



Deep Learning  
week 6

# 텍스트 시퀀스를 위한 딥러닝



# 목차

---

## 1. 텍스트 데이터 다루기

-원 핫 인코딩, 단어 임베딩 방법들

## 2. 순환 신경망 이해하기

-RNN연산

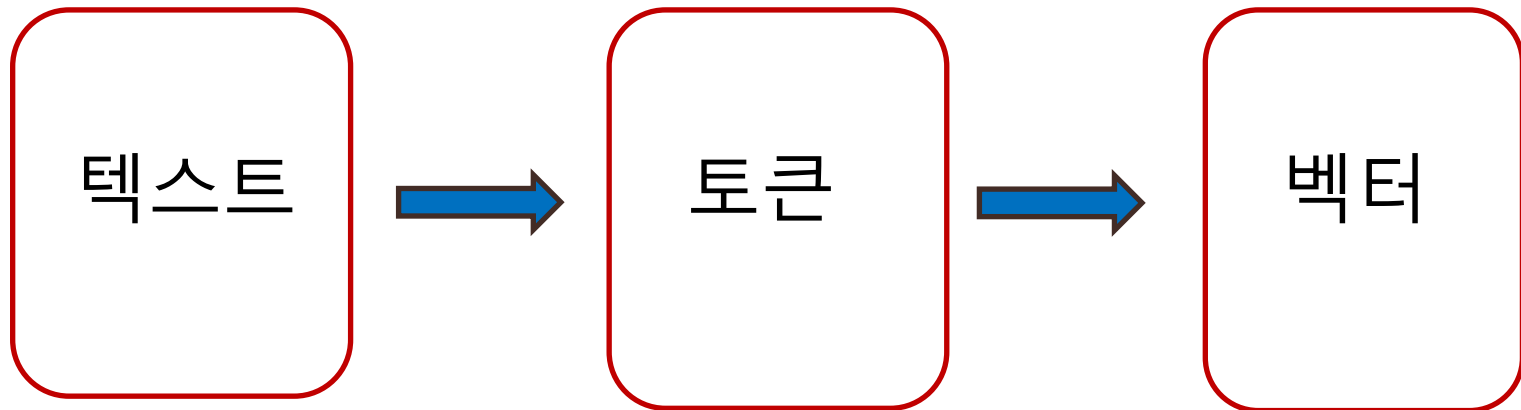
# 시퀀스 데이터 처리: 순환신경망(RNN)과 1D convnet

---

- 문서 분류나 시계열 분류 ex) 글의 주제나 책의 저자 식별하기
- 시계열 비교 ex) 글의 주제나 책의 저자 식별하기
- sequence to sequence 학습 ex) 영어 문장을 프랑스어로 변환하기
- 감성 분석 ex) 트윗이나 영화 리뷰의 긍정, 부정 분류하기
- 시계열 예측 ex) 어떤 지역의 최근 날씨 데이터가 주어졌을 때 향후 날씨 예측하기

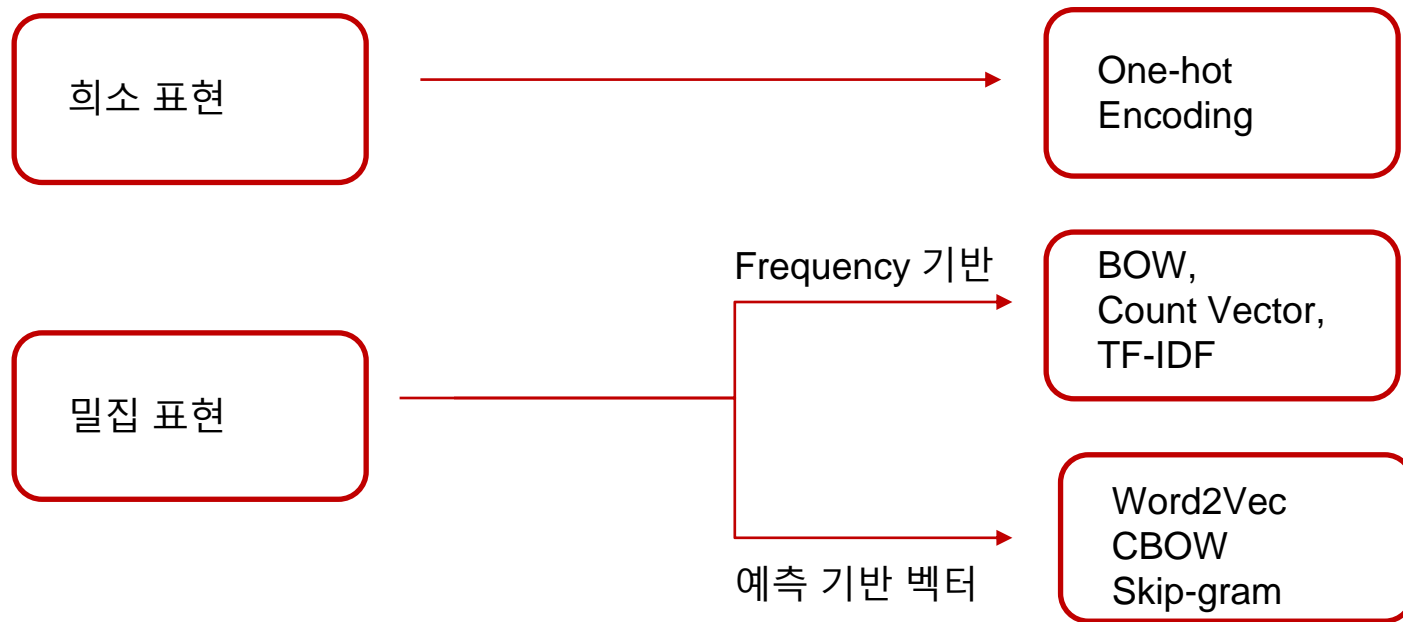
# 텍스트 벡터화

---



# 벡터화 표현 방법

---



# One – Hot Encoding

---

문장의 토큰 하나의 값만 1이고 나머지는 0으로 표현하여 만든 벡터

I have an apple >>> [I, have, an, apple]

Terms	Index
I	[1,0,0,0]
have	[0,1,0,0]
an	[0,0,1,0]
apple	[0,0,0,1]

한계:

1)문장에 단어가 10,000개가 있다면 10,000차원의 벡터를 구해야 한다.>>>공간적 낭비

2)단어의 유사도를 표현하지 못한다.

>>검색 시스템 등에서의 문제점

# BoW(Bag of Words)

---

-단어들의 순서는 고려하지 않고, 출현 빈도에만 집중하는 텍스트 데이터의 벡터화 표현 방법

BoW를 만드는 과정

- 1) 각 단어에 고유한 정수 인덱스를 부여합니다.
- 2) 각 인덱스의 위치에 단어 토큰의 등장 횟수를 기록한 벡터를 만듭니다.

**“정부가 발표하는 물가상승률과 소비자가 느끼는 물가상승률은 다르다.”**

>>('정부': 0, '가': 1, '발표': 2, '하는': 3, '물가상승률': 4, '과': 5, '소비자': 6, '느끼는': 7, '은': 8, '다르다': 9)#각 단어에 대해 인덱스 부여한 결과

>>**bow**

[1, 2, 1, 1, 2, 1, 1, 1, 1, 1]

-단어의 등장 횟수에 따라 문서의 성격을 분류

>>'달리기', '체력', '근력': 체육 관련 문서

>>'미분', '방정식', '부등식': 수학 관련 문서

# Word2Vec

---

CBOW(Continuous Bag of Words) 와 Skip-Gram 두 가지 방식 존재

언어학의 분포 가설에 기반하여, 비슷한 분포를 가진 단어들은 비슷한 의미를 가진다.  
>>같이 등장하는 횟수가 많을 수록 두 단어는 비슷한 의미를 가진다 라는 핵심 아이디어

CBOW에서는 주변 단어를 통해 중심 단어를 예측

Skip-gram은 중심 단어에서 주변 단어를 예측  
>>데이터가 많아지고 있어, 큰 데이터셋에 적합한, 주로 Skip-gram방식 채택

여러 논문에서 성능 비교를 진행하였을 때, 전반적으로, Skip-gram이 CBOW보다 성능이 좋다고 알려짐.



# Word2Vec 우리말 적용 예시

---

한국-서울+도쿄			사랑+이별	
QUERY			QUERY	
+한국/Noun	+도쿄/Noun	-서울/Noun	+사랑/Noun	+이별/Noun
RESULT			RESULT	
일본/Noun			추억/Noun	

# CBOW(Continuous Bag of Words)

---

주변에 있는 단어들을 가지고, 중간에 있는 단어들을 예측 하는 방법

**예문 ” The fat cat sat on the mat ”**

{"The", "fat", "cat", "on", "the", "mat"}으로부터 sat을 예측

예측하는 단어: 중심 단어(center word), 예측에 사용되는 단어 : 주변 단어(context word)

중심 단어를 예측 하기 위해 , 앞 뒤로 몇 개의 단어를 볼지에 대한 범위: 윈도우

윈도우의 크기가  $n$  이라면, 실제 중심 단어를 예측하기 위해 참고하는 주변 단어의 개수:  $2n$

# 슬라이딩 윈도우 (window 크기:2)

중심 단어      주변 단어

↓      ↓

The fat cat sat on the mat

The fat cat sat on the mat

The fat cat sat on the mat

The fat cat sat on the mat

The fat cat sat on the mat

The fat cat sat on the mat

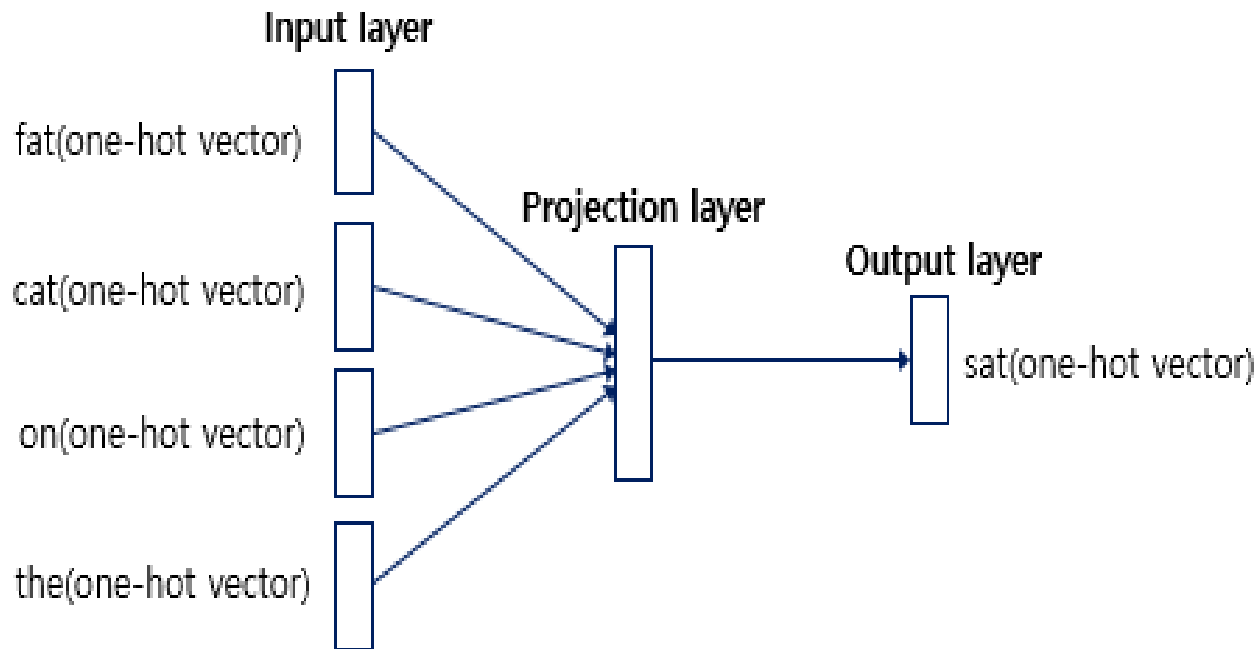
The fat cat sat on the mat

중심 단어	주변 단어
[1, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0]
[0, 1, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0]
[0, 0, 1, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0]
[0, 0, 0, 1, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 1, 0]
[0, 0, 0, 0, 1, 0, 0]	[0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 0, 1, 0], [0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 1, 0]	[0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 0, 1]	[0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 1, 0]

윈도우의 크기를 정한 뒤  
윈도우를 계속 움직여서 주변 단어와 중심 단어 선택을 바꿔가며 학습을 위한 데이터 셋을 만드는 방법

# CBOW의 인공 신경망

---



# GloVe

---

카운트 기반의 LSA와 예측 기반 Word2Vec의 문제점을 개선하기 위해 탄생

LSA는 단어간 유사도를 측정하기 어려움

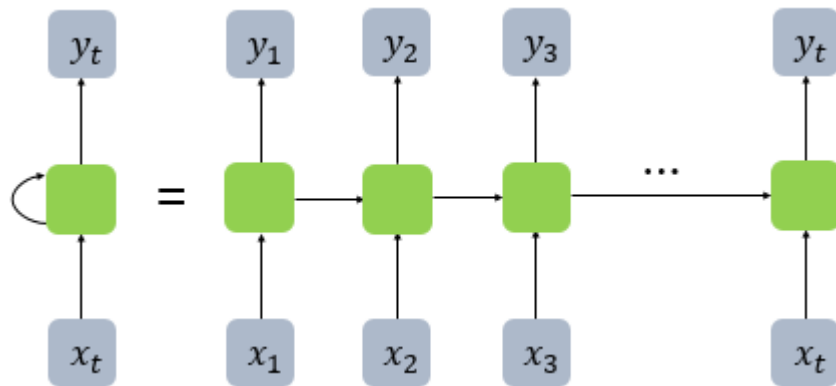
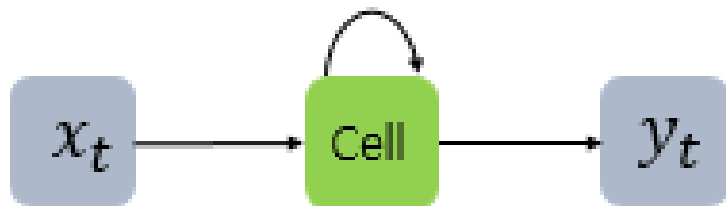
Word2Vec는 문서 전체의 단어 정보가 반영되기 어려움

>>윈도우 기반동시 등장 행렬을 도입하여 연산함

Word2Vec 와 GloVe 중 어느 것이 더 낫다고 말할 수 없음.

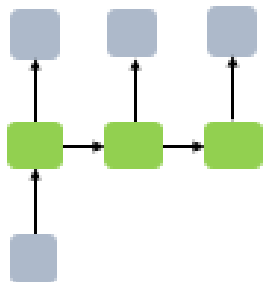
# RNN(Recurrent Neural Network, 순환신경망)

---

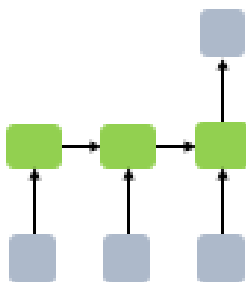


# RNN(Recurrent Neural Network, 순환신경망)

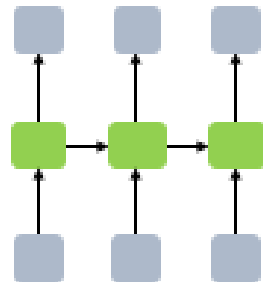
---



일 대 다(one-to-many)



다 대 일(many-to-one)



다 대 다(many-to-many)

RNN은 입력과 출력의 길이를 다르게 설계 할 수 있으므로, 다양한 용도로 사용 가능

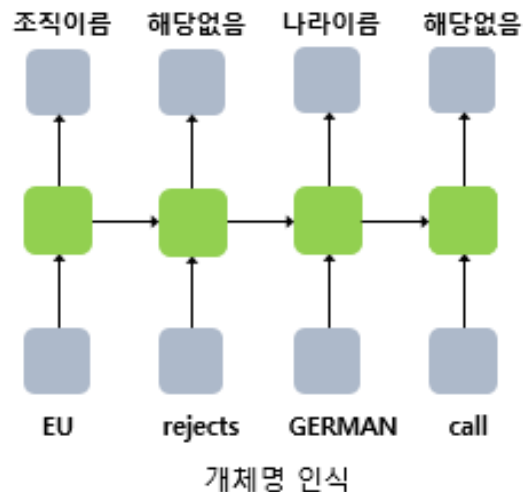
입력과 출력의 길이에 따라서 달라지는 RNN의 다양한 형태

# RNN(Recurrent Neural Network, 순환신경망)

스팸 메일 분류



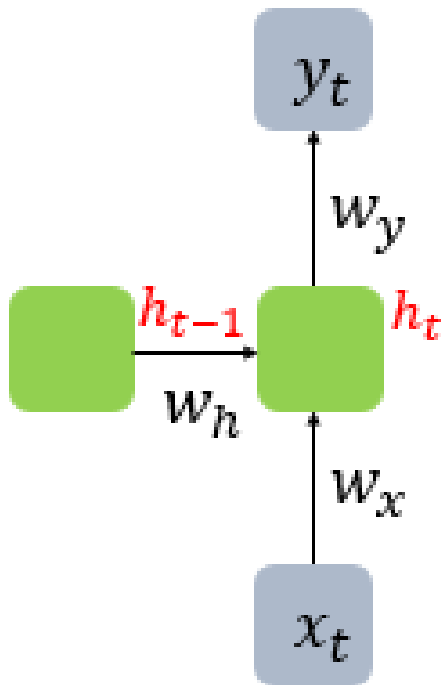
개체명 인식





# RNN(Recurrent Neural Network, 순환신경망)

---



은닉층 :  $h_t = \tanh(W_x x_t + W_h h_{t-1} + b)$

출력층 :  $y_t = f(W_y h_t + b)$

단,  $f$ 는 비선형 활성화 함수 중 하나.

# RNN의 벡터와 행렬 연산

배치 크기가 1이고,  $d$ 와  $D_h$  두 값 모두를 4로 가정하였을 때, RNN의 은닉층 연산

$d$ : 단어 벡터의 차원

$D_h$ : 은닉 상태의 크기

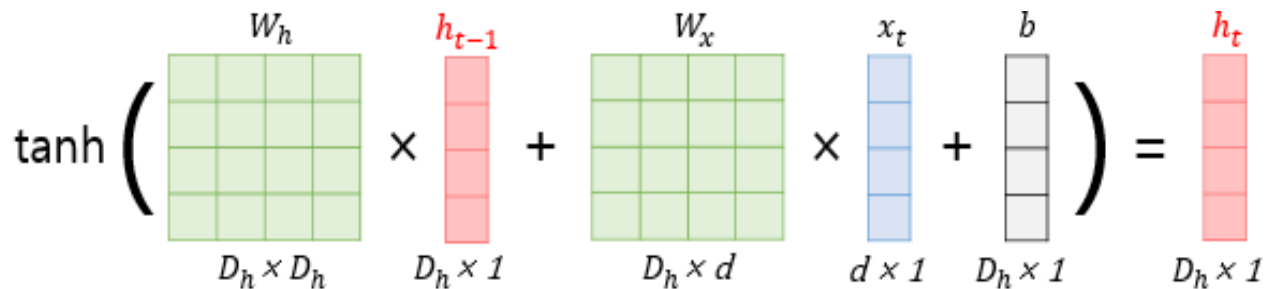
$x_t : (d \times 1)$

$W_x : (D_h \times d)$

$W_h : (D_h \times D_h)$

$h_{t-1} : (D_h \times 1)$

$b : (D_h \times 1)$



The diagram illustrates the hidden layer calculation of an RNN. It shows the following components and their dimensions:

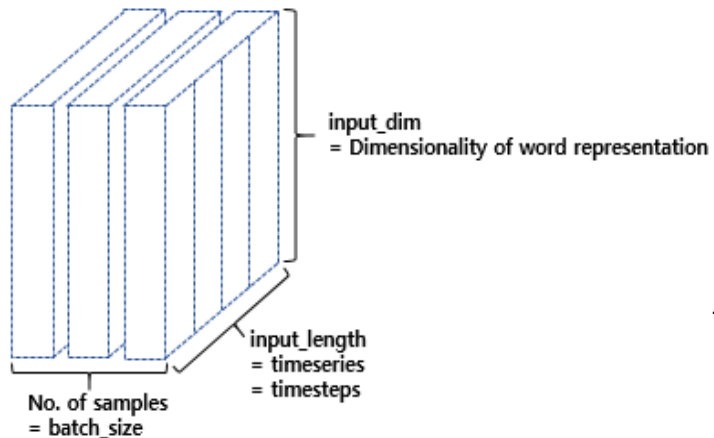
- $W_h$ : A green 4x4 matrix with dimension  $D_h \times D_h$ .
- $h_{t-1}$ : A red 4x1 vector with dimension  $D_h \times 1$ .
- $W_x$ : A green 4x4 matrix with dimension  $D_h \times d$ .
- $x_t$ : A blue 4x1 vector with dimension  $d \times 1$ .
- $b$ : A grey 4x1 vector with dimension  $D_h \times 1$ .
- $h_t$ : A red 4x1 vector with dimension  $D_h \times 1$ .

The calculation is represented by the equation:

$$\tanh \left( W_h \times h_{t-1} + W_x \times x_t + b \right) = h_t$$

# 케라스로 RNN구현하기

```
# RNN 층을 추가하는 코드. model.add(SimpleRNN(hidden_size))
# 추가 인자를 사용할 때
model.add(SimpleRNN(hidden_size, input_shape=(timesteps, input_dim)))
# 다른 표기
model.add(SimpleRNN(hidden_size, input_length=M, input_dim=N))
# 단, M과 N은 정수
```



hidden\_size = 은닉 상태의 크기를 정의. 메모리 셀이 다음 시점의 메모리 셀과 출력층으로 보내는 값의 크기(output\_dim)와도 동일.

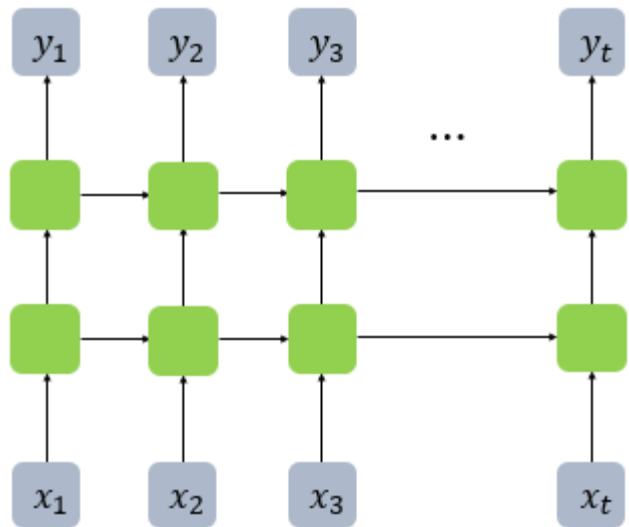
RNN의 용량(capacity)을 늘린다고 보면 되며, 중소형 모델의 경우 보통 128, 256, 512, 1024 값

timesteps = 입력 시퀀스의 길이(input\_length)라고 표현하기도 함. 시점의 수.

input\_dim = 입력의 크기.

# 깊은 순환 신경망(Deep Recurrent Neural Network)

---



RNN도 다수의 은닉층을 가질 수 있음

기존 RNN에 은닉층이 1개 더 추가되어  
2개의 깊은 순환신경망 형성

# 양방향 순환 신경망(Bidirectional RNN)

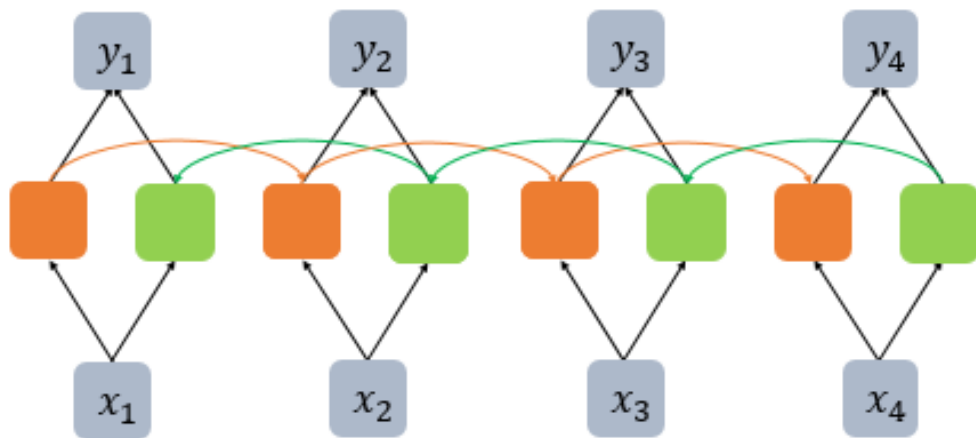
---

Exercise **is** very effective at [     ] belly fat.

- 1) reducing
- 2) increasing
- 3) multiplying

과거 시점의 데이터와 향후 시점의 데이터도 활용하여 정답을 예측 할 수 있다는 아이디어에 기반

# 양방향 순환 신경망(Bidirectional RNN)

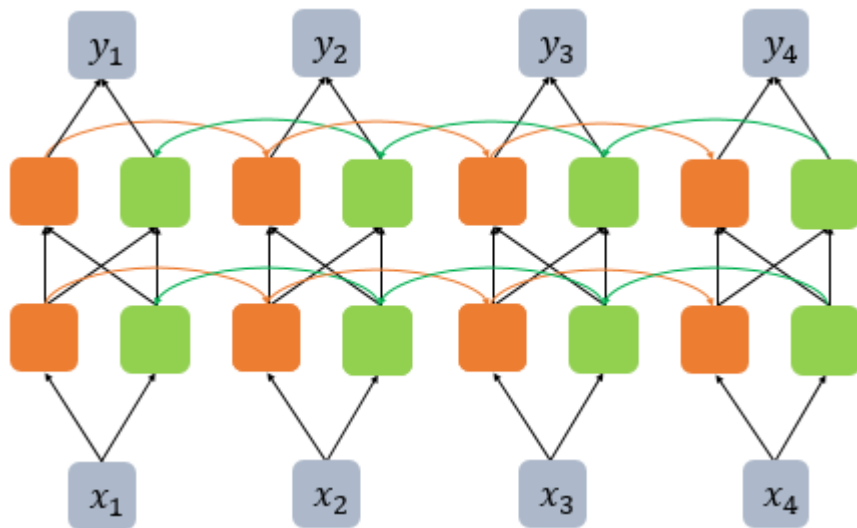


하나의 출력값을 예측하기 위해  
기본적으로 두개의 메모리 셀을 사용

주황색 메모리 셀:  
앞 시점의 은닉 상태를 전달 받아  
현재의 은닉상태 계산

초록색 메모리 셀:  
앞 시점이 아니라 뒤 시점의  
은닉상태를 전달 받아 현재의 은닉상태  
계산

# 깊은 양방향 순환 신경망(Bidirectional RNN)



기본 은닉층 1개에서 1개의 은닉층이 더 추가된 깊은 양방향 순환 신경망

은닉층이 많아진다고 해서 모델의 성능이 좋아지는 것은 아님

은닉층이 많아지면 학습할 수 있는 양이 많아지지만, 반대로 훈련 데이터 또한 그만큼 많이 필요하게 됨.

# Reference

---

<https://wikidocs.net/22650>

자연어처리(NLP)-컴퓨터가 자연어를 이해하는 방법(벡터화)

<https://khann.tistory.com/28>

한국어 Word2Vec적용

<https://word2vec.kr/>



---

**감사합니다.**