LSTM 실습

In [1]:
```
%run ../chap11/rnn_lstm_model.ipynb
%run ../chap11/dataset_sounds.ipynb
```

In [2]:
```
%run ../chap10/dataset_automata.ipynb

ad = AutomataDataset()

am_64 = RnnLstmModel('am_64', ad, ['lstm', {'recur_size':64, 'outseq':False}])
am_64.exec_all(epoch_count=10, report=2)
```

```
Model am_64 train started:
    Epoch 2: cost=0.669, accuracy=0.587/0.630 (44/44 secs)
    Epoch 4: cost=0.634, accuracy=0.622/0.670 (44/88 secs)
    Epoch 6: cost=0.403, accuracy=0.824/0.820 (43/131 secs)
    Epoch 8: cost=0.289, accuracy=0.892/0.910 (44/175 secs)
    Epoch 10: cost=0.254, accuracy=0.910/0.930 (43/218 secs)
Model am_64 train ended in 218 secs:
Model am_64 test report: accuracy = 0.902, (1 secs)

Model am_64 Visualization
zk4/b-(uzh+3)/1*n/3)-(66): 잘못된 패턴 => 합격추정(0.92) : X
1803-q1of*6/260+12-(7): 올바른 패턴 => 합격추정(0.91) : O
9/e1i*1*0/0x9-n*fie*xxk: 잘못된 패턴 => 탈락추정(0.00) : O
```

In [3]:
```
usd_10_10 = UrbanSoundDataset(10, 10)
usd_10_100 = UrbanSoundDataset(10, 100)
```

```
loaded from cache
loaded from cache
```

In [4]:
```
conf_basic = ['rnn', {'recur_size':20, 'outseq':False}]
conf_lstm  = ['lstm', {'recur_size':20, 'outseq':False}]
conf_state = ['lstm', {'recur_size':20, 'outseq':False, 'use_state':True}]

us_basic_10_10 = RnnLstmModel('us_basic_10_10', usd_10_10, conf_basic)
us_lstm_10_10  = RnnLstmModel('us_lstm_10_10',  usd_10_10, conf_lstm)
us_state_10_10 = RnnLstmModel('us_state_10_10', usd_10_10, conf_state)

us_basic_10_100 = RnnLstmModel('us_basic_10_100', usd_10_100, conf_basic)
us_lstm_10_100  = RnnLstmModel('us_lstm_10_100',  usd_10_100, conf_lstm)
us_state_10_100 = RnnLstmModel('us_state_10_100', usd_10_100, conf_state)
```

In [5]:
```
us_basic_10_10.exec_all(epoch_count=10, report=2)
```

```
Model us_basic_10_10 train started:
    Epoch 2: cost=2.120, accuracy=0.240/0.180 (10/10 secs)
    Epoch 4: cost=2.012, accuracy=0.290/0.230 (9/19 secs)
    Epoch 6: cost=1.937, accuracy=0.325/0.290 (10/29 secs)
    Epoch 8: cost=1.889, accuracy=0.328/0.210 (9/38 secs)
    Epoch 10: cost=1.846, accuracy=0.345/0.230 (8/46 secs)
Model us_basic_10_10 train ended in 46 secs:
Model us_basic_10_10 test report: accuracy = 0.348, (0 secs)

Model us_basic_10_10 Visualization


        0:00 / 0:00


siren: wrong(jackhammer)


        0:00 / 0:00


engine_idling: wrong(air_conditioner)


        0:00 / 0:00


children_playing: wrong(street_music)
```

In [6]:
```
us_lstm_10_10.exec_all(epoch_count=10, report=2, show_cnt=0)
```

```
Model us_lstm_10_10 train started:
    Epoch 2: cost=2.130, accuracy=0.249/0.230 (30/30 secs)
    Epoch 4: cost=2.099, accuracy=0.250/0.240 (30/60 secs)
    Epoch 6: cost=2.018, accuracy=0.287/0.290 (31/91 secs)
    Epoch 8: cost=1.974, accuracy=0.288/0.220 (30/121 secs)
    Epoch 10: cost=1.983, accuracy=0.293/0.250 (30/151 secs)
Model us_lstm_10_10 train ended in 151 secs:
Model us_lstm_10_10 test report: accuracy = 0.325, (1 secs)
```

In [7]:
```
us_state_10_10.exec_all(epoch_count=10, report=2, show_cnt=0)
```

```
Model us_state_10_10 train started:
    Epoch 2: cost=2.060, accuracy=0.252/0.200 (30/30 secs)
    Epoch 4: cost=1.962, accuracy=0.284/0.270 (30/60 secs)
    Epoch 6: cost=1.903, accuracy=0.305/0.300 (30/90 secs)
    Epoch 8: cost=1.834, accuracy=0.345/0.300 (30/120 secs)
    Epoch 10: cost=1.764, accuracy=0.367/0.380 (31/151 secs)
Model us_state_10_10 train ended in 151 secs:
Model us_state_10_10 test report: accuracy = 0.353, (0 secs)
```

In [8]:
```
us_basic_10_10.exec_all(epoch_count=100, report=20, show_cnt=0)
us_lstm_10_10.exec_all(epoch_count=100, report=20, show_cnt=0)
us_state_10_10.exec_all(epoch_count=100, report=20, show_cnt=0)
```

```
Model us_basic_10_10 train started:
    Epoch 20: cost=1.627, accuracy=0.414/0.260 (78/78 secs)
    Epoch 40: cost=1.491, accuracy=0.448/0.340 (80/158 secs)
    Epoch 60: cost=1.429, accuracy=0.465/0.340 (85/243 secs)
    Epoch 80: cost=1.414, accuracy=0.481/0.270 (84/327 secs)
    Epoch 100: cost=1.319, accuracy=0.497/0.330 (83/410 secs)
Model us_basic_10_10 train ended in 410 secs:
Model us_basic_10_10 test report: accuracy = 0.374, (0 secs)

Model us_lstm_10_10 train started:
    Epoch 20: cost=1.749, accuracy=0.378/0.290 (311/311 secs)
    Epoch 40: cost=1.561, accuracy=0.448/0.340 (313/624 secs)
    Epoch 60: cost=1.425, accuracy=0.492/0.430 (318/942 secs)
    Epoch 80: cost=1.383, accuracy=0.539/0.380 (321/1263 secs)
    Epoch 100: cost=1.275, accuracy=0.582/0.390 (306/1569 secs)
Model us_lstm_10_10 train ended in 1569 secs:
Model us_lstm_10_10 test report: accuracy = 0.486, (1 secs)

Model us_state_10_10 train started:
    Epoch 20: cost=1.538, accuracy=0.450/0.460 (300/300 secs)
    Epoch 40: cost=1.247, accuracy=0.547/0.520 (303/603 secs)
    Epoch 60: cost=1.119, accuracy=0.600/0.490 (308/911 secs)
    Epoch 80: cost=0.991, accuracy=0.646/0.430 (310/1221 secs)
    Epoch 100: cost=0.889, accuracy=0.678/0.530 (311/1532 secs)
Model us_state_10_10 train ended in 1532 secs:
Model us_state_10_10 test report: accuracy = 0.577, (1 secs)
```

In [9]:
```
us_basic_10_100.exec_all(epoch_count=100, report=20, show_cnt=0)
us_lstm_10_100.exec_all(epoch_count=100, report=20, show_cnt=0)
us_state_10_100.exec_all(epoch_count=100, report=20, show_cnt=0)
```

```
Model us_basic_10_100 train started:
    Epoch 20: cost=1.138, accuracy=0.602/0.580 (91/91 secs)
    Epoch 40: cost=1.015, accuracy=0.649/0.550 (75/166 secs)
    Epoch 60: cost=0.978, accuracy=0.667/0.590 (74/240 secs)
    Epoch 80: cost=0.784, accuracy=0.711/0.690 (73/313 secs)
    Epoch 100: cost=0.845, accuracy=0.700/0.650 (74/387 secs)
Model us_basic_10_100 train ended in 387 secs:
Model us_basic_10_100 test report: accuracy = 0.590, (0 secs)

Model us_lstm_10_100 train started:
    Epoch 20: cost=1.229, accuracy=0.593/0.520 (290/290 secs)
    Epoch 40: cost=0.910, accuracy=0.690/0.650 (291/581 secs)
    Epoch 60: cost=0.746, accuracy=0.756/0.680 (293/874 secs)
    Epoch 80: cost=0.704, accuracy=0.772/0.660 (294/1168 secs)
    Epoch 100: cost=0.618, accuracy=0.806/0.680 (295/1463 secs)
Model us_lstm_10_100 train ended in 1463 secs:
Model us_lstm_10_100 test report: accuracy = 0.722, (1 secs)

Model us_state_10_100 train started:
    Epoch 20: cost=0.979, accuracy=0.682/0.640 (288/288 secs)
    Epoch 40: cost=0.746, accuracy=0.747/0.690 (290/578 secs)
    Epoch 60: cost=0.609, accuracy=0.794/0.710 (292/870 secs)
    Epoch 80: cost=0.482, accuracy=0.837/0.740 (294/1164 secs)
    Epoch 100: cost=0.461, accuracy=0.848/0.730 (295/1459 secs)
Model us_state_10_100 train ended in 1459 secs:
Model us_state_10_100 test report: accuracy = 0.758, (1 secs)
```

결론

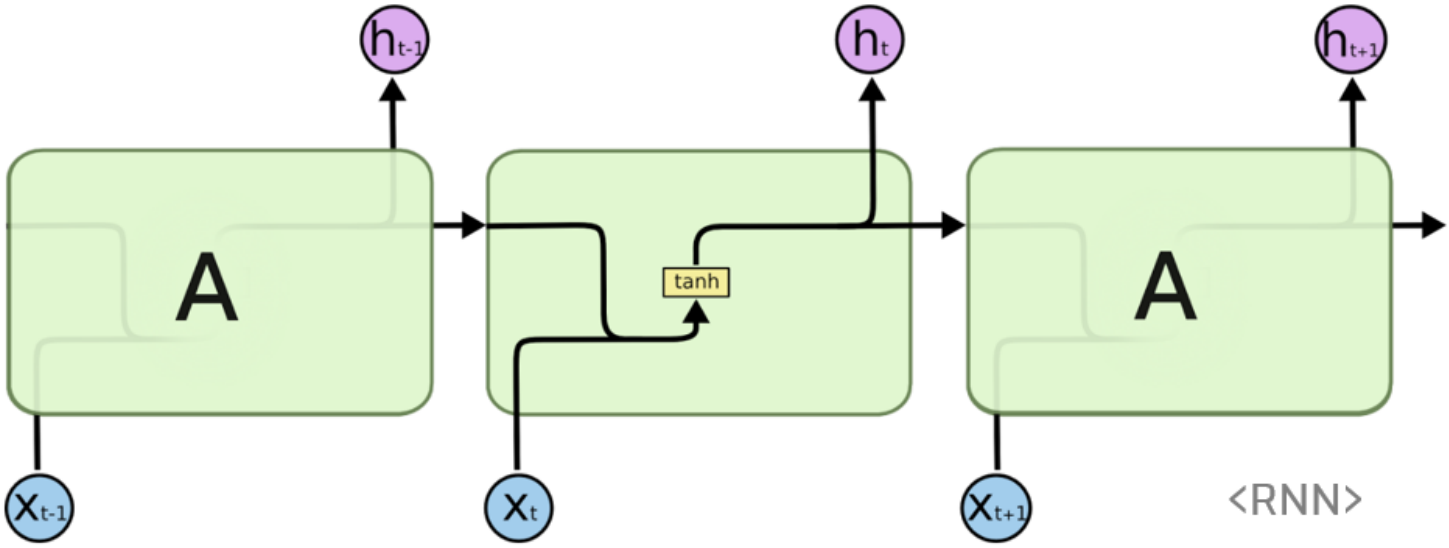
LSTM은 순환벡터 ht와 상태 벡터 ct를 이용하여 세 개의 sigmoid 게이트와 1개의 tanh 를 이용한다.

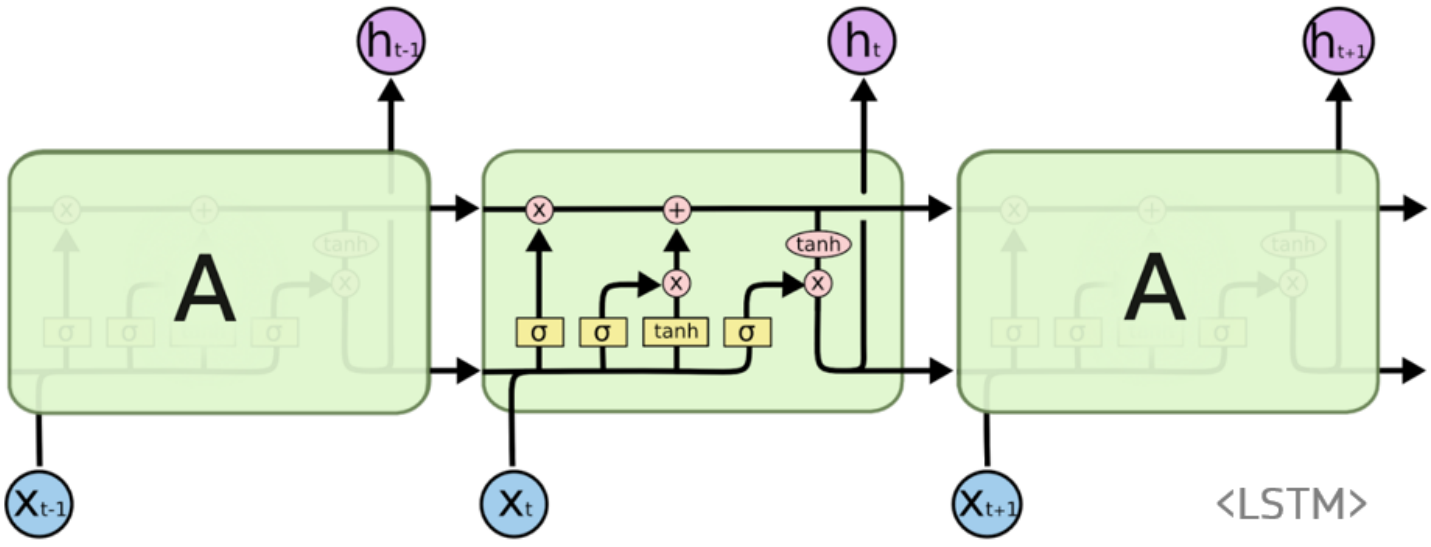
음원 데이터에서는 정보의 개수가 작아도 학습이 가능하지만, 좋은 결과를 얻기 위해서는 많은 학습이 필요하다. 시간대 수를 줄이더라도 시간대별 정보를 좀 더 의미있고 풍성한 내용으로 대체할 수 있는 전처리가 필요하다

- 평균값 등의 구간 대표값 정하기 소리는 파동의 일종이며 파동은 여러 주파수의 사인파의 혼합 평균값을 구하면 대부분의 정보가 사라져 오히려 손해 -> 다른 대표값 정할 필요성
- 주파수 스펙트럼 주파수 별 분포
- 푸리에 변환을 통해서 주파수 강도를 계산할 수 있다.
- 고속 푸리에 변환 : 분할 정복, 슬라이딩 윈도우

LSTM과 RNN 차이

RNN은 관련 정보와 그 정보를 사용하는 지점 사이의 거리가 멀 경우 Back Propagation 시에 Gradient 가 점차적으로 줄어들어 학습 능력이 크게 저하된다. 또한 RNN은 곱으로 이를 평가하기 때문에 Gradient vanishing 과 Gradinet Explode또한 발생한다. 이런 RNN의 문제점을 해결하기 위해서 RNN의 Hidden state에 cell state를 추가하여 LSTM을 구성할 수 있다.





LSTM 각 Gate의 기능

- forget gate : 과거 정보를 잊기 위한 gate이다. ht−1과 xt를 sigmoid를 이용하여 0이면 이전 상태의 정보를 잊고, 1이면 이전 상태의 정보를 온전히 기억할 수 있다.
- input gate : 현재 정보를 기억하기 위한 gate이다. ht−1과 xt를 받아 sigmoid를 취하고, 또 같은 입력으로 tanh를 취해준 다음 Hadamard product 연산을 한 값이 Input Gate가 내보내는 값이 된다.
- output gate : cell state를 바탕으로 필터링 된 값이 되는데. sigmoid에 input 데이터를 넣어서 어떤 output으로 내보낼지를 결정한다. 그리고나서 cell state를 tanh layer에 넣어서 -1과 1 사이의 값을 받은 뒤에 방금 전에 계산한 sigmoid

gate의 output과 곱해준다. 그렇게 하면 우리가 output으로 보내고자 하는 부분만 내보낼 수 있게 된다.

In [ ]: