LSTM 신경망

부산대학교 정보컴퓨터공학부

15 학번 이연걸

1. 개요

LSTM 신경망은 시계열 데이터에 대한 새로운 처리 방식이다. RNN 계층에서는 순환 벡터나 그 기울기 정보에 정보의 소멸 및 폭주 현상 때문에 정보의 장거리 전달이 어려웠다. LSTM 은 이 문제를 해결했다. 전형적인 LSTM 은 다음과 같은 구조를 지닌다.

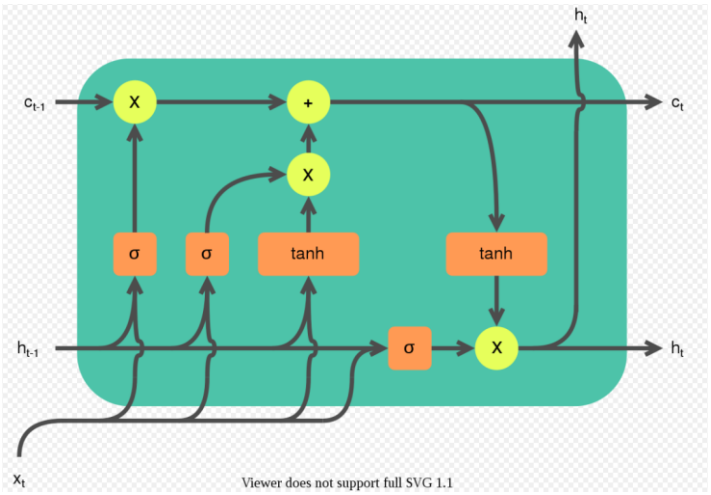

그림 1 – LSTM 셀의 기본 구조

가장 왼쪽의 게이트는 망각 게이트(forget gate)이다. 이 게이트는 시그모이드 함수를 활성화 함수로 이용하여 출력이 0 에서 1 사이이고 학습이 잘 수행되면 0 이나 1 로 수렴하는 경향을 보인다. 0 과 1 사이의 값을 상태 정보에 곱한다는 의미는 상태 정봇값의 일부만 이용한다는 의미다. 또한 학습이 잘 진행되어 시그모이드 함숫값이 0 이나 1 로 수렴하면 어중간한 처리보다는 0 을 곱해 완전히 잊어버리거나 1 을 곱해 완전히 보존하는 양극단의 처리를 선호한다는 의미가 된다. 이런 효과는 대체로 상황에 따라 불필요한 기억을 지우고 필요한 기억을 보존하면서도 전반적으로는 기억이 차츰 희미해져가는 인간의 뇌에 비유될 수 있다. 이런 효과 때문에 이 게이트는 망각 게이트라는 이름을 얻게 되었다. 이런 망각 효과가 확장 입력에 대한 퍼셉트론 처리의 결과로 제어된다. 따라서 항상 일정ㅎ나 방식으로 일어나는 것이 아니라 시간대마다 잊어야 할 기억과 보존해야 할 기억이 무엇인지 상황에 맞게 다시 따진다.

두 번째 게이트는 입력 게이트(input gate)이다. 이 게이트에서 만든 벡터는 입력 블록이 만든 내부 입력 정보와 성분별로 곱해진 후 망각 게이트 처리를 거친 상태 정보에 성분별로 더해진다. 이 게이트 역시 시그모이드 함수를 이용하는 덕분에 입력의 일부만 통과시키되 어중간한 처리보다는 완전 차단이나 완전 통과를 선호하는 쪽으로 학습된다. 이런 효과는 입력 내용 중 필요한 부분만 상태에 반영하여 출력 생성이나 기억 전달에 이용하겠다는 것이다. 한편 입력 게이트 출력과 곱해질 내부 입력 정보를 생산하는 입력 블록 장치는 다른 세 장치와 달리 쌍곡탄젠트 함수를 활성화 함수로 사용한다.

입력 블록에서 생성한 값은 입력 게이트 출력값에 의해 걸러진 후 망각 게이트 출력값으로 걸러진 상태 정보에 합해지며 그 결과는 새로운 상태 정보가 되어 다음 시간대로 전달된다. 이 상태 정보는 다시 한 번 쌍곡탄젠트 함수의 처리를 거쳐 (-1, 1) 사이의 값으로 변환된 후 세 번째 게이트인 출력 게이트 출력에 의해 걸러져 새로운 순환 벡터 성분이 된다.

이 순환 벡터 성분은 LSTM 계층의 출력을 구성하기도 하며 다음 시간대로 전달되어 확장된 입력의 일부로서 세 개의 게이트와 입력 블록의 동작을 결정하는 요인으로 이용되기도 한다.

이번 실험에서는 도시 소음 음원 파일들을 분석한다. rnn 계층과 lstm 계층을 각각 사용하고 결과를 비교한다. 또한 lstm 에서는 출력할 때 순환 벡터인 경우 상태 벡터인 경우로 나누어서 비교한다.

2. 구현 과정

위에서 언급한대로 3 가지로 나누어 진행하고 학습 횟수는 100 으로 한다.

2.1 rnn 계층을 이용한 경우

```
Model us_basic_10_100 train started:
    Epoch 20: cost=1.122, accuracy=0.618/0.520 (638/638 secs)
    Epoch 40: cost=0.992, accuracy=0.653/0.680 (668/1306 secs)
    Epoch 60: cost=0.899, accuracy=0.689/0.650 (544/1850 secs)
    Epoch 80: cost=0.876, accuracy=0.701/0.620 (391/2241 secs)
    Epoch 100: cost=0.707, accuracy=0.754/0.640 (182/2423 secs)
Model us_basic_10_100 train ended in 2423 secs:
Model us_basic_10_100 test report: accuracy = 0.600, (1 secs)
```

그림 2 – rnn 계층을 이용한 경우

2.2 lstm 계층을 이용한 경우

2.2.1 출력으로 순환 벡터를 이용한 경우

```
Model us_lstm_10_100 train started:
    Epoch 20: cost=1.142, accuracy=0.623/0.600 (931/931 secs)
    Epoch 40: cost=0.865, accuracy=0.724/0.710 (751/1682 secs)
    Epoch 60: cost=0.725, accuracy=0.771/0.690 (952/2634 secs)
    Epoch 80: cost=0.635, accuracy=0.787/0.780 (970/3604 secs)
    Epoch 100: cost=0.519, accuracy=0.835/0.700 (867/4471 secs)
Model us_lstm_10_100 train ended in 4471 secs:
Model us_lstm_10_100 test report: accuracy = 0.714, (4 secs)
```

그림 3 – lstm 계층을 이용한 경우 (출력으로 순환 벡터를 이용한 경우)

2.2.2 출력으로 상태 벡터를 이용한 경우

```
Model us_state_10_100 train started:
    Epoch 20: cost=1.126, accuracy=0.598/0.610 (672/672 secs)
    Epoch 40: cost=0.863, accuracy=0.731/0.670 (506/1178 secs)
    Epoch 60: cost=0.827, accuracy=0.724/0.660 (503/1681 secs)
    Epoch 80: cost=0.720, accuracy=0.774/0.640 (541/2222 secs)
    Epoch 100: cost=0.554, accuracy=0.817/0.630 (566/2788 secs)
Model us_state_10_100 train ended in 2788 secs:
Model us_state_10_100 test report: accuracy = 0.688, (1 secs)
```

그림 4 – lstm 계층을 이용한 경우 (출력으로 상태 벡터를 이용한 경우)

3. 총평

	rnn	lstm(순환벡터 출력)	lstm(상태벡터 출력)
정확도	60.0%	71.4%	68.8%

표 1 – 정확도 비교

단순 rnn 계층에 비해 lstm 계층을 이용할 때가 더욱 좋은 정확도를 보여주었다. lstm 의 경우 보통 상태벡터 출력이 순환벡터 출력보다 높은 정확도를 보이는 것으로 알고있지만 실험 결과는 반대로 나왔다. 실험 시간이 워낙 오래 걸려서 더 이상 확인은 힘들었다.

실험을 통해 확실히 정보의 소멸 및 폭주 현상을 막는 lstm 신경망이 기존 rnn 보다 성능이 뛰어나다는 것을 알 수 있었다.