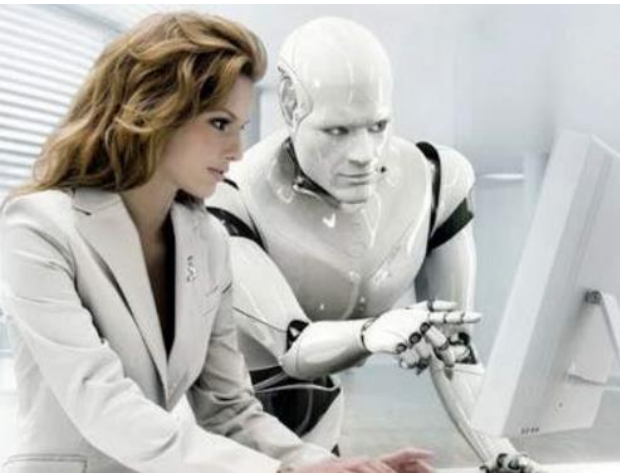


# **DSAC Module4**

## **Deep Learning (4)**

**2019, 2020**

**KPC**



**RNN**

# 순환 신경망 (RNN) – CNN

- 합성곱 신경망(CNN)
  - 2차원(공간) 필터에서 좋은 성과 입증, 이미지 학습 분야에서 널리 사용
- 그러나 CNN은 시간적인 차이를 두고 연속적으로 발생하는 데이터 분석에는 성능이 부족
  - 연속된 단어로 구성되는, 문장 분석, 자연어 처리, 자동 번역, 시계열 데이터 분석 등
- 이런 데이터에는 **순환신경망**(Recurrent Neural Network)이 잘 동작

# 순환 신경망 (RNN) – 텍스트 처리

- 기존 텍스트 분석에서 BOW를 이용
  - 문장 내에 어떤 단어들이 존재하는지는 파악하지만
  - 단어의 배열 정보는 사라지고 이용하지 못함
- 단어의 **순서 정보**도 분석하려면
  - BoW, 단어 벡터의 도입만으로는 부족
  - 단어의 발생 순서 정보를 활용할 수 있는 신경망 모델을 사용해야 한다
  - 이를 위해서 **순환신경망**(RNN)이 널리 사용

# 순환 신경망 (RNN) – 과거 정보 기억

- MLP, CNN에서는 입력 데이터를 한 번 신경망에 통과시키면 데이터가 한 방향으로만 흘러가는 구조
- RNN은 과거의 데이터에서 추출한 정보를 지속적으로 모델 내에서 저장하고 이를 학습에 재사용
  - 사람이 대화를 나누거나 책을 읽을 때 단어를 순차적으로 듣게 되고 최신의 입력 단어들 뿐 아니라 과거의 정보를 계속 내부적으로 사용해야 하는 것과 같은 원리
- 즉, CNN에서는 과거의 기억(상태정보)을 사용하지 않지만 RNN은 과거의 기억을 사용하는 구조를 제공

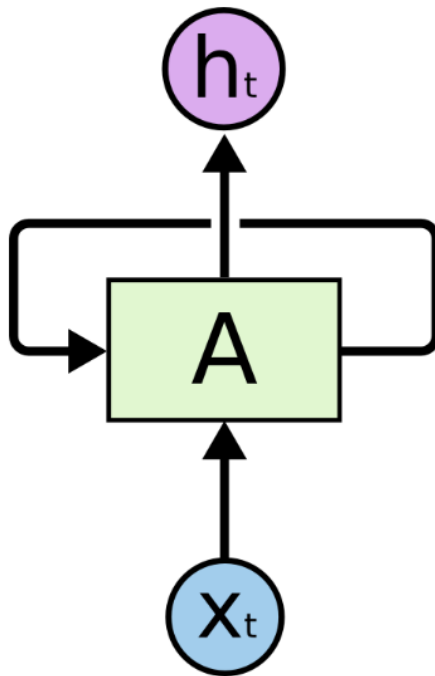
# 순환 신경망 (RNN) – 응용

- 주요 응용
  - 음성인식
  - 텍스트 분석
  - 자연어 처리
  - 챗봇
  - 감성 분석
  - 언어 모델링(language modeling)

등에 널리 사용

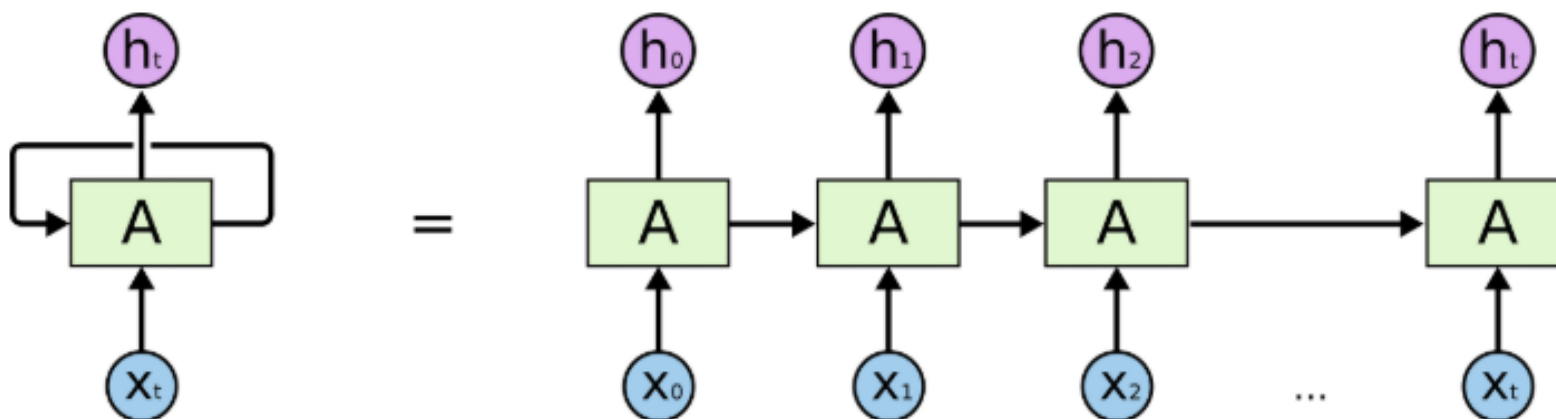
# RNN 기본 구조

- RNN은 loop가 들어 있고
- 과거의 데이터가 미래에 영향을 줄 수 있는 구조



# RNN unrolled

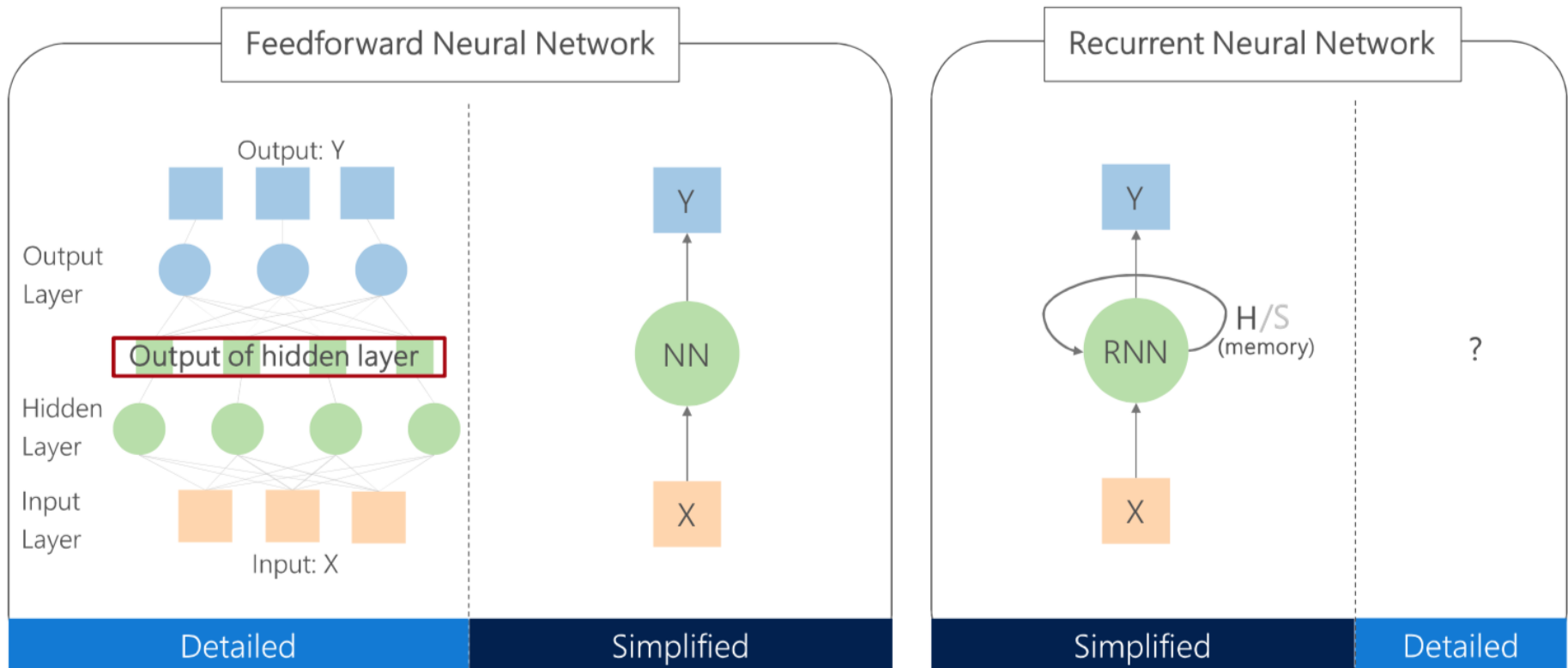
- 좌측
  - 순환신경망 하나의 유닛(셀)을 그린 것
  - 현재의 상태값이 다음번 상태값을 구하는데 순환적으로 사용
- 우측
  - 이해하게 쉽게 시간축으로 나누어 그린 것





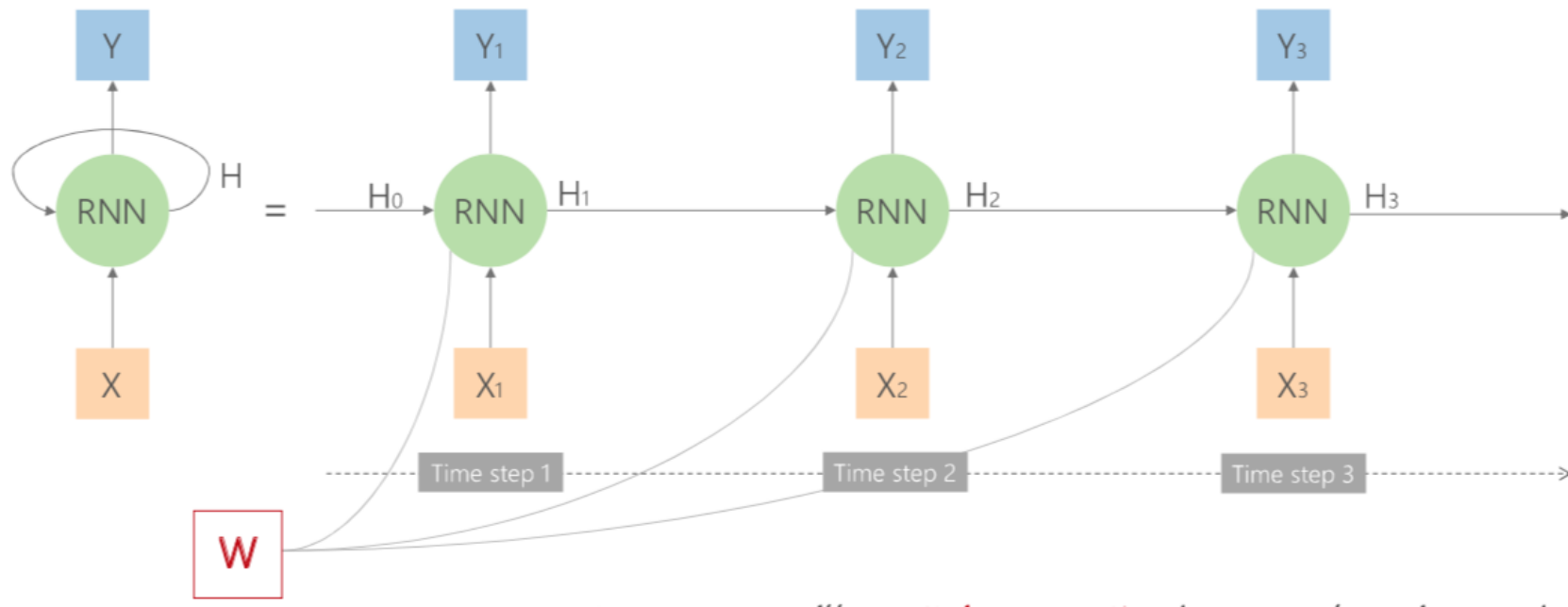
# RNN vs FF 신경망

RNN has internal hidden state which can be fed back to network

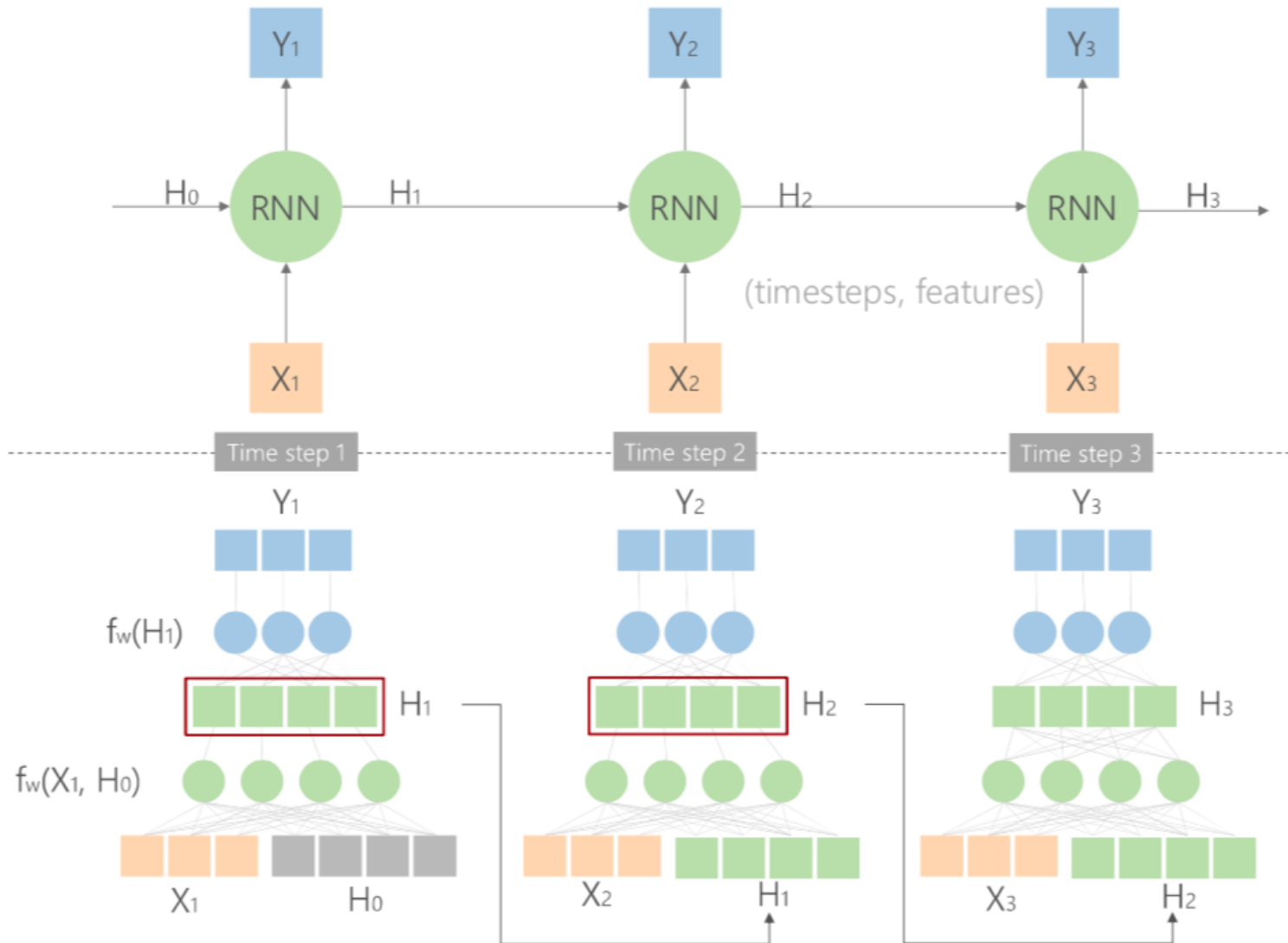


# RNN unrolled

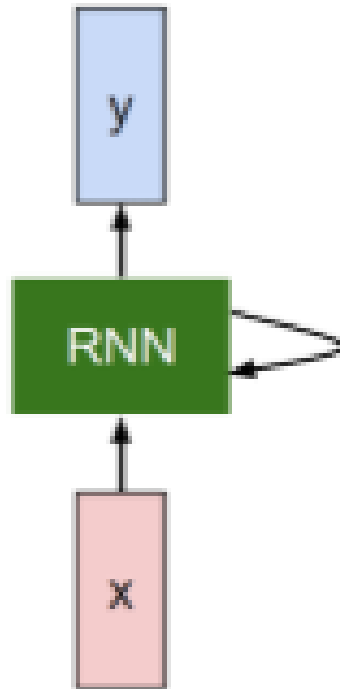
- 매 timesteps에 동일한 가중치를 사용



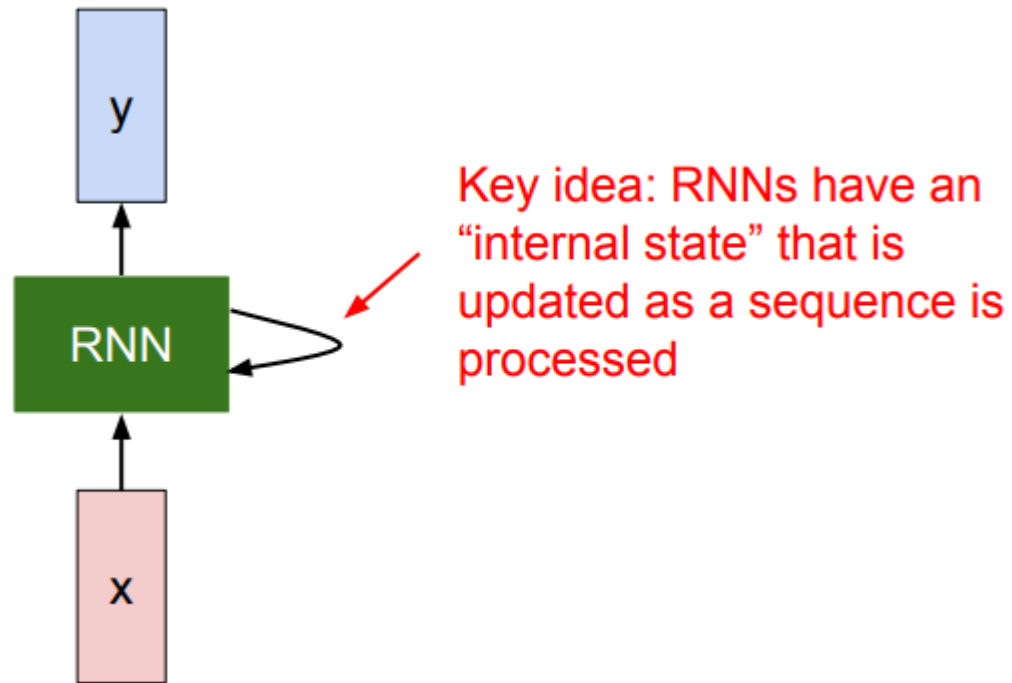
# 상세 동작



# RNN 기본 구조



# RNN 기본 구조 - 내부 상태 보유



# RNN 기본 구조 - 내부 상태 보유

We can process a sequence of vectors  $\mathbf{x}$  by applying a **recurrence formula** at every time step:

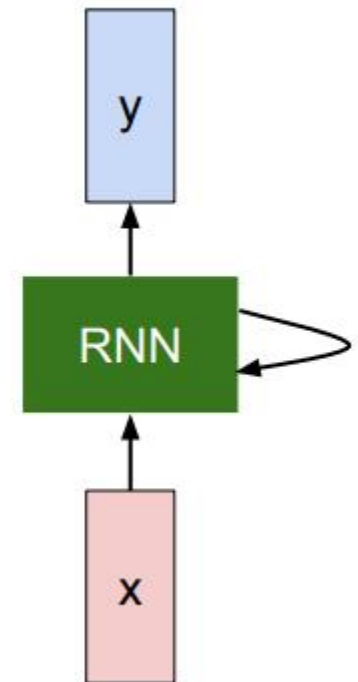
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

some function with parameters  $W$

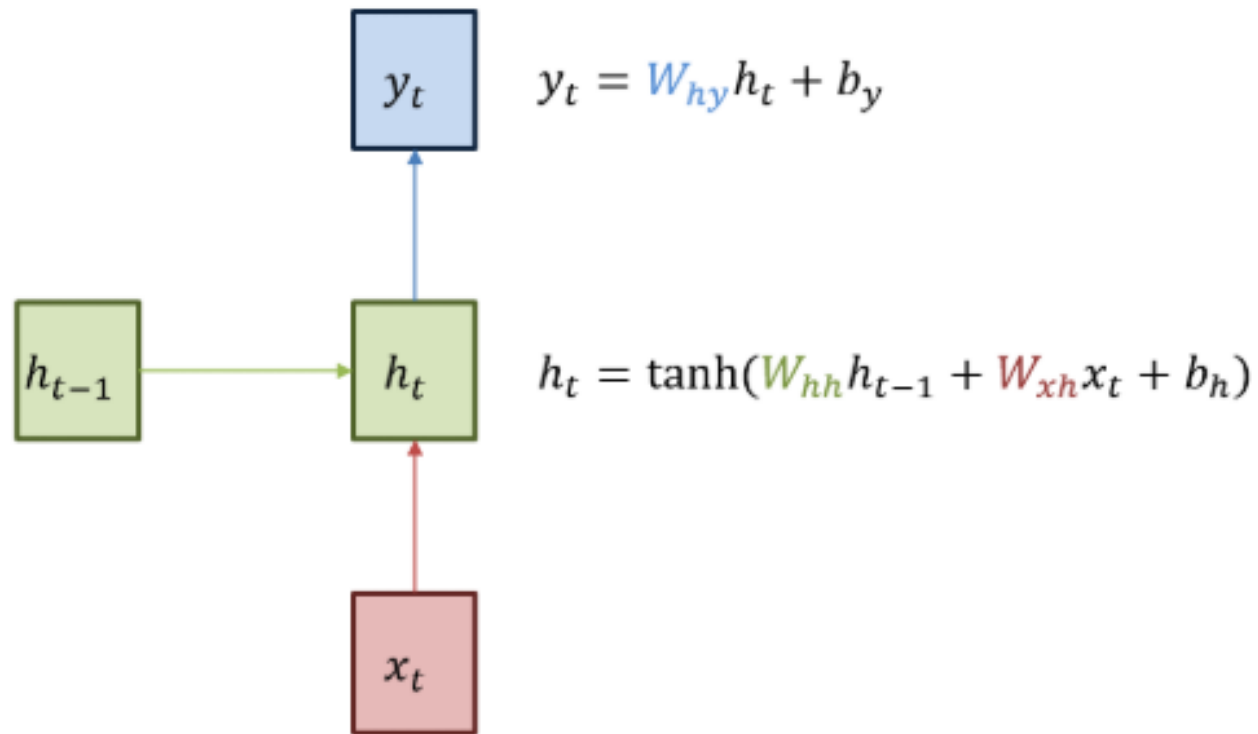
old state

input vector at some time step



# 기본 순환 신경망

- 현재의 출력
  - 현재의 상태값으로부터 구한다



# 기본 순환 신경망 – 과거 정보 사용

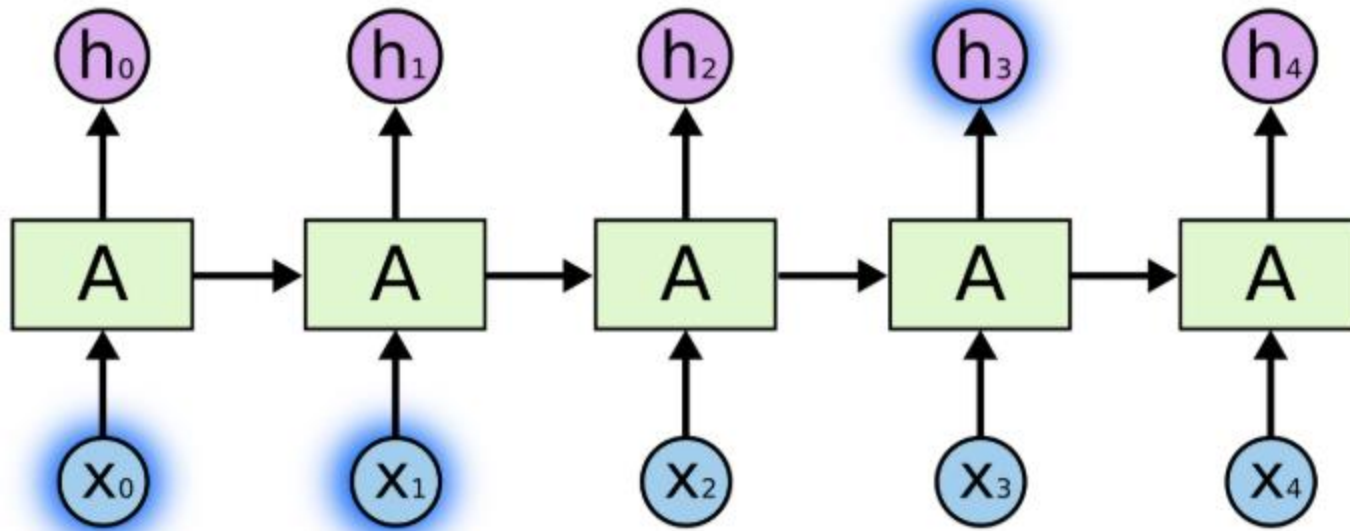
- 모델의 현재 상태값을 얻기 위해서
  - 현재의 입력정보 뿐 아니라 모델의 이전의 상태 정보도 동시에 사용
- 수식으로 표현

$$H_t = UX_t + WH_{t-1}$$

- $X(t)$  : 현재 입력 정보
  - $H(t-1)$  : 이전 상태 정보
  - $U, V$  : 가중치(계수) 매트릭스
- 이로써 과거의 정보를 내부적으로 재사용



# 기본 순환 신경망 - 과거 정보 사용



# 기본 순환 신경망 – 동일한 가중치 사용

- 모든 step에서 **동일한 필터(가중치)**를 사용
  - 계수  $U$ ,  $V$ ,  $W$ 는 시각( $t$ )의 함수가 아니라, 모든 step에서 동일
- 순환신경망을 학습
  - 가중치 매트릭스  $U$ ,  $V$ ,  $W$ 의 값이 최적화되도록 학습하는 것

# 기본 순환 신경망 – 상태, 출력

- 현재 상태 상태 활성화 함수
  - Tanh함수를 주로 사용 (참고로 CNN에서는 ReLU)
  - $W$  : 과거의 상태 정보를 얼마나 많이 활용하는지를 결정

$$H_t = \tanh(WH_{t-1} + UX_t)$$

- 현재의 출력
  - 현재의 상태값으로부터 구한다

$$y_t = VH_t + c$$

# 기본 순환 신경망 – 상태, 출력

- 우리가 다른 사람의 말을 듣고 대답을 한다면,
  - 먼저 머릿속에 상대방의 말을 이해하는 상태를 만들어야 하고(즉,  $H_t$ )
  - 이 상태에 기반하여 출력( $y_t$ )를 만들어내는 것과 유사한 개념
- 입력(X)의 단위
  - 한 단어일 수도 있고
  - 10개의 단어일 수도 있고, 한 문장일 수도
- 현재 출력을 구하는데 이전의 상태 정보를 순환적으로 사용
  - 결국 최초의 상태값을 포함해서 오래된 상태 정보를 조금이라도 활용한다고 볼 수 있다

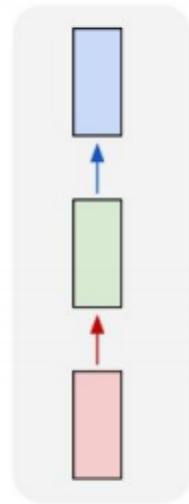
# 기본 순환 신경망 – 기울기 소실 문제

- 과거의 오래된 신호를 **적은 비중**으로 일괄적으로 **반영**하면 신호의 업데이트가 너무 적어져서 **기울기 소실** 문제가 발생
- 과거의 신호를 **너무 많이 반영**하면 신호가 **발산**하는 문제가 발생
- 해결책
  - LSTM(Long-Short Term Memory) 기법이 널리 사용
  - 오랜 기간 중요도를 유지할 정보와 그럴 필요 없이 망각해야 할 신호를 구분하여 따로 처리

# RNN 기본 모델 유형

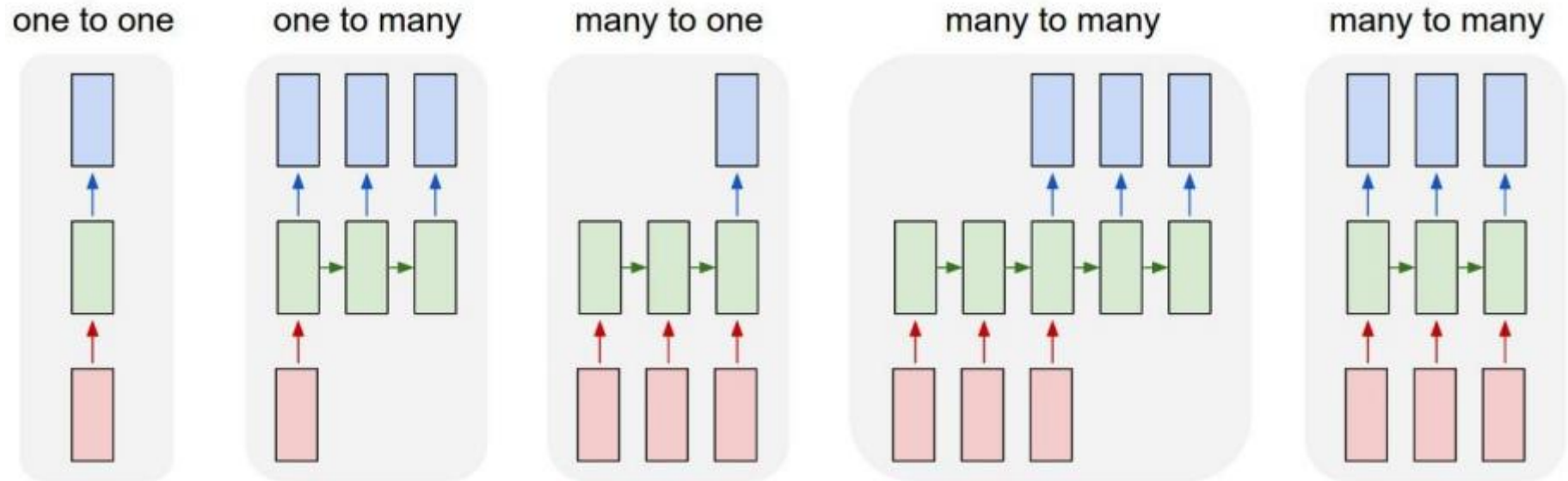
## “Vanilla” Neural Network

one to one



Vanilla Neural Networks

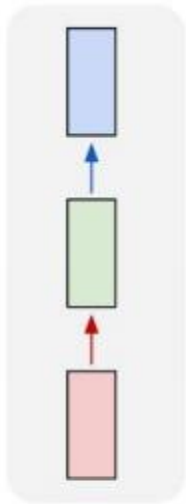
# RNN 기본 모델 유형



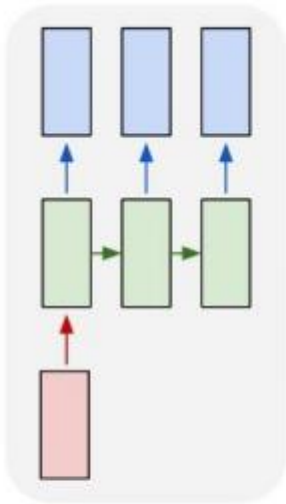
e.g. **Image Captioning**  
image -> sequence of words

# RNN 기본 모델 유형

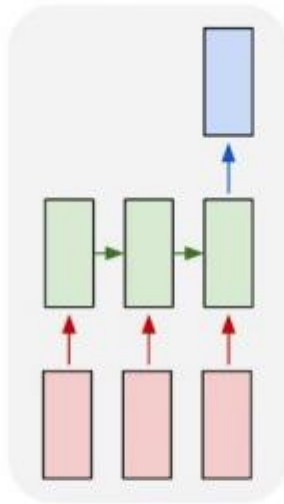
one to one



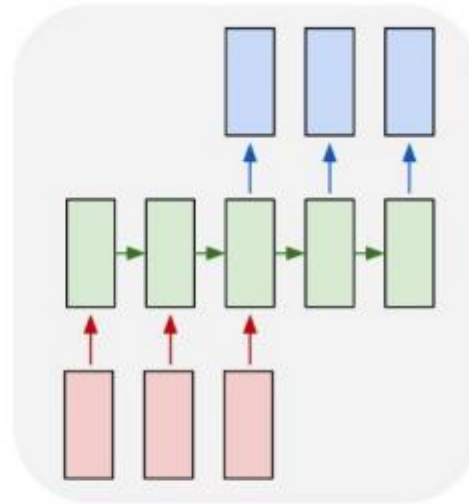
one to many



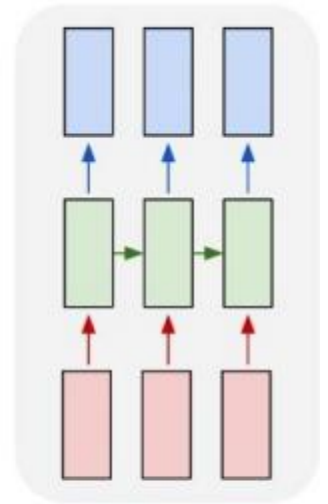
many to one



many to many



many to many



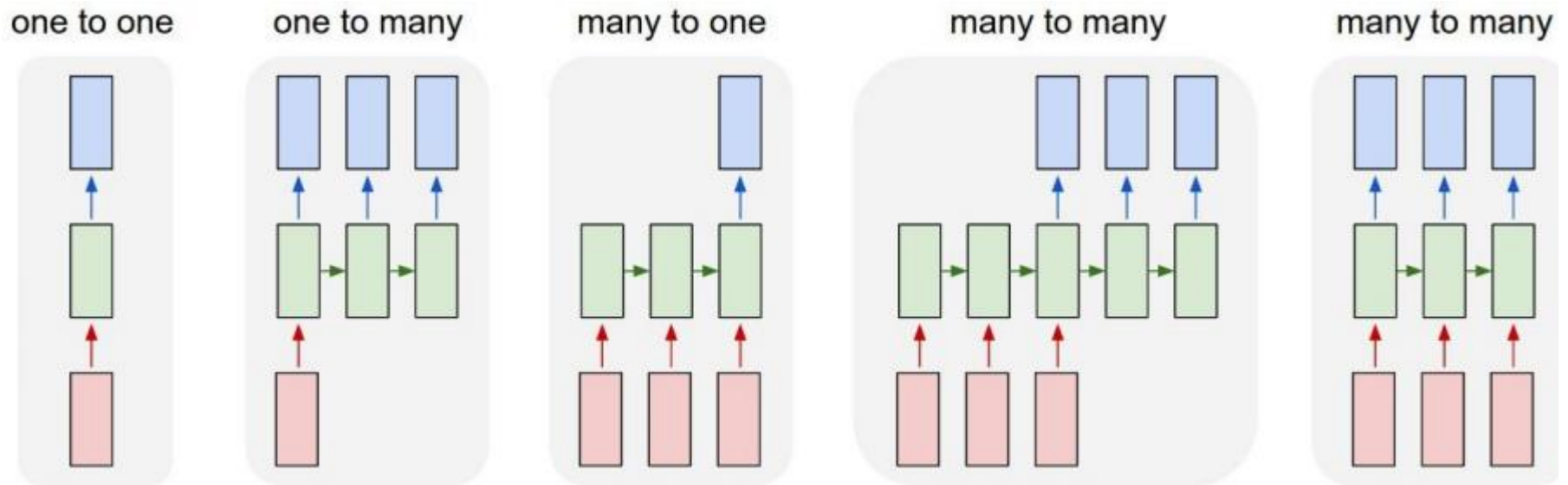
e.g. **Sentiment Classification**  
sequence of words -> sentiment



# RNN 기본 모델 유형

- 언어 모델(language model)

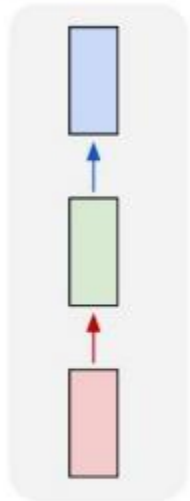
- 일정 크기의 문장을 입력하면 이에 상응하여 한 글자씩 다음 위치에 있는 글자나 단어를 출력
- 매 입력마다 출력이 발생



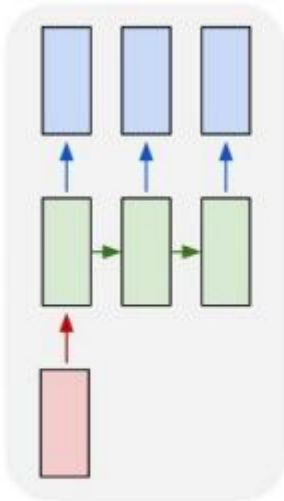
↖ e.g. **Machine Translation**  
seq of words -> seq of words

# RNN 기본 모델 유형

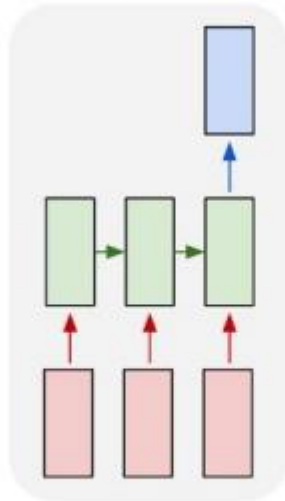
one to one



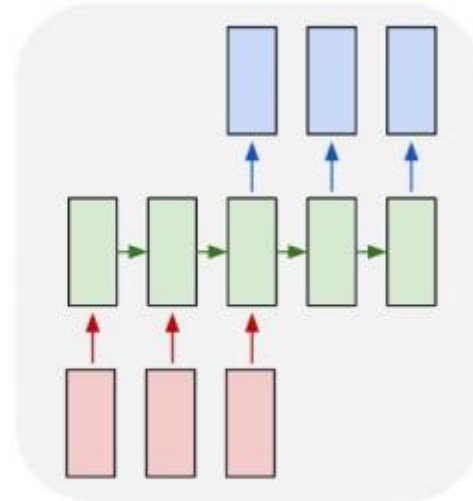
one to many



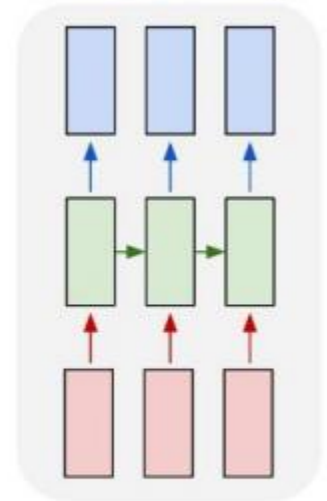
many to one



many to many



many to many



e.g. Video classification on frame level



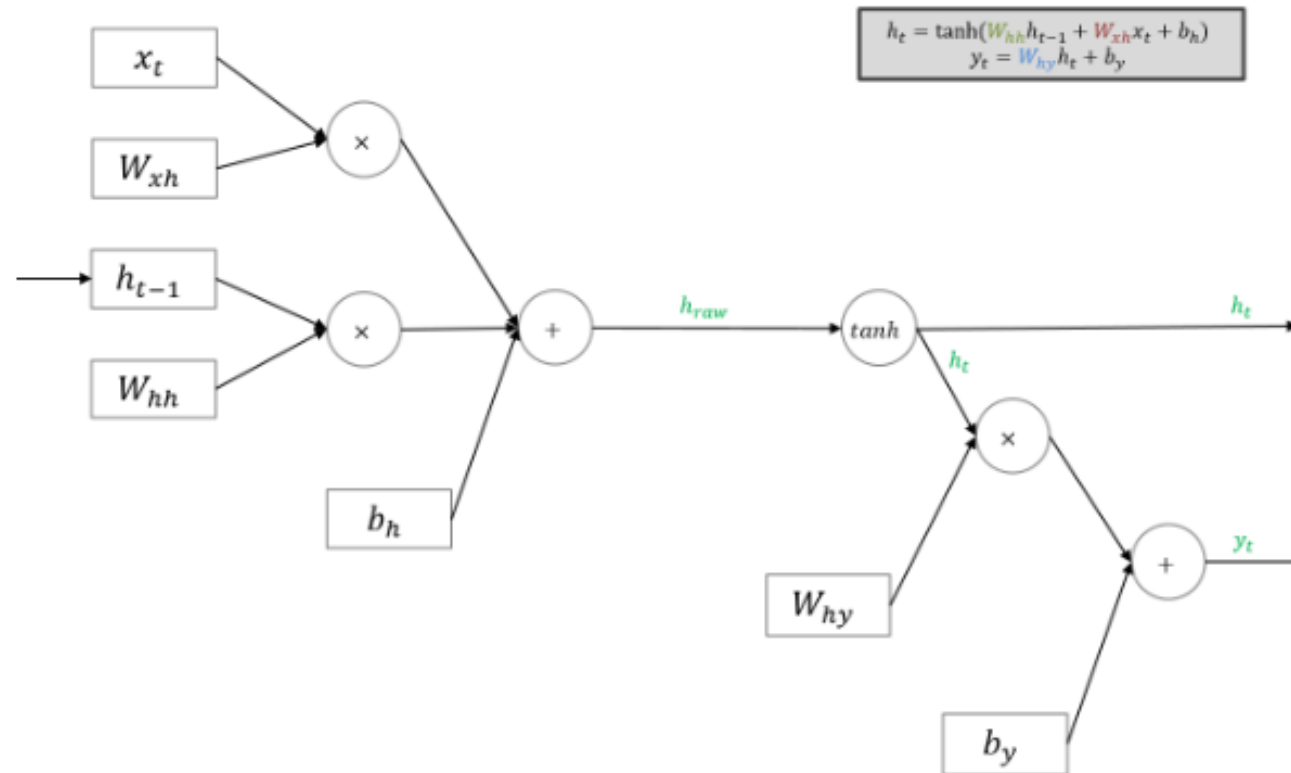
# RNN 모델 – 인코더-디코더 (many to many)

- 인코더-디코더 모델
  - 일종의 시퀀스-시퀀스 모델
  - 하나의 문장을 모두 듣고 이를 번역 또는 다른 대답을 하는 모델 등에 사용
- 시퀀스-시퀀스 모델
  - 하나의 시퀀스를 모두 입력하면 그 이후에 출력이 시퀀스 형태로 나오는 것
- 인코딩 된 결과는 디코더로 전달
  - context vector 또는 machine thought 벡터 형태로 전달
- teacher forcing
  - 예측한 출력이 아니라 실제 레이블 값으로 학습시키는 방법

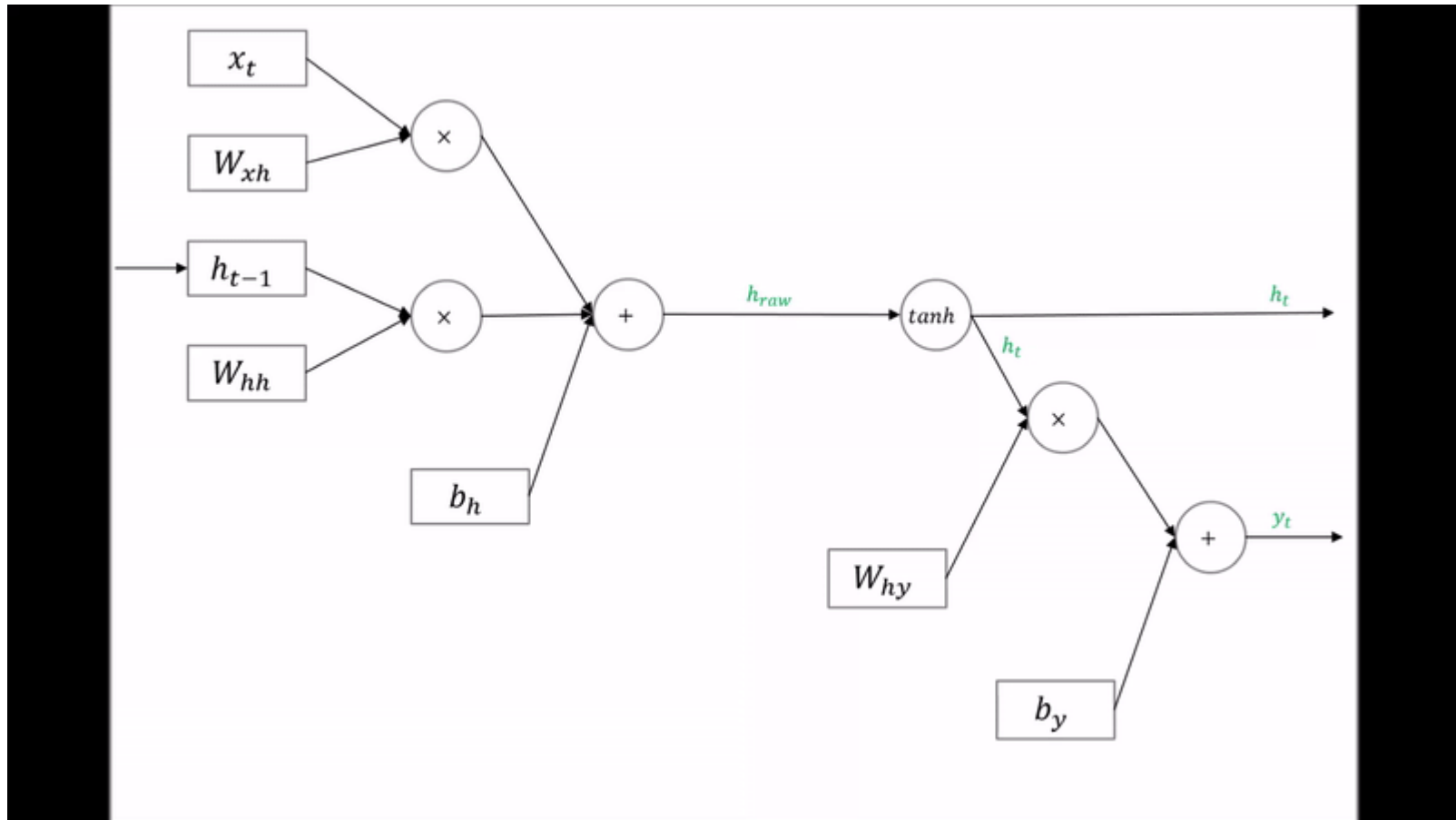
# RNN 학습 – BPTT

- 경사하강(gradient descent) 방식의 학습을 주로 사용
  - 오류역전파 방식을 이용
  - 모델의 출력과 레이블 사이에 오류가 발생하며 이로부터 계산된 손실의 경사값을 앞 단쪽의 계층으로 전달하여 가중치를 조절
- 학습에 의해서 가중치 매트릭스  $W$ ,  $U$ ,  $V$ 가 업데이트 되는데 RNN에서는 모든 타임 스텝에서 **동일한 가중치**를 사용한다는 것을 주의해야 한다.
- 즉, **현재의 경사(gradient) 변화**는 **과거 스텝의 계산에도 영향**을 미친다. 이러한 현상을 **BPTT**(backpropagation through time)이라고 함

# RNN – 순전파



# RNN – 역전파



# RNN의 문제점

- RNN에서 여러 계층을 거치는 경우, 예를 들어 단어가 수십 개로 된 문장을 해석하는 경우, 오차 역전파를 하면서 경사값이 거의 사라지거나 또는 너무 큰 값으로 발산하여 RNN이 제대로 동작하지 못하기 쉽다
  - 오래된 정보를 모두 중요시하면 정보가 너무 많이 축적되어 발산할 우려
  - 오래된 정보를 약하게 반영하면 오래되었지만 중요한 정보를 캐치하지 못하는 즉, 소실되는 우려
- 이를 해결하기 위해서
  - LSTM(Long-Short Term Memory), GRU 기법이 제안
  - LSTM에서는 오랜기간 중요도를 유지할 정보와 그럴 필요 없이 망각해야 할 신호를 구분하여 따로 처리

**LSTM**

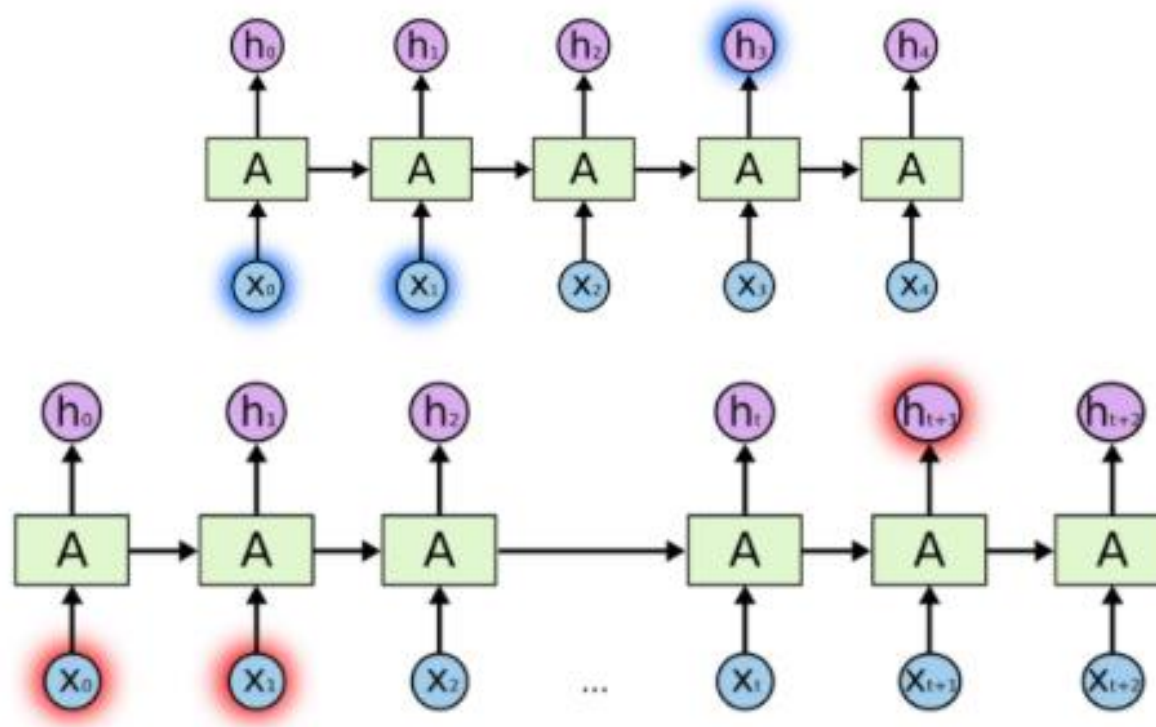


# LSTM – 개념

- 단순 RNN 구조 (케라스의 SimpleRNN)
  - 실제로는 **경사 소실-발산** 등의 문제로 인해 잘 사용하지 않는다
  - 즉, 오래된 정보가 마지막 출력단에서는 매우 약해져서 학습에 사용하기 부족해진다. 따라서 **step 수가 늘면 학습이 잘 되지 않는다.**
- LSTM
  - RNN의 단점을 극복하기 위해서 제안
  - 여러 스텝 앞의 정보를 놓치지 않고 따로 뒷단으로 보내주는 **채널**을 하나 더 **추가**(오래된 정보가 스텝을 지나면서 사라지지 않고 뒤에 영향을 미치도록 하는 것이 목적)
  - 우리가 대화를 할 때에도 바로 최근의 단어들을 듣고 뜻을 파악하지만 오래 전에 한 말을 통해서 전체적인 맥락이나 목적 등을 꾸준히 파악하는 것과 같은 의미
  - 즉, 시퀀스로 입력되는 데이터의 단기(short) 정보와 함께, 오래된(long) 정보를 병행해서 사용하고 학습한다는 의미로 LSTM이라는 이름을 붙임

# LSTM – Vanishing gradient problem

- RNN은 관련 정보와 그 정보를 사용하는 지점 사이의 거리가 멀 경우
  - 역전파 시, gradient 정보가 점차 줄어 들어 **학습능력**이 **현저히 저하**



# LSTM – 개념

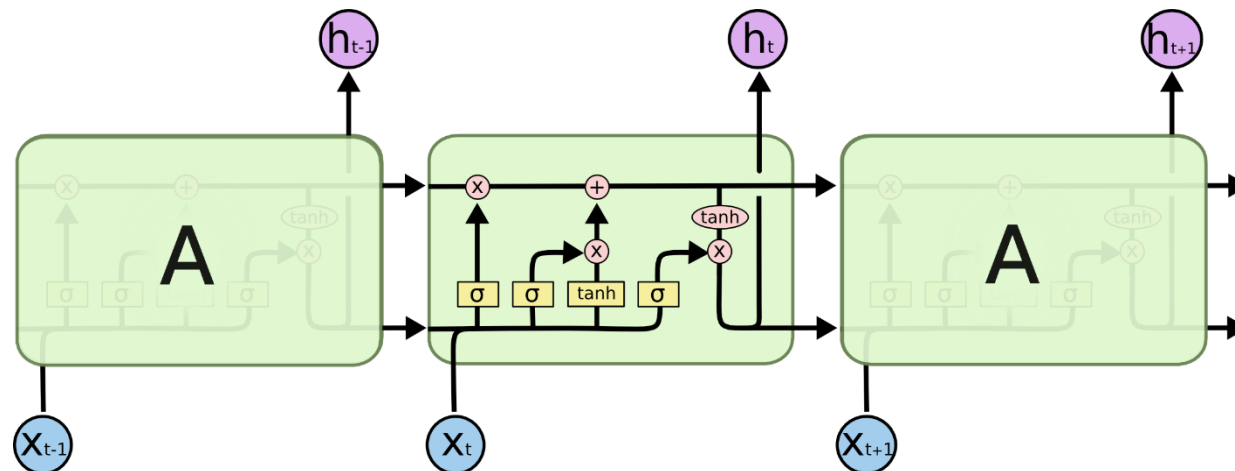
- 각 cell은 장, 단기 2개의 채널 사용
  - 단기적으로 학습한 정보가 전달되는 채널  
(현재의 입력 정보와 상태 정보에서는 필요한 부분을 선택)
  - 장기적으로 살아남아 전달되는 채널  
(과거의 전달 정보에서도 필요한 정보를 필터링하는 작업을 수행)
- LSTM에서 선택해야 할 하이퍼 파라미터
  - 임베딩 차원
  - L출력 차원
- LSTM의 장점
  - 문서 번역
  - 질의 응답(QA)
  - 대화 서비스 (챗봇)등에서 좋은 성능

# LSTM – 개념

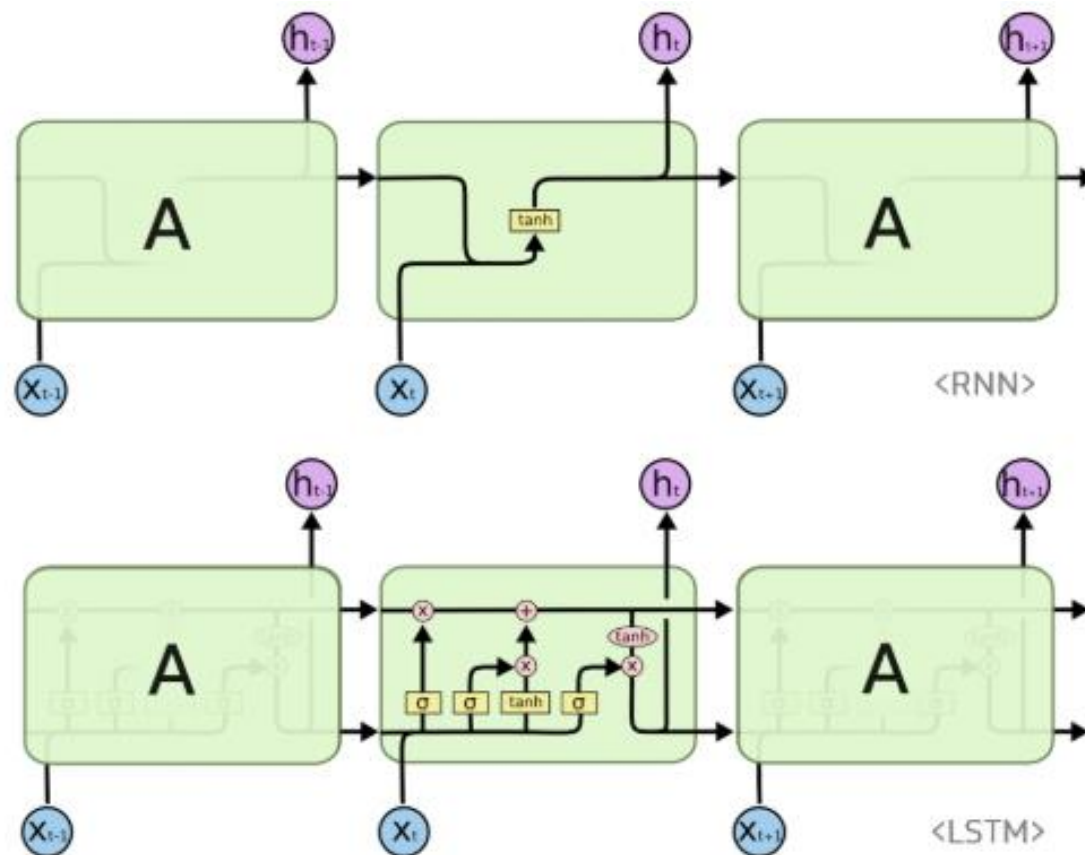
- 내부 히든 정보 외에 **Cell state 정보**를 추가로 전달
  - 선형 자기연결 정보
- 망각(forget), 입력(input), 출력(output) 게이트를 사용
  - 장기적인 상호작용을 학습시키는데 유용
  - 새롭고 관련성이 있는 정보를 선호하여 기억하도록 하고, 관련이 적은 정보를 잊도록 학습

# LSTM – 구조

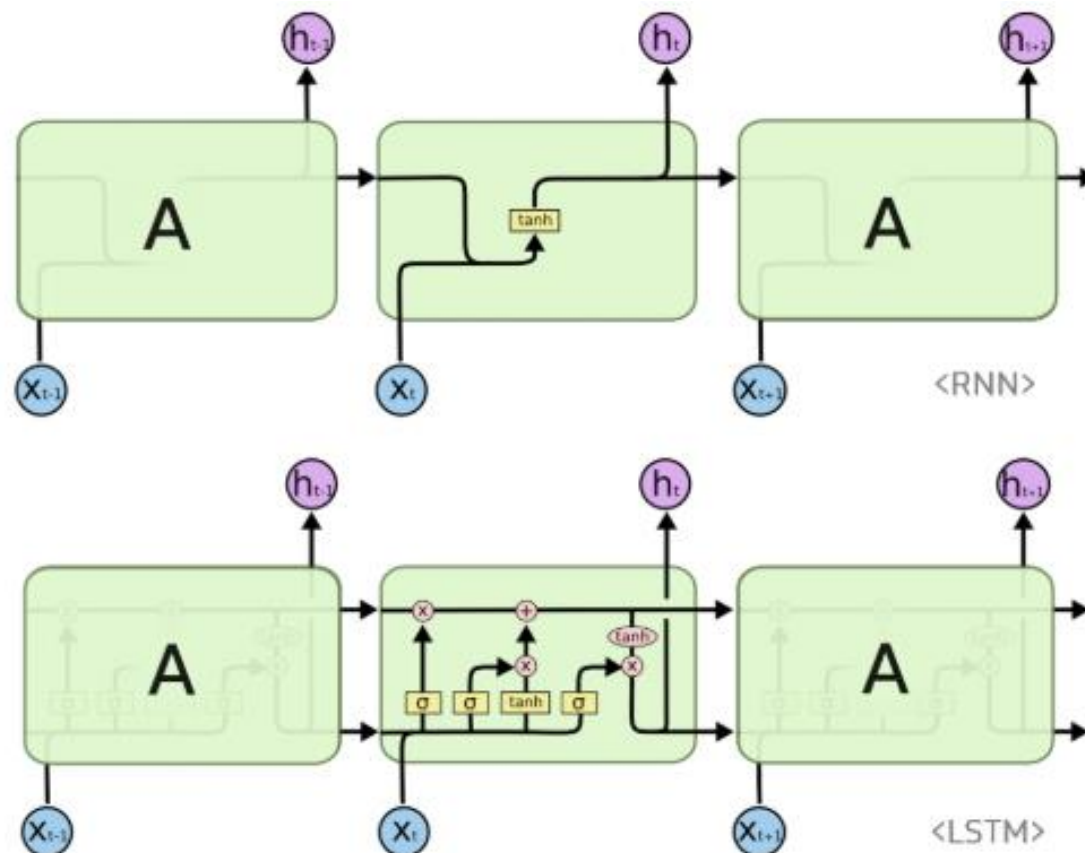
- 오래된 정보를 전달하는(carry) 별도의 채널이 있다
- 각 셀에서는
  - 입력 신호( $x$ ), 이전 단계의 상태 정보( $t$ ), 그리고 이 전달 정보( $c$ ) 세 가지 정보의 가중치 합을 구하고
  - 활성화 함수를 통과하여 출력( $t$ )을 만든다



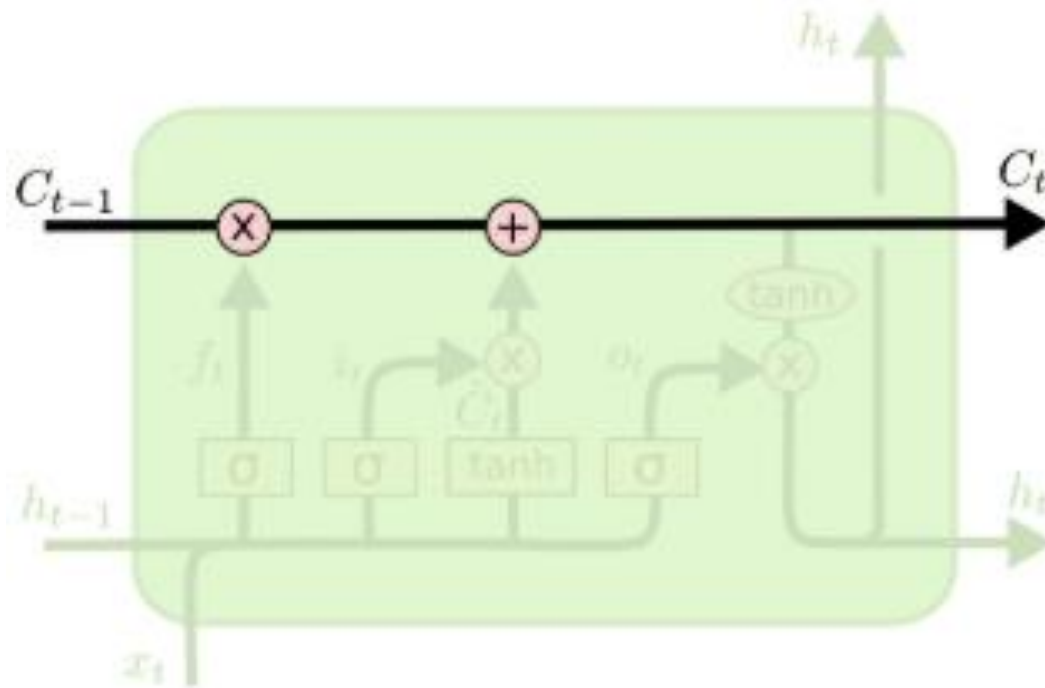
# LSTM – RNN과의 비교



# LSTM – RNN과의 비교



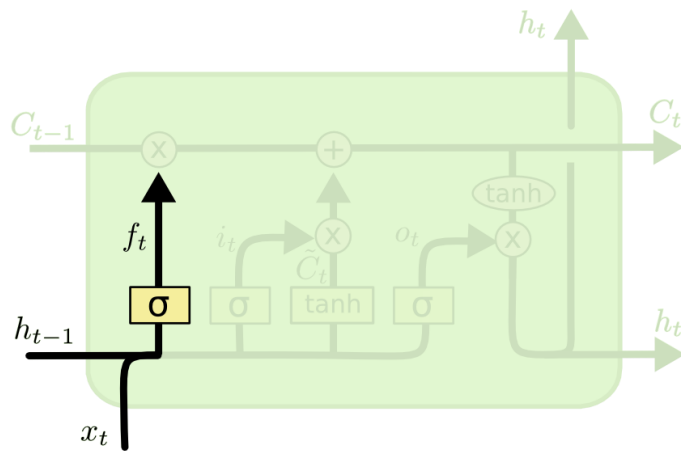
# LSTM – cell state





# LSTM – forget gate

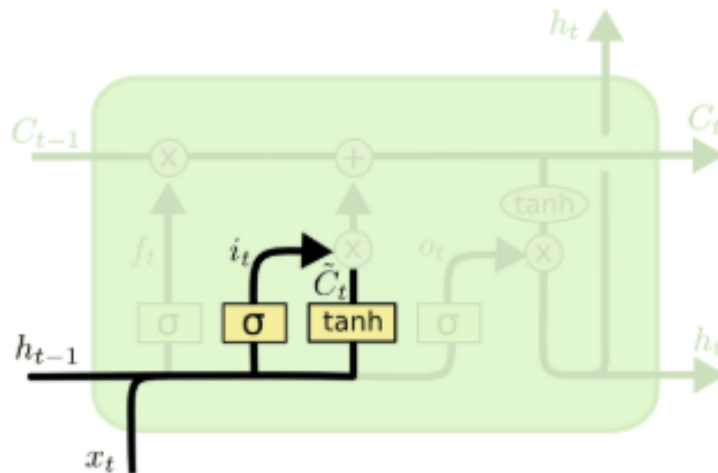
- cell state로부터 어떤 정보를 버릴 것인지를 결정
  - sigmoid layer 사용



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

# LSTM – input gate

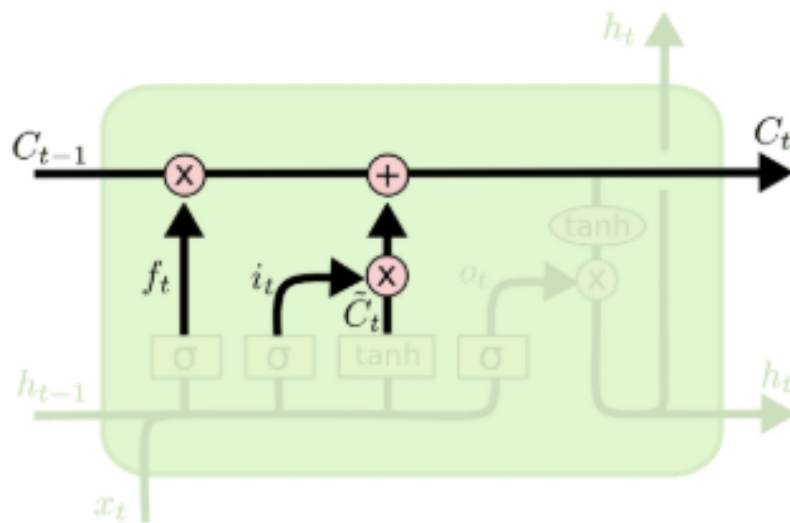
- 앞으로 들어오는 새로운 정보 중 어떤 것을 cell state에 저장할 것인지를 결정



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

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

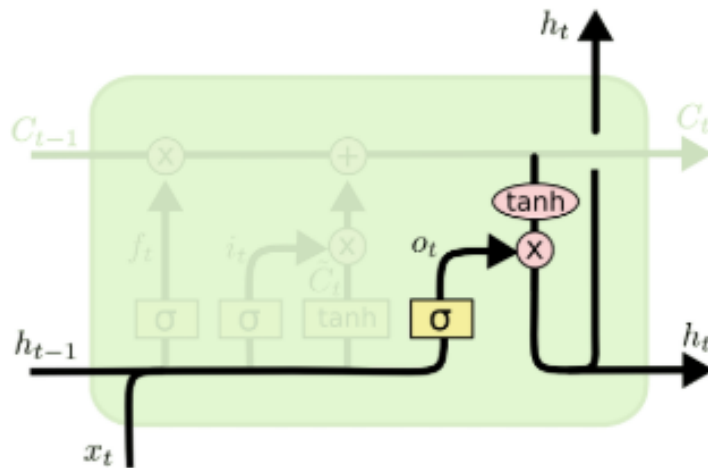
# LSTM – input gate



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

# LSTM – output gate

- 무엇을 output으로 내 보낼 지를 결정

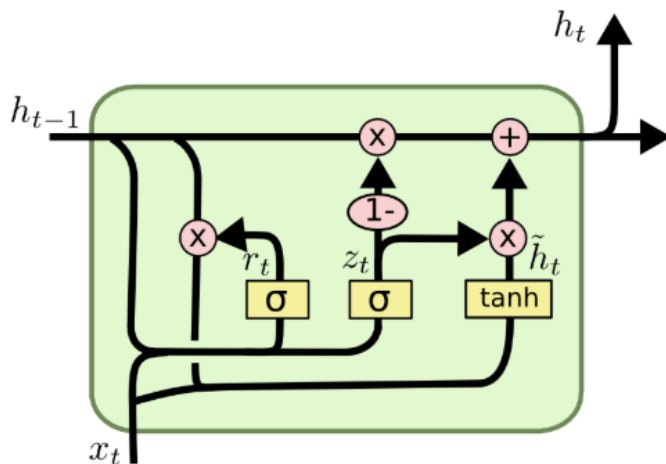


$$o_t = \sigma (W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh (C_t)$$

# GRU - 구조

- LSTM과 같은 기능을 수행, 간단한 구조 (2014)
  - 응용에 따라서 LSTM보다 성능이 우수하기도 하고 떨어지기도 함
- 2개의 게이트 사용
  - 리셋 게이트
  - 업데이트 게이트 : forget과 input 게이트를 합한 기능을 수행



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

# 텍스트 처리 – 원 핫 (One-hot) 인코딩

- 원 핫 인코딩

- 토큰에 고유 번호를 배정
- 모든 고유번호 위치의 한 컬럼만 1, 나머지 컬럼은 0인 벡터로 표시

텍스트: “어제 러시아에 갔다가 러시아 월드컵을 관람했다”

토큰 사전: { “어제” :0, “러시아” :1, “갔다” :2, “월드컵” :3, “관람” :4}

원핫 인코딩:

어제 = [1, 0, 0, 0, 0]

러시아 = [0, 1, 0, 0, 0]

갔다 = [0, 0, 1, 0, 0]

월드컵 = [0, 0, 0, 1, 0]

관람 = [0, 0, 0, 0, 1]

# BOW (Bag of Word, 단어 모음)

- 원핫 인코딩 방식으로 단어(토큰)을 표현하면
  - 단어의 수가 적을 때에는 문제가 안되지만
  - 단어가 모두 10만개이면
    - 모든 단어가 항목이 10만개인 (0과 1로 구성된) 벡터로 표시
  - 주어진 텍스트가 20개의 단어로 구성되어 있다면
    - 20 x 100,000개 크기의 벡터가 필요
- 텍스트 분석은 “문장” 을 단위로 하는 경우가 많다
- 단어 모음(BOW) 방식 : 한 문장을 하나의 벡터로 만드는 방법
  - 한 문장을 단어 사전 크기의 벡터로 표현하고 그 문장에 들어 있는 단어의 컬럼만 1로, 단어가 없는 컬럼은 모두 0으로 표현
- 먼저 단어 사전을 만들고 각 문장에 어떤 단어가 들어 있는지 조사하여 해당 컬럼만 1로, 나머지는 0으로 코딩

- 단어 사전: { “어제” :0, “오늘” :1, “미국” :2, “러시아” :3, “갔다” :4, “축구” :5, “월드컵” :6, “올림픽” :7, “관람” :8, “나는” :9, ..., “중국” :4999 }
- Text\_1: “어제 러시아에 갔다가 러시아 월드컵을 관람했다” 를

## BOW로 표현하면

문장번호	0	1	2	3	4	5	6	7	8	9	10	...	4998	4999
Text_1	1	0	0	1	1	0	1	0	1	0	0	0	0	0
Text_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Text_3	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Text_4	0	0	0	0	0	0	0	1	0	0	0	0	1	0
...														
Text_50	0	0	0	0	0	0	1	0	0	0	0	0	0	0



# 문서-단어 행렬

- 문장 단위로 어떤 단어들이 있는지를 나타내는 BOW의 확장
- 문서(document) 단위로 어떤 단어들이 있는지를 표현
- 같은 단어가 여러번 등장하면 1 이상의 값을 갖는다

문서번호	0	1	2	3	4	5	6	7	8	9	10	...	4998	4999
Doc_1	1	2	3	1	4	0	2	0	1	3	0	0	0	0
Doc_2	0	0	0	0	2	0	0	0	0	0	0	0	2	0
Doc_3	0	0	0	1	0	0	0	0	3	0	0	0	0	1
Doc_4	4	0	0	0	0	0	0	1	0	0	4	0	1	0
...														
Doc_100	0	2	0	0	0	0	1	4	0	1	0	0	0	0

- term frequency-inverse document frequency
- **tf** : 단어가 각 문서에서 발생한 빈도
- **df**(document frequency) : 그 단어가 등장한 ‘문서’의 빈도
- 적은 문서에서 발견될수록 가치 있는 정보
- 많은 문서에 등장하는 단어일수록
  - 일반적인 단어
  - 이러한 공통적인 단어는 tf가 크다고 하여도 비중을 낮추어야 분석이 제대로 이루어질 수 있다.
- 따라서 단어가 특정 문서에만 나타나는 희소성을 반영하기 위해서 **idf(df의 역수)**를 tf에 곱한 값을 tf 대신 사용

# 단어 임베딩의 정의

- 앞에서 소개한 세 가지 텍스트 코딩 방식인 원핫 인코딩, BOW(단어모음), 문서-단어 행렬방식은 단어마다 고유번호를 배정하여 사용
- 그러나 이 고유 번호 숫자에는 아무런 의미가 들어 있지 못하며 단지 인덱스의 성격만 갖는다.
- 단어를 인덱싱이 아니라, 의미 있는 숫자 들의 집합, 즉, 벡터로 표현하는 방법이 단어 임베딩 (Word Embedding)이다.

# 단어 벡터

- 단어 벡터

- 각 단어를 50~300개 정도의 차원으로 구성된 벡터로 표현

학교 = [0.23, 0.58, 0.97, ... , 0.87, 0.95]

바다 = [0.45, 0.37, 0.81, ... , 0.22, 0.64]

- 단어 벡터를 사용하면

- 각 단어들 사이의 “거리” 를 계산이 가능
- 거리를 기반으로 유의어/반대어 등을 찾아낼 수 있다
- 동물의 성별, 단수/복수, 동사/명사를 구분할 수도 있다
- 그러나 각 벡터 값의 의미는 알 수 없다

# 단어 벡터

- 단어 벡터는 **대형 말뭉치로부터 학습**
  - 말뭉치의 문장들을 계속 입력하여 학습을 시키면 단어 벡터를 얻을 수 있다
  - 예를 들어 음식과 관련된 다음과 같은 문장들로 학습을 시키면 다음과 같은 단어 벡터를 얻을 수 있을 것이다.
  - 학습에 사용된 문장 예:

“나는 어제 바나나를 맛있게 먹었다”

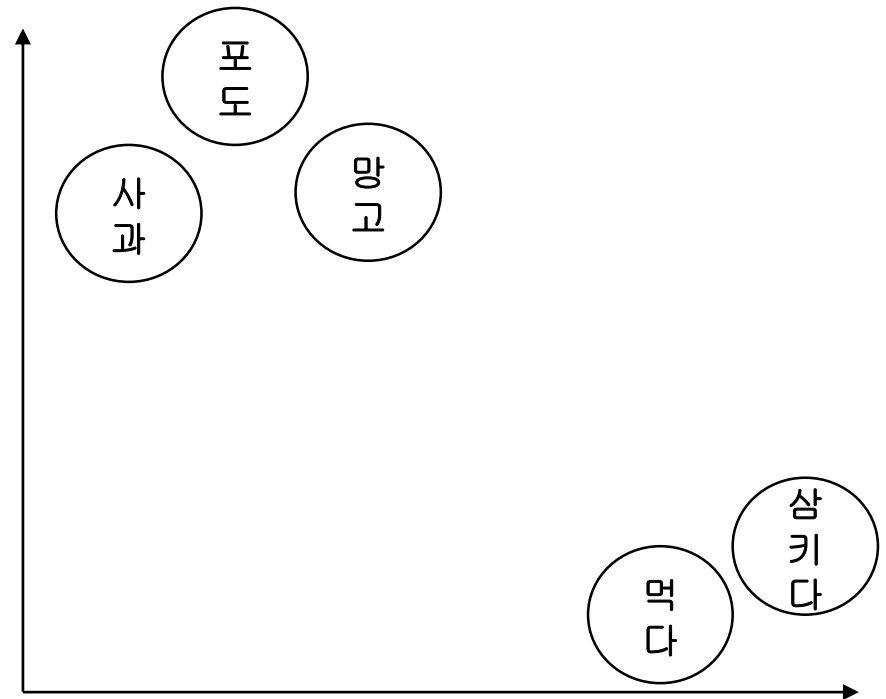
“이 망고는 먹기가 힘들다”

“이 사과는 씹는 맛이 아주 좋다”

“바나나가 사과보다 맛있다”

“잘 씹어야 맛있게 먹을 수 있다”

...



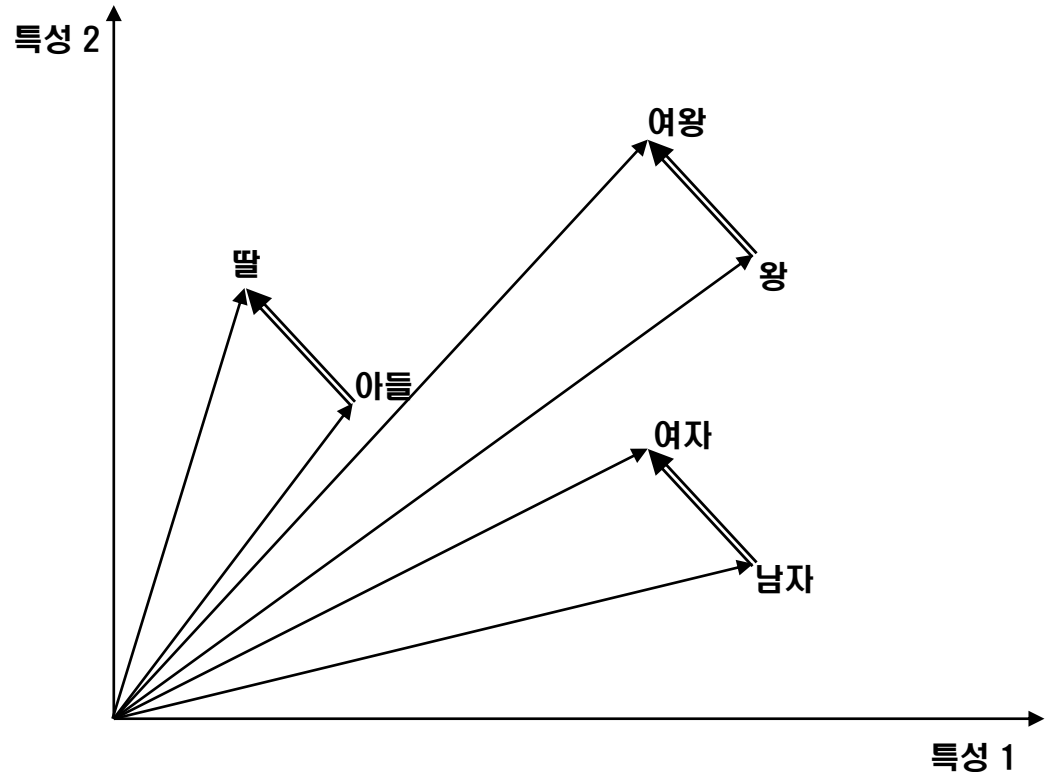
# 단어 벡터

- 이미 만들어져 있는 단어 벡터를 가져다 사용할 수도 있다.
- **glove**
  - 2014년 스탠포드에서 만든 Global Vectors for Word Representations
  - 위키피디아 데이터로부터 학습
  - 40만개 단어를 100차원으로 임베딩
  - [nlp.stanford.edu/projects/glove](http://nlp.stanford.edu/projects/glove) 에서 다운로드

# 단어 벡터

- 단어 벡터를 사용한  $A:B = C: ?$ 의 관계를 만족하는 ?를 찾을 수 있다.
  - 왕 : 여왕 = 아들 : ?  $\rightarrow$  ? 부분 : 딸
  - 이러한 연산은  $(B-A)$  벡터, 즉 (왕 - 여왕) 성분을 구한 후 이를 벡터 C(아들)에 더하면 딸을 구할 수 있다.
  - 이들의 관계는 아래와 같다.

<http://word2vec.kr>



# 단어 벡터 생성

- 단어 벡터 만드는 과정을 소개
  - 가장 널리 사용되는 라이브러리 : **Gensim**
    - pip install gensim

```
from gensim.models.word2vec import Word2Vec  
model = Word2Vec(sentence_list, min_count=1)  
model.most_similar(positive="조선")
```

```
##  
[('일본', 0.9953970909118652),  
 ('관련', 0.9941188097000122),  
 ('인물', 0.9938454031944275),  
 ('러시아', 0.9931197166442871),  
 ('주요', 0.9918481111526489),  
 ('대원군', 0.9915156960487366),  
 ...
```



# 텍스트 처리 - 단어 임베딩

- 문서 단어 행렬
  - 원-핫 인코딩을 사용하여 단어사전 크기의 벡터를 사용
  - 이 벡터는 크기가 매우 크고 희소한 구조
  - 근본적으로 문장 내에서 단어의 존재 여부만 코딩을 함으로써 문장 내에서 단어가 가지고 있는 정보 즉, 문장 구조 정보를 모두 잃어버림
- 단어 임베딩
  - 단어 벡터를 사용하여 단어를 표현하는 방법을 개선
  - 희소 벡터가 아닌 밀집 벡터
  - 단어 벡터를 사용함으로써 id를 구분하는 용도가 아니라 공간상의 점을 매핑함에 따라 단어 사이의 거리를 계산, 단어 간의 관계를 벡터로 표현할 수 있게 됨
  - 자연어 처리, 문서 분류, 감성 분석, 저작 식별, QA, 챗봇 등에서 BoW나 문서 단어 행렬에 비해 높은 성능
  - 텍스트 분석에서 널리 사용

## 3.5 어텐션 기법 - 개념

- CNN과 RNN에서 모두 어텐션 기법이 널리 사용
- 이미지 분석에서는 이미지의 어떤 영역이 이미지 분류에 중요한 역할을 했는지를 파악하는 것이 필요한데, 이 때 사람이 더 집중하여 보는 영역의 개념으로 컴퓨터가 더 중요하게 사용한 부분을 어텐션이라고 부름
- 자연어 처리에서는 긴 문장의 어떤 부분이 문장의 뜻을 파악하는데 더 중요한 역할을 하고 있는지를 파악하는 것을 어텐션 기법으로 처리



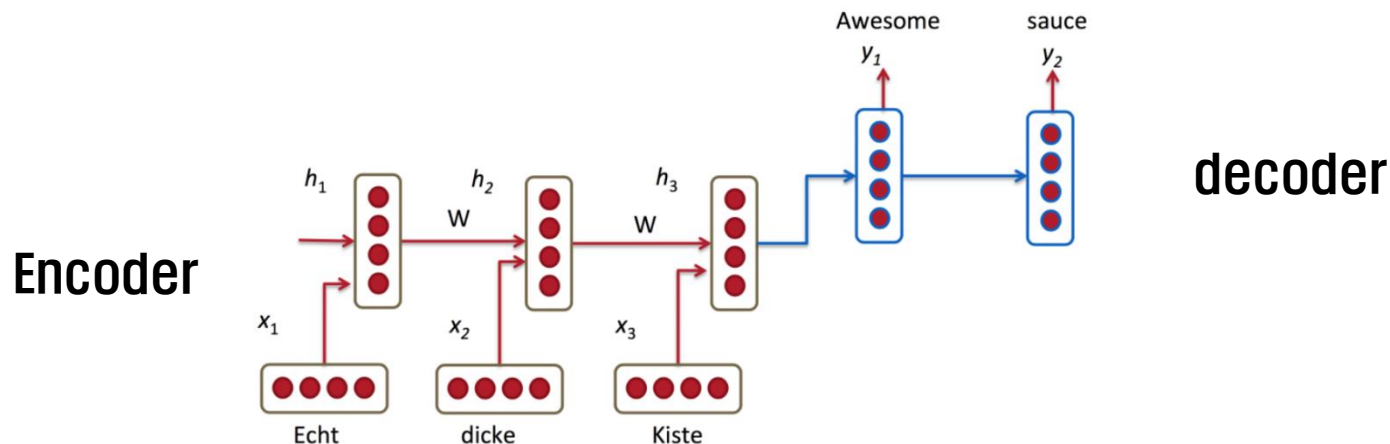
Fig. 1. A Shiba Inu in a men's outfit. The credit of the original photo goes to Instagram @mensweardog.



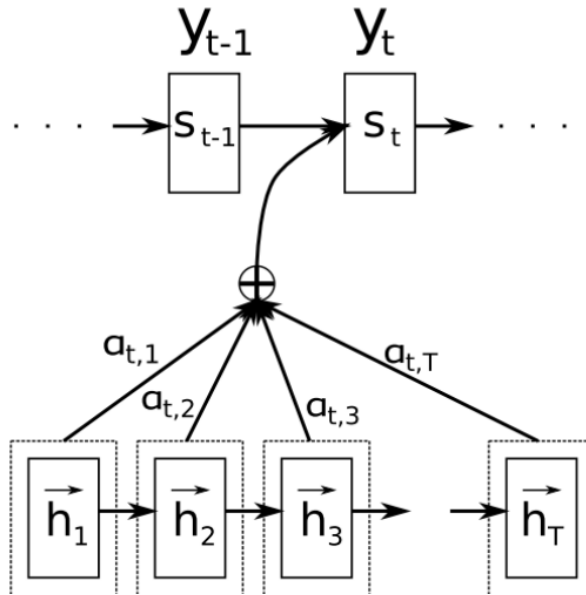
Fig. 2. One word "attends" to other words in the same sentence differently.

# 어텐션 기법 – 응용 예 [자동 번역기]

- RNN의 seq2seq 모델을 사용 가정
  - 디코더는 마지막 단계의 상태값인  $h_3$ 에만 의존해서 번역
  - 정보가 매우 제한적
- 문장이 길어지면 오류의 문제가 커진다
  - 문장 임베딩(sentence embedding)을 수행하여 전체 문장을 하나의 벡터로 만들고 이를 디코더가 사용하는 셈
- 이를 해결하기 위해서 어텐션 기법을 도입
  - 즉, 중요한 정보에 좀 더 집중하도록 하는 방법



# 어텐션 기법 – 응용 예 [자동 번역기]



- 이제 디코더는 하나의 상태값에만 의존하는 것이 아니라, 출력을 얻는 각 스텝마다 입력 신호의 다른 부분에 더 주의를 집중할 수 있도록 허용
- 출력은 이제 마지막 스텝의 값만 이용하는 것이 아니라, 모든 스텝의 입력 값의 가중합(weighted sum)을 이용, 즉 shortcut connection 허용.

# 어텐션 기법 – 학습

- 모델은 어느 부분에 주의를 해야 하는지를, 입출력 데이터를 보고, 학습을 통해서 배운다
- 가중치 계수는 전체 합이 1이 되도록 정규화 한다.
  - 이 계수들을 보면 출력이 어떤 부분의 입력에 주의를 하고 있는지를 알 수 있다
  - 즉, 번역을 하면서 어떤 부분의 단어를 중요하게 활용하는지를 알 수 있다
- 입력과 출력의 차원이 증가하면
  - 어텐션 계수의 수가 이들의 곱에 비례하여 증가
  - 어텐션으로 계산량을 줄이고 중요한 곳에 집중해야 효과를 얻는데 잘못하면 모든 경우의 수를 다 고려하게 되어 계산량이 늘어날 수가 있다.
- 직관적으로는 인간의 주의(어텐션)와 상반되지만 이러한 기법은 현재 좋은 성능을 내고 있고 더 발전할 것으로 예상

# 어텐션 기법 – 응용

- 어텐션 기법은 이미지를 보고 이미지를 설명하는 캡션을 자동으로 생성하는 이미지 캡션에서도 사용(<https://arxiv.org/abs/1502.03044>)
  - 이미지를 인코딩할 때에는 CNN을 사용
  - 캡션을 생성할 때에는 어텐션을 도입한 RNN을 사용
- 어텐션의 종류
  - 글로벌 어텐션 – 전체 step을 모두 사용
  - 로컬 어텐션 : 일부 step만 사용
- align 벡터를 사용하여 인코더와 디코더의 입력을 모두 고려하기도 함
- 셀프 어텐션
  - 순환 구조를 빼고 자신의 어텐션만 사용하는 방법(어텐션의 발전)
  - 성능이 우수하고 RNN을 대체하는 셈

## 3.6 언어 모델링

- BOW(Bag of Words) 는 일종의 특성 추출 방법임. 이는 딥러닝에 사용하기에는 이미 많은 정보를 잃은 상태 – 간단한 머신러닝에서는 아직도 널리 사용되는 기법임.
- RNN 을 이용해서 언어 모델링을 하는 기법 구현.
- 언어모델링이란? 몇 개의 단어가 주어지면 다음에 나올 가장 그럴듯한 (즉, 확률이 높은) 단어를 추천하는 모델
- 자동 번역, 맞춤법 교정, 문장 생성, 저자 스타일 구분 등에 사용

## 4. 시계열 예측



# 시계열 데이터의 특징

- 시계열 데이터
  - 시간 순으로 배열된 데이터
  - 주가 데이터, 센서 데이터, 클릭 스트림 데이터, 음성, 비디오 등이 포함
- 노이즈가 많고 샘플 수(차원)이 많다.
  - 이를 줄이기 위해서, 차원 축소 기술이 필요
  - 웨이블릿 분석, 디지털 필터링 등 사용
- 시계열 데이터 분석의 어려운 점
  - 분석에 효과적인 특성을 선택하는 것이 어려움(시계열 신호처리에는 도메인 전문 지식이 필요)
  - 입력과 출력의 관계가 고정적이지 않고 시간에 따라 종속(같은 입력 시퀀스라도 다른 시간대에 입력되면 다른 결과, 시간 종속성 기간이 정해져 있지 않음)
  - 주파수 도메인으로도 표현(주파수 특성이 분석에 더 나은 경우에는 주파수 도메인에서 분석해야 한다)

# 시계열 데이터의 특징

- 시 불변성

- 시간 이동에 불변인 특성 추출이 중요  
(영상 처리에서는 이미지의 이동, 스케일링, 회전에 영향을 적게 받는 특성 추출이 필요)
- 이러한 특성을 만족하는 것을 stationarity를 보장한다고 함. 즉, 시간이 이동해도 평균, 분산, 자기상관 값 등이 일정한 것을 말함

- 저장과 유량

- 시계열 데이터 분석은 저장(stock)과 유량(flow)로 나누어 분석한다. 일반적으로 유량을 예측하는 경우가 많다
- 신경망을 사용하면 계절적 요인, 트렌드를 신경망이 학습한다는 장점이 있다.
- 데이터 양이 적을 때에는 전통적인 ARIMA 모델이 신경망보다 성능이 우수함

# 응용 분야

- **주식 예측**

- 주가 변동은 랜덤하다고 알려져 있음
- 신경망을 사용하여 뉴스 키워드 등 수많은 요인을 반영하면서 주식 변화 예측도 어느 정도는 가능해질 것으로 예상
- 관련 데이터를 더 많이 사용하면 예측률이 올라갈 수 있으나 현재는 만족스럽지 못한 수준

- **음성인식**

- 화자 인식, 성별 인식, 음성 인식(텍스트 변환), 음향 모델링 등에 사용
- Mel-frequency cepstral coefficients 등의 신호변환 기술이 사용
- HMM도 오랫동안 음성 인식에 사용되었음
- LSTM RNN이 널리 사용

- **공장의 운영 감시**

- 공장의 운영에서 발생하는 센서 데이터를 보고 장비가 정상적으로 동작하는지, 고장의 징후가 있는지 등을 분석하는 요구가 증가 추세
- 기기의 운영 정보 뿐 아니라 공장의 환경 데이터와 제품의 생산에서 생산품의 품질이 유지되는지를 분석하는 것도 필요

- 냄새 센서 데이터

- 음식이 상한 것, 공기 오염, 가스 유출, 박테리아 검출, 질병 진단 등
- 신경망을 로봇에 사용하여 냄새나는 곳을 찾는 동작 수행하기도 함
- 레이블링 작업이 어려워서 비지도 학습 방법이 필요
- 비지도 학습에 의한 특성 추출 기술로 상한 고기 찾기 예

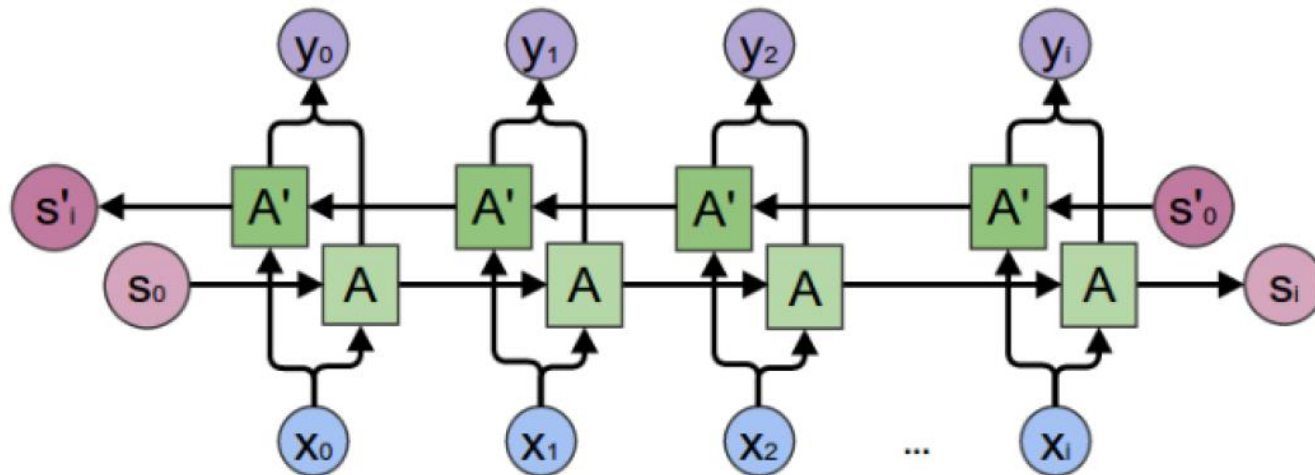
<http://dx.doi.org/10.3390/s130201578>

- 생물학적 데이터 분석

- EEG, ECG, MEG 등 기타 건강 센서 데이터 분석에 필요
- 대개 데이터 분량이 방대, 전문가의 레이블링 작업에 많은 비용 부담
- 비지도 학습에 의한 특성 학습은 주로 비디오 신호에서 연구되었지만 일반 시계열 데이터에서 점차 중요해지고 있음
- 장시간의 변화를 파악하려면 장시간의 데이터가 필요하고 분석이 어려움
- 시계열 데이터는 양이 많아서 전처리된 특성을 사용하기도 함

# 양방향 RNN

- 시퀀스를 양쪽 방향으로 처리하여(즉, 두 번 프로세싱함), 한쪽 방향으로만 볼 때 놓치기 쉬운 패턴을 찾아보는 방법
- 즉, 현재에서 과거로의 추정과 함께 과거로부터 현재로의 추정을 동시에 진행하여 이 중에 좋은 패턴을 활용한다는 개념



# 양방향 RNN

- 이러한 양방향 RNN이 항상 잘 성능을 개선하는 것은 아니나 좋은 성능을 나타내는 경우가 있다. 특히 자연어 처리에서 좋은 성능을 낸다.
- 문장과 같이 단어의 나열의 경우에 반대 방향으로 단어의 순서를 뒤집어서 예측하는 경우도 비슷한 성능을 보임
- 이는 언어에서 반대의 순으로 말을 하여도 컴퓨터를 거의 비슷하게 학습할 수 있다고 볼 수 있다
- 같은 정보를 다른 방법으로 표현하는 것을 활용하여 앙상블을 취하면 더 좋은 성능을 낼 수 있다는 가정에서 양방향 RNN을 도입
- Keras에서는 양방향 RNN을 구축하기 위한 Bidirectional 계층을 지원한다. 그러나 두 배 많은 파라미터를 사용하게 되어 과대적합이 될 가능성도 더 높아진다

**Q&A**