- 따라서, n개의 은닉층으로 이루어진 특징 학습 부분은 아래와 같이 표현할 수 있다.

$$h^{(n)} = F(x) = f^{(n)}_{\theta^{(n)}}(...f^{(k)}_{\theta^{(k)}}...(f^{(2)}_{\theta^{(2)}}(f^{(1)}_{\theta^{(1)}}(x))) \quad (9)$$

F(x) = 특징학습

$f^{(n)}$: n 번째 은닉층

x : $\theta^{(n)}$: n 번째 은닉층의 가중치와 bias 등의 파라미터 집합

- 특징학습 부분과 마찬가지로 예측을 수행하기 위한 예측학습 부분 또한 아래와 같이 표현할 수 있다.

$$\hat{y} = P(h^{(n)}) = f^{(o)}_{\theta^{(o)}}(h^{(n)}) = f^{(o)}(h^{(n)};\theta^{(o)}) = \sigma^{(o)}(W^{(o)}h^{(n)} + b^{(o)}) \quad (10)$$

$\hat{y}$: 모델의 예측값

P(x) : 예측학습 부분

$f^{(o)}$: 출력층

- 결과적으로, 심층신경망은 특징학습을 담당하는 은닉층과 예측학습을 담당하는 출력층의 결합이므로 n개의 은닉층으로 이루어진 DNN은 아래와 같이 표현할 수 있다.

$$\hat{y} = DNN(x) = P(F(x)) = f^{(o)}_{\theta^{(o)}}(f^{(n)}_{\theta^{(n)}}(...f^{(k)}_{\theta^{(k)}}...(f^{(2)}_{\theta^{(2)}}(f^{(1)}_{\theta^{(1)}}(x)))) \quad (11)$$

$\hat{y}$: 모델의 예측값, x : 입력 값

P(x) : 예측학습 부분, F(x) : 특징학습 부분

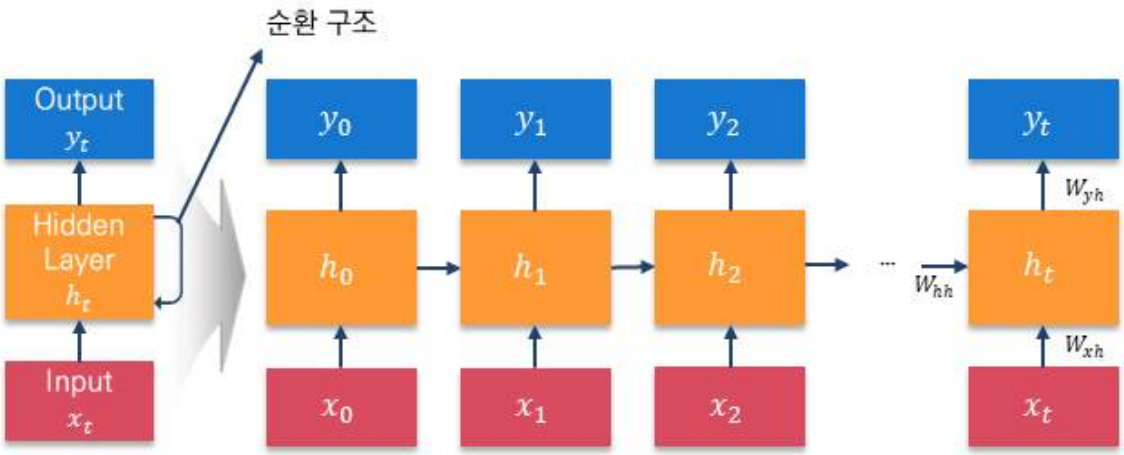
$f^{(n)}$: n 번째 은닉층

$\theta^{(n)}$: n 번째 은닉층의 가중치와 bias 등의 파라미터 집합

$f^{(o)}$: 출력층, $\theta^{(o)}$: 출력 층의 가중치와 bias 등의 파라미터 집합

✓ **순환신경망(Reccurrent Neural Network)**

- 기존의 인공신경망은 모든 입력과 출력이 독립적, 순차적으로 동작하여 데이터의 입력 데이터의 순서와 지속성 등을 고려할 수 없다.
- 순환신경망(Recurrent Neural Network, RNN)은 순환구조를 가진 인공신경망으로써, 순서가 있는 데이터는 시퀀스 데이터(sequence data)의 학습에 있어 입력 데이터의 순서를 반영하여 처리하는 구조를 지닌 신경망 모델이다.
- 순환신경망은 현재의 입력변수를 처리하기 위해 모델에 입력된 과거의 입력변수를 재귀(recursion)하여 현재와 과거의 독립된 입력변수를 연결하여 처리하며, 지난 시점의 은닉층을 현 시점(t)의 은닉층에 누적시켜 계산한다 <그림 2-21>.



<그림 2-21> 순환신경망의 구조

- 순환신경망은 이러한 특징으로 인해 이전의 계산결과가 다음 계산에 영향을 미치기 때문에 데이터의 순서가 정해져 있는 시퀀스 데이터, 특히 시계열 데이터를 처리하는데 적합하다고 알려져 있다.
- 정리하면, 순환신경망은 하나의 인공신경망이 여러 개 복사된 형태로 구성되어 있고 각 인공신경망은 시퀀스 데이터의 특정 시점 t에서 다른 시점인 t+1으로 은닉층 벡터를 넘겨주고 있는 형태로 시퀀스 데이터의 길이에 관계없이 자유롭게 입력변수와 출력변수 설정이 가능하다 <그림 2-22>.
- 이를 수식으로 표현하면 아래와 같으며, f_w는 활성화 함수, h_t는 t 시점의 은닉층으로 이는 t-1 시점의 은닉층과 매 시점마다 적용되는 입력값 벡터 x_t의 활성화 함수로 표현된다.

$$h_t = f_w(h_{t-1}, x_t) \tag{12}$$

f_w : 활성화 함수(Activation function)

h_t : 은닉층

h_{t-1} : 이전 시퀀스의 은닉층

x_t : 입력값

- 이때 순환신경망에서는 일반적으로 활용되는 활성화 함수인 하이퍼볼릭 탄젠트 함수를 사용하여 은닉층은 아래와 같이 표현할 수 있다.

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \tag{13}$$

W_{hh} : 이전 시점의 은닉층 벡터 h_{t-1}를 위한 가중치,

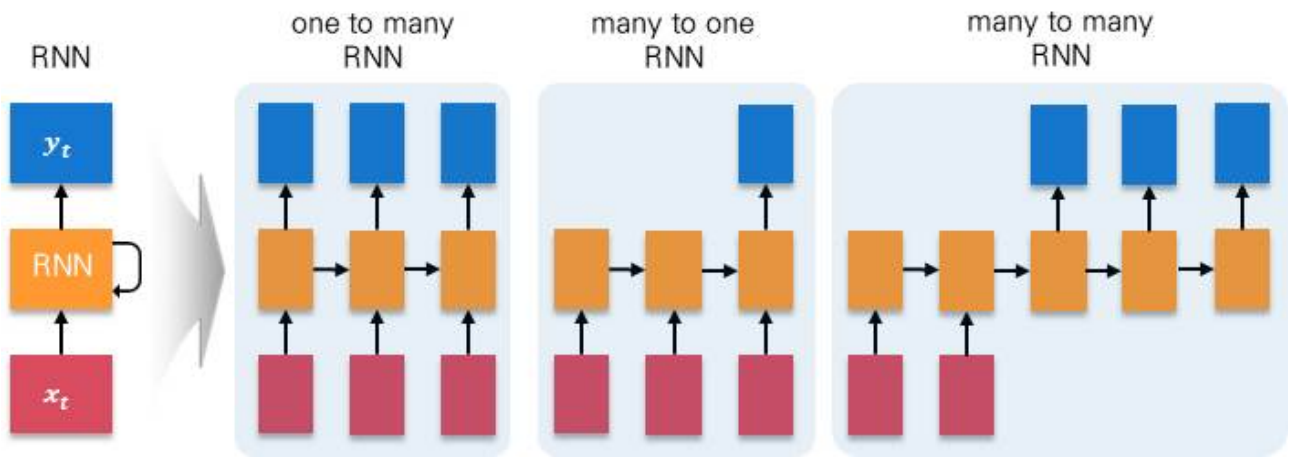
W_{xh} : 입력 벡터 x_t를 위한 가중치

- 이를 토대로 순환신경망의 출력값은 아래와 같이 표현할 수 있다.

$$y_t = W_{hy}h_t \tag{14}$$

y_t : 출력벡터

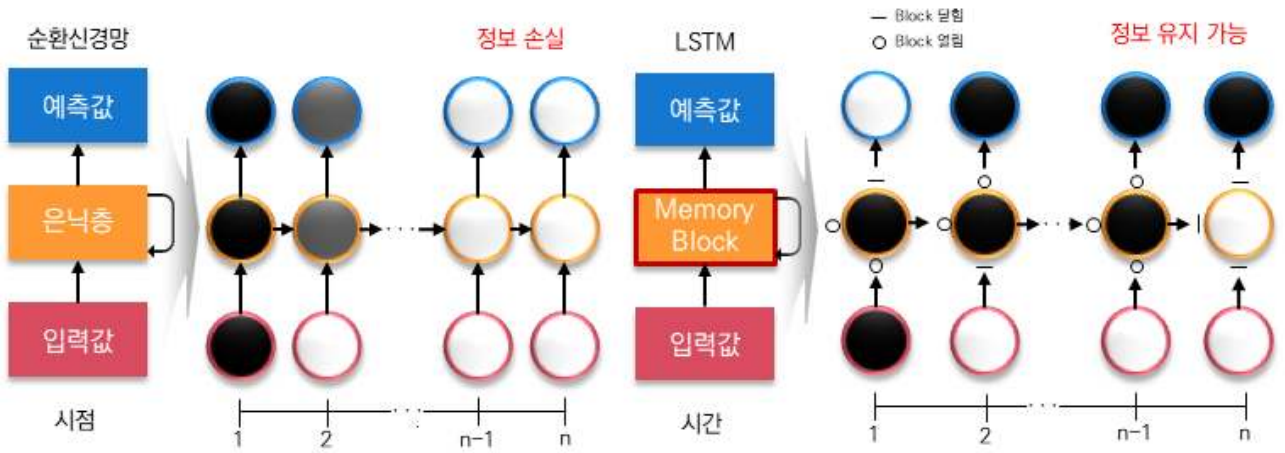
W_{yh} : y_t를 위한 가중치



<그림 2-22> 다양한 순환신경망의 구조 예시

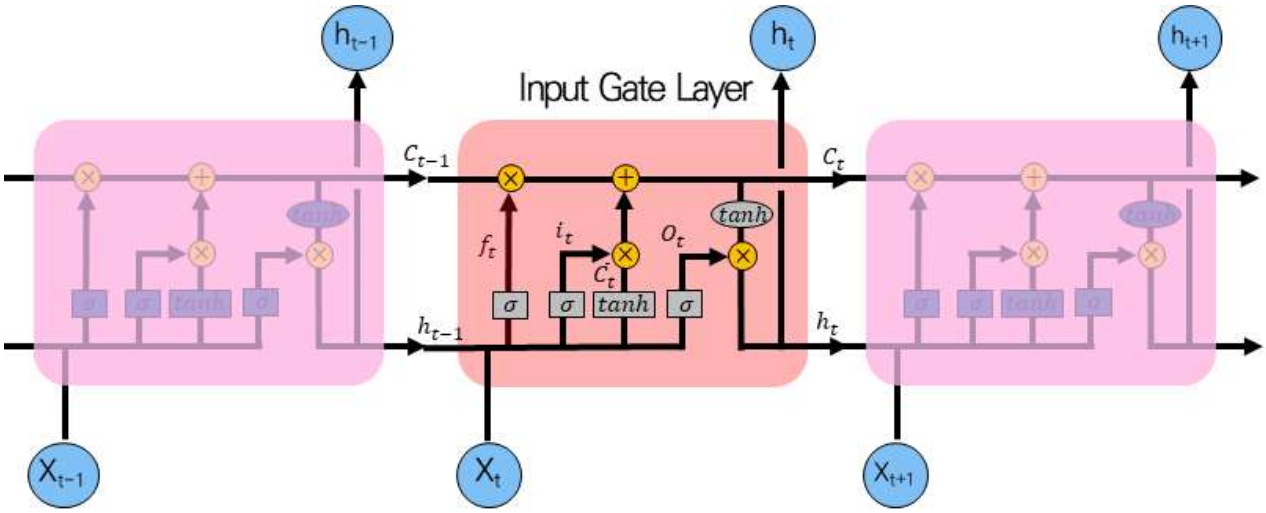
✓ 장단기 메모리(Long Short Term Memory, LSTM)

- LSTM은 순환형 구조로 인한 장기 의존성(long-term dependency)로 인한 과거 순서의 자료에 대한 가중치가 소실되는 가중치 소실(vanishing gradient) 문제를 해결하기 위해 제시되었다 <그림 2-23>.



<그림 2-23> LSTM의 장기의존성 및 가중치 소실 해결 방법

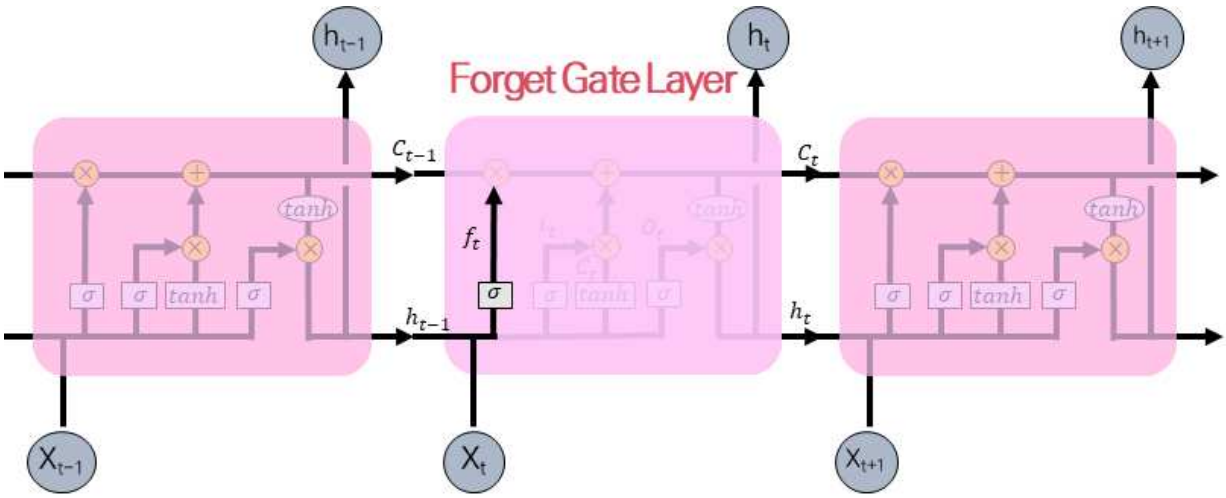
- LSTM은 가중치 반영 및 활성화 함수 변환을 통해 입력값 벡터에서 출력값 벡터로 변환하는 단계를 하나의 셀(cell)로 정의한다 <그림 2-24>.



<그림 2-24> LSTM cell의 구조

- 중단기 순환신경망은 셀 내부의 상태량인 셀 상태(cell state)를 망각, 입력, 갱신, 출력 총 4단계의 계산과정을 통해 가중치 소실, 장기의존성 문제가 발생하지 않도록 조절한다.
- LSTM의 작동 단계 중 첫 번째 단계는 망각단계로써, 특정 정보의 제거 여부를 망각 게이트(forget gate, f_t)를 활용하여 결정한다 <그림 2-25>.
  - 즉, 첫 번째 단계는 셀 상태(cell state)에서 어떤 정보를 버릴지, 유지할지를 결정하는 단계로써 이전 시점의 은닉층 벡터 h_{t-1}와 입력벡터 x_t를 받아 0과 1사이 값을 셀상태 C_{t-1}에 전송한다.
  - 이때 0은 정보의 완전 제거, 1은 정보의 완전 유지를 의미하며 중단기 순환신경망의 셀(cell)내에서 정보를 유지할지 제거할지 선택하게 된다(Jung et al., 2018).

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{15}$$



<그림 2-25> LSTM cell의 구조

- 두 번째 단계는 입력단계로써 입력게이트(input gate, i_t)를 통하여 새로운 정보의 저장여부를 결정하는 단계로 이후에 들어오는 새로운 정보 중 어떤 정보를 셀상태(cell state)에 저장할 것인지 결정하는 단계이다 <그림 2-26>.
  - 입력단계는 세부적으로 1) 입력게이트 층(Input gate layer)라 불리는 Sigmoid 활성화 함수 σ가 어떤 값을 업데이트 할지 정하는 단계; 2) 하이퍼볼릭탄젠트 함수 tanh가 셀상태에 더해질 수 있는 새로운 후보값의 벡터인 $\tilde{C}_t$를 생성하는 2단계로 구성된다.

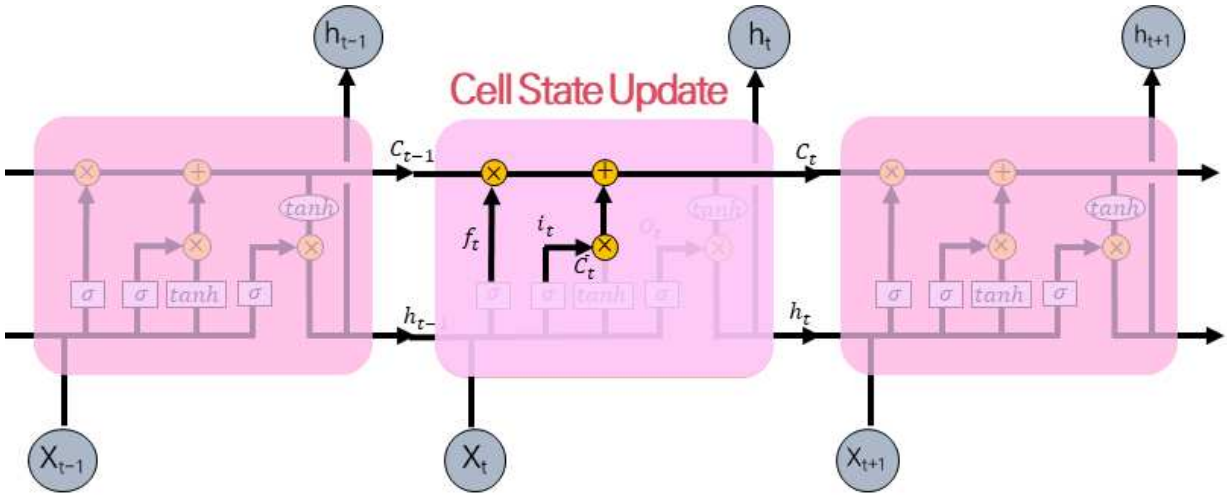
• 결과적으로 입력단계는 아래의 수식을 통해 최종 셀 상태를 갱신할 값을 만든다.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{16}$$

i_t : 0 또는 1의 값을 가지는 입력게이트 값

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_c) \tag{17}$$

$\tilde{C}_t$: tanh로 구성되어있는 셀 상태의 중간값



<그림 2-26> LSTM cell의 구조

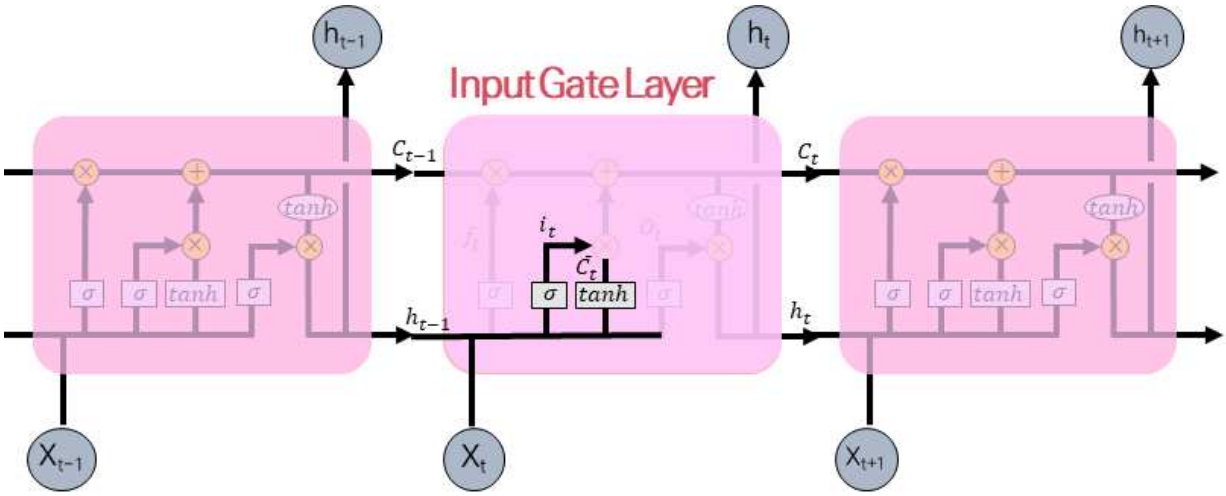
- 세 번째 단계는 갱신단계로써 입력게이트와 출력게이트의 값을 이용하여 셀 상태를 갱신한다.

• 갱신단계에서는 망각단계의 망각게이트 값 f_t를 과거 셀 상태값 C_{t-1}를 곱해서 망각단계에서 결정된 작업을 수행한 후 입력단계의 입력게이트 값 i_t과 셀 상태의 중간값 $\tilde{C}_t$의 곱을 통해 망각단계에서 갱신하기로 한 값을 얼마나 갱신할지 결정한다.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{18}$$

C_t : 현재 셀 상태값

C_{t-1} : 과거 셀 상태값



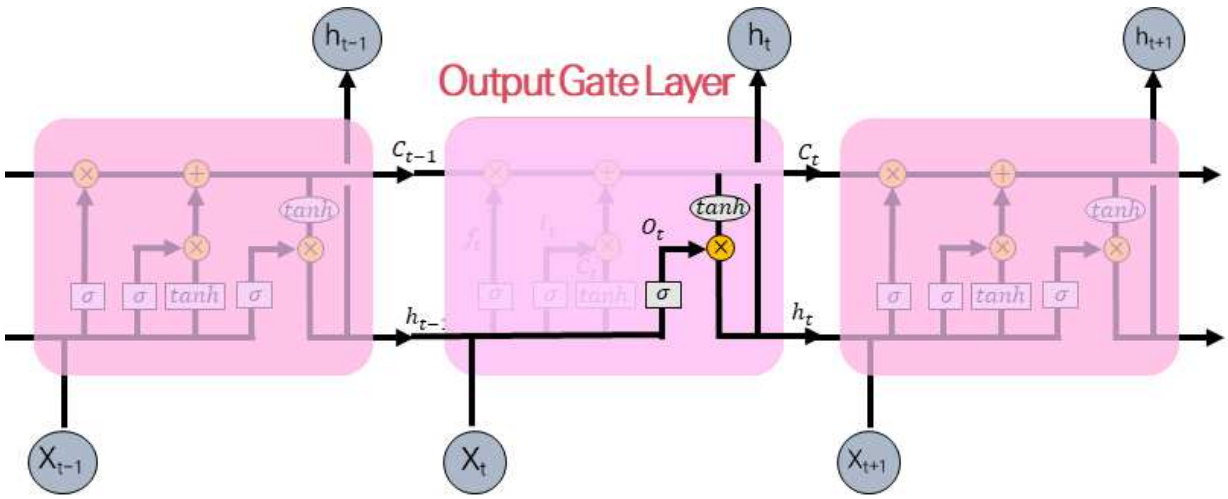
<그림 2-27> LSTM cell의 구조

- 마지막 단계는 출력단계로써 출력게이트와 셀 상태를 이용하여 출력값을 계산한다.
  - 출력단계에서는 tanh를 사용하여 망각, 입력, 갱신단계를 거쳐 활성화된 셀상태 C_t와 곱으로 특정 시점 t의 은닉층 벡터 h_t를 출력한다.

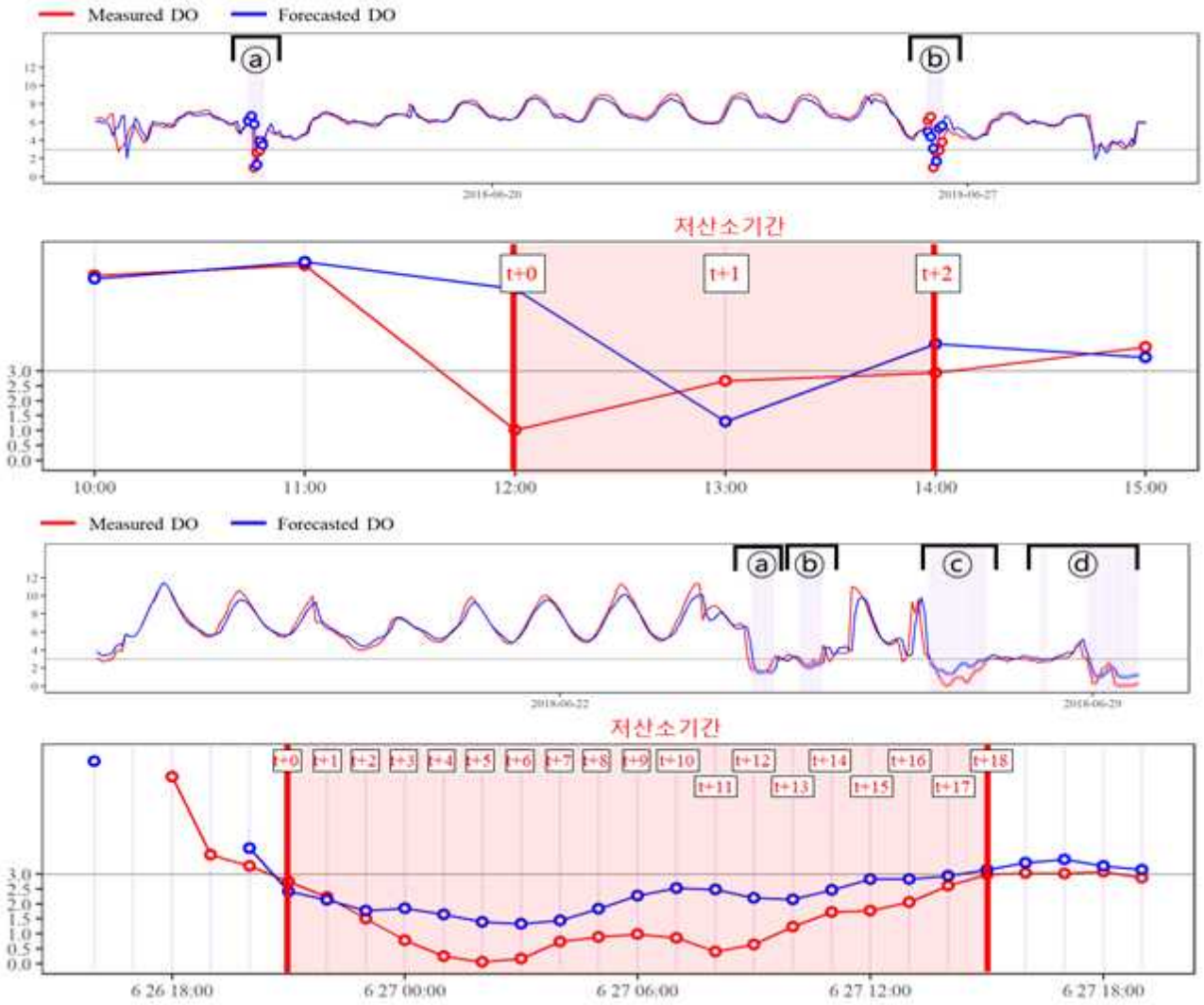
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{19}$$

$$h_t = o_t \bullet \tanh(C_t) \tag{20}$$

o_t : 0 또는 1의 값을 가지는 출력게이트 값



<그림 2-28> LSTM cell의 구조



<그림 2-29> LSTM을 활용한 저산소 시기 용존산소 농도 예측 예시

1-2). 딥러닝 모델의 하이퍼 파라미터 최적화

- 최적화란 함수 $f: X \rightarrow R$ 에 대해 $f(x)$ 를 최소로 하는 해를 찾는 방법을 의미한다. 그 중 가장 좋은 성능 측정값을 갖는 하이퍼파라미터를 선정하는 방식을 하이퍼파라미터 최적화라고 한다(Bergstra et al., 2012).
- 일반적으로 최적의 성능 측정값을 갖는 하이퍼파라미터는 아래 식으로 표현할 수 있다.

$$\lambda^* \equiv argmin\ \Psi(\lambda) \tag{21}$$

λ^*: 최적해
λ: 하이퍼파라미터 셋
Ψ: 하이퍼파라미터 반응함수(목적함수)

- 하지만, $\lambda \in \Lambda$에 대하여 하이퍼파라미터가 존재하는 모든 실수 범위에 대해 Ψ를 평가하는 것은 불가능하다.
- 따라서, 현실적으로 하이퍼파라미터 반응함수를 평가가능한 정도의 적당한 자연수 S를 선정하여 $\lambda \in \{\lambda^{(1)}, ..., \lambda^{(S)}\}$에 대하여 $\lambda^* \approx \hat{\lambda}$인 최적 하이퍼파라미터 추정값 $\hat{\lambda}$를 찾게된다.
  - 이때, Ψ는 일반화된 성능측정값을 나타내기 위해 교차검증(Cross - validation)을 사용한다(Bergstra et al., 2012).
- 하이퍼파라미터를 최적화하는 알고리즘은 다양하게 존재한다. 가장 많이 사용되는 방식은 수동 선택법(Manual search), 격자 탐색법(Grid search), 랜덤 탐색법(Random search)이 있다.
  - 수동 선택법은 가장 기본적인 방식으로 실무자의 경험과 사전 지식에 따라 하이퍼파라미터를 선택하여 평가하고 $\hat{\lambda}$를 찾는 방식으로 불안정하고, 하이퍼파라미터의 차원이 높은 모델의 경우에는 적용이 거의 불가능하다는 단점이 있다.
  - 격자 탐색법은 수동 선택법의 불안정성을 해결하기 위해 각 하이퍼파라미터별로 구간을 임의의 격자로 나누어, 해당 파라미터의 구간을 이산형 집합으로 정의한 뒤, 각 하이퍼파라미터의 곱집합으로 표현되는 경우의 수에 대해 반응함수를 평가하여 $\hat{\lambda}$를 찾는다.

- 이 방식은 하이퍼파라미터의 수가 증가하면 평가에 필요한 비용이 기하급수적으로 증가한다는 문제점과 지정된 경우의 수에 포함되지 않는 격자사이의 값은 반응함수 평가에 포함되지 않는다는 한계점이 있다(Bellman, 1961).
- 랜덤 탐색법은 임의의 수 S만큼 하이퍼파라미터 공간상에서 무작위로 지점을 선정하여 반응함수를 평가하므로 격자 탐색법의 한계점을 보완하면서도 효과적으로 $\hat{\lambda}$를 찾는다(Bergstra et al., 2012).
- 그러나, 무작위성에 의존하여 하이퍼파라미터를 탐색하는 방식은 하이퍼파라미터 수가 많은 경우 적은 평가횟수로는 신뢰도가 낮아 많은 평가횟수를 선정해야한다는 문제점과 전역 최적해를 보장하는 논리적 신뢰도가 부족하다는 한계점이 있다.

- 이에 본 연구에서는 베이지안 최적화를 활용하여 딥러닝 모델의 하이퍼파라미터 최적화를 수행하고, 모델의 성능을 확보하고자 한다.
  - 베이지안 최적화는 최적해 탐색과정에서 이전 하이퍼파라미터 탐색결과를 이용하여, 확률모델 기반으로 다음 탐색지점을 선정하기 때문에 가장 효율적으로 하이퍼파라미터를 탐색하는 알고리즘으로 알려져 있어, 목적함수 평가에 비용이 많이 드는 블랙박스 모델의 하이퍼파라미터 탐색에 적용하기 적합하다고 판단된다(Bergstra et al., 2011; Mockus, 2012).

✓ **베이지안 최적화(Bayesian optimization)**

- 베이지안 최적화는 수치최적화(Numerical optimization)의 한 일종으로 큰 틀에서는 시행착오를 통해 최적화 조건을 만족하는 최적해를 스스로 탐색하는 수치최적화와 과정이 동일하다.
- 베이지안 최적화는 모델기반으로 시행착오 과정에서 다음 탐색할 해를 결정하는 부분에서 차별점이 있으며, 이와 같은 특성 때문에 순차 모델 전역최적화(Sequential Model-Based Global Optimization, SMBO)라고도 한다.
- 베이지안 최적화는 선택함수(selection function 또는 Acquisition function)을 통해 순차적인 하이퍼파라미터 탐색과정에서 이전 탐색결과들을 토대로 다음 탐색지점을 선정한다.
- 선택함수는 다양한 종류가 있으며, 그 중 Expected Improvement(EI)는 함수가 의미하는 바가 직관적으로 이해하기 쉽고 다양한 상황에 적용하여도 그 성능이 안정적이라는 장점을 갖고 있으며, 아래와 같이 표현된다.