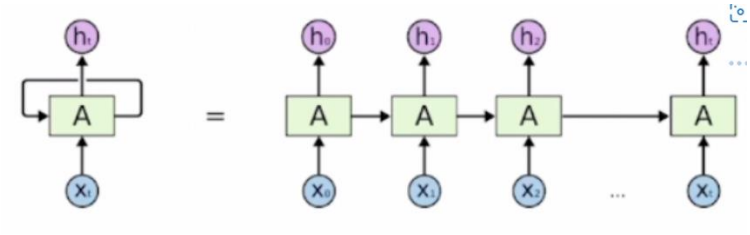
2. 자연어 처리와 딥러닝

Recurrent Neutral Network (RNN)

RNN : 시퀀스 데이터가 입력/출력으로 주어졌을 때, 입력 벡터 X_t와 전 스텝에서 계산한 히든 스테이트 벡터 X_t-1을 입력으로 받아서 현재 타임 스텝에서의 h_t를 출력하는 구조

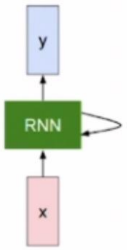


왼쪽 : rolled version의 diagram　　오른쪽 : RNN의 unrolled version

★ 서로다른 타임 스텝에서 들어오는 입력데이터 처리할 때 **동일한 파라미터**를 가진 **반복적 모듈** (recurrent 모듈을 사용한다!)

- hiddenstate 벡터가 다음 타임 스텝의 입력으로 쓰임 + 출력값 계산

$$h_t = f_W(h_{t-1}, x_t)$$



- t : 현재 타임스텝(time step) , w : 웨이트(weight)
- $h_t - 1$: old hidden-state vector
- x_t : input vector at some time step
- h_t : new hidden-state vector
- f_w : RNN function with parameters W
- y_t : output vector at time step t

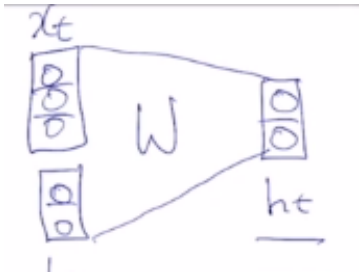
- 파라미터 W : rnn에 필요한 선형 변환 매트릭스를 정의하는 파라미터

1. 위의 변수들에 대하여, h_t = f_w(h_t-1,x_t)의 함수를 통해 매 타임스텝마다 hidden state 를 다시 구한다
2. 이 때, W 와 입력값(x_t, h_t-1)으로 tan_h 를 곱해서 h_t 를 구한다
3. 구해진 h_t, x_t 를 입력으로 y_t 값을 산출한다

- 품사를 예측할 때 => 매 타임 스텝마다 각 단어의 품사 예측값이 나와야 한다
- I hate this movie 가 긍정인지 부정인지 예측할 때 => 마지막 타임스텝에서만 긍정/부정 예측

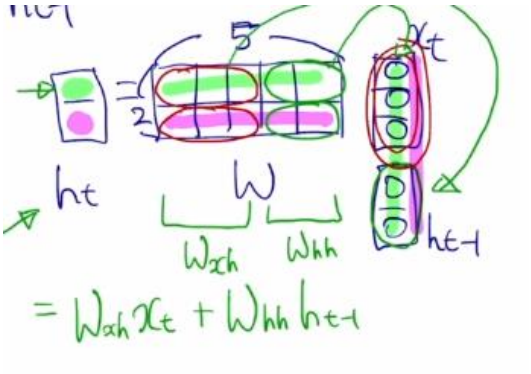
- 매 타임스텝마다 RNN 모듈 정의하는 W : **모든 타임스텝에서 동일한 값 공유** !!

- h_t-1이 입력으로 들어올 때 => 차원이 2차원이라고 가정하면 hidden state vector의 차원수 = 하이퍼파라미터

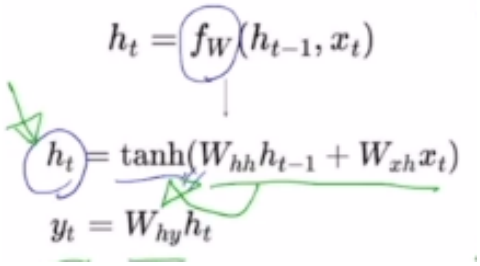


<하나의 완전연결층로 구성된 fW (RNN 모듈) 일 때 h_t 계산 과정>

- h_t의 차원도 h_t-1과 동일한 차원공유 => 차원 여전히 2여야 한다
- nonlinear unit 거치는 것처럼, tanh거쳐서 최종 h_t 계산
- 완전 연결층에서 linear transformation matrix W라고 하면
- W는 2by5 차원
- h_t의 첫번째 노드에서 필요한 연산 : 따로 내적하고 더하기!



- Wxh는 W에서 xt를 ht로 변환하는 역할
- Whh는 W에서 ht-1를 ht로 변환하는 역할
- Why는 W에서 hy를 yt로 변환하는 역할

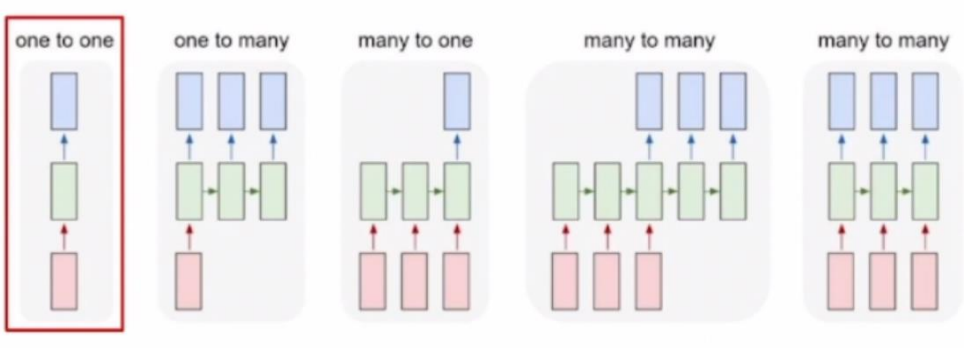


binary class => 출력 벡터는 스칼라, sigmoid 적용해서 binary classification의 확률값이 예측값

multiclass => yt가 클래스 개수만큼 차원가지는 벡터, softmax 층 통과해서 클래스와 동일한 개수의 확률 분포

2. Types of RNN

입력/출력이 sequence 데이터인 경우 적용 가능



1. one to one

• 3차원 벡터가 입력으로 주어지면, 노드를 선형결합과 비선형변환을 통해 2차원벡터의 히든 스테이트 벡터로 변환후, 벡터를 out put으로 출력

• 입출력에 시퀀스 데이터가 아닌 일반 데이터 도식화

ex) [키, 몸무게, 나이]와 같은 정보를 입력값으로 할 때, 이를 통해 저혈압/고혈압인지 분류하는 형태의 태스크

2. one to many

• 입력으로 하나의 이미지 (time step가 아닌)

이미지의 설명글을 예측하기 위해 필요한 단어를 타임스텝별로 순차적으로 생성하는 output

• 입력이 첫번째 스텝에서만 들어간다

• 추가적 입력이 따로 없는 경우에는 같은 사이즈의 벡터가 들어가되, 값이 모두 0으로 채워진 것을 입력으로 들어가게 된다

ex) 이미지 캡션 태스크

3. many to one

• 시퀀스를 입력으로 한다

• 문장이 길이가 달라지면 rnn 셀이 확장되어 반복적으로 수행

ex) 감정분석 : 문장이 입력되면, rnn모듈이 데이터를 처리한 후 마지막 타임스텝에서 나온 ht로 최종 output layer 적용하여 긍정/부정 얘측

4. many to many

1)

• 문장이 주어지면 끝까지 읽은 후, 마지막 타임스텝에서 문장에 해당하는 번역을 예측값으로 출력

위의 그림에서 타임스텝이 5개

ex) machine translation

2) c

• 입력문장을 다 읽고 처리하는 것이 아니라 문장이 주어지면 바로 예측 수행하는 태스크

ex) 단어별로 문장 성분이나 품사를 예측하는 POS tagging, 비디오 classifiacation

해당 프레임이 어떤 scene인지 예측할 때

Character-level Language Model

언어 모델 태스크 : 문자열이나 단어의 순서를 바탕으로 다음 단어가 무엇인지 맞추는 태스크

- wordlevel, character level에서 다 수행 가능

<학습데이터로 hello가 주어졌을 때>

1. character level의 사전 구축

- unique 한 character을 중복없이 모으기

2. 각각의 character은 사전의 개수의 차원을 가지는 원핫벡터로 표현

ex) [1,0,0,0] , [0,1,0,0] ...

3. h e l l 로 주어지면 h 가 주어지면 e를 예측해야하고, h e 가 주어지면 l을 예측해야한다

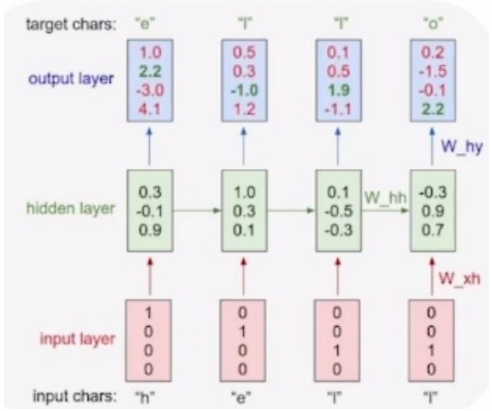
h e l l 이 원핫벡터 형태의 입력으로 주어지면

RNN은 매 타임스텝에서 주어지는 입력벡터 + 전 타임스텝의 h_t-1 을 선형결합하여 h_t를 만든다

h_t, h_t-1의 차원이 3차원이면,

★ h_t-1에서 h_t로의 선형 변환 => x_t에서 h_t로의 선형변환을 담당하는 행렬과 곱함 => bias term b를 더함 => + 비선형변환 tanh 통과=> h_t 구할 수 있다 (fully connected layer의 RNN)

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



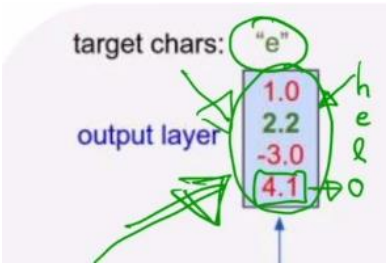
many to many 태스크

- h0는 0인 벡터를 입력으로 한다
- h1 계산하고, 이걸 다음 스텝의 입력으로 사용
- h2=>h3 에서 W_hh 행렬 관여
- x3=>h3 에서 W_xh행렬 관여
- h_t에 output layer 적용하여 최종 출력

• Logit(output) = W_hy * h_t + b

- output layer의 노드 수는 사전의 크기와 동일 (4개)
- multiclass classification을 하기 위해softmax에 입력

  제일 큰 값을 가질 때 해당 확률 값이 가장 크게 나온다



o라고 예측

but e=[0,1,0,0] ground truth 벡터 에 가까워지도록

softmax loss를 적용하여 학습

- h_t-1가 이전 정보를 나타내는 hidden state vector

- 학습을 끝낸 후에 inference 수행 할 때

해당 타임 스텝에서 예측값을 다음 타입 스텝의 입력으로 재사용하여 e를 입력으로 넣으면 다음 캐릭터 또 예측

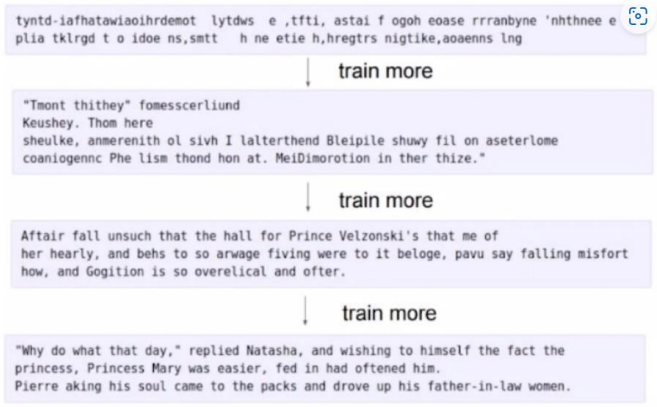
무한한 길이의 시퀀스 생성 예측 가능

ex) 다음날의, 그 다음날의 … 주식 값 예측 / 먼 미래의 주식값도 동일한 모델로 예측 가능

- 문단 학습도 가능

  공백도 특수문자의 하나로 하나의 캐릭터로 생각

  쉼표, 줄바꿈도. 특수문자를 사전에 등록하면 글을 1차원 캐릭터 시퀀스로 봐서 모델에 학습 가능



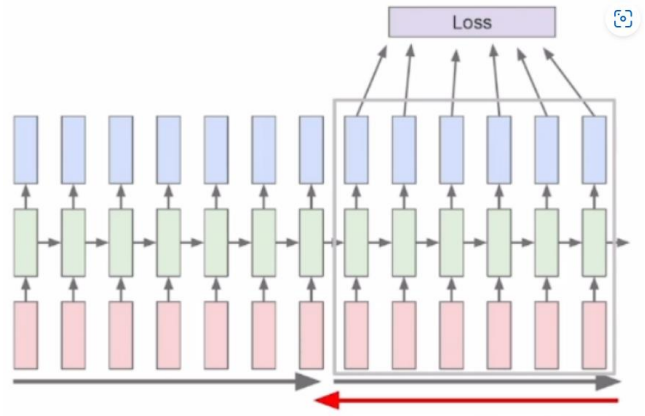
- 등장인물이 한 말로 학습을 진행한 후 한 등장인물을 제시하고 inference 수행하도록 한 결과
- 논문 학습도 가능 (Latex라는 특정 프로그래밍 언어로 작성) RNN에 학습시키고 inference 수행하도록
- C code 생성 가능 (괄호를 열면 줄을 바꾸고, 공백을 언제 사용하는지도 학습)

back propagation through time and Long-Term-Dependency

매 타임 스텝마다 주어지는 캐릭터 존재

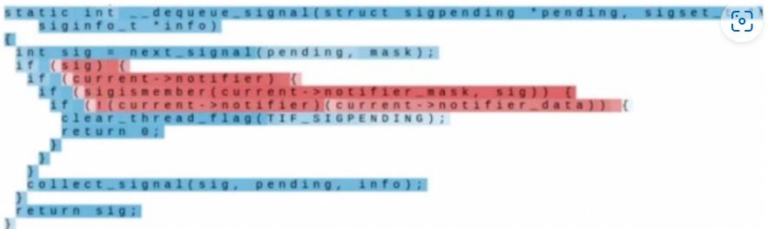
히든 스테이트를 통해 output layer을 통과시켜 예측 값 구하고

다음 캐릭터에 해당하는 groundtruth과의 비교를 통한 loss function을 통해 학습 진행



truncation : 제한된 리소스(메모리) 내에서 모든 시퀀스를 학습할 수 없기 때문에 제한된 길이의 시퀀스로 학습

- rnn에서 필요로 하는 정보를 저장하는 공간 : 매타임 스텝마다 업데이트하는 h_t 히든스테이트 벡터
- 히든 스테이트 벡터의 차원 중 어디에 저장되어있는지 역추적해서 확인 가능



특정한 차원의 히든스테이트 노드가 어떻게 변하는지 시각화

- 값이 양수로 커지면 빨간색
- 값이 음수로 커지면 파란색

- Quote detection cell : 따옴표 닫히는 동안 항상 값이 음수(파랑)였다가 따옴표가 닫히면 빨간색이 된다

  ⇨ RNN 내의 hidden state 벡터의 특정 dimension이 하는 역할 :

  따옴표가 열렸다/닫혔다의 상태 기억

- If statement cell

RNN is excellent but …

fully connected layer 로 구성된 vanilla RNN에서는 동일한 행렬을 매 타임 스텝마다 곱하게 되는데,

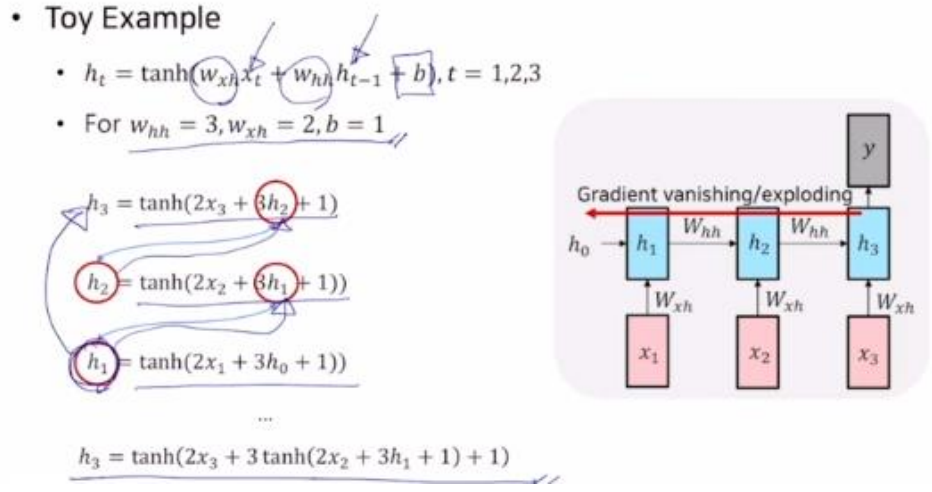
Whh를 곱하고 tanh로 계산되어 정보가 변환되고, 다음 타임 스텝에서도 동일한 Whh를 곱한 후… 반복

⇨ 앞에서 필요로 하는 정보가 여러 타임 스텝 이전에 해당하는 경우 :

Whh가 반복적으로 반영되기 때문에 back propagation이 잘 작동하지 못한다

gradient가 소실/증폭

h1=>h3변환 과정에서 동일한 RNN 구조로 인해 값이 변환, 하나의 식으로 표현 가능



<Whh=3 일 때>

- h1에 대한 편미분 값을 계산하게 되는데, tanh(X)라고 보면, tanh의 접선의 기울기 의미

  괄호 안의 미분값 = 3 => tanh에 해당하는 기울기값 * 3 => ...

  time step 개수 만큼 거듭제곱 => gradient 값이 증폭

  < Whh=0.2일 때>

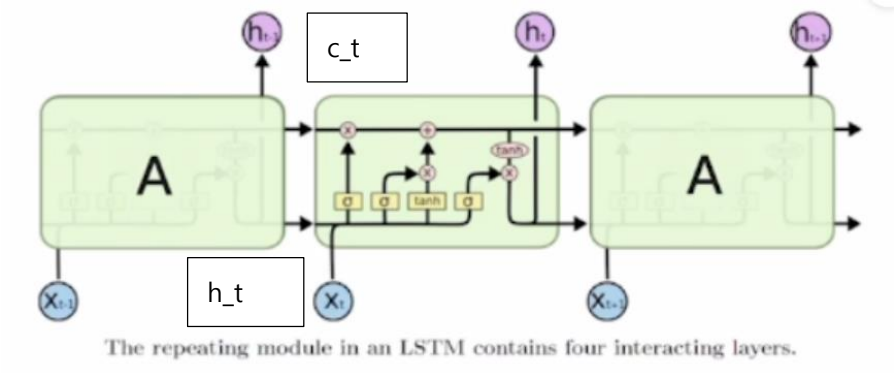
  gradient 소실

Long Short-Term Memory (LSTM)

lstm : gradient 소실/폭발 해결, 타임 스텝이 긴 경우에도 학습 가능

히든 스테이트 벡터: 단기기억 저장 소자 => 단기기억을 더 길게 기억하도록 개선

ht=fw(x_t,ht_t-1)



The repeating module in an LSTM contains four interacting layers.

{c_t,h_t}=LSTM(x_t,c_t-1,h_t-1)

c_t-1 : cell state vector

h_t-1 : hidden state vector

- cell state vector : 완성된 정보를 담는 벡터
- hidden state vector : cell state vector을 한번 더 가공해서 노출할 정보만 남긴 필터링된 벡터

  예측값을 계산하는 output layer 등의 다음 layer의 입력 벡터로 사용

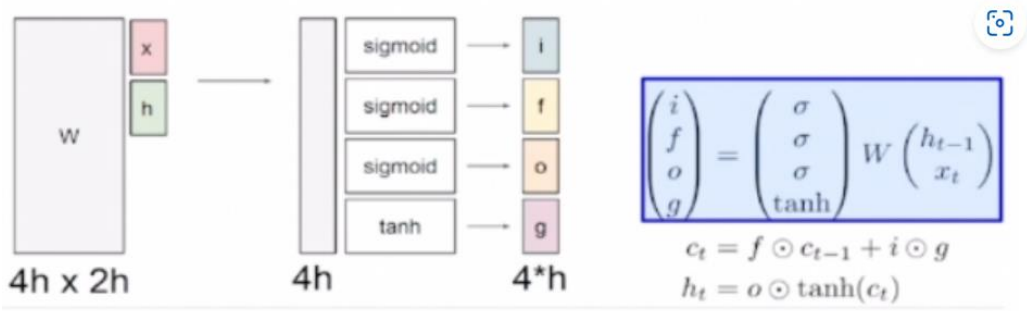
<lstm의 계산 과정>

1. x_t, c_t-1, h_t-1중 **x_t와 h_t-1을 입력으로** 받아서 선형변환

2. 벡터를 4개로 분할하고, 원소별로 sigmoid 나 tanh를 거쳐서 output 값 생성

- forget gate, input gate, output gate, gate gate

- 선형변환한 칼럼의 개수는 2h



• sigmoid를 통해 나온 벡터는 원소별 곱셈을 통해 0~1과 곱해줌 ⇒ 일부의 %만 갖도록 한다

- sigmoid 값이 0.3으로 나오고, 곱해지는 벡터가 5였다면, 5*0.3=1.5 , 30%만 보존

• tanh를 통해 hidden state 벡터를 -1~1 값 ⇒ 현재 타임 스텝에서 계산되는 유의미한 정보

★ I : Input gate로 불리며, cell 에 쓸 지말지를 결정하는 게이트입니다. 즉, 들어오는 input에 대해서 마지막에 sigmoid를 거쳐 0-1 사이 값으로 표현해줍니다. 이 값은 cell state와 hidden state 두 갈래로 흐르게 됩니다.

표현식 : sigmoid(W(xt, ht-1)

★ o : Output gate로 불리며, Cell 정보를 어느정도 hidden state에서 사용해야할 지를 0~1사이 값으로 나타낸다

표현식 : sigmoid(W(xt, ht-1)

★ f : Forget gate 로 불리며, 정보를 어느정도로 지울지를 0~1사이의 값으로 나타낸다

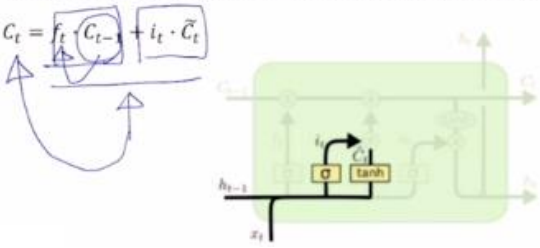
표현식 : sigmoid(W(xt, ht-1)

- c_t-1이 3차원 벡터로 3,5,-2의 값일 때

  forget gate벡터와 곱해져서 선형변환을 해서 만들어지는 output vector (sigmoid를 거친) =0.7,0.4,0.8 일때

  두 값을 원소별 곱셈 ⇒ 2.1,2,-1.6 (3차원으로 나오게 된다)

- Generate information to be added and cut it by input gate
  - $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
  - $\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$ → -1 ~ 1
- Generate new cell state by adding current information to previous cell state
  - $C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$



★ g : Gate gate로 불리며, 어느정도로 Cell state에 반영해야할 지를 -1 ~ 1 사이의 값으로 나타낸다

표현식 : tanh(W(xt, ht-1)

- input gate와 dimension 별로 0~1값과 곱해짐으로써 벡터를 변환
- 필요한 정보만 유지한 부분에 덧셈으로 벡터 변환

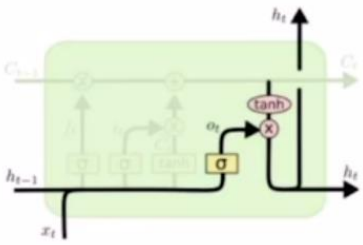
왜 따로 input gate를 만들어서 한번에 c_t~ gate gate를 출력하지 않을까?

한번에 선형변환만으로 c_t-1로 더해줄 정보 만들기 어려운 경우

⇨ gate gate 형태로 만든 후, dimension별로 정보를 덜어내서 c_t-1에 더해주고자하는 정보를 두단계로 나눠서 진행

- Generate hidden state by passing cell state to tanh and output gate
- Pass this hidden state to next time step, and output or next layer if needed
  - $o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$
  - $h_t = o_t \cdot \tanh(C_t)$

-1 ~ 1



- dimension 비율 별로 값을 작게 만들어서 h_t 구성

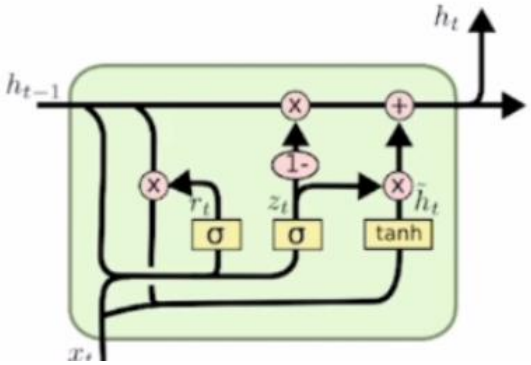
• c_t : 기억해야하는 모든 정보를 담는다

• h_t : 현재 타임스텝에서 output layer의 입력으로 사용, 예측값에 직접적으로 필요한 정보만을 담는다

GRU

- 적은 메모리 요구량, 빠른 계산 시간
- lstm에서 두가지 종류의 벡터로 존재하던 cell state, hidden state를 일원화 => **h_t만 존재**
- h_t가 lstm의 c_t와 비슷한 역할

- $z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$
- $r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$
- $\widetilde{h_t} = \tanh(W \cdot [r_t \cdot h_{t-1}, x_t])$
- $h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \widetilde{h_t}$
- c.f) $C_t = f_t \cdot C_{t-1} + i_t \cdot \widetilde{C_t}$ in LSTM



- input gate 만 사용

  forget gate 자리에는 1- input gate 값을 사용

  ⇨ input gate 가 커질수록 forget gate 값은 작아짐

- h_t-1과 현재 정보인 h_~t 가 lstm에서의 gate gate과 동일한 역할

  두 정보간의 가중 평균을 내는 형태로 계산

  ⇨ 두개의 독립된 게이트가 아닌 하나의 게이트 만으로 계산

cell state vector가 업데이트 되는 과정이 Whh를 계속적으로 곱하는 것이 아니라,

전타임 cell state에서 forget gate를 곱하고 필요로 하는 정보를 **덧셈**을 통해 정보 만든다

⇨ 경사 소실/폭발 문제 해결!