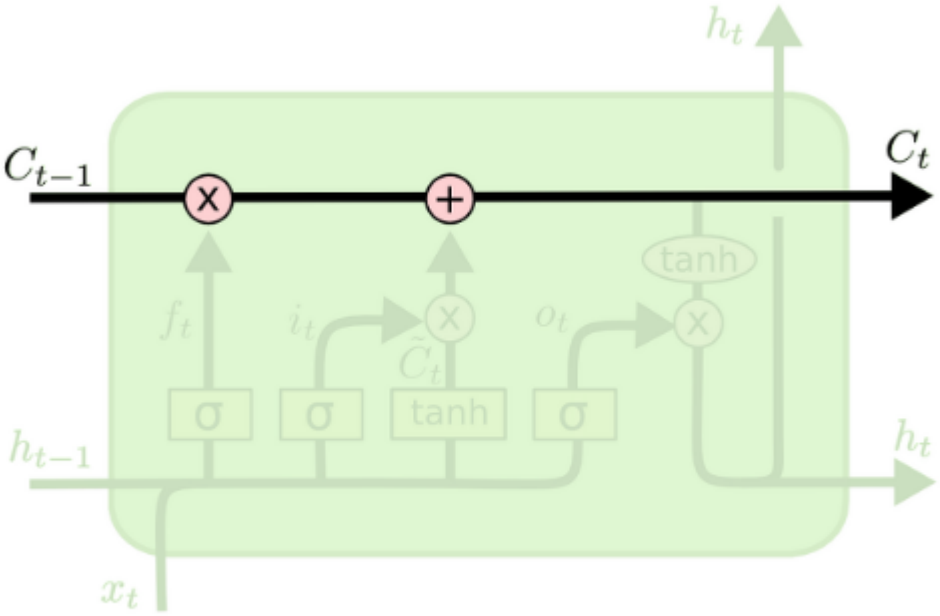
주의할 점 : 이 block은 하나의 노드를 겨냥한게 아니라, 하나의 layer로 봐야된다.
모두 2차원 행렬로,,(batch 제외)
node 하나라고 가정했을 때, 모두 하나의 scalar 값이다

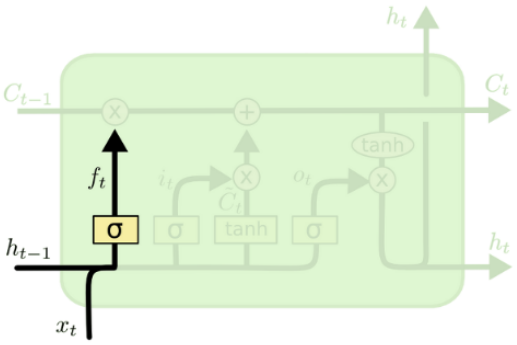
Cell state



LSTM의 cell state

Cell state(t)는 T시점까지의 정보를 담고 있는 역할이다. (forget gate의 값에 따라서 특정 시점부터 담을 수도 있다)

Forget Gate



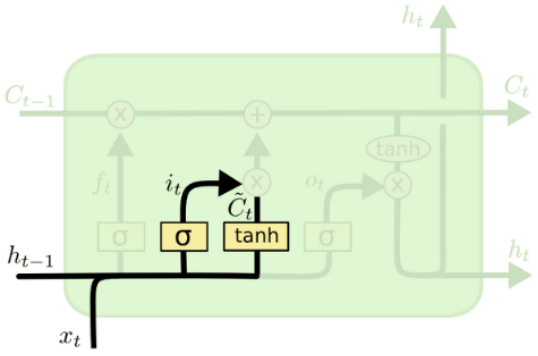
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

LSTM의 forget gate layer

과거 정보와 현재 정보를 weighted sum. sigmoid function을 거치면서, C(t-1)의 값을 얼만큼 사용할 것인지 정한다.

- C(t-1)은 C(t)의 구성요소이다.
- 왜 h(t-1)의 정보까지 포함해서 C(t-1)의 중요도를 결정해야 하는가?
  C(t-1)은 {h(t-2),x(t-1)}의 정보를 가지고 있다. C(t-1)가 C(t)의 구성에 얼만큼의 영향을 줄 수 있는지를 결정하기 위해선, 현재 정보만 가지고는 관계도를 파악하기 어렵다.
  ⇒ 현재까지의(과거 포함) 정보를 가지고 sigmoid를 거쳐서 판단해야 한다. 따라서, x(t)와 h(t-1)을 weighted sum하고 sigmoid를 거침으로써, 과거와 현재의 관계를 판단하고, C(t-1)이 중요도를 결정한다.
- Forget gate의 값이 0이라면?
  필요없는 과거 정보는 버린다. ⇒ input gate에서의 h(t-1)의 weight를 0으로 주면, C(t)를 현재 정보로만 구성시킬 수 있다.
- Forget gate의 값이 1이라면?
  C(t)를 구성하는데 이전 정보가 그대로 들어간다.
  input gate에서 x(t)의 weight값이 0이라면, 현재 노드를 과거 정보로만 구성할 수도 있다.

Input Gate

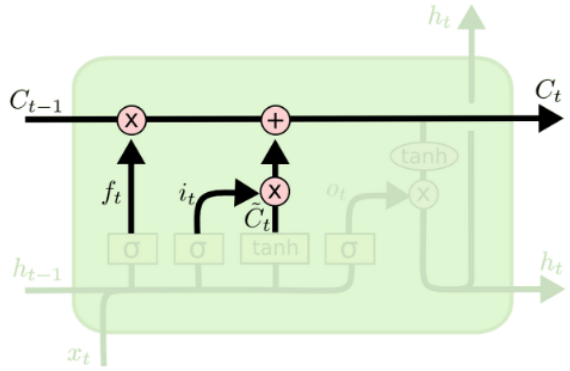


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

LSTM의 input gate layer

- C(t)를 구성하는 두 번째 요소이다.
- 현재 정보의 중요도를 판단할 수 있는 gate이다.

Cell state update

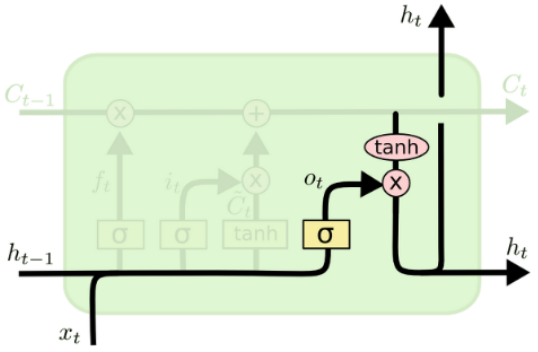


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM의 cell state 업데이트

C(t)를 update하는 구간. 과거와 현재 정보를 조합해서, 구성한다.

Output gate

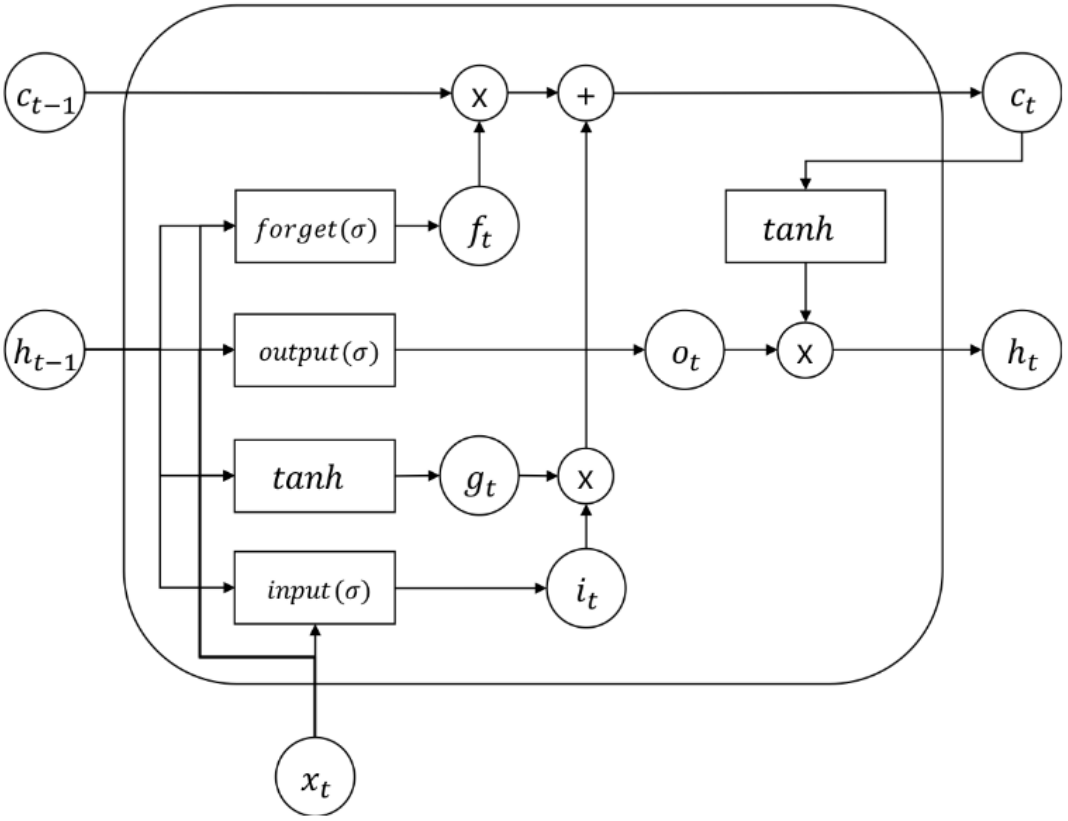


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM의 output gate layer

output을 내보내는 gate. sigmoid를 또 왜 해야하는 지는 잘 모르겠다. 각 activation 값의 중요도를 다시 판단하기엔, 어차피 다음 state에서 weight가 존재하지 않는가?
완성된 C(t)로 output을 내보낸다. 이때, 다시 한 번 non-linear를 보장.
이때 C(t)값을 그대로 사용하는 것이 아닌, 필요한 만큼 뽑아내서 사용한다. (sigmoid)

최종



LSTM은 RNN보다 먼 시점의 정보를 전달할 수 있고, 그래디언트 값도 전달 가능하다.
(RNN은 시점마다 다른 네트워크를 통과시키는 게 아니라, Weight가 시점마다 동일. ⇒ 먼 시점으로부터의 gradient 전파가 안되는 문제 발생 = gradient vanishing)

시점마다 다른 네트워크를 사용 X ⇒ 한 Layer는 같은 weight를 사용 ⇒ 각 시점의 gradient 값이 더해진다. ⇒ deep 한 network에서의 gradient vanishing도 어느 정도 해결

If t = 1인 시점의 input → input gate의 weight의 gradient를 구하고자 할때,

C(t)를 미분 시, 편미분에 직접적으로 관여하는 것은 update gate에서의 activation & output gate에서의 +(행렬 덧셈) & input gate에서의 X(아다마르 곱 = 원소 곱) & input gate의 activation 이다.
행렬 덧셈은 편미분에 관여를 하지 않는다.
아다마르 곱은 affine 변환으로 하나의 연산으로 계산 가능
따라서 실질적으로, 아마마르 곱 & activation 2개가 관여한다고 볼 수 있다.

> if input gate의 값들이 1이라면, (먼 과거의 정보가 엄청 중요한 data라고 가정)
> 아마다르 곱도 관여 X ⇒ activation 2개만 관여하기 때문에, 먼 시점의 정보도 온전히 전달 가능하다.

연산속도가 느리다. (parameter가 많다)

여전히 긴 state의 정보는 기억하지 못한다. (아다마르 곱이 연속적으로 곱해질떄, 1보다 작은 수가 계속해서 곱해지기 때문에)

Cell state(database)에서 필요없는 정보는 지우고(forget gate), 새로운 정보를 update(input gate)한 뒤에 해당 시점에서 필요한 정보만 빼서(output gate) 사용하는 방식이다.