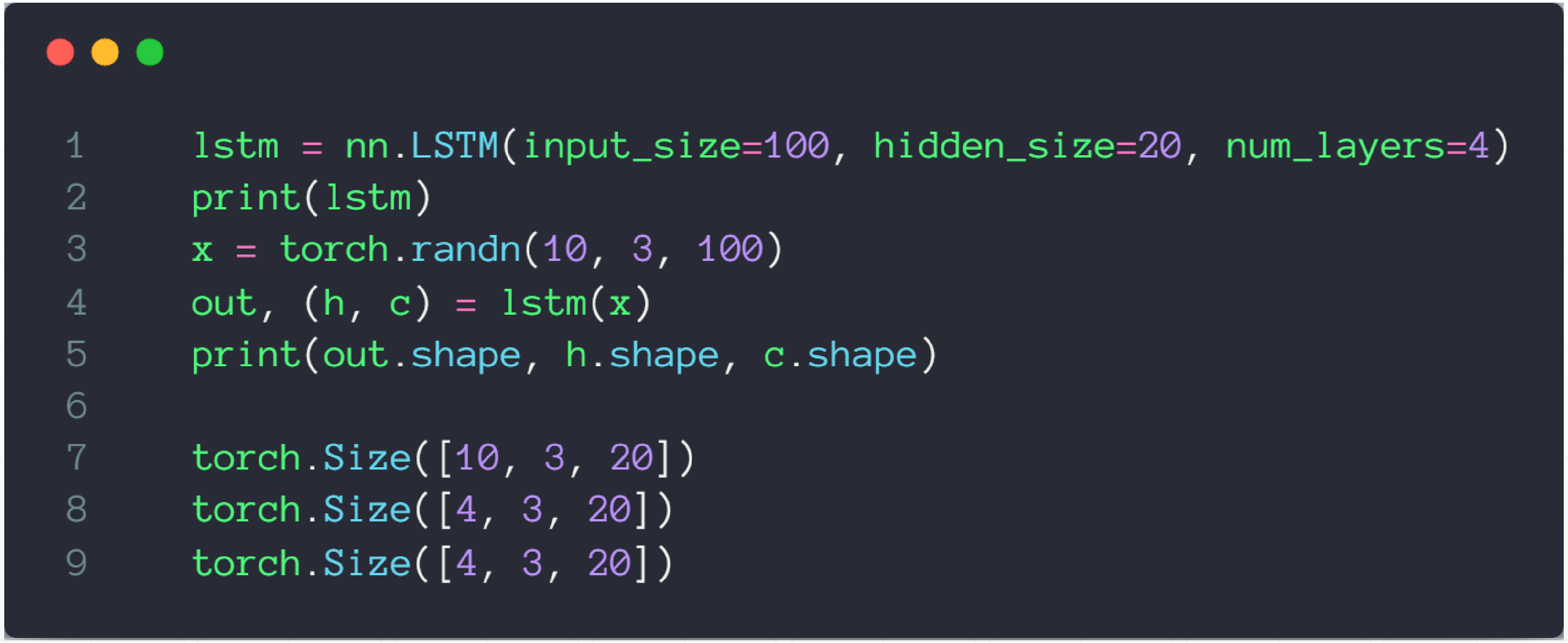


- **input_size** – The number of expected features in the input *x*
- **hidden_size** – The number of features in the hidden state *h*
- **num_layers** – Number of recurrent layers. E.g., setting `num_layers=2` would mean stacking two LSTMs together to form a *stacked LSTM*, with the second LSTM taking in outputs of the first LSTM and computing the final results. Default: 1

LSTM.foward()

- out, (ht, ct) = lstm(x, [ht_1, ct_1])
  - x: [seq, b, vec]
  - h/c: [num_layer, b, h]
  - out: [seq, b, h]

nn.LSTM

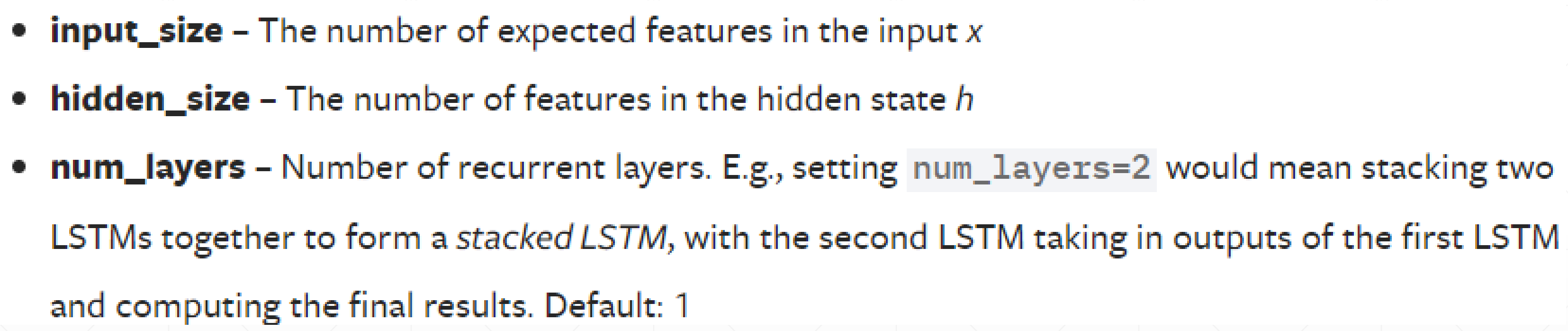


```
lstm = nn.LSTM(input_size=100, hidden_size=20, num_layers=4)
print(lstm)
x = torch.randn(10, 3, 100)
out, (h, c) = lstm(x)
print(out.shape, h.shape, c.shape)

torch.Size([10, 3, 20])
torch.Size([4, 3, 20])
torch.Size([4, 3, 20])
```

nn.LSTMCell

- __init__

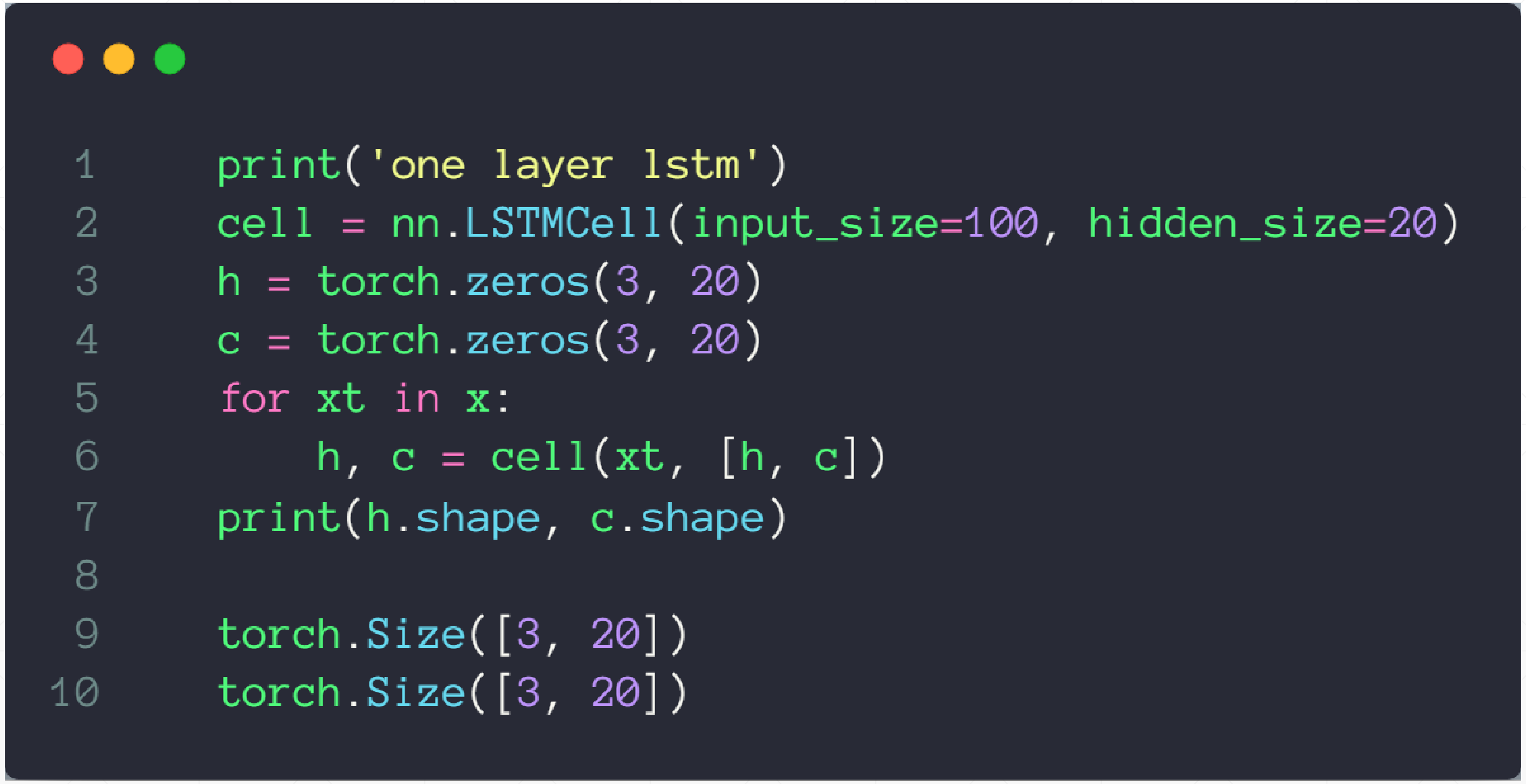


- **input_size** – The number of expected features in the input *x*
- **hidden_size** – The number of features in the hidden state *h*
- **num_layers** – Number of recurrent layers. E.g., setting `num_layers=2` would mean stacking two LSTMs together to form a *stacked LSTM*, with the second LSTM taking in outputs of the first LSTM and computing the final results. Default: 1

LSTMCell.forward()

- ht, ct = lstmcell(xt, [ht_1, ct_1])
  - xt: [b, vec]
  - ht/ct: [b, h]

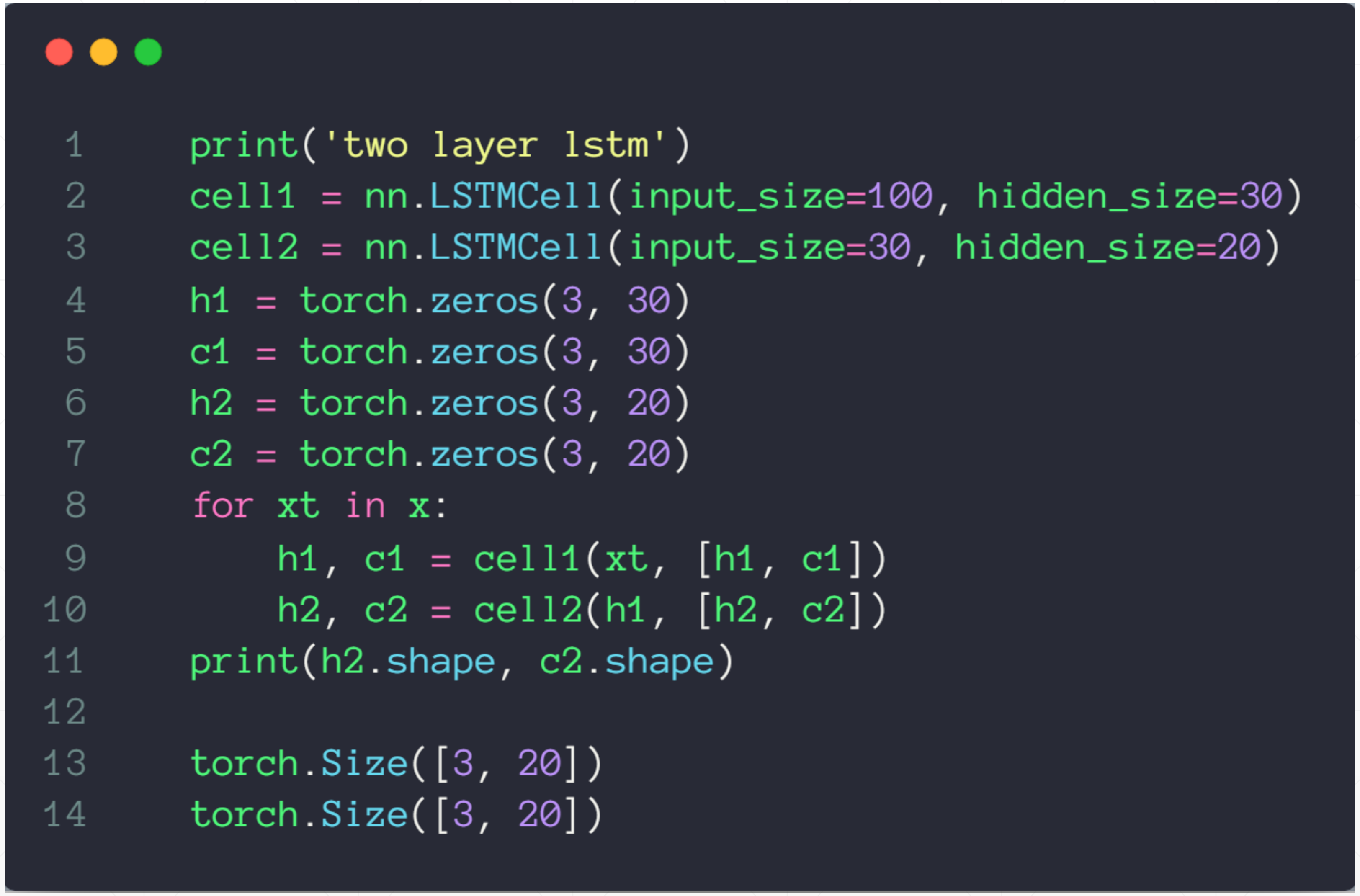
Single layer



```python
print('one layer lstm')
cell = nn.LSTMCell(input_size=100, hidden_size=20)
h = torch.zeros(3, 20)
c = torch.zeros(3, 20)
for xt in x:
    h, c = cell(xt, [h, c])
print(h.shape, c.shape)

torch.Size([3, 20])
torch.Size([3, 20])
```

Two Layers



```
print('two layer lstm')
cell1 = nn.LSTMCell(input_size=100, hidden_size=30)
cell2 = nn.LSTMCell(input_size=30, hidden_size=20)
h1 = torch.zeros(3, 30)
c1 = torch.zeros(3, 30)
h2 = torch.zeros(3, 20)
c2 = torch.zeros(3, 20)
for xt in x:
    h1, c1 = cell1(xt, [h1, c1])
    h2, c2 = cell2(h1, [h2, c2])
print(h2.shape, c2.shape)

torch.Size([3, 20])
torch.Size([3, 20])
```

下一课时

情感分类问题实战

Thank You.