Recurrent Neural Network(RNN)

Sequential Data Modeling

Sequential Data

- 순차 데이터란 '**데이터 집합 내의 객체들이 어떤 순서를 가진 데이터**'로 그 순서가 변경될 경우 고유의 특성을 잃어버리는 특징이 있다.
- Most of data are sequential
- Speech,Text,Image,...

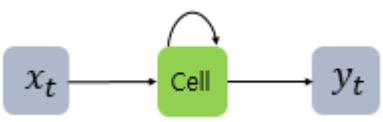
Deep Learnings for Sequential Data

- Convolution Neural Networks(CNN)
  - Try to find local features from a sequence
- Recurrent Neural Networks : LSTM,GRU
  - Try to capture the feature of the **past**

Recurrent Neural Networks(순환신경망)

앞서 배운 신경망들은 전부 hidden layer에서 activation func를 지닌 값은 오직 출력층 방향으로만 향했다. >> Feed Forward Neural Network

RNN은 은닉층의 노드에서 활성화 함수를 통해 나온 결과값을 출력층 방향으로도 보내면서 ,다시 은닉층 노드의 다음 계산의 입력으로 보내는 특징이 있다.

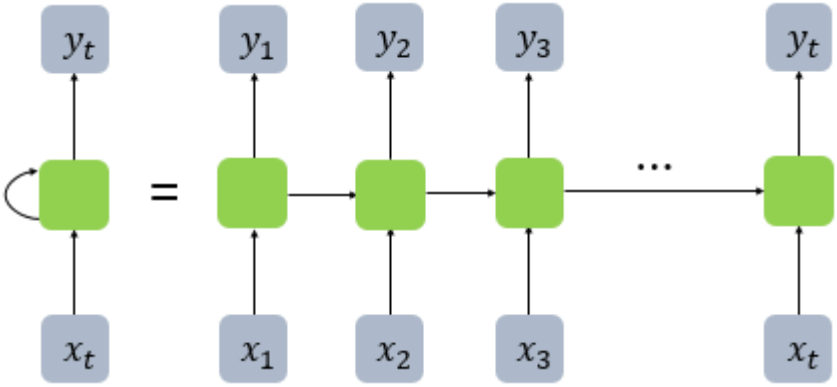


이를 그림으로 표현하면 위와 같습니다. x는 입력층의 입력 벡터, y는 출력층의 출력 벡터입니다. 실제로는 편향 b도 입력으로 존재할 수 있지만 앞으로의 그림에서는 생략합니다.

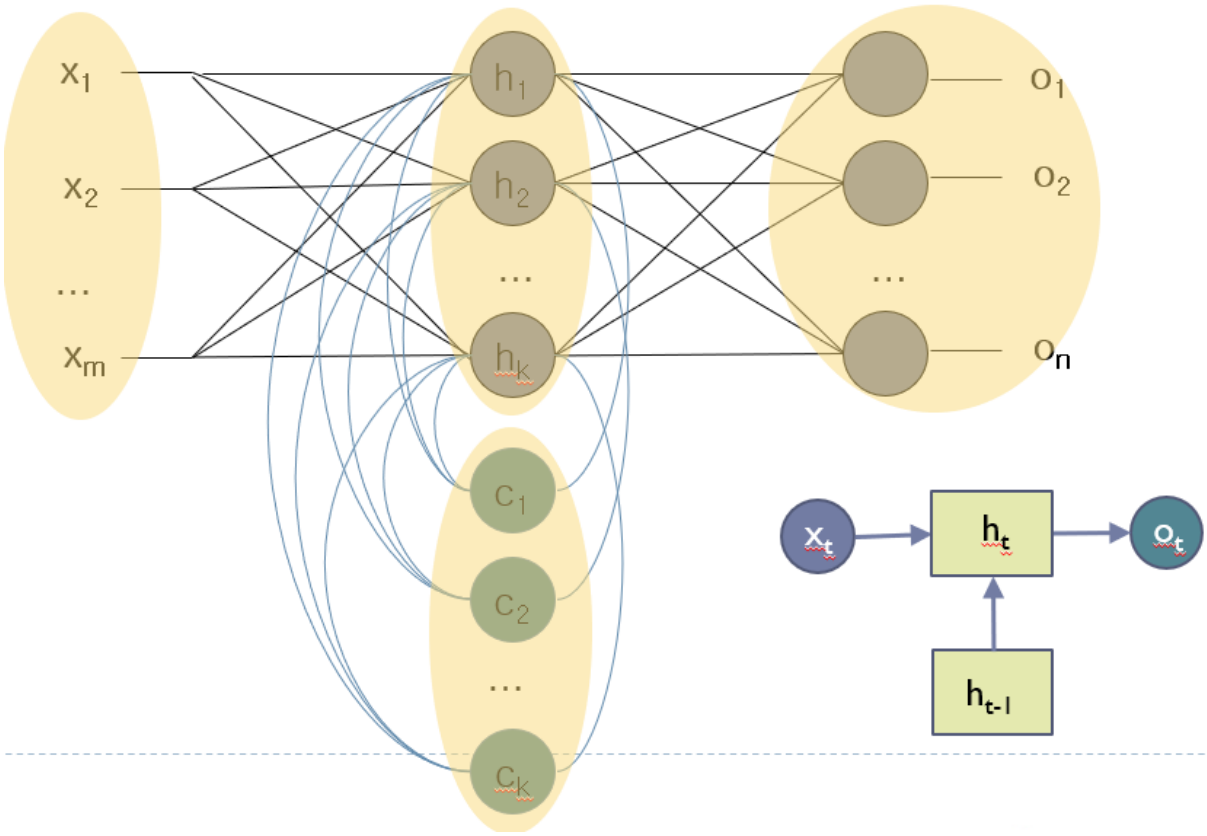
RNN에서 은닉층에서 활성화 함수를 통해 결과를 내보내는 역할을 하는 노드를 셀(cell)이라고 합니다. 이 셀은 이전의 값을 기억하려고 하는 일종의 메모리 역할을 수행하므로 이를 메모리 셀 또는 RNN 셀이라고 표현합니다.

은닉층의 메모리 셀은 각각의 시점(time step)에서 바로 이전 시점에서의 은닉층의 메모리 셀에서 나온 값을 자신의 입력으로 사용하는 재귀적 활동을 하고 있습니다. 앞으로는 현재 시점을 변수 t로 표현하겠습니다. 이는 현재 시점 t에서의 메모리 셀이 갖고있는 값은 과거의 메모리 셀들의 값에 영향을 받은 것임을 의미합니다. 그렇다면 메모리 셀이 갖고 있는 이 값은 뭐라고 부를까요?

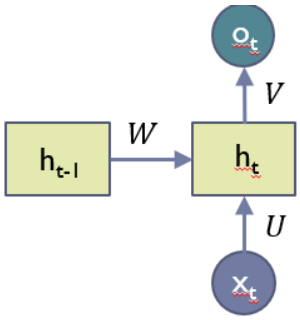
메모리 셀이 출력층 방향으로 또는 다음 시점 t+1의 자신에게 보내는 값을 은닉 상태(hidden state)라고 합니다. 다시 말해 t 시점의 메모리 셀은 t-1 시점의 메모리 셀이 보낸 은닉 상태값을 t 시점의 은닉 상태 계산을 위한 입력값으로 사용합니다.



RNN을 표현할 때는 일반적으로 위의 그림에서 좌측과 같이 화살표로 사이클을 그려서 재귀 형태로 표현하기도 하지만, 우측과 같이 사이클을 그리는 화살표 대신 여러 시점으로 펼쳐서 표현하기도 합니다. 두 그림은 동일한 그림으로 단지 사이클을 그리는 화살표를 사용하여 표현하였느냐, 시점의 흐름에 따라서 표현하였느냐의 차이일 뿐 둘 다 동일한 RNN을 표현하고 있습니다.



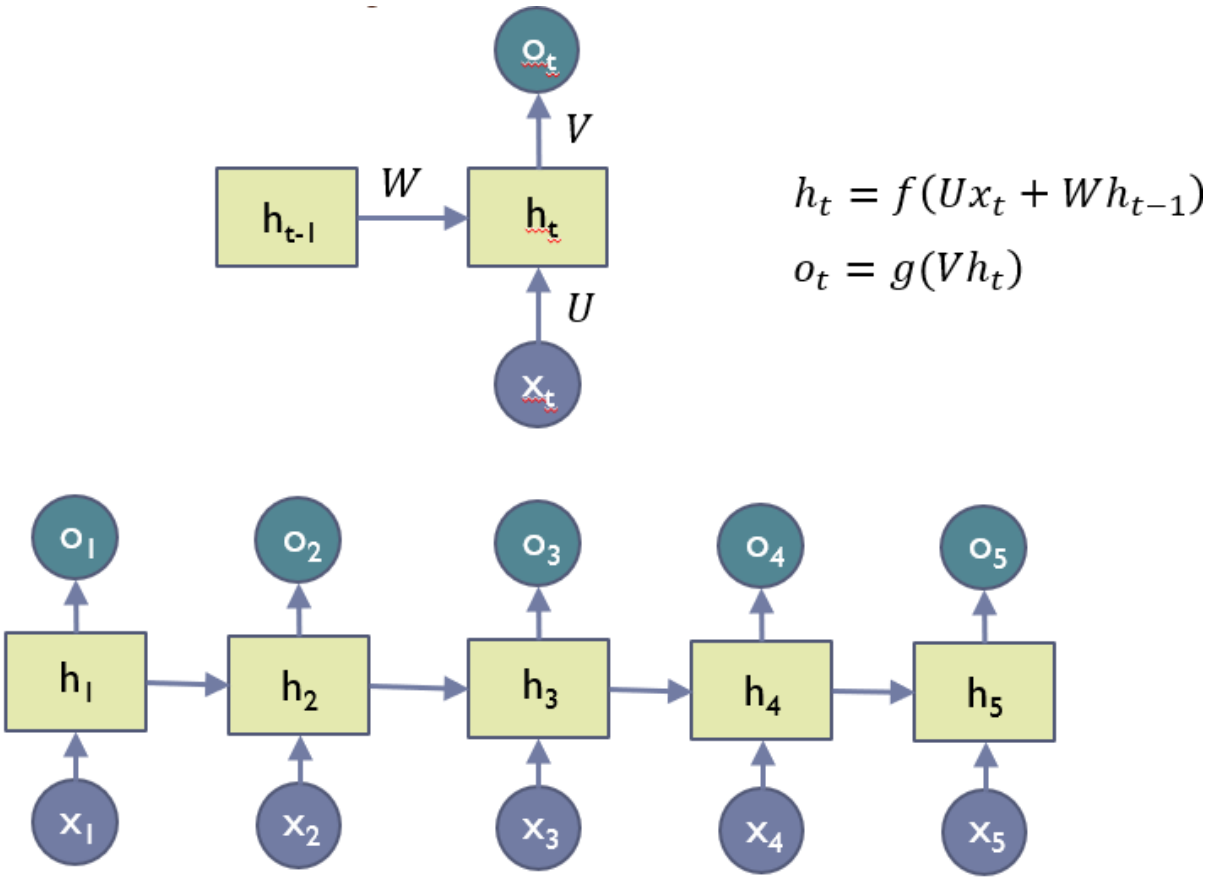
u



$$h_t = f(Ux_t + Wh_{t-1})$$
$$o_t = g(Vh_t)$$

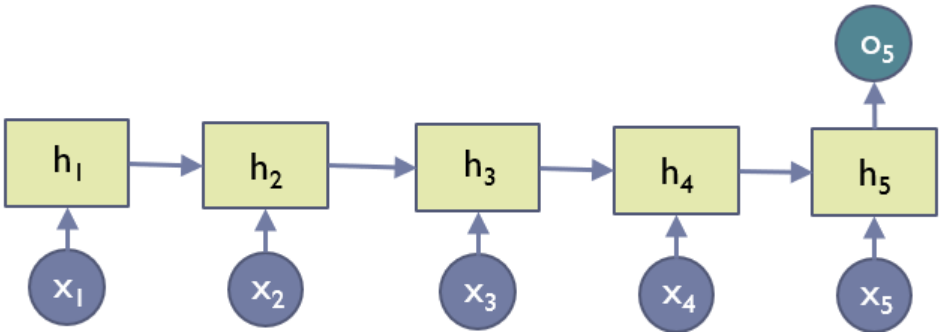
- x_t: input at time t
- h_t: hidden state at time t
- f: is an activation function
- U, V, W: network parameters
  - RNN shares the same parameters across all time steps
- g: activation function for the output layer

U,V,W는 모든 시점에서 값을 동일하게 공유한다.



Long Term Dependency

- $x_1 \sim x_{t-1}$ are encoded into h_{t-1}
- h_{t-1} has the information on the past
- It is a context to process x_t



Long Term Dependency of Standard RNN

- However, it may exponentially decade or grow
- Usually it is limited to 10 dteps

Long Short-Term Memory(LSTM)

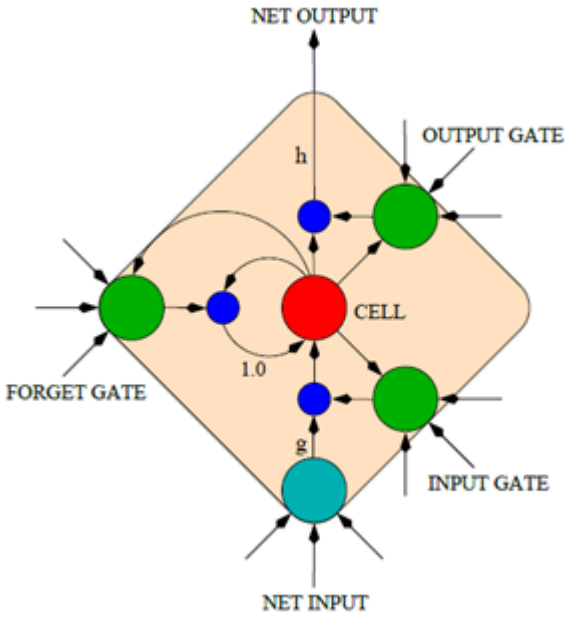
Capable of learning long-term dependencies

- LSTM networks introduce a new structure called a memory cell
  - An LSTM can learn to bridge time intervals in excess of 1000 steps
- Gate units that learn to open and close access to the past
  - Input gate
  - Forget gate
  - Output gate
  - Neuron with a self-recurrent

RNN은 관련 정보와 그 정보를 사용하는 지점 사이 거리가 멀 경우 역전파시 gradient가 점차 줄어 학습능력이 크게 저하되는 것으로 알려져 있다. 이 문제를 극복하기 위해서 고안된 것이 바로 LSTM이다. LSTM은 RNN의 히든 state에 **cell-state를 추가한 구조**이다.

cell state는 일종의 컨베이어 벨트 역할을 한다. 덕분에 state가 꽤 오래 경과하더라도 gradient가 비교적 전파가 잘 되게 된다.

1. **"+" Node : 상류에서 전해지는 기울기를 그대로 흘릴 뿐, 따라서 기울기 변화(감소)는 일어나지 않음**
2. **" X(element wise)" Node : 여기서 사용한 Node는 MatMul(행렬곱)이 아닌, "원소별 곱"을 계산**



$$i = \sigma\left(x_t U^i + s_{t-1} W^i\right)$$
$$f = \sigma\left(x_t U^f + s_{t-1} W^f\right)$$
$$o = \sigma\left(x_t U^o + s_{t-1} W^o\right)$$
$$g = \tanh\left(x_t U^g + s_{t-1} W^g\right)$$
$$c_t = c_{t-1} \circ f + g \circ i$$
$$s_t = \tanh(c_t) \circ o$$
$$y = softmax(V s_t)$$

forget gate f는 '과거 정보를 잊기'를 위한 게이트이다. h_t-1과 x_t를 받아 시그모이드를 취해준 값이 바로 forget gate가 내보내는 값이 된다.

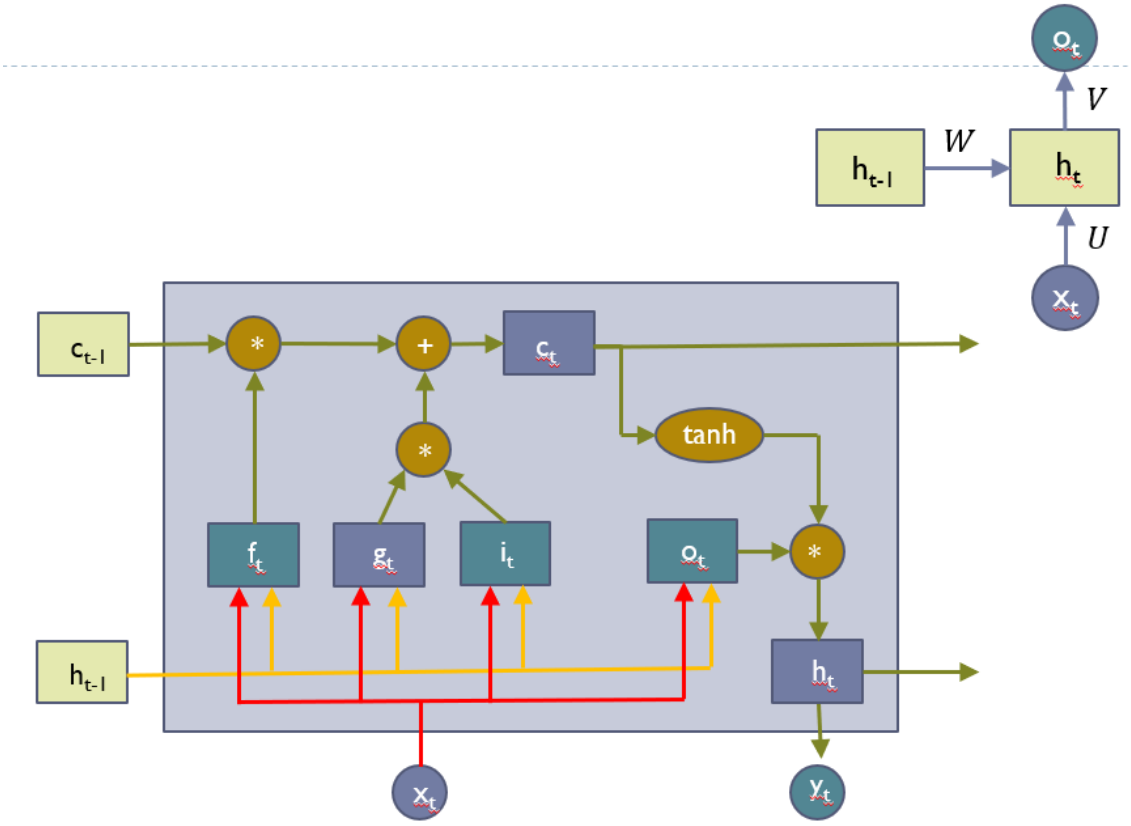
시그모이드 함수의 출력 범위는 0에서 1사이이기 때문에 그 값이 0이라면 이전 상태의 정보는 잊고, 1이라면 이전 상태의 정보를 온전히 기억하게 된다.

input gate it⊙gt는 '현재 정보를 기억하기' 위한 게이트이다. ht−1과 xt를 받아 시그모이드를 취하고, 또 같은 입력으로 하이퍼볼릭탄젠트를 취해준 다음 Hadamard product

연산을 한 값이 바로 input gate가 내보내는 값이 된다. 개인적으로 it의 범위는 0~1, gt의 범위는 -1~1이기 때문에 각각 강도와 방향을 나타낸다고 이해했다.

>>앞으로 들어오는 새로운 정보 중 어떤 것을 cell state에 저장할 것인지를 정한다.

gate들을 수도꼭지라 생각하면 편하다. 데이터량을 조절하니까!!



$$i = \sigma\left(x_t U^i + s_{t-1} W^i\right)$$
$$f = \sigma\left(x_t U^f + s_{t-1} W^f\right)$$
$$o = \sigma\left(x_t U^o + s_{t-1} W^o\right)$$
$$g = \tanh\left(x_t U^g + s_{t-1} W^g\right)$$
$$c_t = c_{t-1} \circ f + g \circ i$$
$$s_t = \tanh(c_t) \circ o$$
$$y = softmax(V s_t)$$

- **Equations**
  - i: input gate to accept the new
  - f: forget gate to forget the past
  - o: output gate, how much of the information will be passed to expose to the next time step.
  - g: self-recurrent which is equal to standard RNN
  - c_t: internal memory
  - s_t: hidden state
  - y: final output

Gated Recurrent Unit (GRU)

>>한국인이 개발...ㄷㄷ

GRU는 게이트 메커니즘이 적용된 RNN프레임워크의 일종으로 LSTM에 영감을 받았고, 더 간략한 구조를 가지고 있다.

Reset Gate

Reset Gate는 과거의 정보를 적당히 리셋시키는게 목적으로 sigmoid 함수를 출력으로 이용해 (0,1)값을 은닉층에 곱해준다.

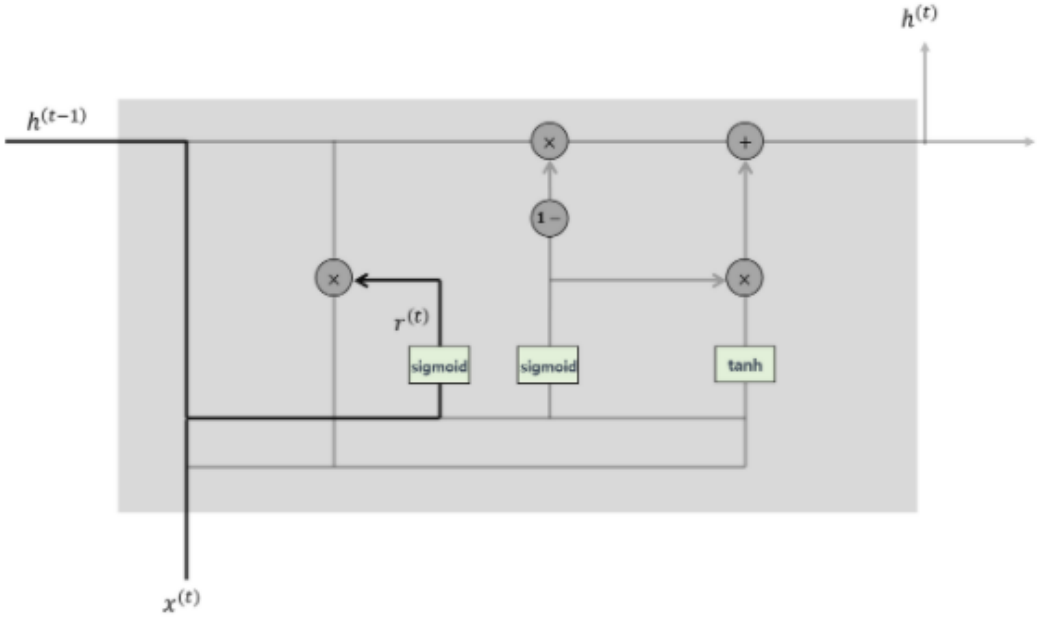


Figure 2: GRU, Reset Gate

$$r^{(t)} = \sigma\left(W_r h^{(t-1)} + U_r x^{(t)}\right)$$

Update Gate

Update Gate는 LSTM의 forget gate와 input gate를 합쳐놓은 느낌으로 과거와 현재의 정보의 최신화 비율을 결정한다. Update gate에서 sigmoid로 **출력된 결과 (u)는 현 시점의 정보의 양을 결정하고, 1 에서 뺀 값 (1-u)는 직전 시점의 은닉층의 정보에 곱해주며,** 각각이 LSTM의 input gate와 forget gate와 유사하다.

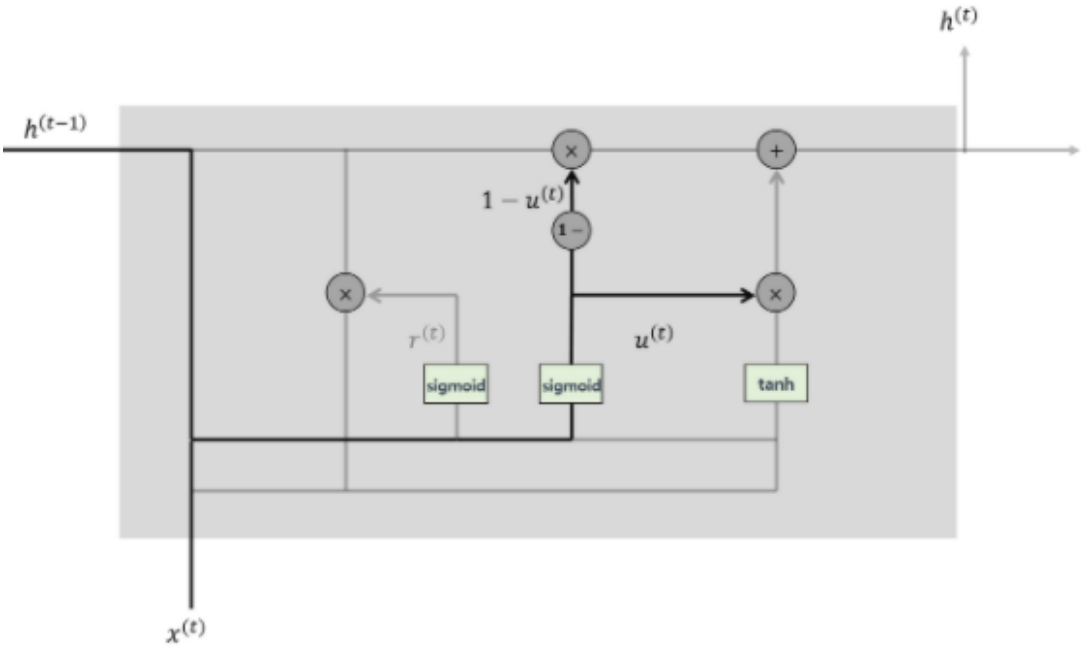


Figure 3: GRU, Update Gate

이를 수식으로 나타내면 다음과 같습니다.

$$u^{(t)} = \sigma\left(W_u h^{(t-1)} + U_u x^{(t)}\right)$$

Candidate

현 시점의 정보 후보군을 계산하는 단계이다. 핵심은 과거 은닉층의 정보를 그대로 이용하지 않고 리셋 게이트의 결과를 곱하여 이용해준다.

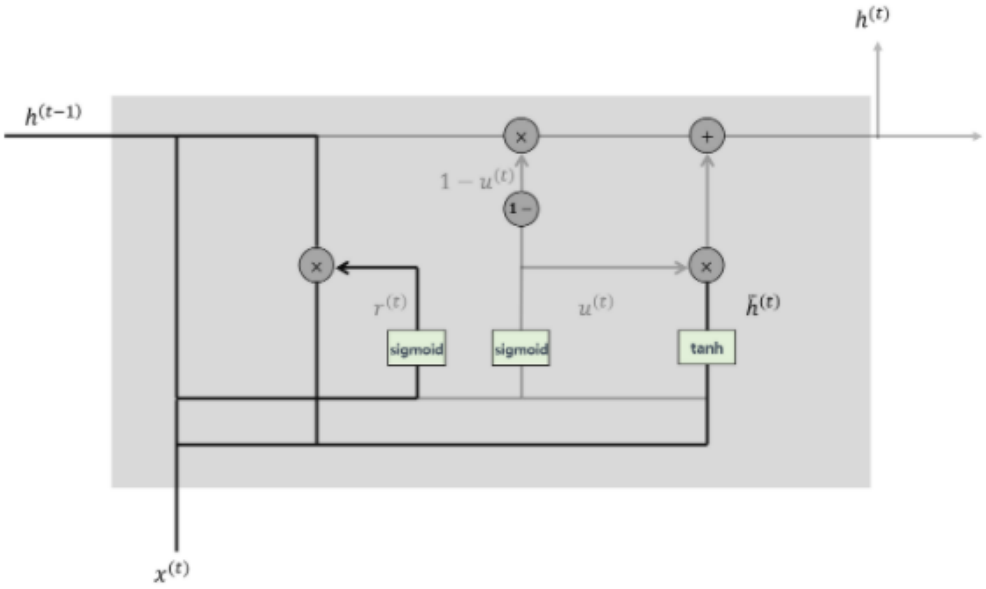


Figure 4: GRU, Candidate

수식으로 나타내면 다음과 같습니다.

$$\tilde{h}^{(t)} = \tau\left(Wh^{(t-1)} * r^{(t)} + Ux^{(t)}\right)$$

여기에서 τ는 tangent hyperbolic이고, $*$은 pointwise operation입니다.

은닉층 계산

마지막으로 update gate결과와 candidate 결과를 결합하여 현 시점의 은닉층을 계산하는 단계이다. 앞에 말했다 시피 **sigmoid 함수의 결과는 현시점 결과의 정보의 양을 결정하고, 1-sigmoid 함수의 결과는 과거 시점의 정보 양을 결정한다.**

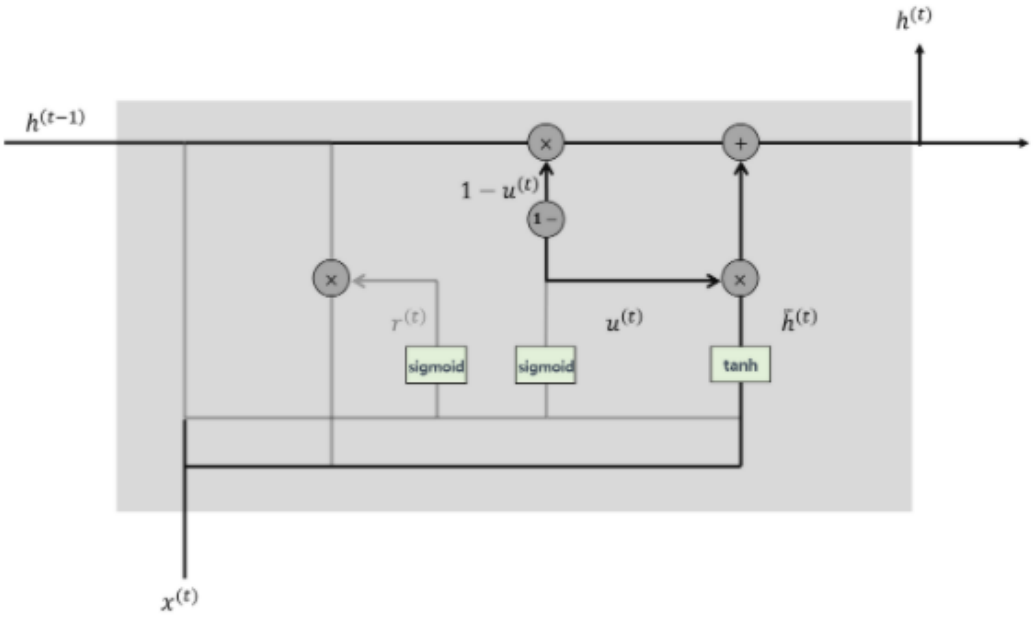


Figure 5: GRU, hidden layer

수식으로 정리하면 아래와 같습니다.

$$h^{(t)} = (1 - u^{(t)}) * h^{(t-1)} + u^{(t)} * \tilde{h}^{(t)}$$

GRU 정리

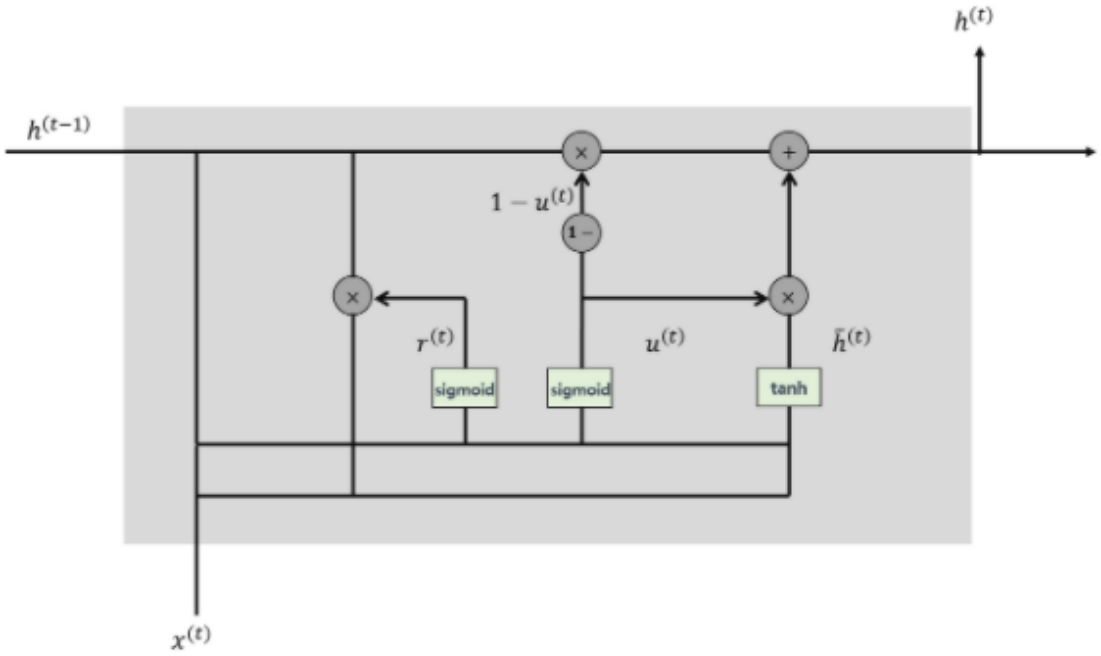


Figure 6: GRU

LSTM과 구조상 큰 차이도 없고, 분석결과도 큰 차이가 없는 것으로 알려져있다. 분석결과가 큰 차이가 없다는 것은 같은 목적에서 성능상의 큰 차이가 없다는 의미가 아니라, 주제별로 LSTM이 좋기도, GRU가 좋기도 하다는 의미이다. 다만 **GRU가 학습할 가중치가 적다는 것은 확실한 이점이다.**

$$
\begin{aligned}
r^{(t)} &= \sigma\left(W_r h^{(t-1)} + U_r x^{(t)}\right) \\
u^{(t)} &= \sigma\left(W_u h^{(t-1)} + U_u x^{(t)}\right) \\
\tilde{h}^{(t)} &= \tau\left(W h^{(t-1)} * r^{(t)} + U x^{(t)}\right) \\
h^{(t)} &= \left(1 - u^{(t)}\right) * h^{(t-1)} + u^{(t)} * \tilde{h}^{(t)}
\end{aligned}
$$