RNN, LSTM

이번 장에서는 사진 위주로 피피티를 정리하고 그에 대한 설명은 각각의 페이지에 따로 적어두겠습니다.

학부연구생 이현재

RNN (Recurrent Neural Network)

Sequence Data

· 순서가 있는 data

· 예 : Text에는 문맥이라는 순서가 있고, 시계열 데이터에는 시간이라는 순서가 있음. 마찬가지로 영상, 음성 등도 모두 전부 순서와 함께 흘러가는 데이터

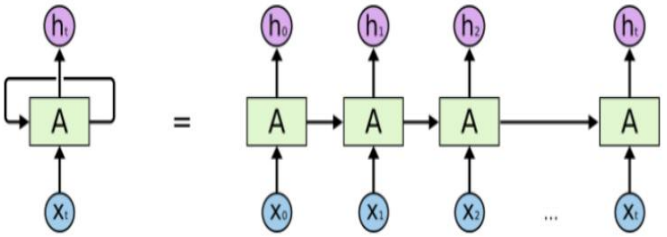
· Sequence Model : 순서가 있는 Sequence Data에서 특징들을 추출하여 문제를 해결하고 예측(RNN, LSTM, GRU)

· 이전에 공부한 CNN과 일반적인 Neural Network는 순차적으로 과거 정보를 반영하지 못함(모든 input들이 독립적)

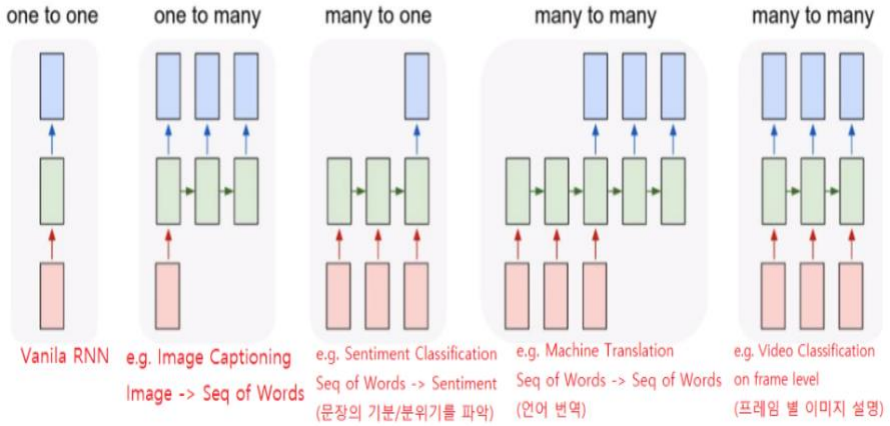
RNN

· 순차열(Sequence), 순서 있는 일련의 값들을 처리하는데 특화(더 긴 순차열로 확장 가능하며 대부분의 RNN은 가변 길이 순차열도 처리 가능)

· 과거 정보를 담을 수 없었던 문제점을 RNN은 루프를 통해 해결



< RNN 형태 >



< RNN 학습 유형 >

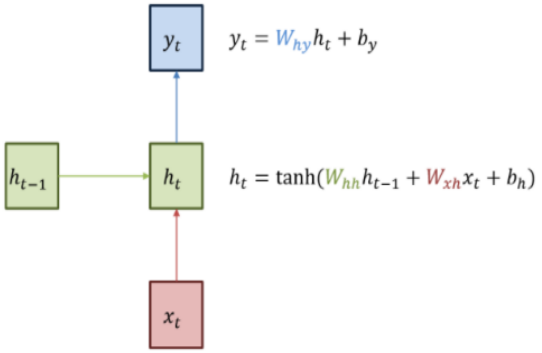
Prof. Hyerim Bae (hrbae@pusan.ac.kr)

RNN (Recurrent Neural Network)

이와 같은 형태가 재귀적으로 반복

· 매 time step마다 동일한 function과 동일한 parameters 사용되어야 합니다.

=> input, output size에 영향을 받지 않고, 무관하게 적용가능하게 되어집니다.

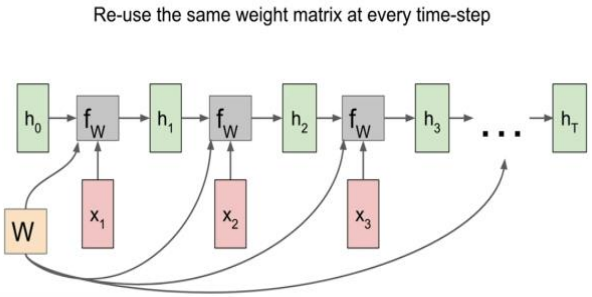


$$h_t = f_W(h_{t-1}, x_t)$$
$$h_t = tanh(W_{hh} * h_{t-1} + W_{xh} * x_t)$$
$$y_t = W_{hy}h_t$$

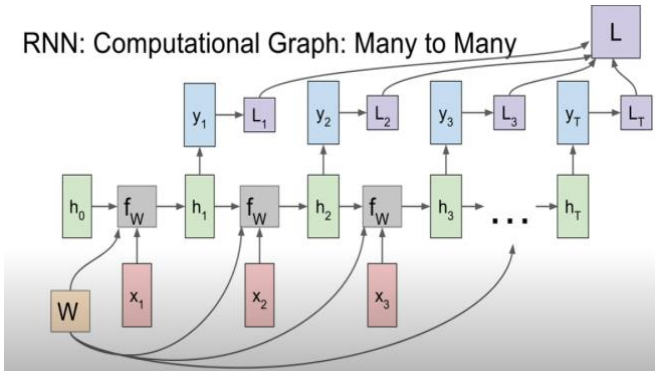
처음보는 tanh 사용

즉, RNN은 **가중치가 3개(W_{hh}, W_{xh}, W_{hy}) 존재한다.**

- W_{hx} : 입력 x에 곱해지는 가중치
- W_{hh} : 이전 hidden state에 곱해지는 가중치
- W_{hy} : y 출력 전에 곱해지는 가중치



< Shared Weights>



< RNN 학습 >

RNN의 역전파 : https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/

RNN (Recurrent Neural Network)

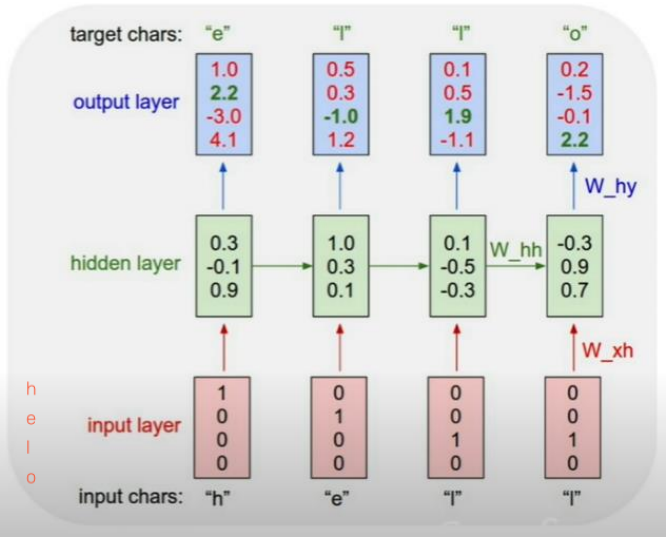
RNN 학습 과정 예시

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

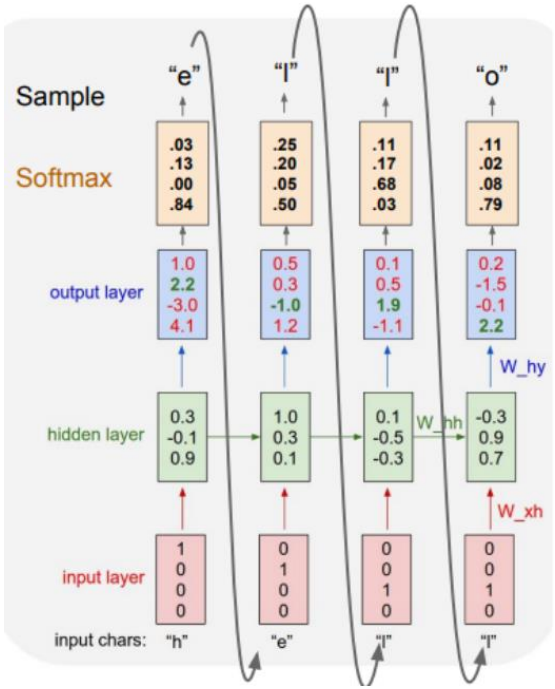
Example:
Character-level
Language Model

Vocabulary:
[h,e,l,o]

Example training sequence:
"hello"



Train



Test

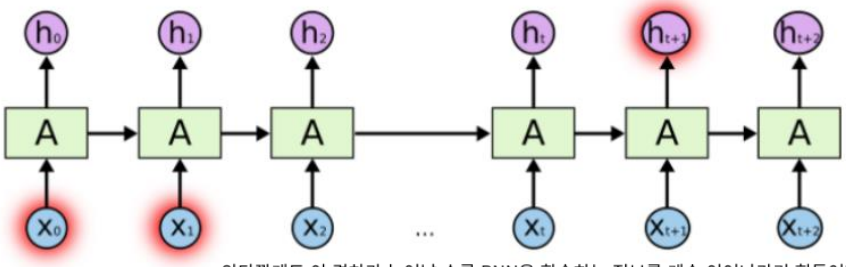
Prof. Hyerim Bae (hrbae@pusan.ac.kr)

RNN 문제점

· RNN은 구조 상 최근의 정보일수록 더 예측에 반영할 수 있게 설계

· 오래된 과거 정보를 이용할 경우, 학습이 잘 안되는 단점(long-term dependency problem)

· 나는 한국에서 자랐다. ....., 나는 __?__를 유창하게 말할 수 있다. ← ? 자리는 한국어가 적절하지만 너무 많은 문장 뒤에 이 ?가 왔기 때문에 예측을 잘하지 못한다.



안타깝게도 이 격차가 늘어날 수록 RNN은 학습하는 정보를 계속 이어나가기 힘들어한다.

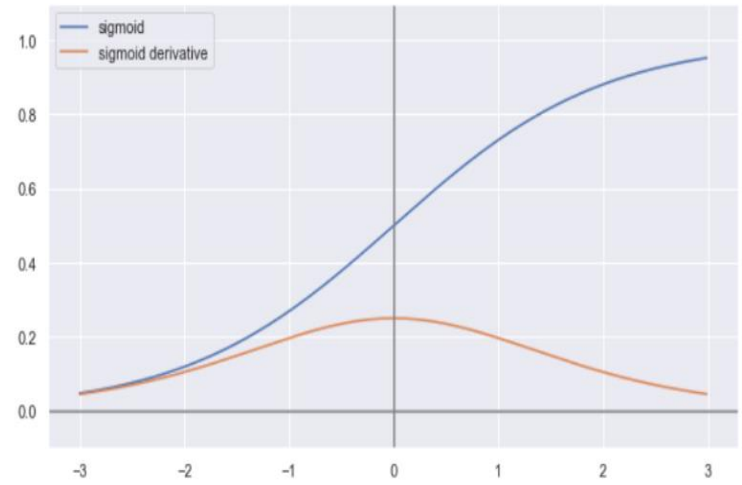
· 수식적 이해

- RNN은 입력 정보와 그 입력 정보를 사용하려는 출력지점 거리가 멀 경우 기울기가 점차 줄어들거나 커져서 학습 능력이 저하된다.
- 역전파시 기울기가 전 층 기울기와 계속 곱하는 연산으로 구해지기 때문.(연쇄법칙)

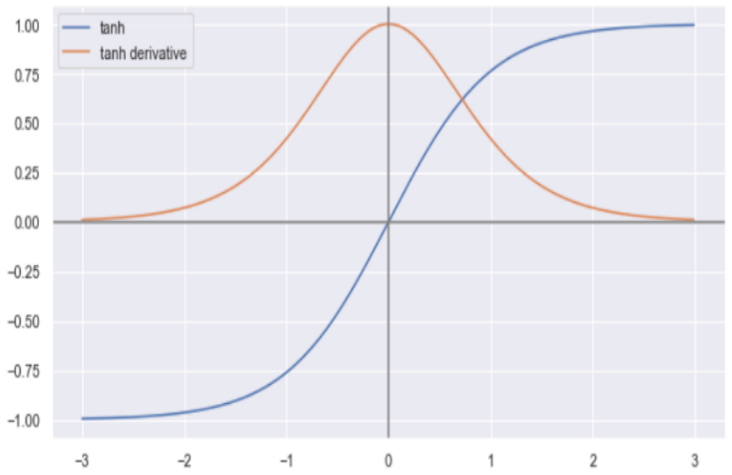
RNN에서 활성화함수로 tanh를 사용하는 이유

RNN의 Vanishing gradient 문제를 예방하기 위해서 gradient가 최대한 오래 유지될 수 있도록 해주는 역할로 tanh가 적합하기 때문

sigmoid



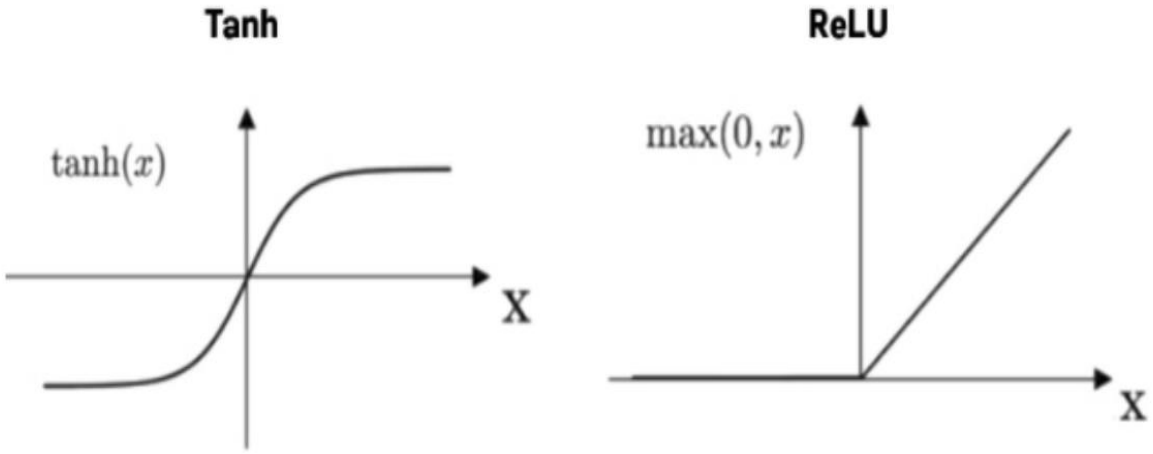
tanh



Prof. Hyerim Bae (hrbae@pusan.ac.kr)

RNN에서 활성화함수로 relu를 사용하지 않는 이유

과거의 값들이 끊임없이 재귀(반복적으로)로 사용되기때문에, -1~1 사이로 normalizing이 필요합니다.



RNN의 내부는 계속 순환하는 구조로 값이 1보다 크게 되면, ReLU 특성상 값이 발산할 수 있기 때문에 적합하지 않다고 합니다.

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

LSTM (Long Short Term Memory)

LSTM

· RNN의 단점을 보완하는, 긴 의존 기간을 필요로 하는 학습을 수행할 능력인 long-term dependency를 가지고 있음

· Cell State : 정보를 기억(저장)하는 역할

· RNN과의 차이점 : Cell State가 추가되어 LSTM의 state는 hidden state와 cell state 총 2개

-> input이 Sequence(=timestep)에 따라 이전 상태(old hidden state)와 현재 상태(hidden state)에 저장되기 앞서 기억 상태(cell state)를 활성화하는 처리과정

-> 이 처리과정에서 3개의 gate(i, f, o)가 cell state를 변화

LSTM

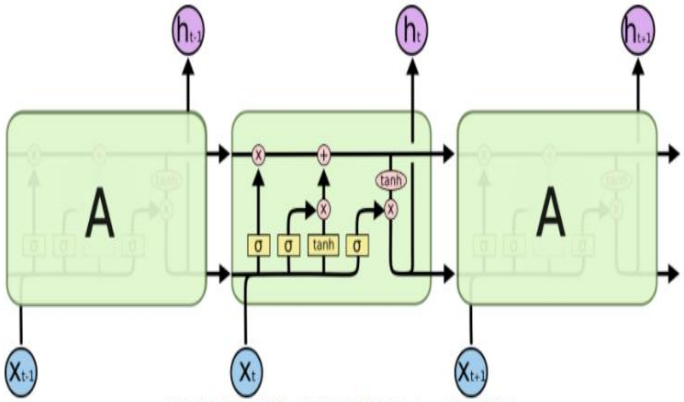
f: Forget gate, Whether to erase cell
i: Input gate, whether to write to cell
g: Gate gate (?), How much to write to cell
o: Output gate, How much to reveal cell

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

$$f_t = \sigma(W_{xh\_f} x_t + W_{hh\_f} h_{t-1} + b_{h\_f})$$
$$i_t = \sigma(W_{xh\_i} x_t + W_{hh\_i} h_{t-1} + b_{h\_i})$$
$$o_t = \sigma(W_{xh\_o} x_t + W_{hh\_o} h_{t-1} + b_{h\_o})$$
$$g_t = \tanh(W_{xh\_g} x_t + W_{hh\_g} h_{t-1} + b_{h\_g})$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$
$$h_t = o_t \odot \tanh(c_t)$$



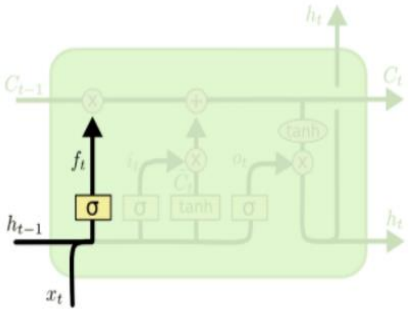
LSTM의 반복 모듈에는 4개의 상호작용하는 layer가 들어있다.

LSTM (Long Short Term Memory)

단계별로 알아보는 LSTM

⊙는 요소별 곱셈

· forget gate layer



$$f_t = \sigma(W_{xh\_f}x_t + W_{hh\_f}h_{t-1} + b_{h\_f})$$ - f ⊙ c(t-1)

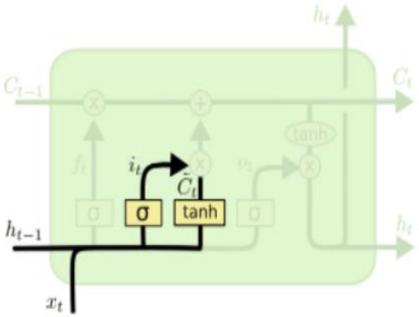
기존의 정보를 얼마나 잊어버리는가?

h(t-1)와 X(t)를 받아서 시그모이드를 취해준 값이 forget gate가 내보내는 값

(출력범위가 0에서 1사이이기 때문에 값이 0에 가까우면 이전 상태의 정보는 거의 잊는 것이고, 1이라면 이전 상태의 정보를 많이 기억)

f의 의미는 old cell state(이전 cell state)x를 얼마나 포함시켜줄거냐에 대한 여부 의미

· input gate layer



- i ⊙ g

새로운 정보를 얼마나 기억할 것인가?

h(t-1)와 X(t)를 받아서 시그모이드를 취해준 값과, 같은 입력의 tanh를 취해준 값을 product연산하여 input gate를 통해 내보내는 값입니다. (i 값의 범위는 0~1이고 g의 범위는 -1~1의 값)

즉 i의 의미는 현재 cell state를 얼마나(정도) 포함시켜줄거냐에 대한 여부 의미입니다.

즉 g의 의미는 새로운 정보를 의미합니다. -1 ~ 1 사이의 정보를 추출하는 것이며,

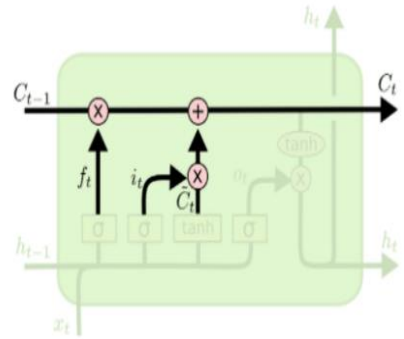
이것은 RNN에서 정보를 추출하는 일반적인 방법과 같다고 합니다.

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

LSTM (Long Short Term Memory)

단계별로 알아보는 LSTM

· cell state update

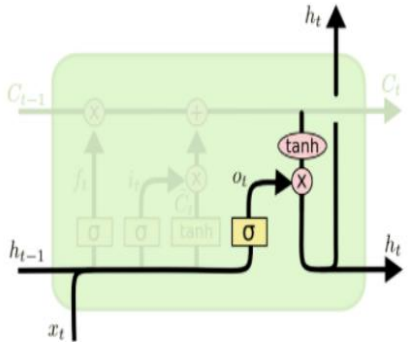


$$c_t = f \odot c_{t-1} + i \odot g$$

기존의 정보를 얼마나 잊고, 새로운 정보를 얼만큼 대체할 것인가?

생존해있던 old cell state(c(t-1))에 얼만큼 i ⊙g를 더해줄지

· output gate layer



$$h_t = o \odot \tanh(c_t)$$

C(t)는 기억만 하는 것이다. h(t)로 출력해야 의미가 있는 것이다.

계산된 c(t) ,즉 최종 cell state를 얼마나(정도) hidden state(현재 상태)에 포함시켜줄거냐에 대한 여부 의미

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

LSTM (Long Short Term Memory)

LSTM이 gradient vanishing에 강한 이유?

RNN의 h_t에 대한 계산식은 다음과 같다.

$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}X_t + b_h)$$

이때, h_T를 h_t에 대해 미분한다고 하면(T>t), 이는 chain rule에 의해 다음과 같이 표현될 수 있다.

$$\frac{\partial h_T}{\partial h_t} = \frac{\partial h_T}{\partial h_{T-1}} * \frac{\partial h_{T-1}}{\partial h_{T-2}} * \ldots * \frac{\partial h_{t+1}}{\partial h_t}$$

LSTM의 Cell State C_t에 대한 계산식은 다음과 같다.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

이때, C_T를 C_t에 대해 미분한다고 하면(T>t), 이는 chain rule에 의해 다음과 같이 표현될 수 있다.

$$\frac{\partial C_T}{\partial C_t} = \frac{\partial C_T}{\partial C_{T-1}} * \frac{\partial C_{T-1}}{\partial C_{T-2}} * \ldots * \frac{\partial C_{t+1}}{\partial C_t}$$

위 식의 f는 sigmoid함수의 output이기 때문에 (0,1)의 값을 갖게 되는데, 이 값이 1에 가까운 값을 갖게되면 미분값(gradient)이 소멸(vanished)되는 것을 최소한으로 줄일 수 있게된다. f값이 1에 가깝다는 것은, Cell State 공식에 의하면 오래된 기억(long term memory)에 대해 큰 비중을 둔다는 것과 같은데, 이로인해 gradient 또한 오래 유지된다는 것은 꽤나 흥미로운 현상이다.

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

LSTM이 gradient vanishing에 강한 이유?

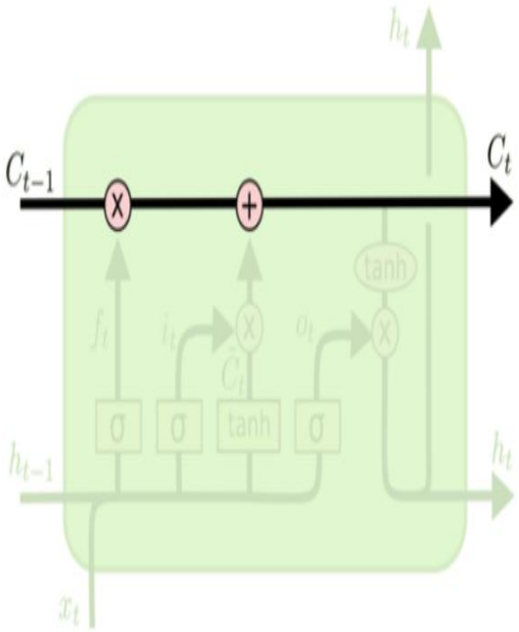
$$c^{(t)} = f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} \qquad h^{(t)} = o^{(t)} \circ \tanh c^{(t)}$$

만약 forget gate가 1로 설정, input gate가 0으로 설정하면, t시점에서의 cell state는 이전의 정보가 완전히 보존되는 채로 hidden state를 update하기 때문에 cell의 정보가 완전하게 보존될 것입니다. => 장기 의존성 문제 해결!!

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

LSTM (Long Short Term Memory)

LSTM이 gradient vanishing에 강한 이유?



먼저 맨 위를 가로지르는 선 C를 **cell state**라고 부릅니다.

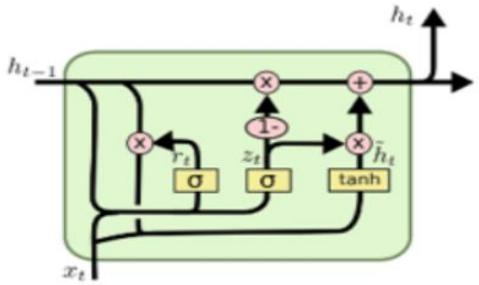
가장 눈에 띄는 부분은 **더하기(+)기호**입니다. 일반적인 RNN에서는 곱하기 연산으로만 이루어져 있었는데 LSTM에서는 피드백을 더하기(+)로 이음으로써 **Vanishing Gradient**와 같은 문제를 해결할 수 있습니다.

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

GRU (Gated Recurrent Unit)

GRU

· LSTM을 간소화한 모델



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right) \quad \text{---(1)}$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right) \quad \text{---(2)}$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right) \quad \text{---(3)}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \quad \text{---(4)}$$

· LSTM의 경우 forget gate, input gate, output gate 3개의 gate가 있었지만, GRU에서는 reset gate, update gate 2개의 gate만을 사용

· LSTM의 cell state, hidden state가 합쳐져 하나의 hidden state로 표현

· Reset Gate

- (2)번 식에 해당. 이전 시점의 hidden state와 현시점의 x를 시그모이드를 적용
- 이전 hidden state의 값을 얼마나 활용할 것인가에 대한 정보
- Reset gate에서 나온 값은 그대로 사용되는 것이 아니라 (3)번 식으로 다시 활용. (3)번 식에서는 전 시점의 hidden state에 reset gate를 곱하여 계산

· Update Gate

- LSTM의 input, forget gate와 비슷한 역할을 하며 과거와 현재의 정보를 각각 얼마나 반영할지에 대한 비율을 구하는 것이 핵심
- (1)번 식을 통해서 구한 결과 z는 현재 정보를 얼마나 사용할지 반영, (1-z)는 과거정보에 대해서 얼마나 사용할지 반영
- 최종적으로 (4)식을 통해 현 시점의 출력값 hidden state 구함

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

attention

Lstm의 단점을 보완하는 attention 모델이 있긴한데 추후에 시간이 남으면 정리해보겠습니다.

RNN, LSTM 관련 모델 코드 구현

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

참고 자료

- 밑바닥부터 시작하는 딥러닝