RNN(Recurrent Neural Networks)

- **Sequential Model**

  소리, 문자열, 주가 등 이벤트의 발생 순서가 중요한 데이터

  시계열(time-series) 데이터

  i.i.d(독립동등분포) 가정 자주 위배 => 순서를 바꾸거나 과거 정보 손실 발생 시, 데이터 확률분포 바뀜

  이전 시퀀스 정보 가지고 앞으로 발생할 데이터 확률분포 다룸

  => 조건부확률 이용

  $$P(X_1, \dots X_t) = P(X_t|X_1, \dots, X_{t-1})P(X_1, \dots, X_{t-1})$$
  $$= \prod_{s=1}^{t} P(X_s|X_1, \dots, X_{s-1})$$

  => $X \sim P(X_t|T_{t-1}, \dots, X_1)$로 과거의 정보를 담은 확률분포 표현

  과거의 모든 정보 필요한 것 X, 맥락에 따라 잘 truncate 하는 것이 중요

  - **Naive sequence model**

    $X_t \sim P(X_t|X_{t-1}, \dots, X_1)$

    $X_{t+1} \sim P(X_{t+1}|X_t, \dots, X_1)$

    이전의 모든 정보 담는 방법, 현 시점에서 과거의 모든 정보 담는 모델

    길이가 가변적인 데이터를 다룰 수 있는 모델

    X_t는 t-1개, X_t는 t개의 정보 이용

  - **Autoregressive model** (자기회귀 모델, AR 모델)

    $X_t \sim P(X_t|X_{t-1}, \dots, X_{t-\tau})$

    $X_{t+1} \sim P(X_{t+1}|X_t, \dots, X_{t-\tau+1})$

    각 시점마다 가변적인 길이 데이터 다룸, 고정된 길이의 데이터를 다루는 문제로 재정의

    => 고정된 길이 τ만큼 정보를 담는 모델

    - **Markov model (first-rder autoregressive model)**

      AR 모델에서 가장 간단한 모델: 현재의 정보는 바로 이전 단계의 과거의 정보에만 영향을 받는다는 가정에 정의

      $P(X_1, \dots, X_t) = P(X_t|X_{t-1})P(X_{t-1}|X_{t-2}) \dots P(X_2|X_1)P(X_1) = \prod_{s=1}^{t} P(X_s|X_{s-1})$

      고정된 길이 $\tau = 1$ 만큼의 정보를 담는 모델

      현 시점의 정보가 단순히 바로 단계의 과거 정보만 가지고 결정

      => 많은 과거의 정보 이용 X

  - **Latent AR Model** (잠재자기회귀 모델)

    고정된 길이의 데이터를 다루는 동시에, 유의미한 과거의 정보를 담는 모델

    $X_t \sim P(X_t|X_{t-1}, H_t)$

    $X_{t+1} \sim P(X_{t+1}|X_t, H_{t+1})$

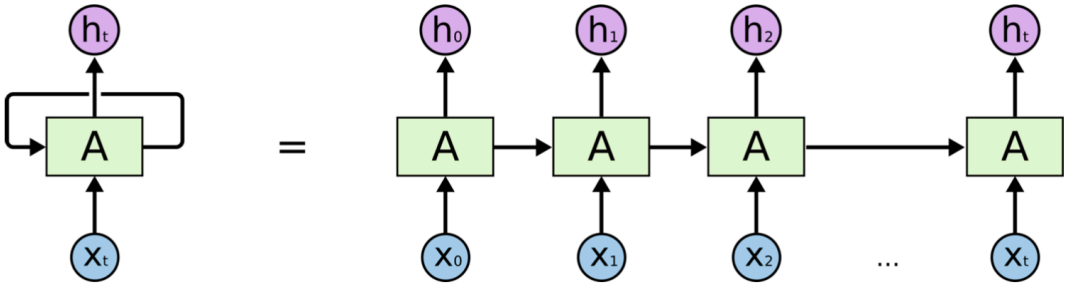
    $H_t = Net_\theta(H_{t-1}, X_{t-1})$

    잠재변수(Latent Variable) H_t 이용하여 정의

H_t를 신경망을 통해 반복해서 사용, 시퀀스 데이터의 패턴을 학습하는 모델 => **RNN(Recurrent Neural Networks)**

- **Recurrent Neural Neetwork**

  MLP와 달리, 자기 자신을 돌아오는 구조 있음

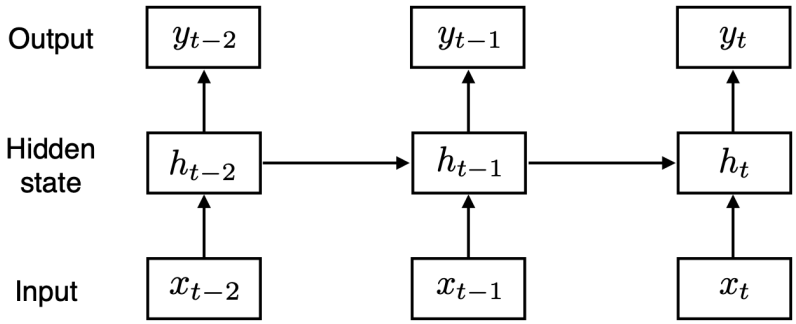


  - **short-term dependencies (단점)**

    RNN: 하나의 fixed rule로 정보들을 계속 취합

    => 먼 과거에 있던 정보가 미래까지 살아남기 어려움

    현재에서 몇 개의 전 과거의 데이터는 잘 고려되지만, 멀리 있는 정보는 고려하기 힘듬



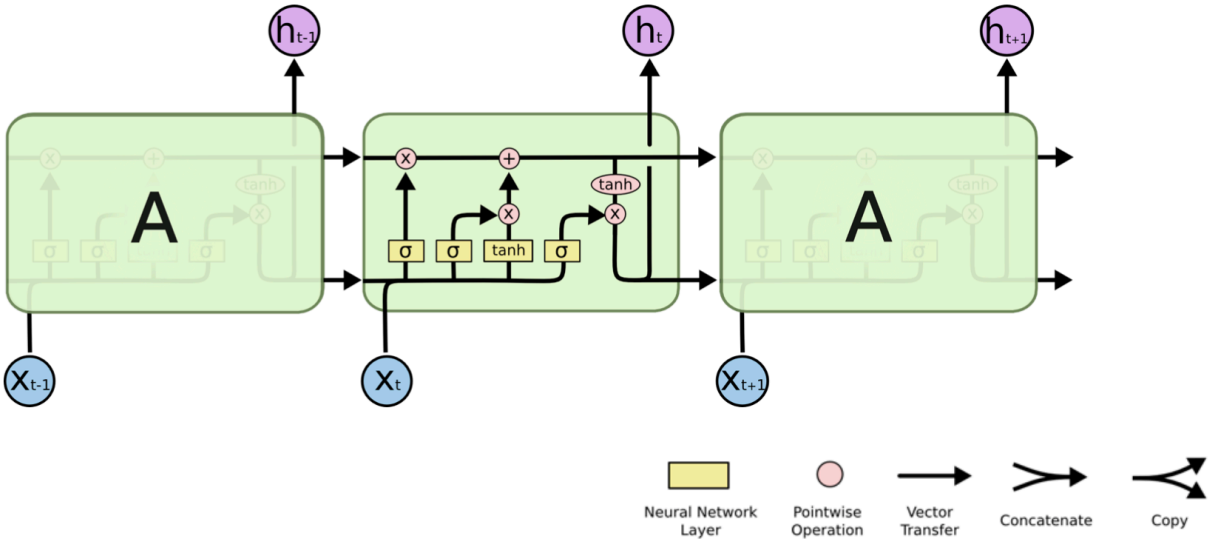
$$h_1 = \phi(W^T h_0 + U^T x_1)$$
$$h_2 = \phi(W^T \phi(W^T h_0 + U^T x_1) + U^T x_2)$$
$$h_3 = \phi(W^T \phi(W^T \phi(W^T h_0 + U^T x_1) + U^T x_2) + U^T x_3)$$
$$h_4 = \phi(W^T \phi(W^T \phi(W^T \phi(W^T h_0 + U^T x_1) + U^T x_2) + U^T x_3) + U^T x_4)$$

Vanishing / exploding gradient

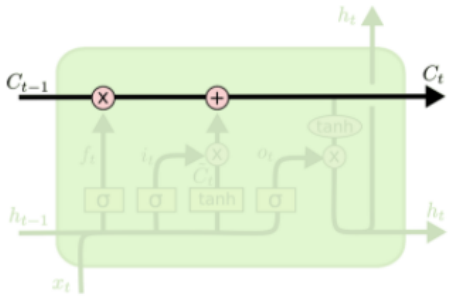
- **LSTM(Long Sort Term Memory)**

  RNN의 멀리 있는 과거 정보 gradient vanshing 문제로 학습 기여 못하는 문제 해결을 위해 고안

  RNN의 hidden statedp cell state을 추가한 구조



- **Cell state**

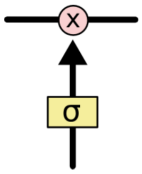


same as 컨테이너 벨트

작은 linear interaction만을 적용시키면서 전체 체인 계속 구동

정보가 전혀 바뀌지 않고 흐르게만 하는 것을 매우 쉽게 함

- **gate**



정보 전달될 수 있는 추가적인 방법

sigmoid layer와 pointwise 곱셈으로 이루어짐

sigmoid layer 0,1 숫자 내보냄 => 각 컴포넌트가 얼마나 정보 전달해야하는지 척도

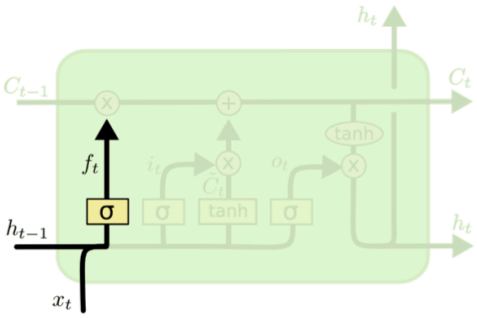
  - 0: 아무 것도 넘기지 말라 // 1: 모든 것을 넘겨라

- **Forget Gate**

어떤 정보를 버릴지 결정, 과거 정보 얼마나 반영하지 결정

(0,1) 값을 가지는 sigmoid가 과거의 정보를 얼마나 반영할지 결정

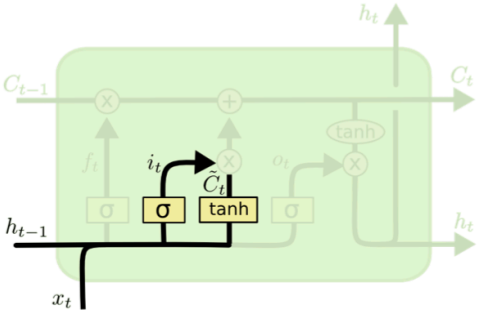
  - 1에 가까울수록 과거 정보 많이 반영 // 0에 가까울수록 과거 정보 적게 반영



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

- **Input Gate**

  어떤 정보를 cell state에 저장할지 결정



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

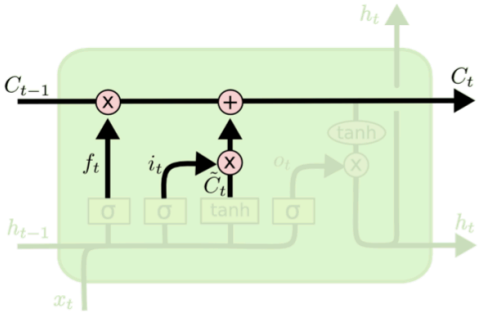
$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

  - i_t: 현시점의 정보를 셀에 얼마나 반영할지 결정
  - $\tilde{C}_t$: 현 시점의 정보 계산

    => 현 시점이 실제 가지고 있는 정보가 얼마나 중요한지 반영

- **Update Cell**

  forget gate, input gate의 값을 잘 취합해 cell state를 update
  과거 정보와 현시점의 정보의 중요도 반영해서 update

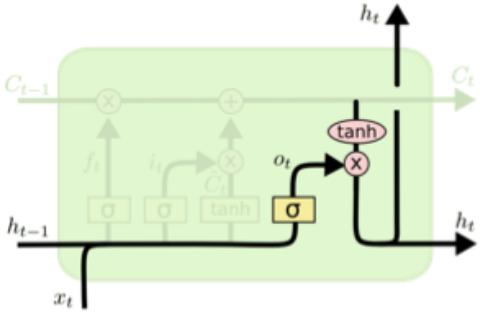


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

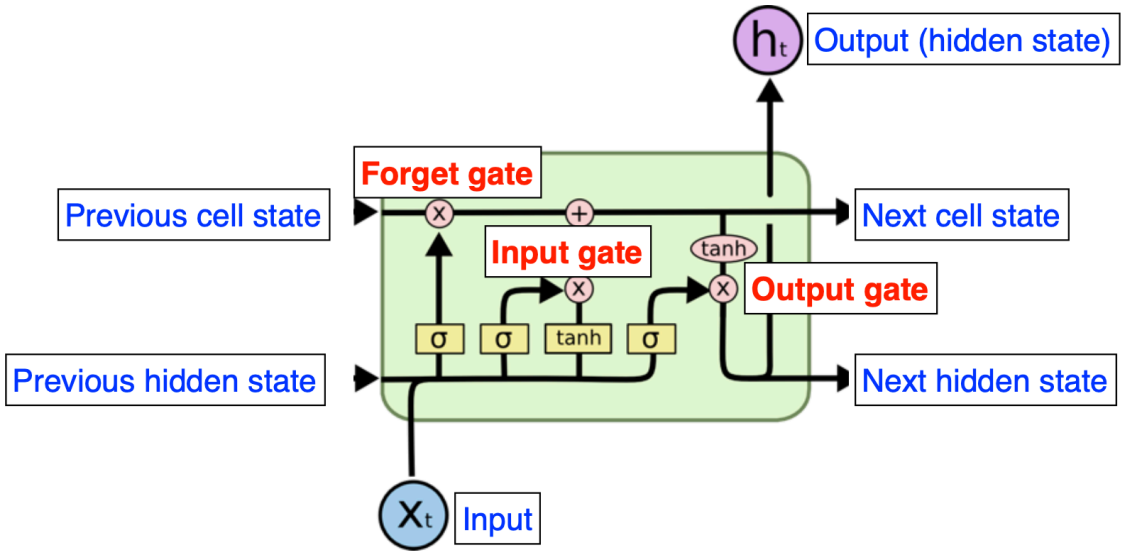
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

- **Output Gate**

  update된 cell stat(과거 중요한 모든 정보)를 hidden state(다음 time step에 당장 필요한 정보)로 출력할 양 결정

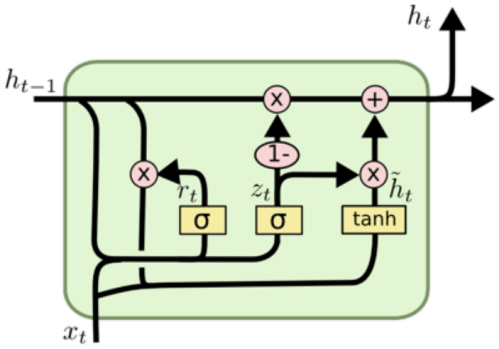


$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * tanh(C_t)$$



- **GRU(Gated Recurrent Unit)**

  LSTM와 유사, but cell state 없이 더 간략한 구조



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

  LSTM와의 차이점: cell state C_t의 역할을 H_t가 같이 수행

  C_t의 forget gate f_t와 input gate i_t의 역할을 z_t의 가중평균으로 대체

  두 개의 독립적인 gate => 1개의 gate, cell state 없이 hidden state만 있음 => **계산량, 메모리 이점**