

CH18. RNN

(Recurrent Neural Network)

SMARCLE Winter Study [RNN team]

17 강신힌

17 싯도싡

19 송혜원

19 오승싡

목차

1. What is RNN ?
2. RNN 활용한 예제 학습1
 - 로이터 뉴스 카테고리 분류
3. RNN 활용한 예제 학습2
 - 영화 리뷰 분류
4. RNN + α



TABLE OF CONTENTS



01

Lore
ante
Sene
Done



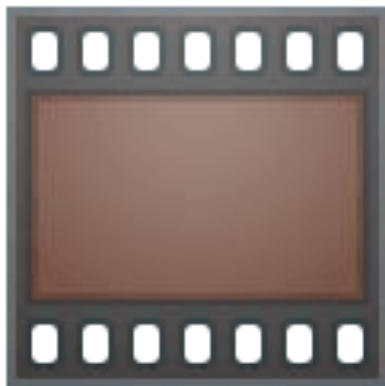
04

Lore
ante
Sene
Done



1. What is RNN?

이제까지와는 조금 다른 데이터, Sequence data



텍스트 데이터 & 시계열 데이터
ex) 말, 영화, 음악, 주가



순서가 존재하는 데이터 !

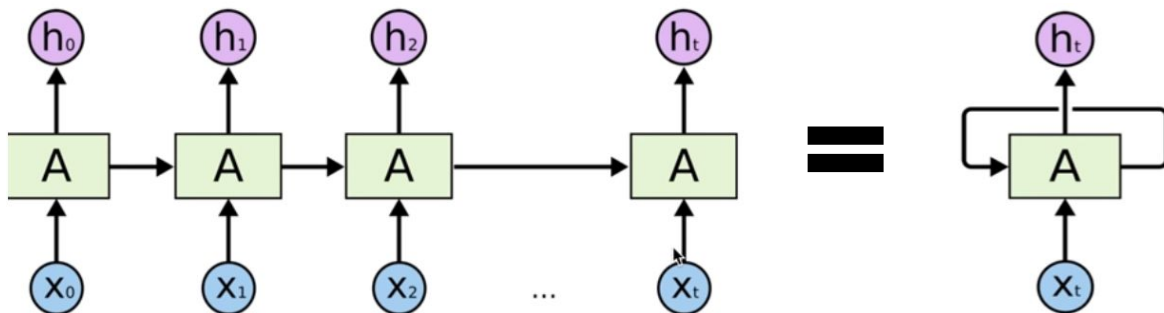
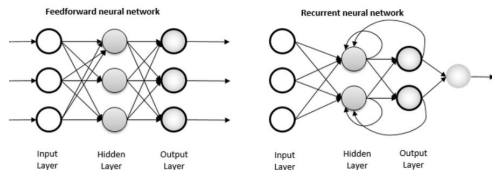
기존 Neural Network로는 이런 데이터를 처리할 수 없음 $\pi \sim \pi$

RNN이란? : Recurrent Neural Network

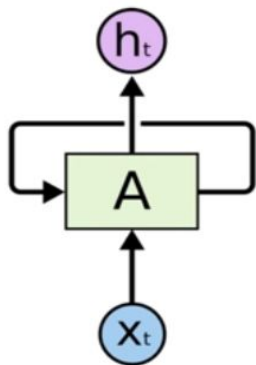
과거에 입력된 데이터 & 나중에 입력된 데이터 사이의 관계를 고려

여러 개의 데이터가 입력되었을 때 앞서 입력 받은 데이터를 잠시 기억

기억된 데이터가 얼마나 중헌지 판단하여 가중치를 주고 다음 데이터로 넘어가는 방식 !



계산 수식을 슬쩍 살펴보자



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

new state / old state input vector at some time step

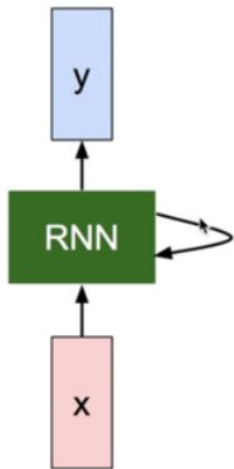
some function with parameters W

※주의※

$h(t)$ 는 output인 y 가 아님 !

$h(t)$ 는 새로운 상태값이자 다음 스텝에서의 입력으로 사용되는 값

그렇다면 y는?



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



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

$$y_t = W_{hy}h_t$$

fw 함수가 모든 상태에서 tanh로 같기 때문에 RNN이라고 하나로 표시 가능~
h(t)에 가중치 W_{hy} 를 곱한 값이 최종 $y(t)$ 가 된다 ‘-’

RNN의 여러가지 구조

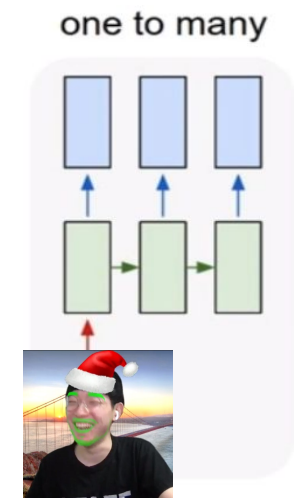


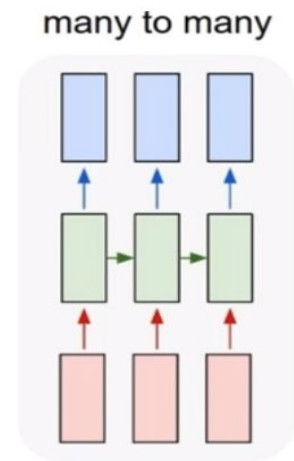
Image Captioning



Sentiment Classification



Translation



Video Frame

RNN의 문제점

RNN의 activation function : \tanh

이전 타임 스텝들의 정보를 다음 타임스텝으로 계속해서 전달 -> **gradient vanishing**

Sequence가 너무 길면 앞 쪽 정보가 뒤 쪽까지 충분히 전달 X : **long term dependencies**

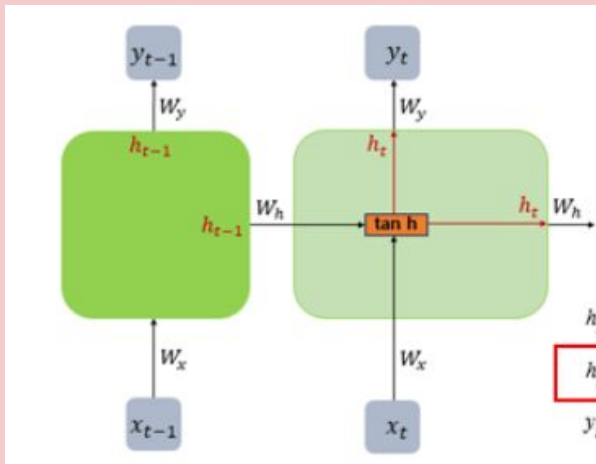
LSTM, GRU 모델 🤖!!

'오늘'에 대한 결과

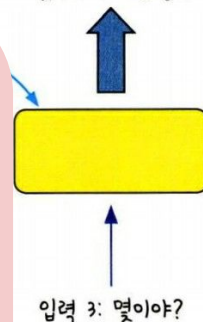
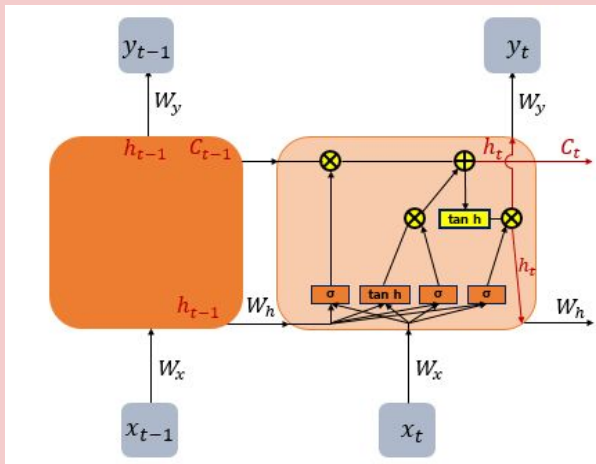
'주가가'에 대한 결과

'몇이야'에 대한 결과

그냥 RNN



LSTM



가

Gate를 지나는데 **forget, input, output**이 있음

- 1) **Forget Gate** : 과거 정보를 기억 or 잊을지 결정
- 2) **Input Gate** : 현재 정보를 기억 or 잊을지 결정
- 3) **Output Gate** : 위의 두 gate를 통해 계산된 출력값을 넘기는 단계

좀 더 발전된 RNN - ②GRU

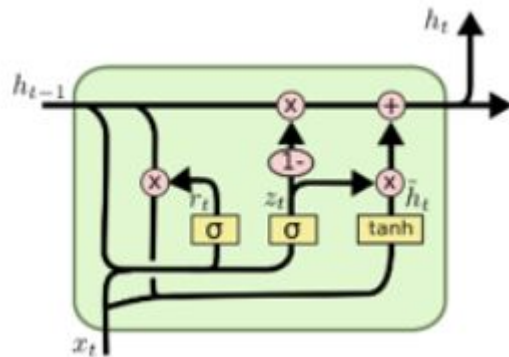
LSTM의 구조를 조금 더 간단하게 개선한 모델

Reset, Update 2개의 gate만을 사용

cell state & hidden state가 합쳐져서 하나의 hidden state로 표현

- 1) **Reset Gate** : 이전 hidden state의 값을 얼마나 활용할 것인지에 대한 정보
- 2) **Update Gate** : 과거와 현재의 정보를 각각 얼마나 반영할지에 대한 비율을 구함

학습할 파라미터가 LSTM보다는 더 적다 **BUT** 성능 면에서 무조건 우월하진X





2. RNN 예제 학습 1

- 로이터 뉴스 카테고리 분류
(LSTM)

입력된 문장의 의미를 파악하는 것

모든 단어를 종합 -> 1개의 카테고리로 분류

광진구는 대체로 맑다가 오후에 눈이 내리겠습니다 ▶[]

연 초부터 GME의 주가변동이 심상치 않았다 ▶[]

이번 서울시장 선거에서 누가 당선될까? ▶[]

오늘 주제는 입력순서와 가중치를 활용한 RNN이래 ▶[]

입력된 문장의 의미를 파악하는 것

모든 단어를 종합 -> 1개의 **카테고리**로 분류

광진구는 대체로 **맑다**가 오후에 **눈**이 내리겠습니다 ▶ 날씨

연 초부터 **GME**의 **주가변동**이 심상치 않았다 ▶ 주식

이번 **서울시장** 선거에서 누가 **당선**될까? ▶ 정치

오늘 주제는 **입력순서**와 **가중치**를 활용한 **RNN**이래 ▶ 딥러닝

로이터 뉴스 데이터셋

`from keras.datasets import reuters`

X


[뉴스 기사0 , 뉴스 기사1, 뉴스
기사2, ...
..., 뉴스 기사 11257]

11,258개의 뉴스기사

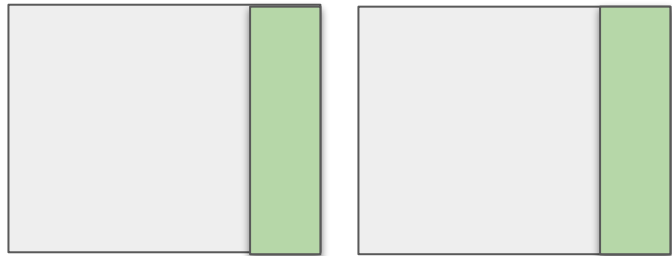
Y

[카테고리 0, 카테고리 1,
카테고리 2, ...
..., 카테고리 45]

46개의 카테고리



```
(X_train, Y_train), (X_test, Y_test) =  
reuters.load_data(num_words=1000, test_split=0.2)
```



num_words=1000 : 1~1000번째 까지 자주 쓰인 단어만 불러온다.
(1001번째부터(나머지) 단어들은 버림)

test_split = 0.2 : 80% train data + 20% test data

데이터 확인

46 카테고리

8982 학습용 뉴스 기사

2246 테스트용 뉴스 기사

```
category = np.max(Y_train) + 1
```

```
print(category, '카테고리')
```

```
print(len(X_train), '학습용 뉴스 기사')
```

```
print(len(X_test), '테스트용 뉴스 기사')
```

```
print(X_train[0])
```

```
[1. 2. 2. 8. 43. 10. 447. 5. 25. 207. 270. 5. 2. 111. 16. 369. 186. 90. 67. 7. 89. 5. 19. 102. 6. 19. 124. 15. 90. 67. 84. 22. 482. 26. 7. 48. 4. 49. 8. 864. 39. 209. 154. 6. 151. 6. 83. 11. 15. 22. 155. 11. 15. 7. 48. 9. 2. 2. 504. 6. 258. 6. 272. 11. 15. 22. 134. 44. 11. 15. 16. 8. 197. 2. 90. 67. 52. 29. 209. 30. 32. 132. 6. 109. 15. 17. 12]
```

데이터 확인

46 카테고리

8982 학습용 뉴스 기사

2246 테스트용 뉴스 기사

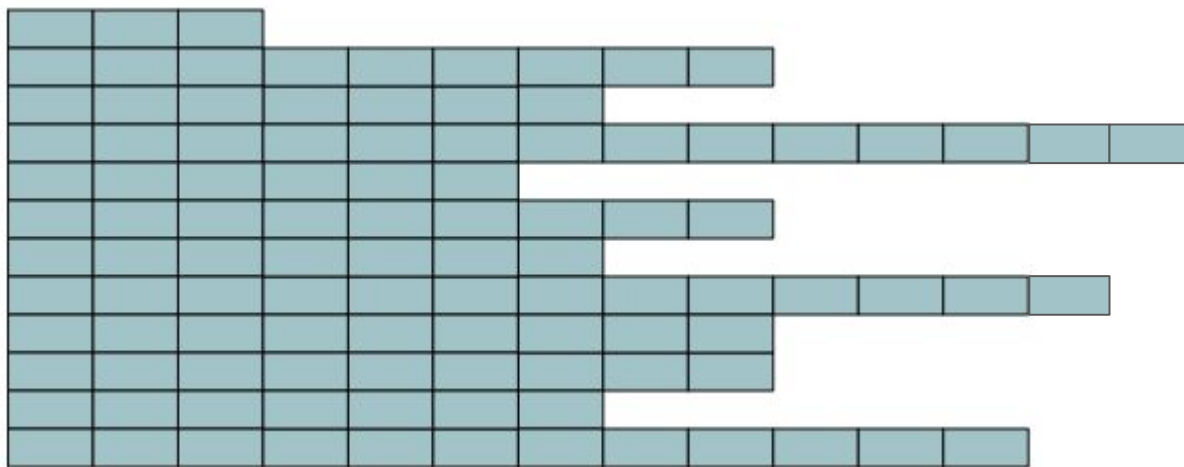
```
category = np.max(Y_train) + 1

print(category, '카테고리')
print(len(X_train), '학습용 뉴스 기사')
print(len(X_test), '테스트용 뉴스 기사')
print(X_train[0])
```

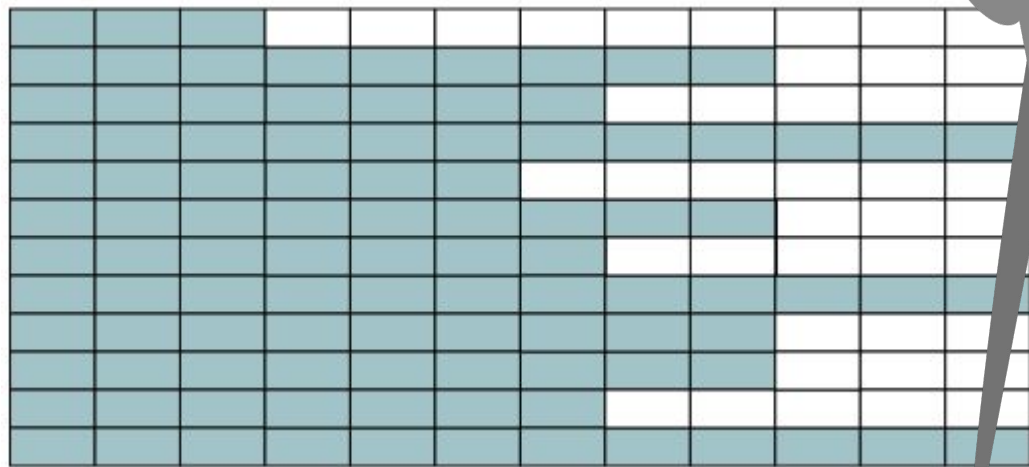
```
[1, 2, 2, 8, 43, 10, 447, 5, 25, 207, 270, 5, 2, 111, 16, 369, 186, 90, 67, 7, 89, 5, 19, 102, 6, 19, 124, 15, 90, 67, 84, 22, 482, 26, 7, 48, 4, 49, 8, 864, 39, 209, 154, 6, 151, 6, 83, 11, 15, 22, 155, 11, 15, 7, 48, 9, 2, 2, 504, 6, 258, 6, 272, 11, 15, 22, 134, 44, 11, 15, 16, 8, 197, 2, 90, 67, 52, 29, 209, 30, 32, 132, 6, 109, 15, 17, 12]
```

▶ **tokenizer()** 작업이 이미 완료된 데이터를 불러온 것 (17장 NLP)

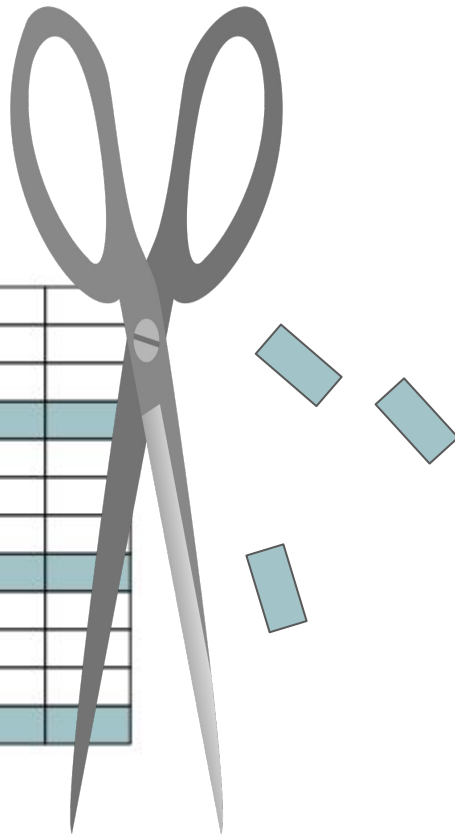
각 기사의 단어수가 제각각



각 기사의 단어수가 제각각



Blue	Blue	Blue	White	White	White	White	White	White	White	White	White
Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	White	White	White
Blue	Blue	Blue	Blue	Blue	Blue	Blue	White	White	White	White	White
Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue
Blue	Blue	Blue	Blue	Blue	Blue	White	White	White	White	White	White
Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	White	White	White
Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue
Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue
Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue
Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue
Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue
Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue



X 데이터 전처리

```
x_train = sequence.pad_sequences(X_train, maxlen= 100)
x_test = sequence.pad_sequences(X_test, maxlen= 100)
```

maxlen = 100

입력된 기사의 단어 수 > 100 : 100번째 단어까지만 선택

< 100 : 부족한 부분 모두 0으로 채움

preview) to_categorical()

정수 인덱스를 0과 1로 이루어진
배열로 바꾸어 주기

```
from keras.utils import to_categorical
```

```
# 인덱스 수에 하나를 추가해서 원-핫 인코딩 배열 만들기
```

```
word_size = len(token.word_index) + 1
```

```
x = to_categorical(x, num_classes=word_size)
```

[[1,2,3,4,5,6]]

[[0. 1. 0. 0. 0. 0. 0.]

[0. 0. 1. 0. 0. 0. 0.]

[0. 0. 0. 1. 0. 0. 0.]

[0. 0. 0. 0. 1. 0. 0.]

[0. 0. 0. 0. 0. 1. 0.]

[0. 0. 0. 0. 0. 0. 1.]]

오랫동안

꿈꾸는

이는

그

꿈을

뒹아간다

Y 데이터 전처리



```
from keras.utils import np_utils
```

```
y_train = np_utils.to_categorical(Y_train)
```

```
y_test = np_utils.to_categorical(Y_test)
```

코드 설명





3. RNN 예제 학습 2

- 영화 리뷰 분류하기
(LSTM + CNN)

LSTM + CNN

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, None, 100)	500000
conv1d (Conv1D)	(None, None, 64)	32064
max_pooling1d (MaxPooling1D)	(None, None, 64)	0
lstm (LSTM)	(None, 55)	26400
dense (Dense)	(None, 1)	56
activation (Activation)	(None, 1)	0

Total params: 558,520
Trainable params: 558,520
Non-trainable params: 0

convolution

1	0	1	0
0	1	1	0
0	0	1	1
0	0	1	0

input data



x1	x0
x0	x1

필터 (커널)



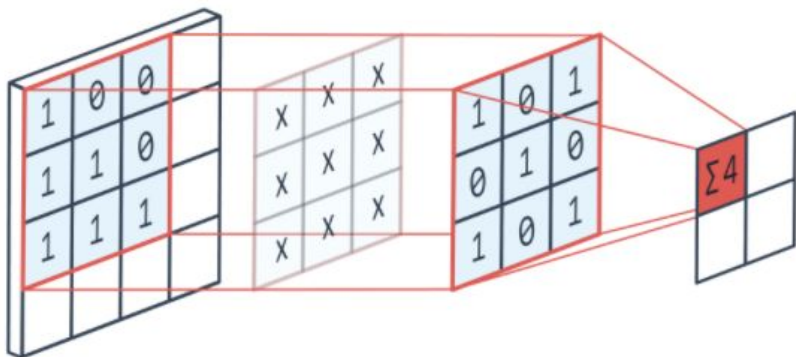
1x1	0x0	1	0	1	0x1	1x0	0	1	0	1x1	0x0
0x0	1x1	1	0	0	1x0	1x1	0	0	1	1x0	0x1
0	0	1	1	0	0	1	1	0	0	1	1
0	0	1	0	0	0	1	0	0	1	0	0
1	0	1	0	0x1	1x0	1	1	0	1	1	0
0x1	1x0	1	0	0x0	0x1	1	0	0	0	1	0
0	0	1	0	1	0	1	1	0	1	1	0
1	0	1	0	0	1	1	0	0	1	1	0
0x1	0x0	1	1	0	0x1	1x0	1	0	0	1x1	1x0
0x0	0x1	1	0	0	0x0	1x1	0	0	0	1x0	0x1

2	1	1
0	2	2
0	1	1

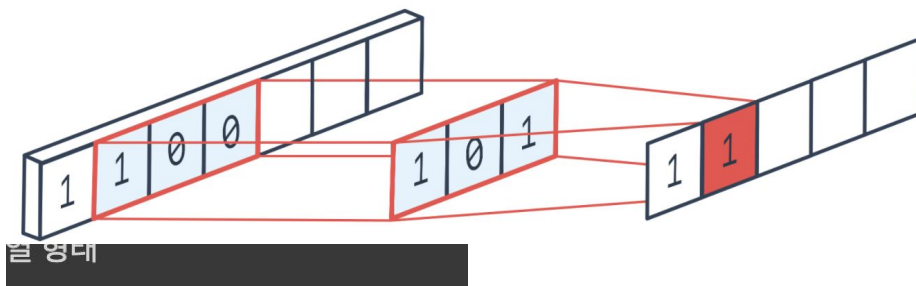
입력 데이터로부터 더욱 정교한 특징 추출!

Conv1D vs 2D

Conv2D (16장)



Conv1D (18장)



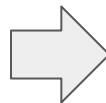
maxpooling

convolution -> 특징 도출 -> but, 결과가 여전히 크고 복잡 -> pooling

maxpooling

average pooling

1	0	1	0
0	4	2	0
0	1	6	1
0	0	1	0

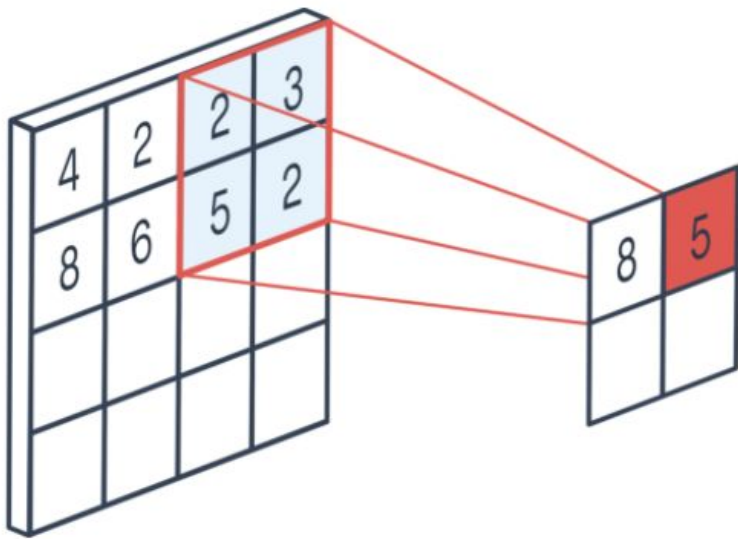


4	2
1	6

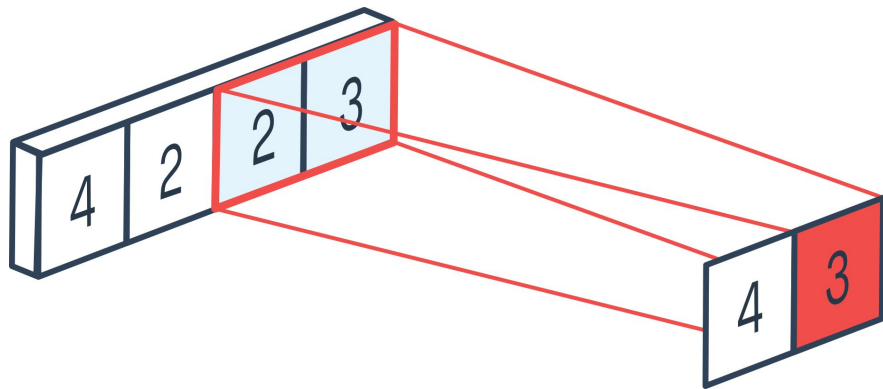
가장 큰 값들을 도출해 불필요한 정보 제거!

MaxPooling1D vs 2D

MaxPooling2D (16장)



MaxPooling1D (18장)



```
model.add(Conv1D(64, 5, padding='valid', activation='relu', strides=1))
```

- "VALID" = 패딩 없음 :

inputs: 1 2 3 4 5 6 7 8 9 10 11 12 13


- "SAME" = 제로 패딩 포함 :

inputs: 1 2 3 4 5 6 7 8 9 10 11 12 13 (0 0 0)

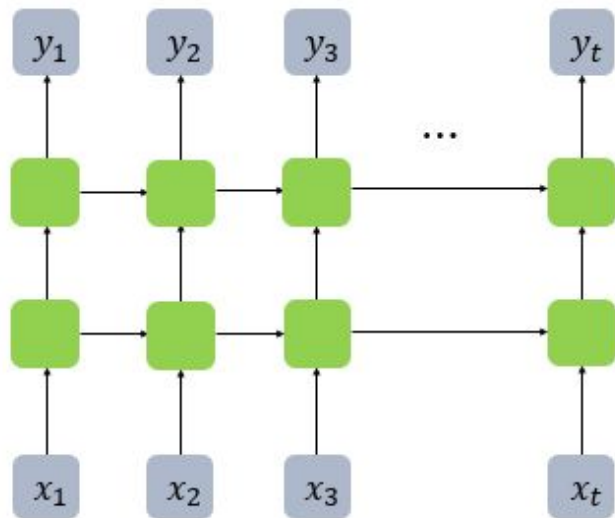

padding : 출력 데이터의 크기를 조정하기 위해 사용



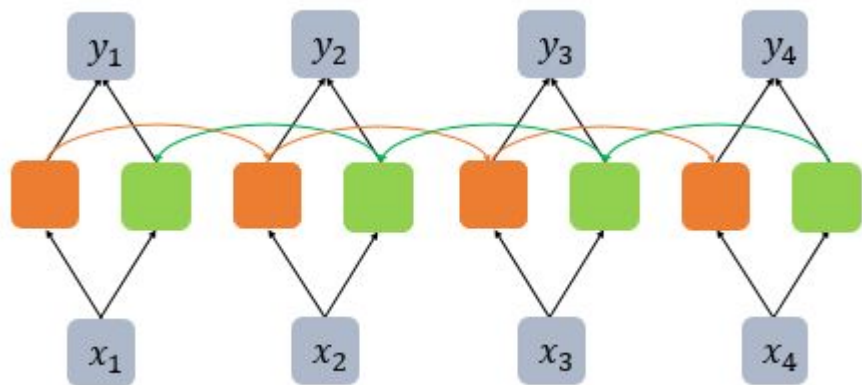


RNN +

깊은 순환 신경망 (Deep Recurrent Neural Network)



양방향 순환 신경망 (Bidirectional Recurrent Neural Network)



I wanna [] at nice restaurant.

-a) eat

-b) punch

-c) fight

기아자동차 주식 예측

기아차

81,500 KRW

+500 (0.62%) ↑

2월 19일 오후 2:55 GMT+9 · 면책조항

KRX: 000270

+ 팔로우

1일 5일 1개월 6개월 ytd 1년 5년 최대



Thank you for watching

SMARCLE Winter Study [RNN team]

질의응답