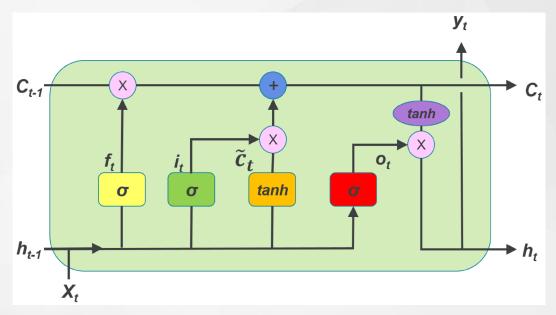


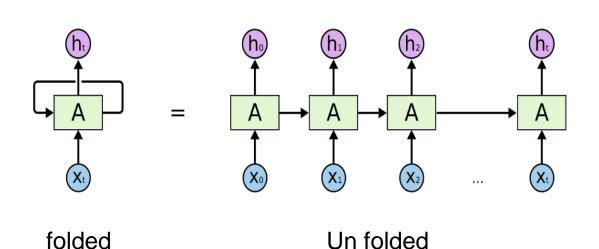
Chapter 09. 시계열을 활용한 딥러닝 (Time Sequence Processing)

RNN, Seq2Seq, LSTM, GRU



Recurrent Neural Network

- Recurrent : 반복되는 -> 시퀀스의 모든 요소에 대해 동일한 작업을 수행하고 출력은 이전 계산에 의존하기 때문에 *반복적* 이라고 합니다.
- RNN은 지금까지 계산 된 것에 대한 정보를 캡처하는 "메모리"가 있다는 것입 니다



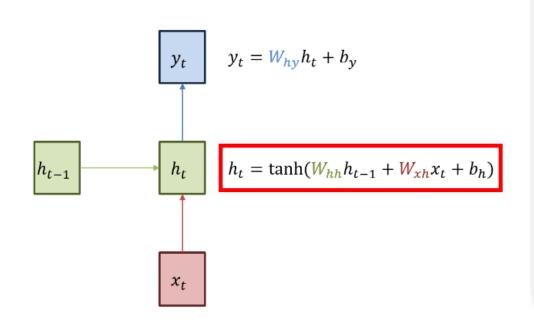
RNN 의 특징:

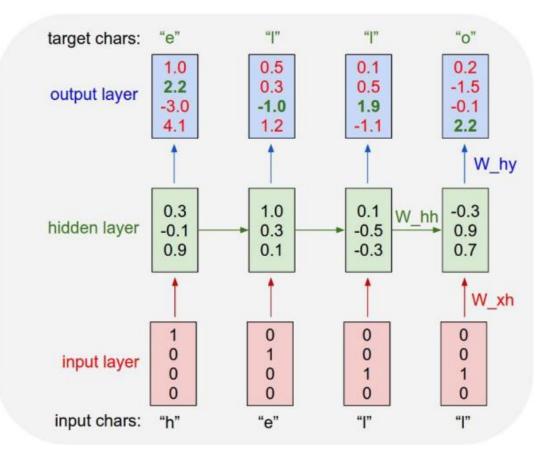
- 1) 장점 :
- 시퀀스 길이에 관계없는 input/output
- 2) 단점 :
- Gradient Vanishing : 상대 적으로 짧은 sequence 학습



RNN forward propagation

• RNN 은 "입력층", "은닉층(Hidden layer)", "출력층" 으로 구성되있다.

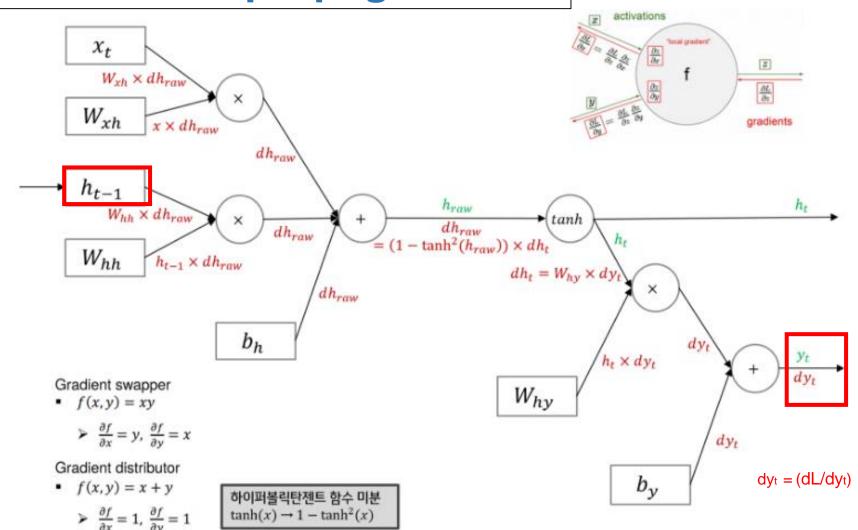






https://ratsgo.github.io/natural%20languag e%20processing/2017/03/09/rnnlstm/

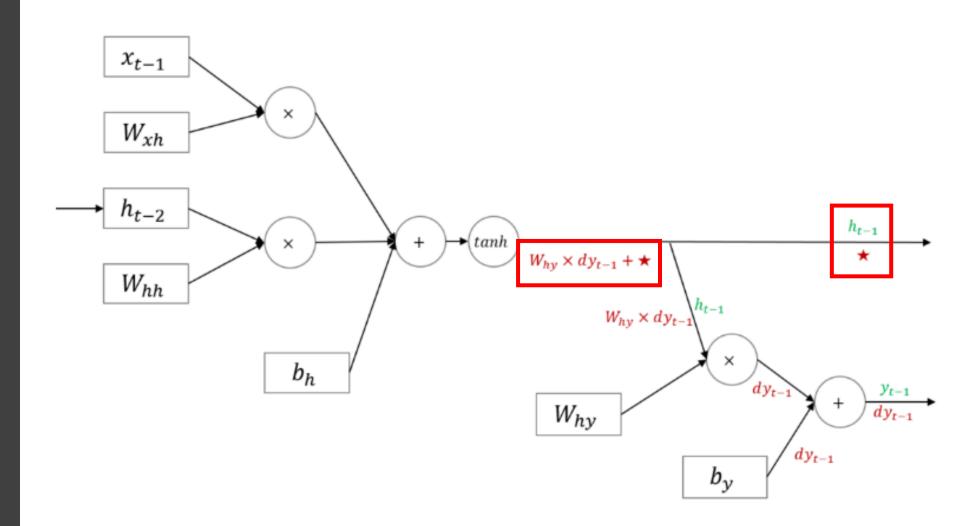
RNN backward propagation





https://ratsgo.github.io/natural%20languag e%20processing/2017/03/09/rnnlstm/

RNN backward propagation



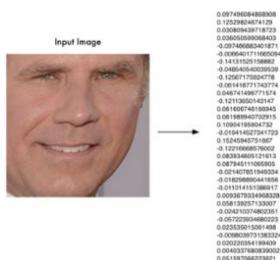


Sequence to Sequence

Encoding: 매우 복잡한 무엇인가를 단순한 것으로 표현할 수있게 해줍 니다.

Input Sentence

"Machine Learning is Fun!"



128 Measurements Generated from Image

0.045223236083984 0.060309179127216 0.17521631717682 -0.01981477253139 0.10801389068365 0.065554238855839 0.0731306001544 0.1226262897253 -0.029626874253154 0.036750309169292 -0.15958009660244 0.14114324748516 -0.031351584941149 -0.061901587992903 -0.15042643249036 -0.10568545013666 -0.12728653848171 -0.074287034571171 -0.065365232527256 0.0061761881224811 0.14746543765068 -0.21055991947651 0.0041091227903962 0.11345765739679 0.021352224051952 0.19372203946114 +0.086726233363152 0.084853030741215 0.09463594853878 0.0064811296761036 0.21180312335491 -0.16582328081131 -0.035577941685915 -0.0072777755558491 -0.036901291459799 -0.059730969369411 -0.070026844739914 0.11478432267904 -0.089621491730213 0.14841195940971 0.078333757817745 0.049525424838066 0.13227833807468 -0.051016297191381 -0.14132921397686 -0.062812767922878 -0.13407408598099 0.0048638740554452 -0.039491076022383 -0.11443792283535 0.071997955441475 0.014683869667351 0.05228154733777 -0.081752350867096 -0.031709920614958 0.037022035568953 0.11009479314089 0.12788131833076 0.18632389605045 -0.094398014247417 -0.11768248677254 -0.10034311562777 -0.040877258235216

0.032084941854014 0.020976085215807 -0.00052163278451189 -0.1318951100111 -0.0059557510539889 0.043374512344599 -0.053343612700701 0.078198105096817 -0.076289616525173 0.12369467318058 0.056418422609568 0.089727647602558 -0.0085843298584223 -D 022388197481632 0.020696049556136 0.050584398210041 -0.072376452386379 -0.034365277737379 -0.045013956725597 -0.013955107890069 -0.17898085713587 -0.0726000327432156 0.0050511929275229 -0.014829395338893 +0.043765489012003 -0.012062266469002 0.012774495407939 0.069833380512302 0.11638788878918 -0.015336792916059 0.10281457751989

-0.062041338086128

0.12529824674129 0.030809439718723 0.036050599068403 -0.097486883401871 -0.0066401711665094 -0.14131525158882 -0.048540540009538 -0.12567175924778 -0.061418771743774 0.046741496771574 -0.12113650143147 0.061606746166945 0.061989940702915 0.10904195904732 -0.019414527341723 0.15245945751887 -0.12216668576002 0.083934605121613 0.067945111095905 -0.021407851949334 -0.018298890441656 -0.011014151386917 0.0093679334968326 0.058139257133007 -0.02421037480235 -0.057223934680223 0.023535015061498 -0.000803973138332 0.020220354199409 0.0040337980839000 0.051597066223621

0.007406084868908

0.060309179127216 -0.01981477253139 0.005554238855839 0.1226262897253 0.036750309169290 0.14114324748516 -0.061901587992907 0.10568545013666 -0.074287034571171 0.006176188122481 -0.21055991947651 0.11345765739679 8 19372203946114 0.084853030741215 0.0064811296761036

0.045223239083984

Measurements Generated from Sentence

-0.1281466782093

0.17521031717682

0.10801389008365

-0.029626874253154

-0.15958009660244

0.07313060001544

-0.031351584941149 -0.15042643249035 -0.12728653848171 -0.065365232527256 0.14746543765068 0.0041091227903962 0.021352224051962 -0.086726233363152 0.09463594853878 0.21180312335491 -0.16582328081131 -0.035577941685919 -0.0072777755558491 -0.036901291459799 0.059730968369411 -D.070026844739914 0.11478432267904 -0.089621491730213 0.14841195949971 0.028333252612745 0.049525424838066 0.13227833807468 -0.05101629719138 -0.14132921397686 -0.062812767922878 -0.13407498598099 0.0048638740554452 -0.039491076022383 -0.11443792283535 0.071997955441475 0.014683869667351 0.05228154733777 -0.081752359867096 -0.031709920614958 0.037022035588953 0.11009479314089 0.12788131833076 0.18632389605045 -0.11768248677254 -0.094398014247417 -0.10034311562777 -0.040977258235216

0.032084941864014 0.020070005215807 -0.00052163278451186 -0.1318951100111 -0.0059557510539889 0.043374512344509 -0.053343612700701 0.078198105096817 -0.076289616525173 0.12369467318058 0.0564184229009568 0.089727647602558 -0.0085843298584223 -0.022388197481632 0.020696049556136 -0.050584398210049 -0.072376452386379 -0.034365277737371 -0.045013056725503 -0.013955107890068 -0.17898085713387 -0.0726000377432156 0.0050511928275226 -0.014829395338893 -0.043765489012003 40.012062266469000 0.012774495407939 0.069833360612392 0.11638788878018 -0.015336792916059

0.10281457751989

-0.052041338086128

이러한 얼굴 특징 측정값들은 다른 사람들의 얼굴이 다른 결과값으로 나오도록 훈련 된 신경망에 의 해 생성됩니다.

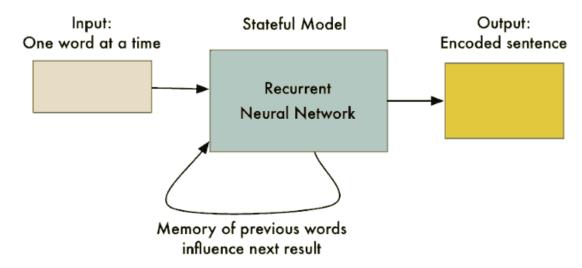
이 숫자 목록은 영어 문장인 "Machine Learning is Fun!"을 나타냅니다. 다른 문장은 다른 숫자의 집합으 로 표현될 것입니다.

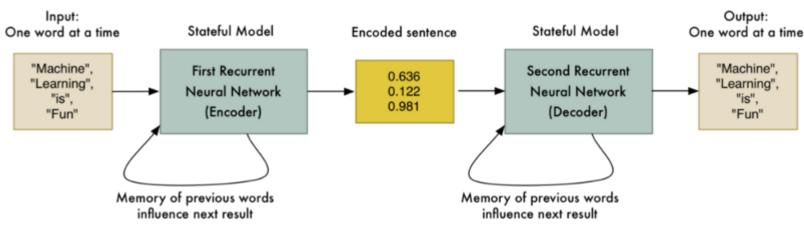
복잡한 그림, 문장을 일련의 고유한 숫자들로 표현 한다!



Sequence to Sequence

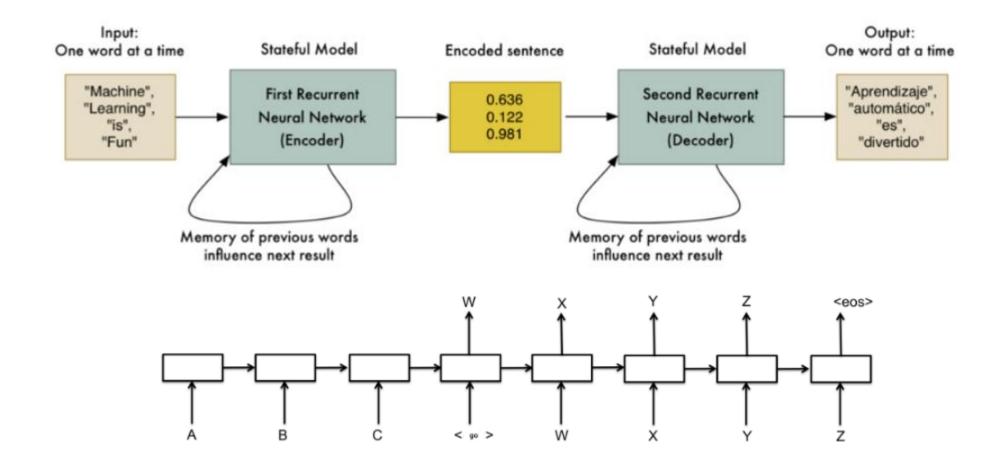
Sequence 2 Sequence : RNN 도 Encoder / Decoder 역할을 할 수 있다.







Sequence to Sequence



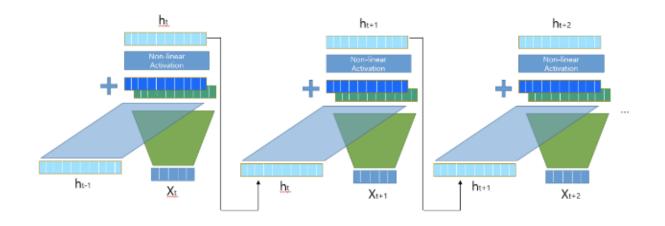


RNN Problem

https://github.com/heartcored98/Standalone-DeepLearning/blob/master/Lec8/Lec8-A.pdf

So many tanh(x)!

Vanishing Gradient Problem



$$h_{t-2} = tanh(W[h_{t-3}, x_{t-2}])$$

$$h_{t-1} = tanh(W[h_{t-2}, x_{t-1}])$$

$$h_t = tanh(W[h_{t-1}, x_t])$$

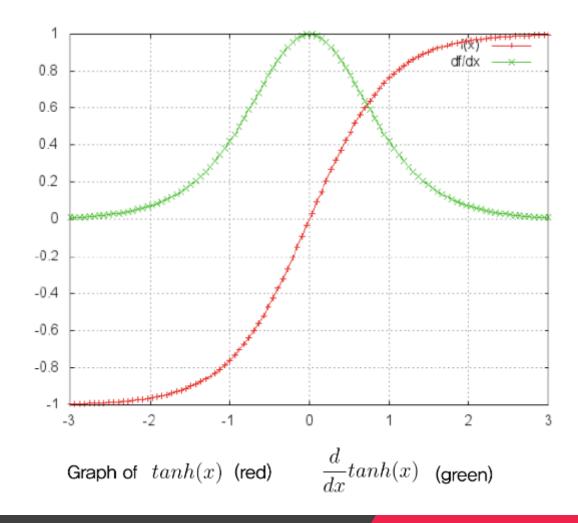
$$h_t = tanh(W[tanh(..tanh(..h_{t-3})), x_t])$$



RNN Problem

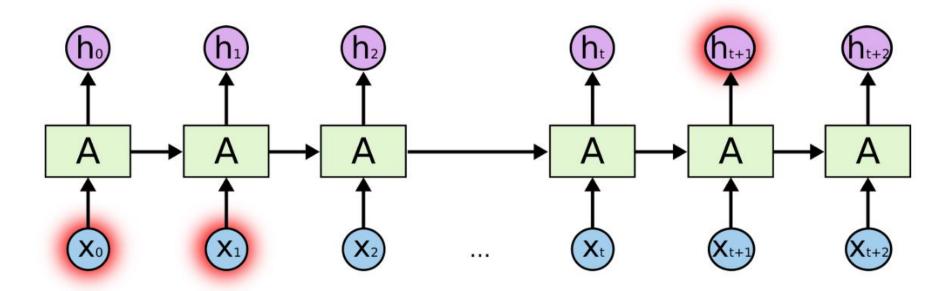
https://github.com/heartcored98/Standalone-DeepLearning/blob/master/Lec8/Lec8-A.pdf

Vanishing Gradient Problem



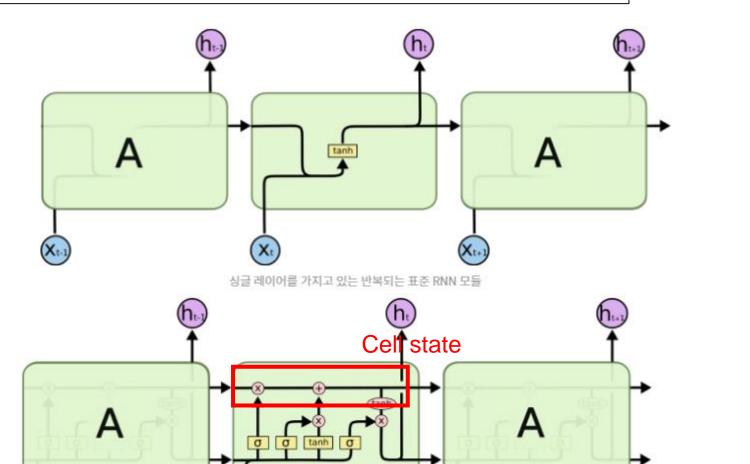


RNN 의 문제점: Gradient Vanishing 문제로 장기간에 걸친시간의존성은 학습시킬 수 없다.



Long Short Term Memory : 장시간에 걸친 시간 의존성, 단 기간에 걸친 시간의존성 모두를 학습시킬 수 있는 기법.





: RNN

: LSTM









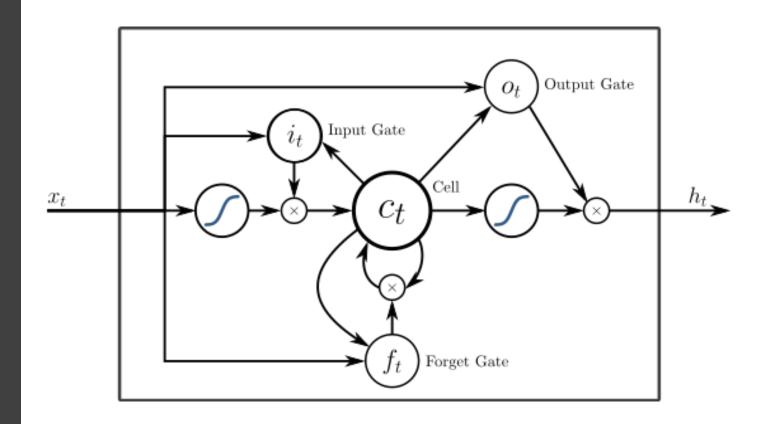




LSTM에 들어있는 4개의 상호작용하는 레이어가 있는 반복되는 모듈

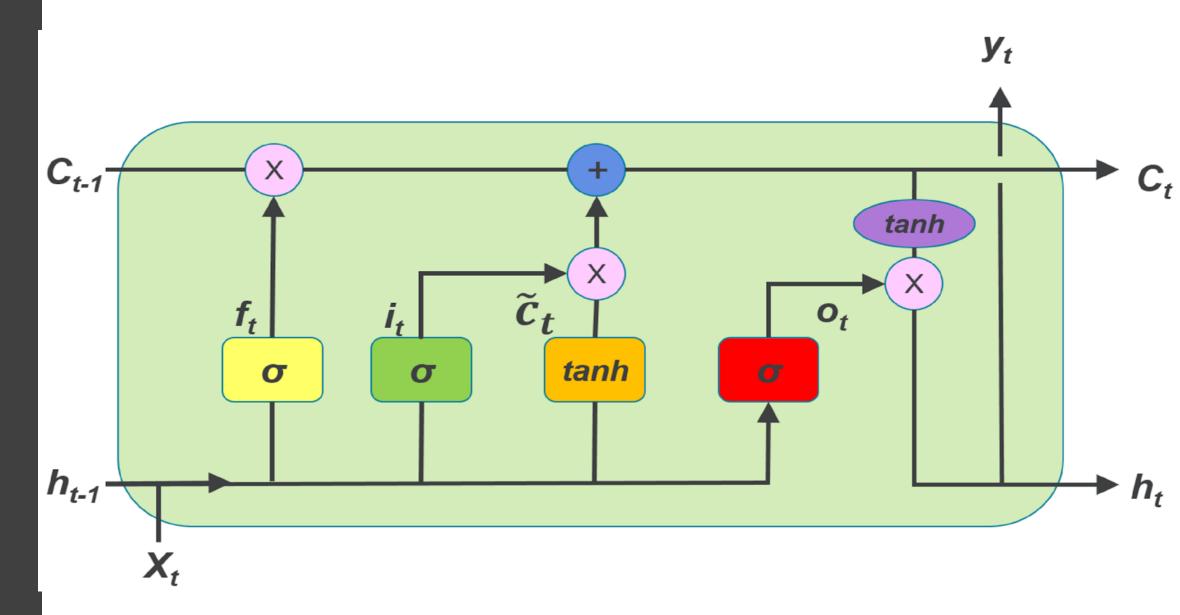
Neural Network Layer Pointwise Operation Vector Transfer

Concatenate



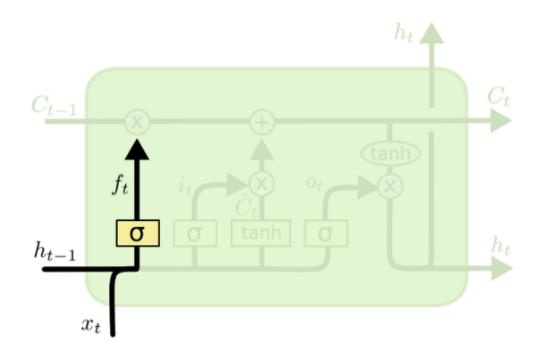
- Input gate
- Forget gate
- Output gate







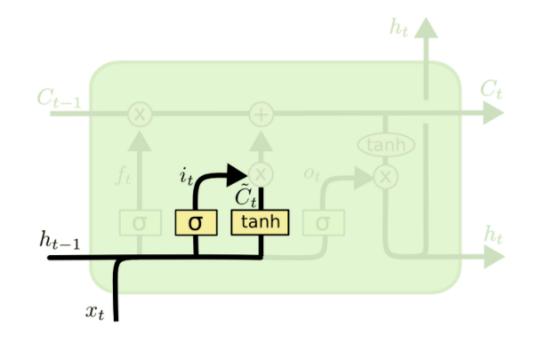
• forget gate : 시간 t 에서의 정보의 중요도에 따른 망각 결정



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



• Input gate: 시간 t 에서의 정보를 받아들이는 양 결정

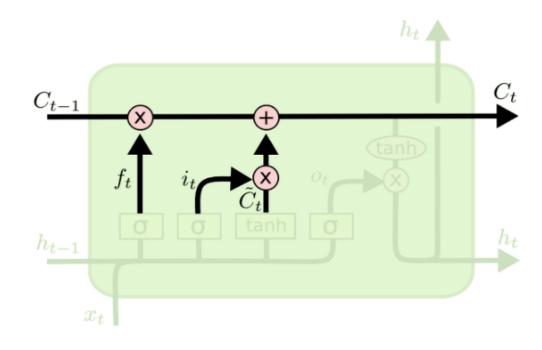


$$i_t = \sigma \left(W_i \cdot [h_{t-1}, x_t] + b_i \right)$$

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



• Cell state : 내부 기억에 관련된 state

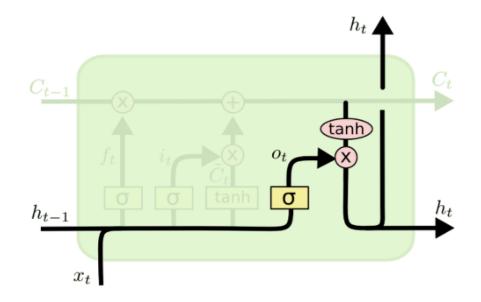


Input 정보를 얼마나 받아들일지 이전 cell state 를 얼마나 망각할 지 결정

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



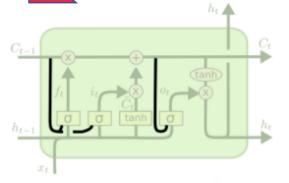
• Output gate : 다음 연속된 모듈에 hidden state 결정



$$o_t = \sigma (W_o [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh (C_t)$$



LSTM and others

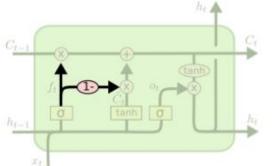


$$f_{t} = \sigma (W_{f} \cdot [C_{t-1}, h_{t-1}, x_{t}] + b_{f})$$

$$i_{t} = \sigma (W_{i} \cdot [C_{t-1}, h_{t-1}, x_{t}] + b_{i})$$

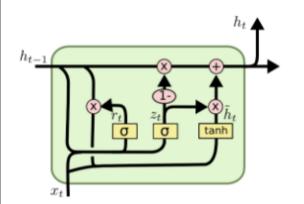
$$o_{t} = \sigma (W_{o} \cdot [C_{t}, h_{t-1}, x_{t}] + b_{o})$$

: **핍홀 변형 –** cell state 의 제어를 보다 용이하게한 구조



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

: merge input / forget gate 변형



$$z_{t} = \sigma (W_{z} \cdot [h_{t-1}, x_{t}])$$

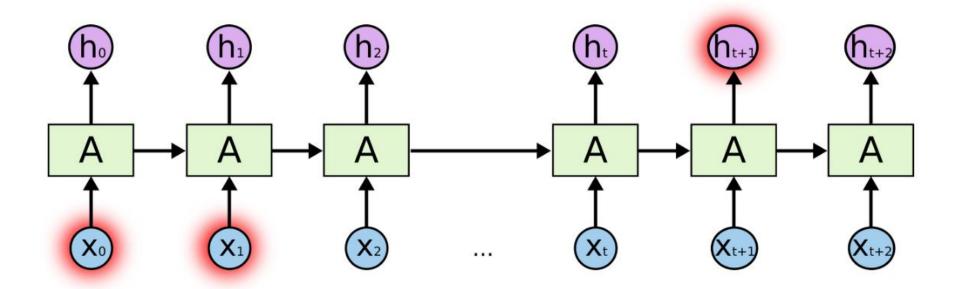
$$r_{t} = \sigma (W_{r} \cdot [h_{t-1}, x_{t}])$$

$$\tilde{h}_{t} = \tanh (W \cdot [r_{t} * h_{t-1}, x_{t}])$$

$$h_{t} = (1 - z_{t}) * h_{t-1} + z_{t} * \tilde{h}_{t}$$

: **GRU 변형** – reset / update gate 만을 사용한 간소화

RNN 의 문제점: Gradient Vanishing 문제로 장기간에 걸친시간의존성은 학습시킬 수 없다.



Long Short Term Memory : 장시간에 걸친 시간 의존성, 단 기간에 걸친 시간의존성 모두를 학습시킬 수 있는 기법.



https://github.com/heartcored98/Standalone-DeepLearning/blob/master/Lec8/Lec8-A.pdf

Long Short Term Memory Network

- 1. Ct-1 에서 불필요한 정보를 지운다. ←
- 2. Ct-1에 새로운 인풋 xt와 ht-1를 보고 중요한 정보를 넣는다. ᢏ
- 3. 위 과정을 통해 Ct를 만든다.
- 4. Ct를 적절히 가공해 해당 t에서의 ht를 만든다.
- 5. Ct와 ht를 다음 스텝 t+1로 전달한다.

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

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

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

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

$$o_t = \sigma \left(W_o \left[h_{t-1}, x_t \right] + b_o \right)$$

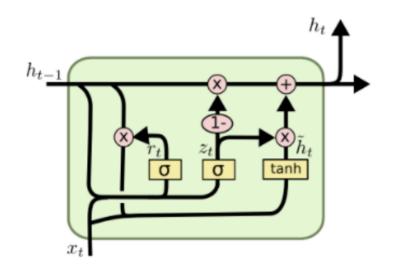
$$h_t = o_t * \tanh(C_t)$$

뭔가 너무 중복되는 느낌..?

→ 더 간단하게 forget gate, input gate, output gate를 해결 할 수는 없을까?



Gated Recurrent Unit (GRU)

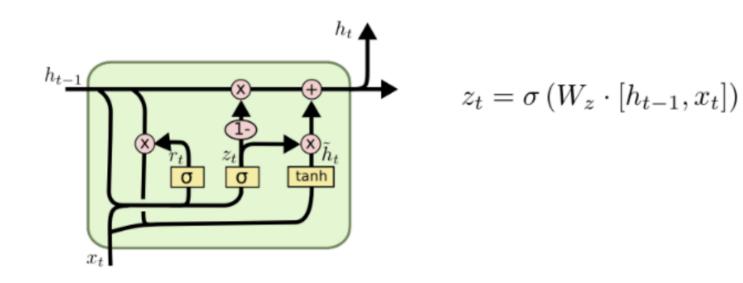


$$r_t = \sigma (W_r \cdot [h_{t-1}, x_t])$$
$$\tilde{h}_t = \tanh (W \cdot [r_t * h_{t-1}, x_t])$$

1. Reset gate를 계산해서 임시 ht를 만든다.

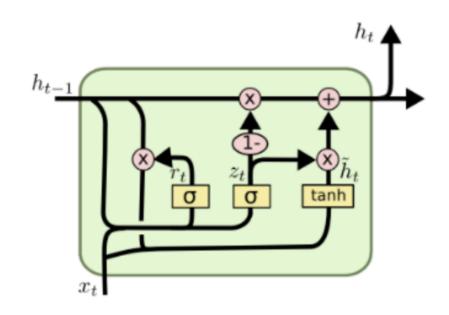


https://github.com/heartcored98/Standalone-DeepLearning/blob/master/Lec8/Lec8-A.pdf



2. Update gate를 통해 ht-1과 ht간의 비중을 결정한다.

https://github.com/heartcored98/Standalone-DeepLearning/blob/master/Lec8/Lec8-A.pdf

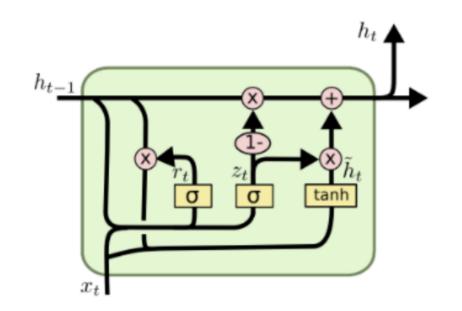


$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

3. zt를 이용해 최종 ht를 계산한다.



https://github.com/heartcored98/Standalone-DeepLearning/blob/master/Lec8/Lec8-A.pdf



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

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

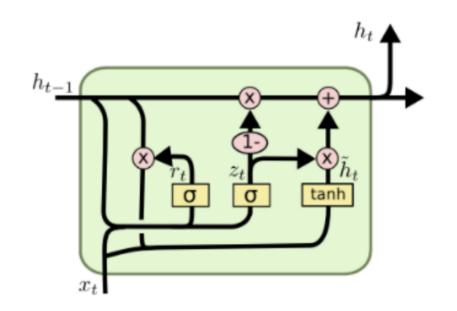
$$z_t = \sigma \left(W_z \cdot [h_{t-1}, x_t] \right)$$

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

- 1. Reset gate를 계산해서 임시 ht를 만든다.
- 2. Update gate를 통해 ht-1과 ht간의 비중을 결정한다.
- 3. zt를 이용해 최종 ht를 계산한다.



https://github.com/heartcored98/Standalone-DeepLearning/blob/master/Lec8/Lec8-A.pdf



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

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

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

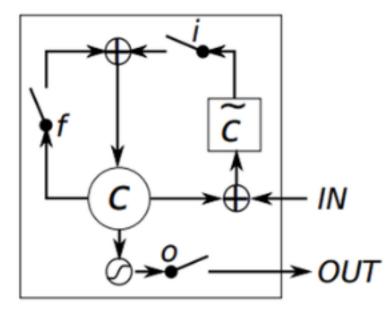
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

- 1. Reset gate를 계산해서 임시 ht를 만든다.
- 2. Update gate를 통해 ht-1과 ht간의 비중을 결정한다.
- 3. zt를 이용해 최종 ht를 계산한다.



https://jay.tech.blog/2016/12/08/lstmlong-short-term-memory-rnn/

GRU



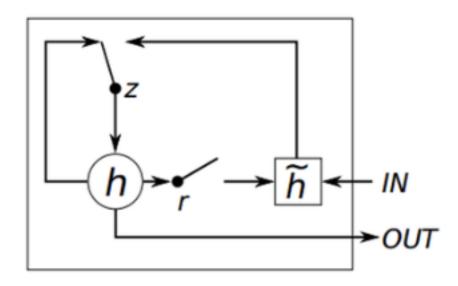
(a) Long Short-Term Memory

$$z = \sigma(x_t U^z + s_{t-1} W^z)$$

$$r = \sigma(x_t U^r + s_{t-1} W^r)$$

$$h = tanh(x_t U^h + (s_{t-1} \circ r) W^h)$$

$$s_t = (1 - z) \circ h + z \circ s_{t-1}$$



(b) Gated Recurrent Unit

$$h_t^j = (1 - z_t^j)h_{t-1}^j + z_t^j \tilde{h}_t^j,$$



Reference

https://towardsdatascience.com/animated-rnn-lstm-and-gru-ef124d06cf45 https://excelsior-cjh.tistory.com/185



• Thank you

