Abstract

일반적으로 머신 러닝, 인공지능에서 사용되는 sample은 강아지, 고양이, 피자와 같은 IID 데이터 (Independent and Identically Distributed Data)이지만, recurrent(반복되는) neural network 모델은 Sequential data를 prediction하기 위한 모델이다. Sequence data는 음성신호, 텍스트와 같이 어떤 순서를 가진 데이터이다. 이 순서가 변경될 경우 순차 데이터는 그 고유의 특성을 잃어버리게 되는 특징을 갖고 있다. 또한, 인공지능에서 순차 데이터를 다룰 때, 데이터의 길이가 고정되어 있지 않은 문제가 존재한다. 이에 따라 RNN 모델에서는 long-term dependencies를 다루는 능력이 꼭 필요하다. 우리는 Recurrent neural network based language model [5]과 LSTM [4] 모델을 통해 Sequential data prediction task에 대해 알아본다.

1 RNNLM

RNNLM이전, 대표적으로 n-gram모델과 같은 기존 모델은 좋은 성능을 내기 위한 여러 가정이 필요하였다. 단어, 구문, 의미 형태를 고려해야한다는 가정, 언어는 원자기호-문장을 형성하는 단어로 구성되야한다는 가정 등이 존재하였다. 이런 가정이 존재함과 더불어 고정된 개수의 단어만을 입력으로 받아야하는 한계까지 존재하였다. 이에 따라 시점(time step)의 개념이 도입된 RNN을 통해 입력의 길이를 고정하지 않는 모델을 만들어 내었고, 이 RNN으로 만든 모델인 RNNLM을 살펴본다.

1.1 Architecture

Recurrent neural network는 주어진 sequence data로부터 단어를 입력 받아 다음 단어를 예측하는 task를 수행하는 역할을 한다. 이는 input layer x, hidden layer s(context layer 또는 state라고도 한다), output layer y로 구성되어 있다. Network의 시간에 대한 변수를 t라고 할 때, Input vector x(t)는 current work를 나타내는 vector w(t)와 t-1의 시간에서 context layer의 출력인 s(t-1)로 구성되어 있다. 입력 x(t)를 받는 hidden layer는 x(t)에 weight가 곱해지고 활성화 함수 sigmoid를 통과하여 s(t)를 출력하고, 이 s(t)를 입력으로 받는 output layer는 s(t)에 weight가 곱해지고 softmax 함수를 적용하여 y(t)를 출력한다. Hideen layer를 A, output을 h로 하여 RNNLM의 model을 도식화한 것은 Figure1과 같다.

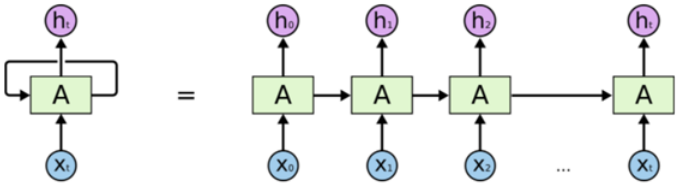


Figure 1: RNNLM Architecture

1.2 BPTT Algorithm

Figure1에서 이전시간(t-1)의 hidden layer의 output이 다음 시간(t)의 hidden layer의 input에 들어가는 것을 확인할 수 있다. 또한 t의 결과가 t+1, t+1의 결과가 t+2에 영향을 미칠 것이다. 이에 따라, recurrent란 이름이 붙은 것을 생각해볼 수 있다. 이는 supervised learning이지만, 현재 시간의 error는 과거 상태와 연관되어 있으므로 Sequence의 weights를 업데이트하기 위해서는 기존과는 다른 학습 알고리즘이 필요한데, truncated-BPTT(Backpropagation Through Time)알고리즘이 사용된다. 학습에서 기존 backpropagation과 동일하게 ground truth와 prediction값을 통해 cross entropy, SGD backpropagation를 사용하는 방식이지만 조금 다르다. 하나의 예를 들어 3개의 word로 구성된 sequence가 존재하고, 입력 x에 곱해지는 weight를 w라고 할 때 3번째의 출력 y에 대한 w의 gradient를 구하는 상황을 가정한다(이때, notation은 위에서 언급한 것으로 한다). 이 gradient는 (1)과 같은 식으로 구해진다.(truncated는 많은 모듈이 반복될 때 어느 정도의 이전시간 까지만 계산을 하는 것이다)

$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial h_3} * \frac{\partial h_3}{\partial w} + \frac{\partial E_3}{\partial h_3} * \frac{\partial h_3}{\partial h_2} * \frac{\partial h_2}{\partial w} + \frac{\partial E_3}{\partial h_3} * \frac{\partial h_3}{\partial h_2} * \frac{\partial h_2}{\partial h_1} * \frac{\partial h_1}{\partial w} \quad (1)$$

즉, sequence data에서 이전의 data가 현재의 data를 prediction할 때 영향을 미치는 것을 확인할 수 있다.

1.3 Limitation

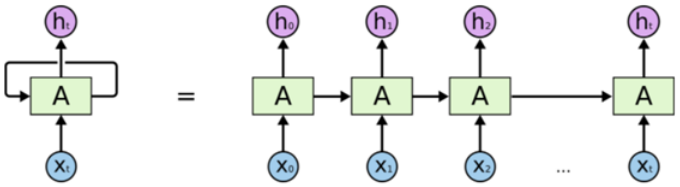
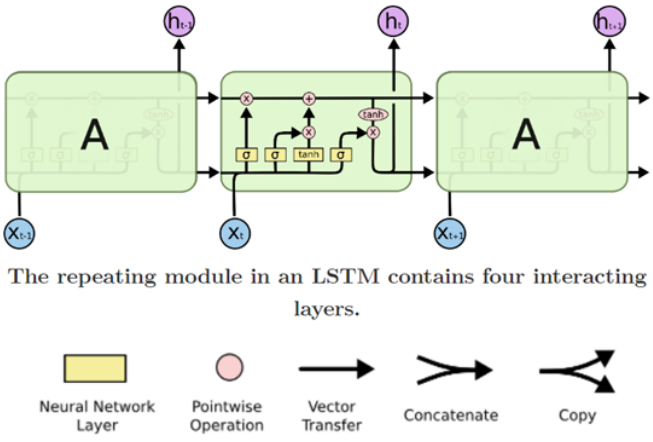


Figure 2: RNNLM Limitation

긴 Sequence data를 RNNLM의 architecture에 대입하였을 때, Figure2로 표현할 수 있는 문제가 존재한다. 만약 h_{t+1}의 work를 예측하려고 할 때, x_0와 x_1의 단어의 정보가 필요하다고 가정한다. 이때, Sequence가 점점 길어져 h_{t+1}와 x_0, x_1사이 gap이 커진다면, RNNLM 방식으로는 학습이 불가능할 것이다. 위 BPTT 알고리즘을 살펴보았을 때, chain-rule에 따라 과거의 시간에 대한 weight의 크기가 점점 줄어드는 Gradient Vanishing 문제가 발생하기 때문이다. 따라서 이 모델은 long-term dependencies를 다루지 못한다.

2 LSTMs



The repeating module in an LSTM contains four interacting layers.

Figure 3: Architecture of LSTM

LSTM Networks는 long-term dependencies의 학습이 가능한 모델로 RNNLM과 동일하게 neural network의 module이 반복되는 chain형태의 구조를 갖는다. 특히, 이는 모든 Recurrent neural networks에 적용되는 특징이다. 기존 RNN의 반복되는 module 하나의 layer만 존재하는 것과 달리, 4개의 layer가 존재하고, 이 layer 사이 interaction이 존재한다. 이 interaction을 자세히 살펴보고 전체 구조에 대해 파악해본다.

LSTM의 전체 구조를 살펴보면 위, 아래에 각각 Cell state와 Hidden state가 존재하고, 이는 지속해서 입력과 출력으로 사용된다(다음 module로 전달된다). 또한 Figure3의 위 diagram에서 각각의 line은 하나의 entire vector를 의미하고, 이는 어느 node의 출력이며 다른 nodes의 inputs이 된다. 분홍색 원은 multiply와 addition 연산을 수행하는 것을 의미한다. 노란 박스는 neural network를 통과하는 것을 의미한다.

2.1 Architecture of LSTM

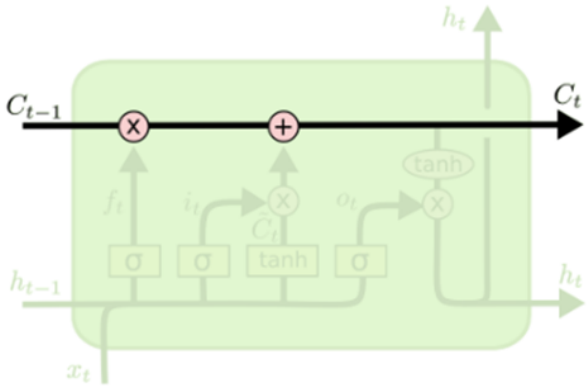


Figure 4: Interaction1 of module

LSTM의 diagram에서 전체 chain을 통과하는 수평의 선인 cell state이며, linear interactions(multiply, addition)이 수행된다. 이전 module에서 들어온 cell state에 대해 어떤 연산이 수행되는지 아래 Interaction2와 Interaction3에서 알 수 있다.

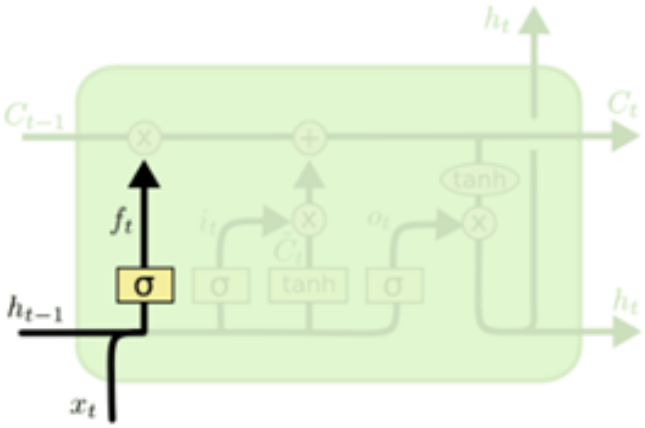


Figure 5: Interaction2 of module

Muliply : 새로운 정보 x_t와 과거의 hidden state h_{t-1}이 concatination되어 sigmoid neural net layer를 통과한다. 이때, sigmoid의 출력은 0 1사이의 값으로, cell state에 곱해져 과거의 정보에 대해 제거 또는 기억을 얼마나 할지 결정한다. 즉 cell state의 정보를 다음 cell State에 전달을 해줄 때 input을 바탕으로 어느 정도의 비중을 전달해줄 지 결정을 한다. 이 layer를 통과한 output을 f_t, sigmoid function을 σ, weight와 bias를 각각 W_f, b_f라 할 때, f_t는 아래와 같이 정의된다.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2}$$

Addition : 새로운 정보를 cell state에 추가하기 위한 layer이다. hidden state와 현재 input이 각각 sigmoid layer와 tanh layer를 통과하여 생성된 변수 i_t와 C_t가 곱해져 cell state에 더해진다. i_t와 C_t는 각각 아래와 같이 정의된다.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3}$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{4}$$

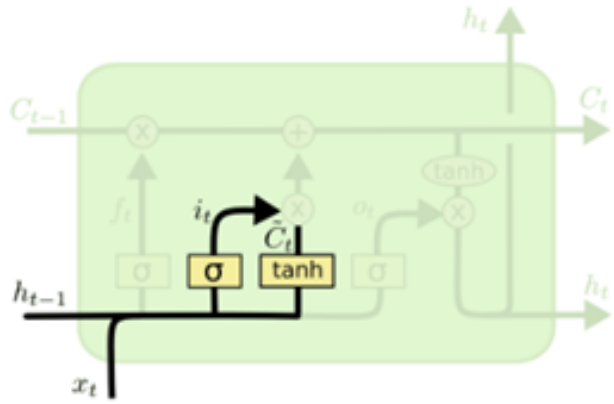


Figure 6: Interaction3 of module

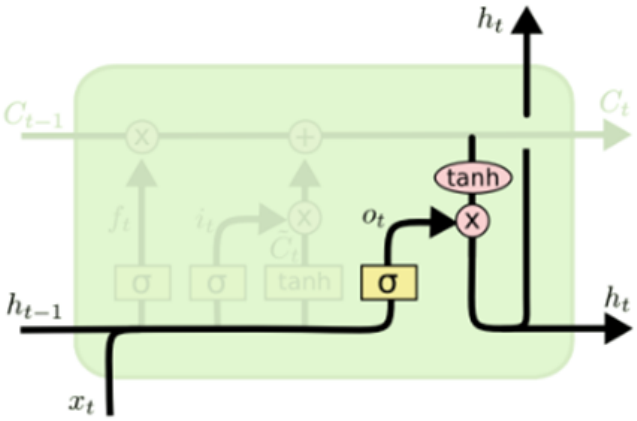


Figure 7: Interaction4 of module

마지막 Interaction은 어떤 input에 대한 outputh_t을 결정하는 layer로 prediction과 이후 module에서 쓰일 hidden state를 결정한다. cell state를 $C_t|'L, D\beta t\tilde{}\phi.o_t = \sigma(W_o[h_{t-1}, x_t] + b_o(5)$

$$h_t = o_t * tanh(C_t) \tag{6}$$

3 Conclusion

LSTM은 Input, cell state, hidden state사이 다양한 방식으로 Interaction를 할 수 있다. 하나의 예로, Figure과 같이 cell state가 다른 operation에 의해 줄어들고, 새로운 정보가 정해지는 방식이 아닌 이전의 cell state가 Interaction에 더해져 현재의 cell state를 함께 결정하는 방식이 있다. 이처럼 LSTM의 varient를 각 역할과 상황에 맞게 적용시켜 다양하게 활용할 수 있다. LSTM은 [6], [3], [2], [1] 의 연구에서 변형을 통해 RNN연구에 사용되고 있다. 특히 올해, 한국은행에서 LSTM 모델을 통해 2020년 상반기 GDP 급격한 성장률 하락 폭을 상대적으로 정확히 잡아내는 등 많은 곳에 활용되고 있다.

[1] Justin Bayer and Christian Osendorfer. Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*, 2014.

[2] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. *Advances in neural information processing systems*, 28, 2015.

[3] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *International Conference on Machine Learning*, pages 1462–1471. PMLR, 2015.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[5] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari, 2010.

[6] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.