chapter 8 – 어텐션

8.1 어텐션의 구조

- 이번 장에서는 지금까지 배운 seq2seq를 한층 더 강력하게 하는 [어탠션 메커니즘]이라는 아이디어를 소개합니다.
- 이 어텐션이라는 메커니즘 덕분에 seq2seq는 (우리 인간처럼) 필요한 정보에만 '주목'할 수 있게 됩니다.
- 앞 장에서도 수행한 seq2seq 개선은 '작은 개선'이었습니다. 이와 달리, 지금부터 설명하는 어텐션 기술은 지금까지의 seq2seq가 안고 있던 근본적인 문제를 해결하는 '큰 개선'입니다.

8.1.1 seq2seq 문제점

- seq2seq에서는 Encoder가 시계열 데이터를 인코딩합니다. 그리고 인코딩된 정보를 Decoder로 전달하죠. 이때 Encoder의 출력은 '고정 길이의 벡터'였습니다. 그런데 실은 이 '고정 길이'라는 데에 큰 문제가 잠재해 있습니다. 고정 길이 벡터라 함은 입력 문장의 길이에 관계없이(아무리 길어도), 항상 같은 길이의 벡터로 변환한다는 뜻입니다. 앞 장의 번역 예로 설명하면, 아무리 긴 문장이 입력되더라도 항상 똑같은 길이의 벡터에 밀어 넣어야 합니다.

8.1.2 Encoder 개선

- 지금까지 우리는 LSTM 계층의 마지막 은닉 상태만을 Decoder에 전달했습니다. 그러나 Encoder 출력의 길이는 입력 문장의 길이에 따라 바꿔주는게 좋습니다. 이 점이 Encoder의 개선 포인트입니다.
- [그림 8-2]처럼 각 시각(각 단어)의 은닉 상태 벡터를 모두 이용하면 입력된 단어와 같은 수의 벡터를 얻을 수 있습니다. [그림 8-2]의 예에서는 5개의 단어가 입력되었고, 이때 Encoder는 5개의 벡터를 출력합니다. 이것으로 Encoder는 '하나의 고정 길이 벡터'라는 제약으로부터 해방됩니다.
- 많은 딥러닝 프레임워크에서는 RNN 계층(혹은 LSTM 계층과 GRU 계층 등)을 초기화할 때, '모든 시각의 은닉 상태 벡터 반환'과 '마지막 은닉 상태 벡터만 반환' 중 선택할 수 있습니다.
- [그림 8-5]에서 보듯 앞 장의 Decoder는 Encoder의 LSTM 계층의 마지막 은닉 상태만을 이용합니다. hs에서 마지막 줄만 빼내어 Decoder에 전달한 것이죠.
- 기계 번역의 역사를 보면 '고양이 = cat'과 같은 단어의 대응 관계 지식을 이용하는 연구는 많이 이뤄져 왔습니다. 단어(혹은 문구)의 대응 관계를 나타내는 정보를 [얼라인먼트]라 하는데, 지금까지는 얼라인먼트를 주로 사람이 수작업으로 만들었습니다. 그러나 지금부터 설명하는 어텐션 기술은 얼라인먼트라는 아이디어를 seq2seq에 자동으로 도입하는 데 성공했습니다. 이번에도 다시 한번, '수작업'에서 '기계에 의한 자동화'라는 흐름이 이어지고 있습니다.
- 다시 말해, 필요한 정보에만 주목하여 그 정보로부터 시계열 변환을 수행하는 것이 목표입니다. 이 구조를 어텐션이라 부르며, 이번 장의 핵심 주제입니다.
- [그림 8-6]처럼 여기에서는 새롭게 '어떤 계산'을 수행하는 계층을 추가할 겁니다. 이 '어떤 계산'이 받는 입력은 두 가지로, 하나는 Encoder로부터 받는 hs이고, 다른 하나는 시각별 LSTM 계층의 은닉 상태입니다. 그리고 여기에서 필요한 정보만 골라 위쪽의 Affine 계층으로 출력합니다. 참고로, 지금까지와 똑같이 Encoder의 마지막 은닉 상태 벡터는 Decoder의 첫 번째 LSTM 계층에 전달합

니다.

- 그런데 [그림 8-6]의 신경망으로 하고 싶은 일은 단어들의 얼라이먼트 추출입니다. 각 시각에서 Decoder에 입력된 단어와 대응 관계인 단어의 벡터를 hs에서 골라내겠다는 뜻이죠. 그리고 이러한 '선택' 작업을 '어떤 계산'으로 해내겠다는 겁니다.
- 하지만 여기서 문제가 발생합니다. 바로 선택하는 작업(여러 대상으로부터 몇 개를 선택하는 작업)은 미분할 수 없다는 점입니다.
- 신경망의 학습은 (일반적으로) 오차역전파법으로 이뤄집니다. 따라서 미분 가능한 연산으로 신경망을 구축하면 오차역전파법의 틀 안에서 학습을 수행할 수 있습니다. 반대로 미분 가능한 연산을 이용하지 않으면 (기본적으로는) 오차역전파법을 사용할 수 없습니다.
- '선택한다'라는 작업을 미분 가능한 연산으로 대체할 수는 없을까요? 사실 이 문제를 해결할 아이디어는 아주 단순하답니다. 그 아이디어란 '하나를 선택'하는 게 아니라, '모든 것을 선택'한다는 것입니다. 그리고 이때 [그림 8-7]과 같이 각 단어의 중요도(기여도)를 나타내는 '가중치'를 별도로 계산하도록 합니다.
- [그림 8-7]에서 보듯, 여기에서는 각 단어의 중요도를 나타내는 '가중치'를 이용합니다. a는 확률분포처럼 각 원소가 0.0 ~ 1.0 사이의 스칼라이며, 모든 원소의 총합은 1이 됩니다.
- 그리고 각 단어의 중요도를 나타내는 가중치 a와 각 단어의 벡터 hs로부터 가중합을 구하여, 우리가 원하는 벡터를 얻습니다.
- 그 결과를 [맥락 벡터'라고 부르고, 기호로는 c로 표기합니다.
- 그런데 이 그림을 잘 보면 '나'에 대응하는 가중치가 0.8입니다. 이것이 의미하는 바는 맥락 벡터 c에는 '나' 벡터의 성분이 많이 포함되어 있다는 것입니다. 즉, '나' 벡터를 '선택'하는 작업을 이 가중합으로 대체하고 있다고 할 수 있습니다.

8.1.4 Decoder 개선

- [그림 8-12]에서는 Decoder의 LSTM 계층의 은닉 상태 벡터를 h라 했습니다. 지금 목표는 h가 hs의 각 단어 벡터와 얼마나 '비슷한가'를 수치로 나타내는 것입니다. 방법은 여러 가지가 있습니다만, 여에서는 가장 단순한 방법인 벡터의 '내적'을 이용하고자 합니다.
- 내적의 직관적인 의미는 '두 벡터가 얼마나 같은 방향을 향하고 있는가'입니다. 따라서 두 벡터의 '유사도'를 표현하는 척도로 내적을 이용하는 것은 자연스러운 선택이라고 할 수 있습니다.

8.1.5 Decoder 개선 (2)

- 우리는 이 계산을 Weight Sum 계층과 Attention Weight 계층, 2개로 나눠 구현했습니다. 다시 말하지만, 이 계산에 따르면 Attention Weight 계층은 Encoder가 출력하는 각 단어의 벡터 hs에 주목하여 해당 단어의 가중치 a를 구합니다. 이어서 Weight Sum 계층이 a와 hs의 가중합을 구하고, 그 결과를 맥락 벡터 c로 출력합니다. 우리는 이 일련의 계산을 수행하는 계층을 Attention 계층이라고 부르겠습니다.
- Attention Weight 계층 = Weight Sum + Attention Weight를 합한 계층
- 이제 이 Attention 계층을 우리는 LSTM 계층과 Affine 계층 사이에 삽입하면 됩니다.

8.4 어텐션에 관한 남은 이야기

8.4.1 양방향 RNN

- [그림 8-30]에서 보듯, 양방향 LSTM에서는 지금까지의 LSTM 계층에 더해 역방향으로 처리하는 LSTM 계층도 추가합니다. 그리고 각 시각에서는 이 두 LSTM 계층의 은닉 상태를 연결시킨 벡터를 최종 은닉 상태로 처리합니다.
- 이처럼 양방향으로 처리함으로써, 각 단어에 대응하는 은닉 상태 벡터에는 좌와 우 양쪽 방향으로부터의 정보를 집약할 수 있습니다. 이렇게 해서 균형 잡힌 정보가 인코딩되게 됩니다.
- LSTM 계층 하나는 지금까지와 똑같습니다. 즉, 입력 문장을 '왼쪽부터 오른쪽으로' 처리하는 일반적인 LSTM 계층입니다. 한편, 또 하나의 LSTM 계층에는 입력 문장의 단어들을 반대 순서로 나열합니다. 두 번째 LSTM 계층은 입력문을 '오른쪽에서 왼쪽으로' 처리하게 됩니다. 마지막으로 이 두 LSTM 계층의 출력을 연결하기만 하면 양방향 LSTM 계층이 완성됩니다.

8.4.3 seq2seq 심층화와 skip 연결

- 번역 등 현실에서의 애플리케이션들은 풀어야 할 문제가 훨씬 복잡합니다. 그렇다면 어텐션을 갖춘 seq2seq에도 더 높은 표현력이 요구될 것입니다. 이때 우선 생각해야 할 것은 RNN 계층(LSTM 계층)을 깊게 쌓는 방법입니다.
- 층을 깊게 할 때 사용되는 중요한 기법 중 [skip 연결]이라는 게 있습니다.
- [skip 연결]은 [그림 8-34]처럼 계층을 넘어 '선을 연결'하는 단순한 기법입니다. [그림 8-34]에서 보듯, skip 연결은 '계층을 건너뛰는 연결'입니다. 이때 skip 연결의 접속부에서는 2개의 출력이 '더해'집니다. 이 덧셈이 핵심입니다. 왜냐하면 덧셈은 역전파 시 기울기를 '그대로 흘려'보내므로, skip 연결의 기울기가 아무런 영향을 받지 않고 모든 계층으로 흐르기 때문이죠. 따라서 층이 깊어져도 기울기가 소실 (혹은 폭발)되지 않고 전파되어, 결과적으로 좋은 학습을 기대할 수 있습니다.
- RNN의 깊이 방향 기울기 소실에는 여기서 설명한 skip 연결이 효과적입니다.

8.5 어텐션 응용

8.6 정리

- 번역이나 음성 인시 등, 한 시계열 데이터를 다른 시계열 데이터로 변환하는 작업에서는 시계열 데이터 사이의 대응 관계각 존재하는 경우가 많다.
- 어텐션은 두 시계열 데이터 사이의 대응 관계를 데이터로부터 학습한다.
- 어텐션에서는 (하나의 방법으로서) 벡터의 내적을 사용해 벡터 사이의 유사도를 구하고, 그 유사도를 이용한 가중합 벡터가 어텐션의 출력이 된다.
- 어텐션에서 사용하는 연산은 미분 가능하기 때문에 오차역전파법으로 학습할 수 있다.
- 어텐션이 산출하는 가중치(확률)를 시각화하면 입출력의 대응 관계를 볼 수 있다.
- 외부 메모리를 활용한 신경망 확정 연구 예에서는 메모리를 읽고 쓰는 데 어텐션을 사용했다.