Slide2

RNN은 한국어로는 순환 신경망으로, 단어, 문장과 같은 순서가 존재하는 sequential data와 주식과 같이 수서가 존재하지만 특별히 시간에 따른 의미가 존재하는 시계열 데이터를 다루기 위해 만들어진 신경망입니다. 이 그림과 같이 위에 순환성이 없는 일반 인공 신경망과 달리 RNN은 순환을 하기 때문에 순환성이 있습니다. 은닉층의 노드들은 어떠한 초기값을 가지고 있고 계산이 시작되면 이전 시점에서의 결과값과 새로 들어온 입력값을 가지고 새로운 결과값을 계산하는데 이때 이전 노드들에서 계산이 끝난게 아니라 계속 t=0시점에서 다시 입력값이 계속 들어오기 때문에 결과값이 다시 계산되는데 이 과정이 지정한 시간만큼 반복되므로 순환성을 가진다고 합니다. 이 식은 Cell A에서 계산되는데 기본적으로 순환 신경망에서는 하이퍼볼릭 탄젠트 활성화 함수를 사용하기 때문에 이와 같은 수식이 사용됩니다.

Slide3

RNN cell의 문제점은 RNN의 역전파 방법인 BPTT(BackPropagation Through Time)은 이 그림과 같이 모든 타임스텝마다 처음부터 끝까지 역전파합니다. 그렇기 때문에 타임 스텝이 클 경우, 위의 그림과 같이 RNN을 펼치게(unfold)되면 매우 깊은 네트워크가 될것이며, 이러한 네트워크는 **그래디언트 소실 및 폭주**(vanishing & exploding gradient) 문제가 발생할 가능성이 큽니다. 그리고, 계산량 또한 많기 때문에 한번 학습하는데 아주 오랜 시간이 걸리는 문제가 있습니다. BPTT의 이러한 문제를 해결하기 위해 아래의 그림과 같이 타임 스텝을 일정 구간(보통 5-steps)으로 나누어 역전파를 계산하여, 전체 역전파로 근사시키는 방법인 **Truncated BPTT**를 대안으로 사용할 수 있습니다. 하지만 truncated-BPTT의 문제는 만약 학습 데이터가 장기간에 걸쳐 패턴이 발생한다고 하면, 이러한 장기간의 패턴을 학습할 수 없는 문제가 있습니다.  
  
RNN은 타임 스텝 ​에서 이전 타임 스텝의 상태를 입력으로 받는 구조이기 때문에, 이전의 정보가 현재의 타임 스텝 ​에 영향을 줄 수 있습니다. RNN은 이론적으로 모든 이전 타임 스텝이 영향을 주지만 앞쪽의 타임 스텝은 타임 스텝이 길어질 수록 영향을 주지 못하는 문제가 발생하는데 이를 **장기 의존성(Long-Term Dependency) 문제**라고 합니다. 이러한 문제가 발생하는 이유는 입력 데이터가 RNN Cell을 거치면서 특정 연산을 통해 데이터가 변환되어, 일부 정보는 타임 스텝마다 사라지기 때문입니다. 이러한 문제를 해결하기 위해 장기간의 메모리를 가질 수 있는 여러 종류의 셀이 만들어졌는데, 그 중에서 대표적인 셀들이 LSTM과 GRU 셀입니다.

Slide4

LSTM은 RNN 셀의 장기 의존성 문제를 해결할 뿐만 아니라 학습 또한 빠르게 수렴합니다. LSTM 셀에서는 상태가 두 개의 벡터 ​ht와ct​로 나누어 진다는 것을 알 수 있습니다. ht​를 단기 상태(short-term state), ​ct를 장기 상태(long-term state)라고 볼 수 있습니다. LSTM의 핵심은 네트워크가 장기 상태(​)에서 기억할 부분, 삭제할 부분, 그리고 읽어 들일 부분을 학습하는 것이다. 핍홀 연결(peephole connection)은 LSTM의 변종이다. 기존의 LSTM에서 gate controller(ft, it, ot​)는 입력 ​xt와 이전 타임스텝의 단기 상태 ​ht-1만 입력으로 받았었는데 핍홀 연결을 같이 연결 해주면서 gate controller에 이전 타임스텝의 장기 상태 ct-1​가 입력으로 추가되며, 좀 더 많은 맥락를 인식할 수 있습니다.

GRU는 LSTM 셀의 **간소화된 버전입니다.** LSTM Cell에서의 두 상태 벡터 ​ct와 ht​가 하나의 벡터 ht​로 합쳐졌으며GRU 셀은 output 게이트가 없어 전체 상태 벡터 ​가 타임 스텝마다 출력되며, 이전 상태 ​의 어느 부분이 출력될지 제어하는 새로운 gate controller인 ​rt가 있습니다.

Cell의 복잡도는 LSTM, GRU, RNN 순으로 복잡하며 구조가 복잡해질수록 chainability가 감소하여 좋은 성능을 낼 수 있지만 더 많은 학습차원이 필요합니다.

\*LSTM 보충설명

장기 기억 ct​은 셀의 왼쪽에서 오른쪽으로 통과하게 되는데 **forget gate**를 지나면서 일부를 기억(정보)을 잃고, 그 다음 덧셈(+) 연산으로 **input gate**로 부터 새로운 기억 일부를 추가한다. 이렇게 만들어진 ct​는 별도의 추가 연산 없이 바로 출력되며, 이러한 장기 기억 ​는 타임 스텝마다 일부의 기억을 삭제하고 추가하는 과정을 거치게 된다. 그리고 덧셈 연산 후에 ct​는 복사되어 **output gate**의 ​함수로 전달되어 단기 상태 ht​와 셀의 출력인 ​yt를 만든다. 이번에는 위에서 설명한 forget, input, output 게이트(gate)가 어떻게 작동하는지에 대해 알아보도록 하자.

먼저, 현재 입력 벡터 ​와 이전의 단기 상태 ​이 **네 개의 다른 FC-레이어(Fully-Connected layer)에 주입**되는데, 이 레이어는 모두 다음과 같이 다른 목적을 가진다.

* 주요 레이어는 ​gt를 출력하는 레이어이며, 현재 입력 데이터 ​xt와 이전 타임스텝 ht-1의 단기 상태 ​을 분석하는 역할을 한다. LSTM 셀에서는 이 레이어의 출력인 ​gt가 it​의 곱셈연산 후 장기 상태 ​ct에 일부분이 더해지게 된다. 반면에, [**기본 RNN 셀**](http://excelsior-cjh.tistory.com/183)에서는 이 레이어만 있으며, 바로 ​ht와 yt​로 출력된다.
* ​Ft, it, ot를 출력하는 세 개의 레이어에서는 활성화 함수로 시그모이드(sigmoid, logistic)를 사용한다. 시그모이드 함수의 출력의 범위는 0 ~ 1 이며, 이 출력값은 각 forget, input, output 게이트의 원소별(element-wise) 곱셈연산에 입력된다. 따라서, 출력이 0일 경우에는 게이트를 닫고 1일 경우에는 게이트를 열기 때문에 ft, it, ot​를 **gate controller**라고 한다.
  + **Forget gate** : ​에 의해 제어되며 장기 상태 ​의 어느 부분을 삭제할지 제어한다.
  + **Input gate** : ​에 의해 제어되며 ​의 어느 부분이 장기 상태 ​에 더해져야 하는지 제어한다.
  + **Output gate** : ​는 장기 상태 ​의 어느 부분을 읽어서 ​ 와 ​로 출력해야 하는지 제어한다.

\*GRU 보충설명

하나의 gate controller인 ​zt가 **forget**과 **input** 게이트(gate)를 모두 제어한다. ​zt가 1을 출력하면 forget 게이트가 열리고 input 게이트가 닫히며, zt​가 0일 경우 반대로 forget 게이트가 닫히고 input 게이트가 열린다. 즉, 이전(​t-1)의 기억이 저장 될때 마다 타임 스텝 t​의 입력은 삭제된다.

Slide5

어떤 데이터 형식을 입력하고 출력값들 중 무엇을 취하냐에 따라 one to one, one to many, many to one, many to many를 골라서 사용하는데 many to one은 주로 번역기, many to man는 주로 비디오에 많이 사용됩니다.

Slide6

이 코드는 파이토치에 hello를 rnn으로 학습시킨 것입니다. Output shape와 input shape를 결정할 때 맨 앞의 3은 batch size입니다. 코드에서 batch를 이렇게 3개로 설정했습니다. 두번째 5는 sequence length로 ‘hello’가 다섯글자이기 때문에 5이고 sequence length는 파이토치에서 자동으로 계산해줍니다.. Input shape의 마지막 4는 one-hot-encoding 종류 수인데 ‘hello’에서 ‘h’, ‘e’, ‘l’, ‘o’ 이렇게 네 가지의 문자가 사용되었기 때문에 4입니다. One-hot-encoding은 이와 같이 1과 0 두 숫자로 문자를 나타내는 것인데 1은 하나만 쓰고 나머지는 0으로 채우고 문자마다 1의 순서를 다르게 하면 됩니다. Output의 맨 뒤 숫자 2는 hidden state로 바람직한 output 크기를 설정하면 됩니다. 이 rnn부분이 옆 그림에서 A 부분에서 이루어지는 연산입니다.

Slide7

이 코드는 입력한 string을 rnn으로 학습시켜주는 코드입니다. In[11]번 줄을 이용하면 string에 속해있는 문자들을 list에 담아서 이렇게 임의로 index를 매겨주고 np.eye를 이용하면 identity matrix를 생성해서 각 줄을 one-hot-encoding vector로 이용할 수 있게 됩니다. In[17]번을 보면 optimizer.zero\_grad()는 새로운 gradient를 생성하는 것이고 이를 안쓰면 gradient가 축적되어 제대로 training이 안됩니다. 밑의 \_status는 hidden state이고 loss.backward()는 back propagation gradient를 계산하는 것입니다.

Slide8

긴 문장을 학습시킬 때는 문장을 특정 size로 잘라서 사용하고 한 칸 씩 움직이는 것을 반복하면서 진행해야합니다. 이 코드는 띄어쓰기까지 합쳐서 알파벳 100글자로 이루어진 문장을 학습시키는 것입니다. 여기서 특정 size를 10으로 주었고 앞에서 10글자만 가져온 다음 다음 은닉 층에는 한 칸 씩 움직이면서 학습시킨 것입니다. 그래서 x\_str는 i부터 i+9번째 까지 10글자를 가져오고 y\_str는 한 칸 뒤인 i+1부터 i+10까지만 가져온 것입니다. 이 코드에서는 layer를 2개 주어서 계산하였습니다.

Slide9

Seq2seq는 seq를 입력받고 seq를 출력하는 것으로 chatbot이나 번역에 사용되는 data를 말합니다. Eocoder-Decoder구조로 이루어져 있는데 encoder는 input을 vector형태로 압축시키는 것이고 decoder는 encoder에서 받은 정보를 첫 cell의 hidden state로 넣고 다음 단어를 예측 하는 것입니다. 이 코드는 gru를 이용하였는데 여기서 embedding은 총 단어의 개수만큼 one-hot-encoding을 해주는 것이고 밑에 이를 gru에 넣어주는 것입니다. train에서 encoder내용을 decoder내용으로 이동시켜주어서 학습을 합니다.

Slide10

길이가 다른 sequential data를 하나의 batch로 만드는 방법에는 padding과 packing이 있습니다. Padding은 가장 긴 data에 맞춰 나머지 부분은 padding으로 채우는 방식입니다. 가장 긴 data 길이\* data 수로 tensor가 깔끔하다는 장점이 있으나 padding으로 채운 뒷부분도 계산한다는 단점도 있습니다. Packing은 sequence에 대한 길이 정보도 저장하는 방식입니다. 길이 내림차순으로 저장해야 파이토치에서 작동됩니다. 뒷부분은 계산안하므로 padding보다 효율적이나 구현이 복잡하다는 단점이 있습니다. Pad\_sequence, pack\_sequence로 data를 paddig, packing할 수 있고 padding, packing된 sequence를 one-hot character embedding한 다음 RNN 모델에 이렇게 넣을 수 있습니다.

Slide11

Padding된 sequence를 packing sequence로, packing된 sequence를 padding sequence로 바꿔줄 수 잇습니다. 먼저 packing된 sequence를 padding sequence로 만들 때 pad\_packed\_sequence 함수를 쓰면 되는데 packed sequence는 각 sequence에 대한 길이 정보도 가지고 있기 때문에 이 함수는 tensor와 함께 길이에 대한 리스트르 튜플로 리턴해 줍니다.

Padding sequence를 packing sequence로 만들 때 pack\_padded\_sequence함수를 사용하면 되는데 padding sequence는 실제 sequence길이에 대한 정보를 모르기 때문에 꼭 파라미터로 제공해주어야 합니다. 그리고 pack\_sequence와 같이 길이에 대해 내림차순으로 정렬되어 있어야 packed sequence로 올바르게 변환될 수 있습니다.