11장. LSTM 순환 신경망 : 도시 소음 분류 신경망

부산대학교 항공우주공학과

정대현 201527137

1. 개요

LSTM(Long Short-Term Memory)은 시계열 데이터에 대한 처리 방식입니다. RNN(Recurrent Neural Network, 순환신경망)의 경우 순환 벡터나 기울기 정보에 정보의 소멸(0으로 수렴)하거나 폭주(무한대로 발산)하는 현상이 있습니다. LSTM은 반복적으로 곱하는 과정에서 일어나는 문제를 해결하기 위해 망각 게이트를 추가하여 가중치를 다르게 줍니다. 이번 장에서는 RNN과 LSTM과 차이점을 학습을 진행하게 됩니다.

2. RNN

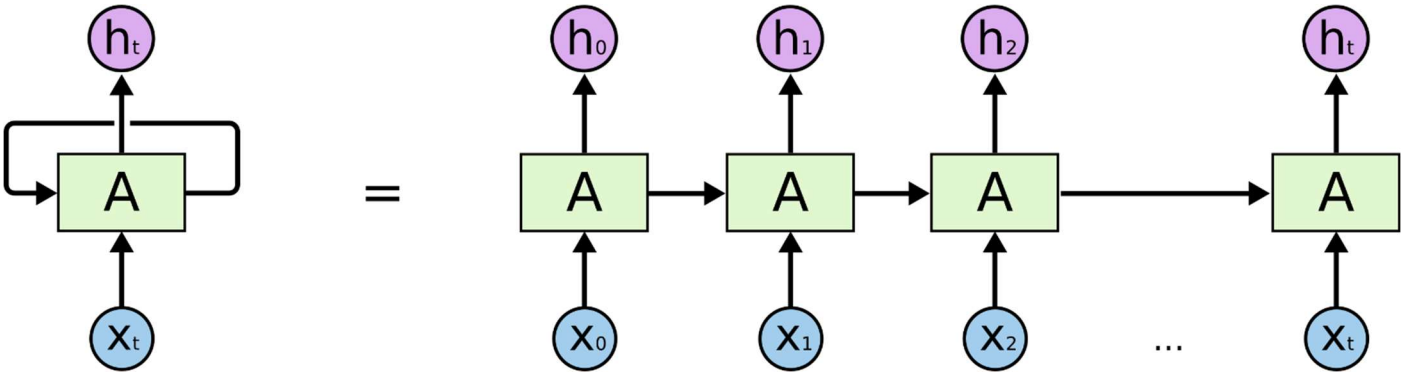


Fig. 1. 순서대로 연결한 RNN

Fig. 1. 그림은 RNN의 체인을 표현한 그림입니다. RNN은 시간의 성질을 가지고 있으며 sequence, list에 따라서 이어지며 이런 시계열을 다루는데 최적화된 구조의 Neural Network입니다. 음성인식, 언어모델링, 번역, 이미지 주석생성 등 광범위하게 사용되었으나 '긴 의존 기간으로 인한 문제점'이 발생했습니다. 순환 신경망에서는 이전 시간대에서 전달된 순환 벡터를 입력의 일부로 이용하면서 다음시간대로 전달한다고 했습니다. 이때 각 시간대 데이터에 대해 동일한 내용의 가중치 행렬이 반복적으로 선형 연산에 이용됩니다. 이 과정에서 절대값이 1보다 큰 값이 반복적으로 곱해지면 기하급수적으로 증가해 폭주를 하게 되며 부호가 다를 경우 크게 롤링 하게 됩니다. 반대로 절대값이 1보다 작은 수가 계속해서 곱해지면 0으로 수렴하여 소멸하게 됩니다. 또한 -1만 곱해지게 된다면 그저 부호만 계속 바뀌고 아무런 의미가 없는 값만 도출할 수 있습니다.

3. LSTM

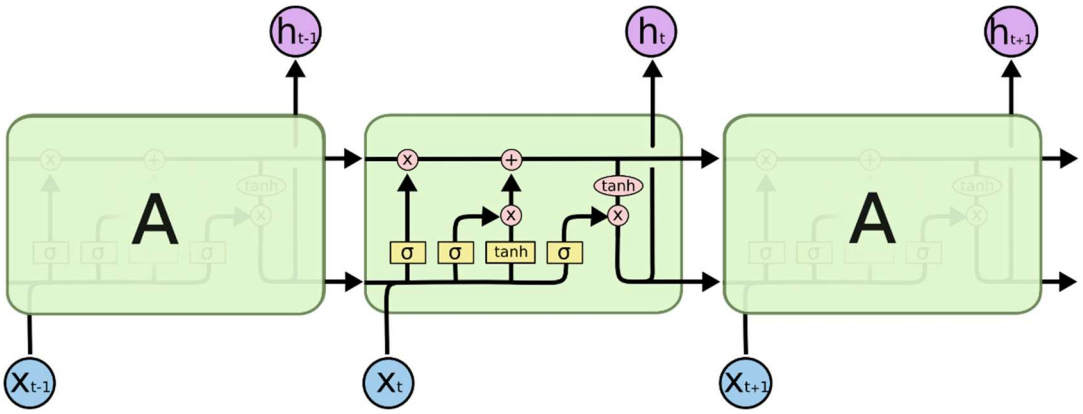
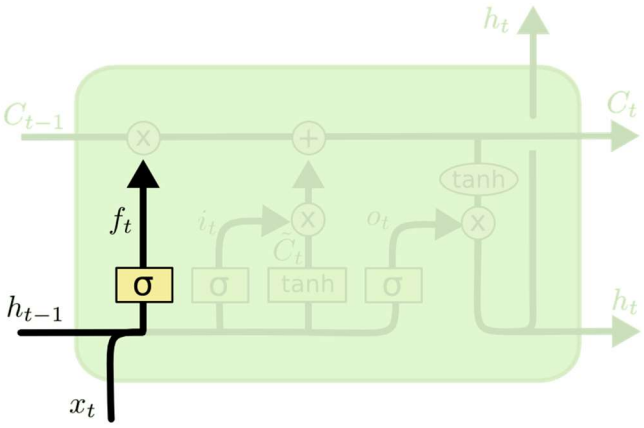


Fig. 2. 4 개의 Layer 이 추가된 LSTM

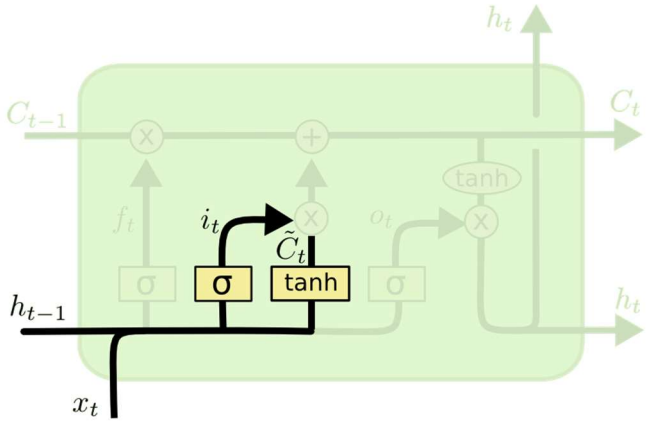
Fig. 2는 LSTM의 내부구조를 보여주고 있습니다. RNN은 tanh하나의 모듈이 있었다면 LSTM은 3개의 시그마가 추가되었습니다. 각 게이트의 의미는 다음과 같습니다.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Fig. 3. LSTM 의 forget gate layer

LSTM 의 첫 단계로는 cell state 로부터 어떤 정보를 버릴 것인지 정하며 sigmoid layer 에 의해 결정합니다. 이 단계의 gate 를 forget gate 라고 합니다. h_{t-1}와 x_t를 받아서 0 고 1 사이의 값을 C_{t-1}에 보내주게 됩니다. 이때 0 은 완전히 잊어버린다는 것이고 1 은 모든 것을 기억한다는(보존하라는) 의미가 됩니다.

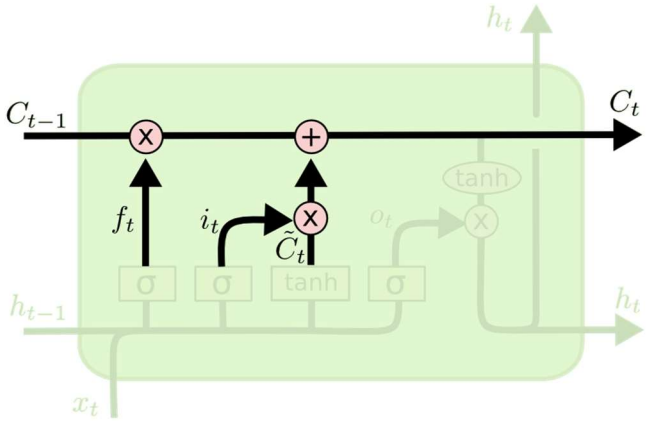


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Fig. 4. LSTM 의 input gate layer

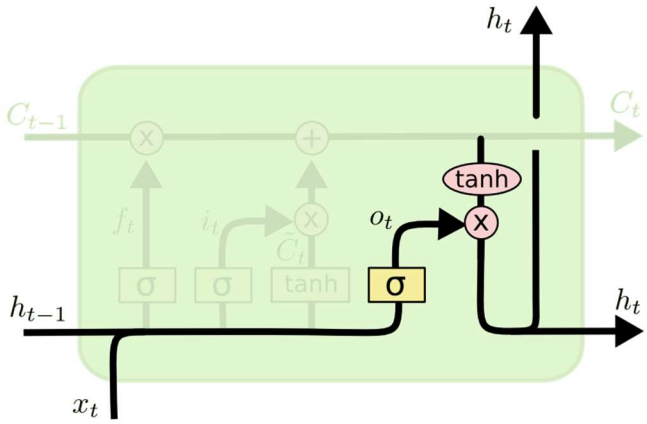
그 다음으로는 새로운 정보 중 어떤 것을 cell state 에 저장할 것인지 정합니다. Input gate layer 라고 불리는 sigmoid layer 이 어떤 값을 업데이트 할지 정하며 tanh layer 가 새로운 후보 값들인 $\tilde{C}_t$ 라는 벡터를 만들고 cell state 에 더할 준비를 합니다. 이렇게 두 단계에서 나온 정보를 합쳐서 state 에 업데이트할 준비를 합니다.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Fig. 5. LSTM 의 cell state 업데이트

이전 단계에서 구한 값들을 바탕을 cell state 를 업데이트 하는 과정입니다.



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Fig. 6. LSTM 의 output gate layer

마지막으로 무엇을 output 로 내보낼지 정하는 일이 남았습니다. Output 는 cell state 를 바탕으로 필터 된 값이 될 것입니다. sigmoid layer 에 input 데이터를 보낸 다음 cell state 의 어느 부분을 output 으로 내보낼 지를 정하게

됩니다. cell state 를 tanh layer 에 보내어 그 범위인 -1 과 1 사이의 값을 받은 뒤에 방금 전에 계산한 sigmoid gate 의 output 과 곱해줍니다. 이제 output 으로 보내고자 하는 부분만 내보낼 수 있게 됩니다.

4. 실험

4.1 작은 윈도우와 RNN 계층을 이용하는 도시 소음 분류 신경망의 학습 epoch 10회

```
Model us_basic_10_10 train started:
    Epoch 2: cost=2.120, accuracy=0.240/0.180 (7/7 secs)
    Epoch 4: cost=2.012, accuracy=0.290/0.230 (8/15 secs)
    Epoch 6: cost=1.937, accuracy=0.325/0.290 (7/22 secs)
    Epoch 8: cost=1.889, accuracy=0.328/0.210 (7/29 secs)
    Epoch 10: cost=1.846, accuracy=0.345/0.230 (7/36 secs)
Model us_basic_10_10 train ended in 36 secs:
Model us_basic_10_10 test report: accuracy = 0.348, (0 secs)
```

4.2 작은 윈도우와 LSTM 계층을 이용하는 도시 소음 분류 신경망의 학습 epoch 10회

```
Model us_lstm_10_10 train started:
    Epoch 2: cost=2.135, accuracy=0.234/0.240 (27/27 secs)
    Epoch 4: cost=2.056, accuracy=0.263/0.250 (27/54 secs)
    Epoch 6: cost=1.983, accuracy=0.296/0.310 (28/82 secs)
    Epoch 8: cost=1.928, accuracy=0.317/0.270 (29/111 secs)
    Epoch 10: cost=1.921, accuracy=0.320/0.300 (28/139 secs)
Model us_lstm_10_10 train ended in 139 secs:
Model us_lstm_10_10 test report: accuracy = 0.361, (1 secs)
```

4.3 LSTM 계층과 상태 벡터 출력 epoch 10회

```
Model us_state_10_10 train started:
    Epoch 2: cost=2.056, accuracy=0.240/0.240 (28/28 secs)
    Epoch 4: cost=1.923, accuracy=0.310/0.340 (29/57 secs)
    Epoch 6: cost=1.893, accuracy=0.320/0.300 (26/83 secs)
    Epoch 8: cost=1.853, accuracy=0.337/0.310 (31/114 secs)
    Epoch 10: cost=1.766, accuracy=0.357/0.340 (30/144 secs)
Model us_state_10_10 train ended in 144 secs:
Model us_state_10_10 test report: accuracy = 0.338, (1 secs)
```

4.4 위의 실험을 각 epoch 100회로 증가

```
Model us_basic_10_10 train started:
    Epoch 20: cost=1.627, accuracy=0.414/0.260 (64/64 secs)
    Epoch 40: cost=1.556, accuracy=0.426/0.330 (62/126 secs)
    Epoch 60: cost=1.461, accuracy=0.459/0.310 (61/187 secs)
    Epoch 80: cost=1.363, accuracy=0.491/0.330 (60/247 secs)
    Epoch 100: cost=1.319, accuracy=0.502/0.320 (61/308 secs)
Model us_basic_10_10 train ended in 308 secs:
Model us_basic_10_10 test report: accuracy = 0.397, (0 secs)

Model us_lstm_10_10 train started:
    Epoch 20: cost=1.627, accuracy=0.425/0.370 (264/264 secs)
    Epoch 40: cost=1.458, accuracy=0.514/0.370 (287/551 secs)
    Epoch 60: cost=1.295, accuracy=0.560/0.480 (313/864 secs)
    Epoch 80: cost=1.182, accuracy=0.607/0.440 (285/1149 secs)
    Epoch 100: cost=1.008, accuracy=0.659/0.480 (271/1420 secs)
Model us_lstm_10_10 train ended in 1420 secs:
Model us_lstm_10_10 test report: accuracy = 0.527, (1 secs)

Model us_state_10_10 train started:
    Epoch 20: cost=1.528, accuracy=0.446/0.420 (284/284 secs)
    Epoch 40: cost=1.327, accuracy=0.535/0.380 (270/554 secs)
    Epoch 60: cost=1.089, accuracy=0.605/0.410 (293/847 secs)
    Epoch 80: cost=1.049, accuracy=0.617/0.440 (289/1136 secs)
    Epoch 100: cost=0.904, accuracy=0.675/0.420 (288/1424 secs)
Model us_state_10_10 train ended in 1424 secs:
Model us_state_10_10 test report: accuracy = 0.569, (0 secs)
```

4.5 넓은 윈도우와 RNN 계층을 이용하는 도시 소음 분류 신경망의 학습 100회

```
Model us_basic_10_100 train started:
    Epoch 20: cost=1.138, accuracy=0.602/0.580 (70/70 secs)
    Epoch 40: cost=1.015, accuracy=0.649/0.550 (65/135 secs)
    Epoch 60: cost=0.870, accuracy=0.695/0.650 (63/198 secs)
    Epoch 80: cost=0.761, accuracy=0.726/0.680 (64/262 secs)
    Epoch 100: cost=0.724, accuracy=0.730/0.670 (65/327 secs)
Model us_basic_10_100 train ended in 327 secs:
Model us_basic_10_100 test report: accuracy = 0.631, (1 secs)
Model us_lstm_10_100 train started:
    Epoch 20: cost=1.118, accuracy=0.630/0.590 (270/270 secs)
    Epoch 40: cost=0.849, accuracy=0.739/0.620 (294/564 secs)
```

```
    Epoch 60: cost=0.729, accuracy=0.759/0.710 (320/884 secs)
    Epoch 80: cost=0.629, accuracy=0.800/0.660 (286/1170 secs)
    Epoch 100: cost=0.593, accuracy=0.815/0.690 (262/1432 secs)
Model us_lstm_10_100 train ended in 1432 secs:
Model us_lstm_10_100 test report: accuracy = 0.691, (1 secs)
Model us_state_10_100 train started:
    Epoch 20: cost=0.933, accuracy=0.684/0.640 (269/269 secs)
    Epoch 40: cost=0.787, accuracy=0.720/0.680 (275/544 secs)
    Epoch 60: cost=0.672, accuracy=0.768/0.660 (276/820 secs)
    Epoch 80: cost=0.460, accuracy=0.848/0.690 (271/1091 secs)
    Epoch 100: cost=0.404, accuracy=0.865/0.780 (277/1368 secs)
Model us_state_10_100 train ended in 1368 secs:
Model us_state_10_100 test report: accuracy = 0.738, (1 secs)
```

4.6 넓은 윈도우와 RNN 계층을 이용하는 도시 소음 분류 신경망의 학습 500회

```
Model us_basic_10_100 train started:
    Epoch 100: cost=0.611, accuracy=0.777/0.630 (375/375 secs)
    Epoch 200: cost=0.826, accuracy=0.751/0.600 (335/710 secs)
    Epoch 300: cost=0.576, accuracy=0.797/0.550 (354/1064 secs)
    Epoch 400: cost=0.594, accuracy=0.793/0.610 (363/1427 secs)
    Epoch 500: cost=0.815, accuracy=0.714/0.590 (349/1776 secs)
Model us_basic_10_100 train ended in 1776 secs:
Model us_basic_10_100 test report: accuracy = 0.519, (1 secs)

Model us_lstm_10_100 train started:
    Epoch 100: cost=0.672, accuracy=0.776/0.680 (1966/1966 secs)
    Epoch 200: cost=0.479, accuracy=0.843/0.710 (1513/3479 secs)
    Epoch 300: cost=0.376, accuracy=0.880/0.710 (1553/5032 secs)
    Epoch 400: cost=0.326, accuracy=0.897/0.700 (1608/6640 secs)
    Epoch 500: cost=0.299, accuracy=0.906/0.710 (1546/8186 secs)
Model us_lstm_10_100 train ended in 8186 secs:
Model us_lstm_10_100 test report: accuracy = 0.709, (1 secs)

Model us_state_10_100 train started:
    Epoch 100: cost=0.441, accuracy=0.855/0.720 (1468/1468 secs)
    Epoch 200: cost=0.292, accuracy=0.899/0.840 (1536/3004 secs)
    Epoch 300: cost=0.174, accuracy=0.944/0.810 (1552/4556 secs)
    Epoch 400: cost=0.135, accuracy=0.957/0.810 (1592/6148 secs)
    Epoch 500: cost=0.161, accuracy=0.953/0.820 (1540/7688 secs)
```

Model us_state_10_100 train ended in 7688 secs:
Model us_state_10_100 test report: accuracy = 0.751, (1 secs)

실험	Epoch	RNN	벡터	정확도
1	10	RNN	순환	0.348
2	10	LSTM	순환	0.361
3	10	LSTM	상태	0.338
4	100	RNN	순환	0.397
5	100	LSTM	순환	0.527
6	100	LSTM	상태	0.569
7	100	넓은 윈도우+RNN	순환	0.631
8	100	넓은 윈도우+LSTM	순환	0.691
9	100	넓은 윈도우+LSTM	상태	0.738
10	500	넓은 윈도우+RNN	순환	0.519
11	500	넓은 윈도우+LSTM	순환	0.709
12	500	넓은 윈도우+LSTM	상태	0.751

5. 결론

정보의 소멸 및 폭주 현상을 막아 순환 신경망 학습의 품질을 높여주는 LSTM 셀 구조를 소개하고 LSTM계층을 구현하여 도시 소음분류 신경망에 이용했습니다. 단순히 Epoch가 작을 때는 RNN, LSTM, 상태벡터, 순환벡터의 유의미한 차이가 없었지만 Epoch를 100회로 늘렸을 때에는 RNN보다 LSTM의 성능이 크게 개선된 것을 확인할 수 있었습니다. RNN에서 0.3대에 머물던 정확도가 0.5이상으로 뛰어오른 것을 볼 수 있습니다. 순환보다는 상태 벡터일 때 전반적으로 모델의 성능이 좋아지는 것을 확인할 수 있습니다. 넓은 윈도우와 Epoch 100회 LSTM에 상태벡터를 사용했을 때 정확도는 0.738로 처음에 실험에 비하면 두배 이상의 성능을 볼 수 있습니다. 데이터의 형태에 따라서 정확도가 크게 증가하였습니다. 이후 이어서 Epoch를 500회로 크게 늘려서 학습하자 학습시간은 노트북 기준 1~2시간으로 크게 증가하였지만 단 1~2%정도의 성능 향상이 있었습니다. RNN의 경우 오히려 정확도가 감소하였습니다. 여기서는 Epoch를 늘리기보다 다른 방식으로 접근해야 할 것입니다.