chapter 5 - 순환 신경망(RNN)

- 지금까지 살펴본 신경망은 피드포워드라는 유형의 신경망입니다.
- 단순한 피드포워드 신경망에서는 시계열 데이터의 성질(패턴)을 충분히 학습할 수 없습니다.
- 그래서 순환 신경망(RNN)이 등장하게 됩니다.

5.1 확률과 언어 모델

5.1.1 word2vec을 확률 관점에서 바라보다

5.1.2 언어 모델

- 음성 인식 시스템의 경우, 사람의 음성으로부터 몇 개의 문장을 후보로 생성할 것입니다. 그런 다음 언어 모델을 사용하여 후보 문장이 '문장으로써 자연스러운지'를 기준으로 순서를 매길 수 있습니다.
- [언어 모델]은 새로운 문장을 생성하는 용도로 이용할 수 있습니다.

5.1.3 CBOW 모델을 언어 모델로?

- 그러나 CBOW 모델에서는 맥락 안의 단어 순서가 무시된다는 한계가 있습니다.
- CBOW란 continuous bag-of-words의 약어입니다. bag-of-words란 '가방 안의 단어'를 뜻하는데, 여기에는 가방 속의 단어 '순서'는 무시된다는 뜻도 내포합니다.
- RNN은 맥락이 아무리 길더라도 그 맥락의 정보를 기억하는 메커니즘을 갖추고 있습니다. 그래서 RNN을 사용하면 아무리 긴 시계열 데이터에라도 대응할 수 있습니다.

5.2 RNN이란

- RNN을 직역하면 '순환하는 신경망'이라고 직역됩니다.

5.2.1 순환하는 신경망

- RNN의 특징은 순환하는 경로(닫힌 경로)가 있다는 것입니다.

5.2.2 순환 구조 펼치기

- RNN 계층의 순환 구조를 펼침으로써 오른쪽으로 성장하는 긴 신경망으로 변신시킬 수 있습니다.
- 다른 관점으로 보면, RNN은 h라는 '상태'를 가지고 있으며, [식 5.9]의 형태로 갱신된다고 해석할 수 있습니다.

5.2.3 BPTT

- 여기서의 오차역전파법은 '시간 방향으로 펼친 신경망의 오차역전파법'이란 뜻으로 BPTT라고 합니다. (Backpropagation Through Time)

5.2.4 Truncated BPTT

- 큰 시계열 데이터를 취급할 때는 흔히 신경망 연결을 적당한 길이로 '끊습니다'. 시간축 방향으로 너무 길어진 신경망을 적당한 지점에서 잘라내어 작은 신경망 여러 개로 만든다는 아이디어죠. 그리고 이 잘라낸 각각의 작은 신경망에서 오차역전파법을 수행합니다. 이것이 바로 Truncated BPTT라는 기법입니다.
- Truncated BPTT에서는 신경망의 연결을 끊습니다만, 제대로 구현하려면 '역전파'의 연결만 끊어야 합니다.
- 한편, 역전파의 연결은 적당한 길이로 잘라내, 그 잘라낸 신경망 단위로 학습을 수행합니다.
- 이처럼 역전파의 연결을 잘라버리면, 그보다 미래의 데이터에 대해서는 생각할 필요가 없어집니다. 따라서 각각의 블록 다누이로, 미래의 블록과는 독립저긍로 오차역전파법을 완결시킬 수 있습니다.
- 여기서 반드시 기억할 점은 역전파의 연결은 끊어지지만, 순전파의 연결은 끊어지지 않는다는 점입니다.
- 가장 먼저 할 일은 첫 번째 블록 입력 데이터를 RNN 계층에 제공하는 것입니다.

5.2.5 Truncated BPTT의 미니배치 학습

- 첫 번째 미니배치 원소는 x0, x1, ... , x9가 되고, 두 번째 미니배치 원소는 x500, ..., x509가 됩니다. 그리고 이 미니배치 데이터를 RNN의 입력 데이터로 사용해 학습을 수행합니다. 이후로는 순서대로 진행되므로 다음에 넘길 데이터는 각각 시계열 데이터의 10~19번째 데이터와 510~519번째의 데이터가 되는 식이죠.

5.3 RNN 구현

- 이때 여러개의 RNN을 구현하는 계층을 Time RNN이라고 규정.
- 먼저 RNN의 한 단계를 처리하는 클래스를 RNN이란 이름으로 구현합니다. 그리고 이 RNN 클래스를 이용해 T개 단계의 처리를 한꺼번에 수행하는 계층을 TimeRNN이란 이름의 클래스로 완성시킵니다.

5.4 시계열 데이터 처리 계층 구현

5.4.1 RNNLM의 전체 그림

- [그림 5-25]의 첫 번째 층은 Embedding 계층입니다. 이 계층은 단어 ID를 단어의 분산 표현으로 변환합니다. 그리고 그 분산 표현이 RNN 계층으로 입력되죠. RNN 계층은 은닉 상태를 다음 층으로 출력함과 동시에, 다음 시각의 RNN 계층으로 출력합니다. 그리고 RNN 계층이 위로 출력한 은닉 상태는 Affine 계층을 거쳐 Softmax 계층으로 전해집니다.
- 여기서 주목할 것은 RNN 계층은 'you say'라는 맥락을 '기억'하고 있다는 사실입니다.
- 이처럼 RNNLM은 지금까지 입력된 단어를 '기억'하고, 그것을 바탕으로 다음에 출현할 단어를 예측합니다.

5.4.2 Time 계층 구현

- 지금까지는 시계열 데이터를 한꺼번에 처리하는 계층을 Time RNN이라는 이름의 계층으로 구현했습니다. 이번 절에서도 마찬가지로, 시계열 데이터를 한꺼번에 처리하는 계층을 Time Embedding, Time Affine 형태의 이름으로 구현하겠습니다. 이 Time XX 계층들을 다 만들면 우리가 원하는 신경망을 [그림 5-27] 형태로 구현할 수 있습니다.
- 예컨대, Time Affine 계층은 [그림 5-28]처럼 Affine 계층을 T개 준비해서, 각 시각의 데이터를 개별적으로 처리하면 됩니다.

5.5 RNNLM 학습과 평가

5.5.1 RNNLM 구현

5.5.2 언어 모델의 평가

- 언어 모델의 예측 성능을 평가하는 척도로 [퍼플렉서티]를 자주 이용합니다.
- [퍼플렉서티]는 간단히 말하면 '확률의 역수'입니다.
- 이 예에서 퍼플렉서티는 작을수록 좋다는 것을 알 수 있습니다.
- 이 값은 '분기수'로 해석할 수 있습니다. 분기 수란 다음에 취할 수 있는 선택사항의 수(구체적으로 말하면, 다음에 출현할 수 있는 단어의 후보 수)를 말합니다.
- 즉, 퍼플렉서티가 작아질수록 분기 수가 줄어 좋은 모델이 됩니다.
- [RNNLM 내부 작용 방식]
  1. 미니배치를 '순차적'으로 만들어
  2. 모델의 순전파와 역전파를 호출하고
  3. 옵티마이저로 가중치를 갱신하고
  4. 퍼플렉서티를 구합니다.