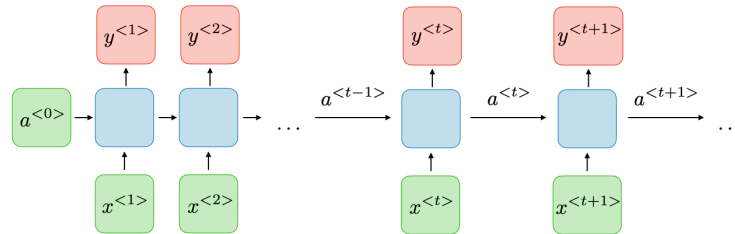


2/10 Recurrent Model (2)

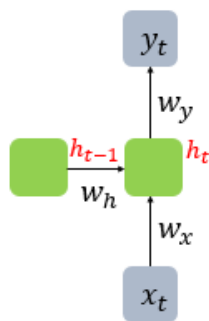
RNN

: 시퀀스 데이터를 다루고, 이전 시점의 출력값을 현재의 입력값으로 사용하는 신경망



RNN 기본 구조

RNN 기본 수식



$$h_t = g_1(W_x x_t + W_h h_{t-1} + b)$$

$$y_t = g_2(W_y h_t + b)$$

• h_t =현재 시점 t에서의 은닉 상태값 (위 그림에서는 a_t)

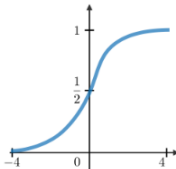
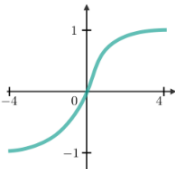
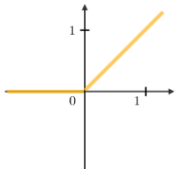
• x_t =입력값, y_t =출력값, b =편향

• W_t, W_h =입력값, t-1시점에서의 은닉 상태값을 위한 가중치

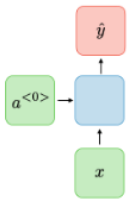
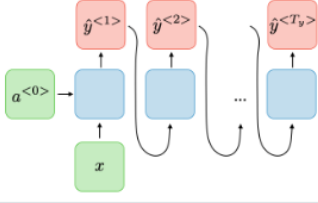
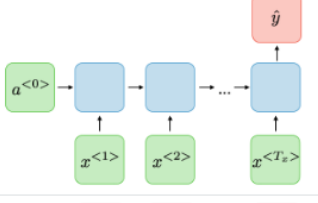
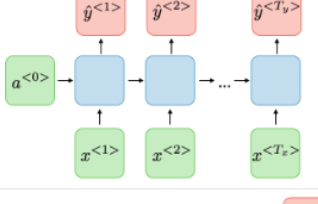
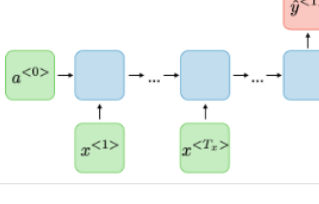
→ g_1, g_2 는 비선형 활성화 함수.

→ h_t 를 계산하기 위한 함수 g_1 는 비선형 함수이고(RNN에서는 tanh, MLP/CNN에서는 ReLU), y_t 를 계산하기 위한 g_2 는 푸는 문제에 따라 다름

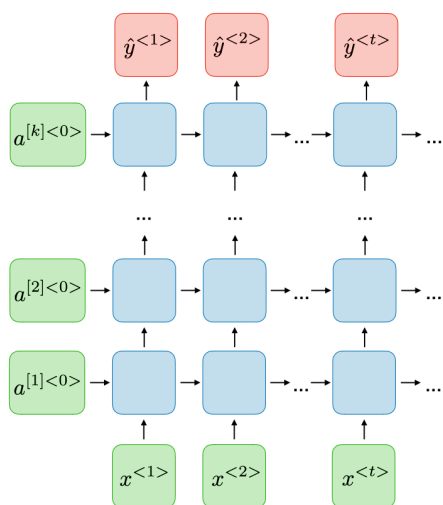
Activation Functions

Sigmoid	Tanh	RELU
$g(z) = \frac{1}{1 + e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$
		

- 회귀: 선형함수
- 이진분류: sigmoid (로지스틱 회귀)
- 다중 클래스 분류: softmax
- 다중레이블 분류: sigmoid

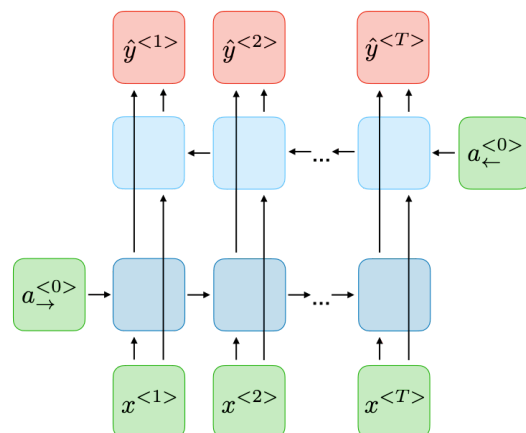
Type of RNN	Illustration	Example
One-to-one $T_x = T_y = 1$		Traditional neural network
One-to-many $T_x = 1, T_y > 1$		Music generation
Many-to-one $T_x > 1, T_y = 1$		Sentiment classification
Many-to-many $T_x = T_y$		Name entity recognition
Many-to-many $T_x \neq T_y$		Machine translation

Deep RNN



- 은닉층이 2개 이상인 RNN

Bi-directional RNN



- 시점 t 에서의 출력값을 예측할 때 이전 시점의 입력 뿐만 아니라 이후 시점의 입력 또한 예측에 기여할 수 있다는 것에 기반함.

- forward states와 backward states 총 두 개의 메모리 셀을 사용해 출력값 예측

RNN의 장점

- 어떠한 길이의 시퀀스 데이터라도 간단한 모델로 처리할 수 있다

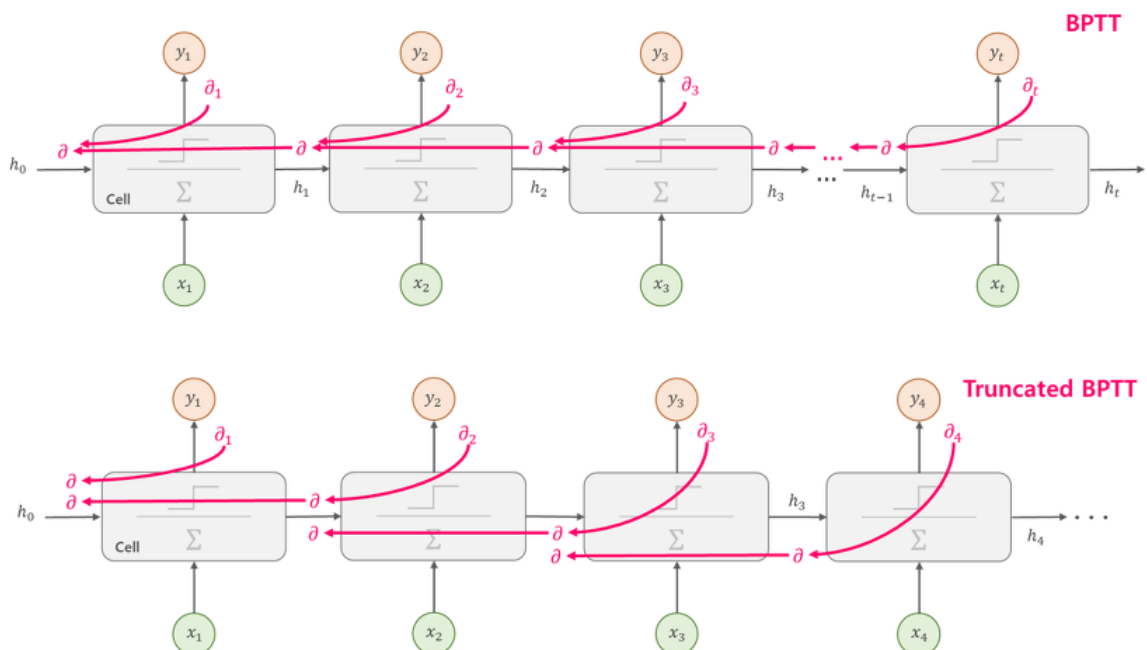
단점

- 처리 속도가 느림
- 오래된 정보를 잘 수용하지 못함
- 역전파 과정에서 vanishing/exploding gradient 문제가 나타남

Backpropagation Through Time (BPTT)

: 시퀀스 데이터를 다루고 은닉층이 존재하는 RNN에서 일어나는 역전파 과정을 일컫는 말. 일반적인 역전파와 같이 먼저 순전파로 각 타임 스텝별 시퀀스를 출력하고, 그 시퀀스와 함수를 사용해 각 loss를 구한 후 역방향으로 전파한다.

- 시퀀스의 처음부터 끝까지 모두 에러를 역전파하는 계산량을 줄이기 위해 일정 시간 이전까지만 보는 Truncated BPTT를 많이 사용한다



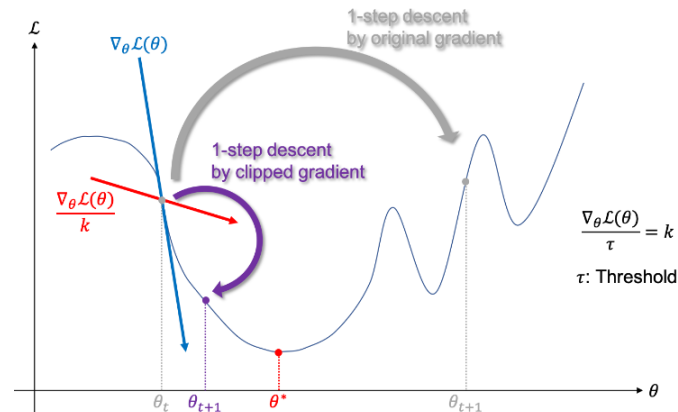
Vanishing/Exploding Gradient

⇒ 레이어의 개수에 따라 gradient가 기하급수적으로 증가/감소하는 현상. 역전파 과정에서 tanh 함수를 미분함으로써 초기 부분의 입력층으로 갈수록 기울기가 점차 작아지거나 커지게 되어 가중치는 최적의 값에 도달하지 못하게 된다.

해결 방법

1) Gradient Clipping

- 역전파시 일정 임계값을 넘어서지 못하게 기울기 값을 자르는 방법



클리핑 (파란색 선)으로 인해 그래디언트가 발산하는 대신 전역 최솟값으로 향하는 모습

2) 수렴하지 않는 활성화 함수 사용

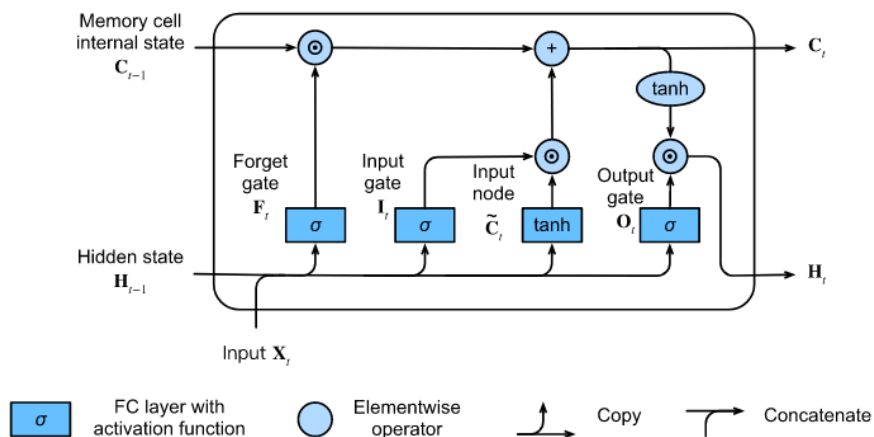
- 은닉층의 활성화 함수로 시그모이드 대신 LeakyReLU 등 변형된 ReLU 함수 활용

3)가중치 초기화

- 세이버 초기화: 균등 분포 또는 정규 분포로 초기화함. 이전 층의 뉴런 개수와 다음 층의 뉴런 개수를 사용하여 가중치를 초기화함. sigmoid 또는 tanh 함수를 사용할 때 효율적
- HE 초기화: 마찬가지로 균등 분포 또는 정규분포로 초기화하지만 다음 층의 뉴런의 수를 반영하지 않음. ReLU 계열 함수를 사용할 때 효율적

LSTM (Long Short-Term Memory)

: 오래된 정보를 더 많이 수용하고자 하는, 긴 의존 기간을 필요로 하는 학습을 수행할 수 있는 RNN모델. 세가지의 gate가 있음 (RNN에 추가될 수 있는 gate들은 vanishing gradient 문제를 해소하기도 한다)



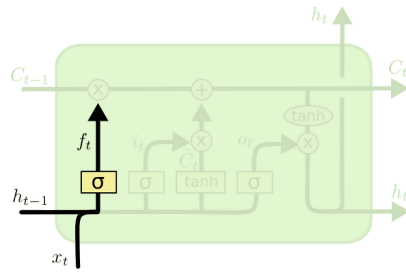
Cell State

: 가장 위에 수평으로 그려진 선. Gate에 의해 뭔가를 더하거나 없애도록 제어된다.

Gate

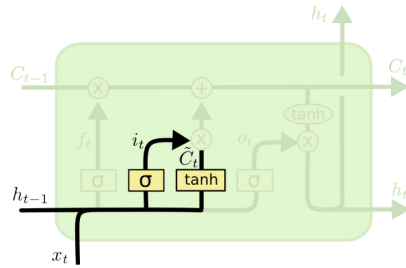
: 정보가 전달될 수 있는 추가 방법이고, sigmoid 레이어와 pointwise 곱셈으로 이루어져 있음. 정보를 전달하는 척도에 따라 sigmoid 레이어가 0에서 1사이의 값을 내보낸다.

1) Forget Gate: cell state로부터 어떤 정보를 버릴 것인지 정한다. h_{t-1} 과 x_t 를 받은 후 보존 정도에 따라 0과 1 사이의 값을 C_{t-1} 에 보낸다.



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

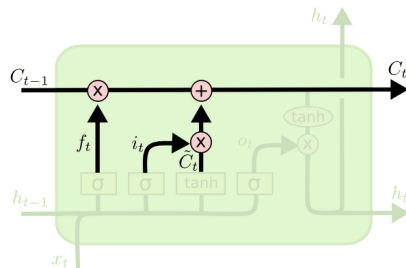
2) Input Gate: 새로 들어오는 정보 중 어떤 것을 cell state에 저장할 것인지 정한다. Sigmoid 레이어가 어떤 값을 업데이트할지 정하고, tanh 레이어는 후보 값들로 벡터 \tilde{C}_t 를 만든다.



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

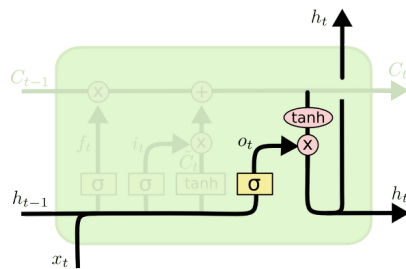
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

C_t 업데이트: Forget gate가 정한 f_t 를 곱함으로써 잊어버리기로 했던 정보를 잊고, input gate에서 정한 i_t 와 \tilde{C}_t 를 곱해 더하면 어떤 값들이 얼마나 업데이트 될 지에 대한 것이 반영된다.



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

3) Output Gate: 우선 sigmoid 레이어를 통해 cell state의 어느 부분을 아웃풋으로 내보낼지 정한다. (o_t) Cell state를 tanh 레이어에 넣어 -1과 1 사이의 값을 받고 o_t 와 곱해주면 정해진 아웃풋만 내보낼 수 있게 된다.

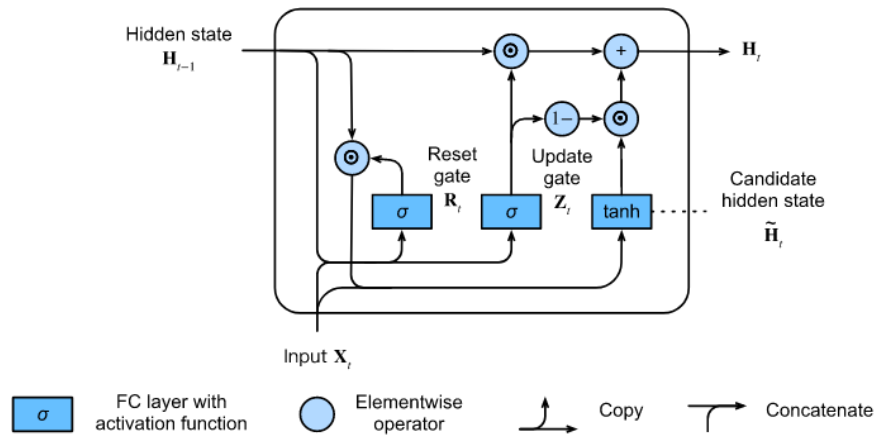


$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$$

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

GRU (Gated Recurrent Unit)

: 두 가지 게이트가 존재하는 RNN. LSTM보다 cell을 간소화하고 매개 변수를 적게 쓰는 버전이라고 볼 수 있다.

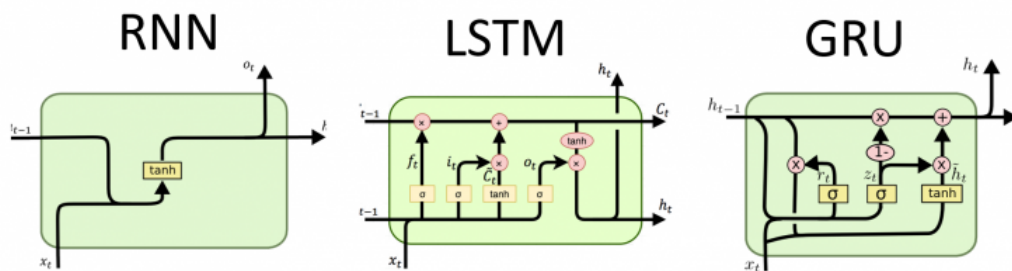


1) **Reset Gate**: 이전 state에서 온 정보를 얼마나 기억할지 정한다.

2) **Update Gate**: 현재 state가 이전 state의 정보를 얼마나 그대로 출력할지 정한다. LSTM에서의 forget gate, input gate를 제어하는 셈

→ GRU는 LSTM과 비슷한 이유로 만들어진 만큼 성능도 비슷하다고 한다. 다만 학습 속도는 GRU가 더 빠르고, 데이터의 양이 많지 않고 적은 경우 GRU가 조금 더 낫다고 알려져 있다.

RNN, LSTM, GRU의 모델 아키텍처 비교



참고

<https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks>

<https://towardsdatascience.com/how-to-choose-the-right-activation-function-for-neural-networks-3941ff0e6f9c>

<https://wikidocs.net/22886>

<https://excelsior-cjh.tistory.com/184>

<https://casa-de-feel.tistory.com/36>

<https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

https://d2l.ai/chapter_recurrent-modern/

<https://wooono.tistory.com/242>