딥러닝을 활용한 자연어 처리(1)

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

07

APPLICATION

ARTIFICIAL INTELLIGENCE

Notice

THINKING

생각해보기

✓ 딥러닝을 활용한 자연어 처리는 어떻게 할까요?

학습목표

GOALS

1 순차 데이터란 무엇인지 설명할 수 있다.

2 순환 신경망에 대해 설명할 수 있다.

3 RNN에서의 Feed Forward 과정을 설명할 수 있다.

4 Keras를 활용해 RNN모델을 구성할 수 있다.

5 RNN에서의 역전파 과정을 설명할 수 있다.

6 Keras를 활용해 RNN모델을 설명할 수 있다.

1 순차 데이터와 순환 신경망

2 RNN에서의 Feed-forward

3 RNN 모델 학습하기

4 실습

CONTENTS

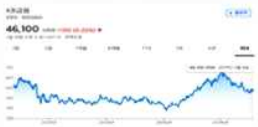
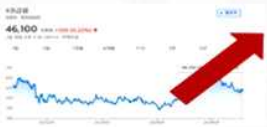
학습내용

APPLICATION

01 순차 데이터와 순환 신경망

순차 데이터(Sequential Data)

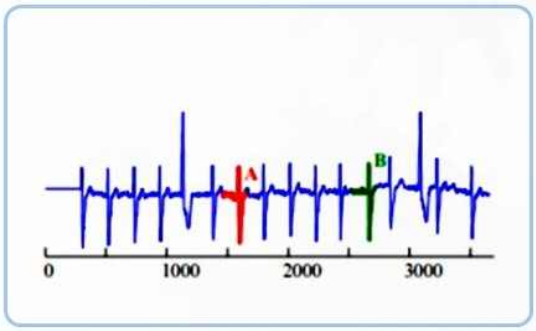
- 데이터 집합 내의 객체들이 어떤 순서를 가지고 있는 데이터
  - 데이터들의 순서에 따라 해당 데이터의 의미가 달라짐
  - 데이터의 길이는 가변적임

Example	X	y
음성인식		"나는 너를 사랑한다."
시 생성		"사랑이 온다는 건 실은 어마어마한 일이다."
감정 분류	"대박맛집! 고기가 친절하고 사장님이 맛있어요!"	만족도 : ★★★★☆
DNA 염기서열	AAGTCTAACGAAACGTCCCTATC	돌연변이 : AAGTCTAACGTAAACGTCCCTATC
기계번역	ನಾನು ನಿನ್ನನ್ನು ಪ್ರೀತಿಸುತ್ತೇನೆ.	"나는 너를 사랑해."
비디오 활동 인식		"타자가 공을 쳤습니다!"
주가 예측		

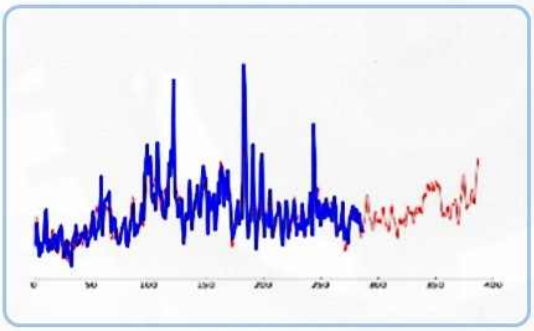
출처: 퍼블릭에이아이(www.publicai.co.kr)

01 순차 데이터

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING



Time series classification
ECG anomaly detection
Human activity recognition



Time series classification
Energy demand prediction
Stock market prediction

출처: https://image.slidesharecdn.com/javier-171204143541/95/state-of-the-art-timeseries-analysis-with-deep-learning-by-javier-ordez-at-big-data-spain-2017-5-638.jpg?cb=1512399195

순환신경망(RNN, Recurrent Neural Network)

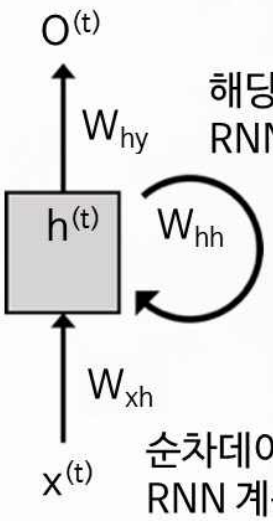
유닛 간의 연결이 순환적 구조를 가짐으로써 순차 데이터가 가지고 있는 "데이터의 순서"의 의미와 가변성을 처리할 수 있도록 모델링 된 인공 신경망

자기 자신의 출력 값이 그 다음의 입력값으로 순환되어 들어가는 순환 경로가 존재

02 순환신경망

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

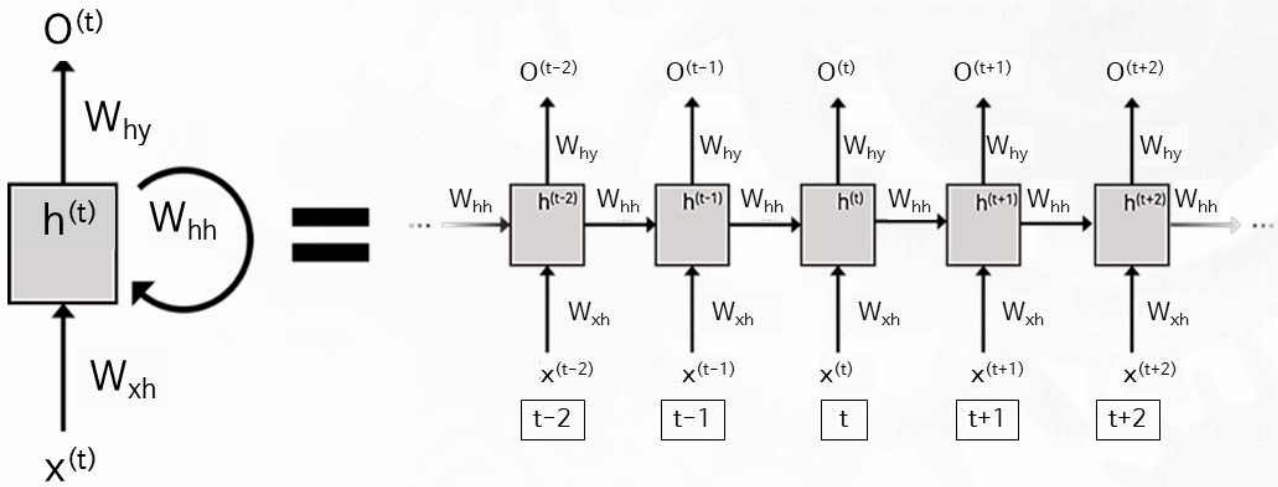
순환신경망의 작동 원리



해당 순환 경로를 따라 데이터를
RNN 계층 안에서 순환시킬 수 있음

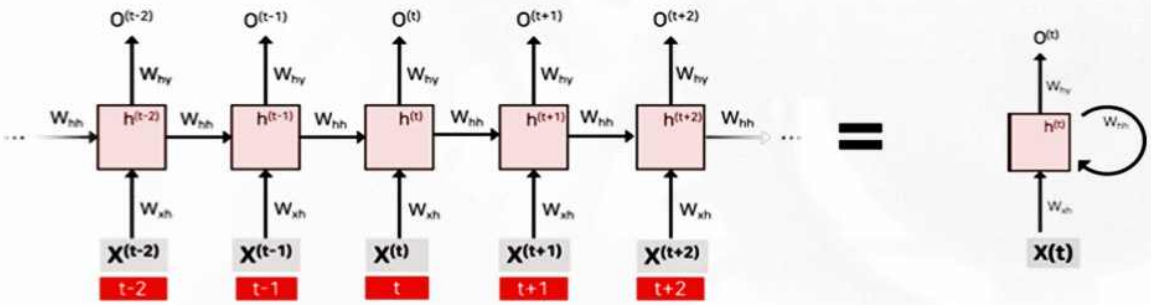
순차데이터 (x0, x1, x2…)가
RNN 계층에 입력됨

순환신경망의 작동 원리



출처: 퍼블릭에이아이(www.publicai.co.kr)

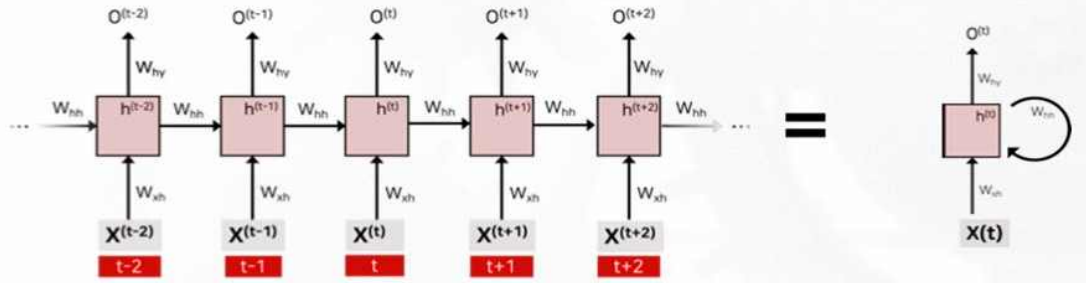
순환신경망의 Time Step



예시)	순차데이터 : … "나는 너를 사랑한다" …				
띄어쓰기로 구분할 경우	"나는"	" "	"너를"	" "	"사랑한다"
자모음으로 구분할 경우	"ㄴ"	"ㅏ"	"ㄴ"	"ㅡ"	"ㄴ"
	주가데이터				
종가로 예측할 경우	1250000	1245000	1265000	1300000	1296000
날씨로 예측할 경우	기온 : 6 습도 : 0.7	기온 : 5 습도 : 0.73	기온 : 2 습도 : 0.53	기온 : 3 습도 : 0.66	기온 : 3.3 습도 : 0.67

출처: 퍼블릭에이아이(www.publicai.co.kr)

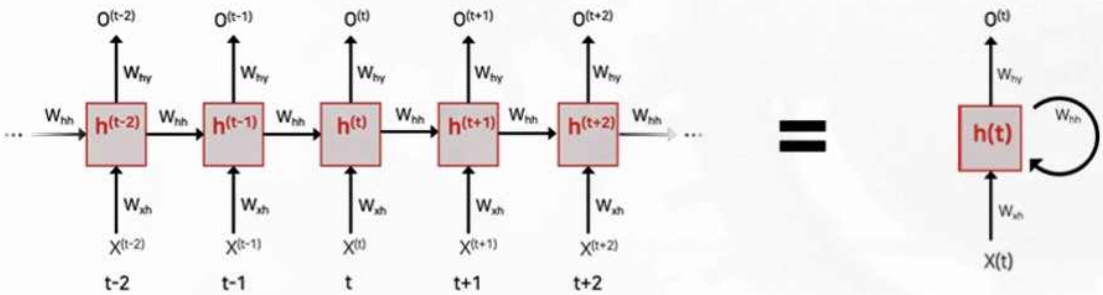
순환신경망의 Time Step



예시) 순차데이터 : … "나는 너를 사랑한다" …

띄어쓰기로 구분할 경우	띄어쓰기로 구분할 경우	"나는"	" "	"너를"	" "	"사랑한다"	"나는"	" "	"너를"	" "	"사랑한다"
자모음으로 구분할 경우	자모음으로 구분할 경우	"ㄴ"	"ㅏ"	"ㄴ"	"ㅡ"	"ㄴ"	"ㄴ"	"ㅏ"	"ㄴ"	"ㅡ"	"ㄴ"
		주가데이터					주가데이터				
종가로 예측할 경우	종가로 예측할경우	1250000	1245000	1265000	1300000	1296000	1250000	1245000	1265000	1300000	1296000
날씨로 예측할 경우	날씨로 예측할경우	기온 : 6 습도 : 0.7	기온 : 5 습도 : 0.73	기온 : 2 습도 : 0.53	기온 : 3 습도 : 0.66	기온 : 3.3 습도 : 0.67	기온 : 6 습도 : 0.7	기온 : 5 습도 : 0.73	기온 : 2 습도 : 0.53	기온 : 3 습도 : 0.66	기온 : 3.3 습도 : 0.67

순환신경망의 State(상태)



상태(h : hidden) : 과거의 상태값($h^{(t)}$)과 현재의 입력값(θ)을 통해 계산 $h^{(t)} = f_\theta(h^{(t-1)}, x^{(t)})$

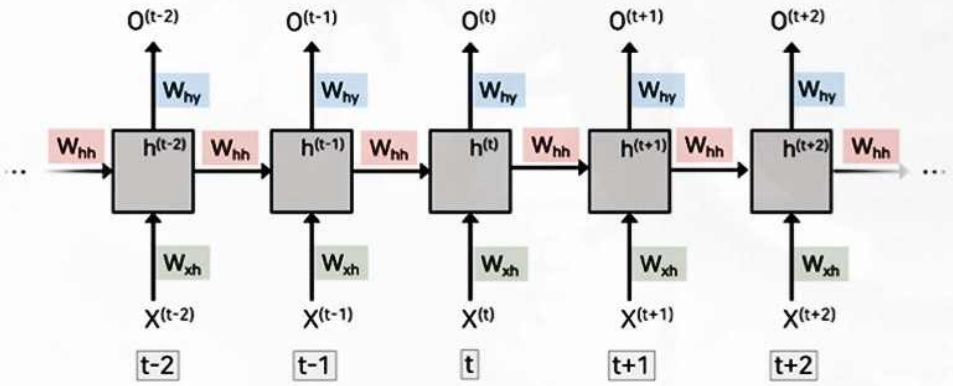
→ Time step에 의해 순차적으로 계산됨에 따라 순서의 의미도 내포

현재의 상태 $h^{(t)}$ 의 사용 :

- 다음 시점의 상태($h^{(t+1)}$)의 계산
- 현재 시점의 출력값(O^t)의 계산

출처: 퍼블릭에이아이(www.publicai.co.kr)

순환신경망에서의 Parameter Sharing



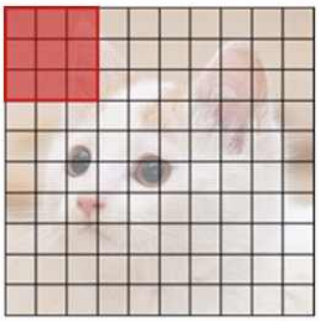
> **기존의 신경망은 위의 가중치가 time step별로 달라야 하지만, 순환 신경망에서는 위의 가중치가 time step마다 동일함**

출처: 퍼블릭에이아이(www.publicai.co.kr)

02 순환신경망

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

순환신경망에서의 Parameter Sharing

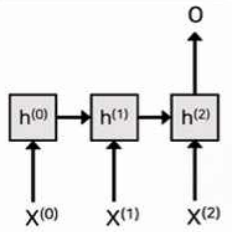


CNN에서 Kernel을 사용한
Parameter Sharing

불	행	한		삶	에	서		벗	어	날		수		있	는
	방	법	은		두		가	지	가		있	다	.		그
것	은		음	악	과		고	양	이	다	.				

출처: 퍼블릭에이아이(www.publicai.co.kr)

Many to One



문장(문체) → 글쓴이 추측

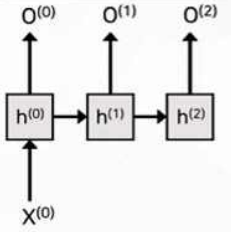
Input

"눈이 푹푹 쌓이는 밤 흰 당나귀 타고
산골로 가자 출출이 우는 깊은 산골로 가
마가리에 살자
눈은 푹푹 나리고
나는 나타샤를 생각하고"

Output

백석

One to Many



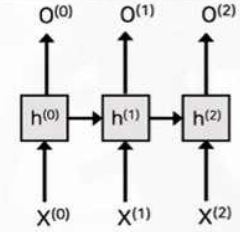
이미지 → 문장 설명

Input

Output

"정장을 입은 다섯명의 회사원이 회의실에서 회의를 진행하고 있다. 회색 정장을 입은 남성이 손에 펜을 쥐고 보드 옆에 서서 무언가를 설명하고 있다."

Many to Many

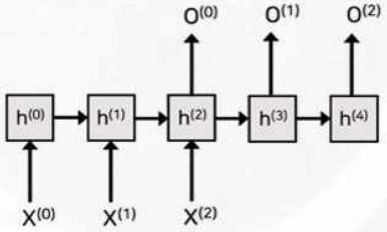


비디오영상 → 매 순간 물체 인식

Input : Video 영상

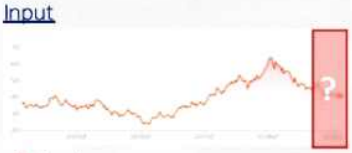
Output

Many to Many

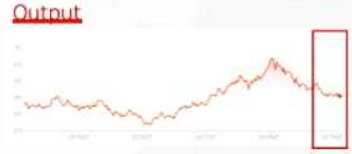


주가예측, 번역, 시 생성기, 기사요약 등

Input



Output



출처: 퍼블릭에이아이(www.publicai.co.kr)

APPLICATION

02 RNN에서의 Feed-forward

03 순환신경망 구현

필요 라이브러리 호출

In [1]:

```
import numpy as np
import tensorflow as tf
import keras

from tensorflow.keras.layers import SimpleRNN
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
import tensorflow.keras.backend as K
```

숫자를 벡터로 표현하기

Char	index	One-Hot Vector
h	0	[1,0,0,0,0]
e	1	[0,1,0,0,0]
l	2	[0,0,1,0,0]
o	3	[0,0,0,1,0]
<EOS>	4	[0,0,0,0,1]

In [2]:
```
char2vec = {'h':np.array([1,0,0,0,0]),
            "e":np.array([0,1,0,0,0]),
            "l":np.array([0,0,1,0,0]),
            'o':np.array([0,0,0,1,0]),
            '<EOS>':np.array([0,0,0,0,1])}

idx2char = ['h','e','l','o','<EOS>']
```

In [3]:
```
print("Character -> Vector : 'l' -> ", char2vec['l'])
print("index -> Character : '2' -> ", idx2char[2])
```

```
Character -> Vector : 'l' ->  [0 0 1 0 0]
index -> Character : '2' ->  l
```


Embedding Layer 사용하기

In [4]:
```
import tensorflow.keras.backend as K
from tensorflow.keras.layers import Embedding, Input
from tensorflow.keras.models import Model
```

In [5]:
```
K.clear_session()
tf.random.set_seed(1)

inputs = Input(shape=())
embeded = Embedding(input_dim=5,output_dim=2)(inputs)

model = Model(inputs,embeded)
```

Embedding Layer의 weights 확인하기

```
In [6]: model.layers[1].get_weights()
```

```
Out[6]: [array([[-0.03348692,  0.04014813],
                [ 0.01309742, -0.00654539],
                [-0.0208061 ,  0.01425021],
                [ 0.04757855, -0.00649005],
                [ 0.01601019,  0.01048958]], dtype=float32)]
```

char	index	1번째 임베딩 벡터 값	2번째 임베딩 벡터 값
"h"	0	-0.033	0.040
"e"	1	0.013	-0.065
"l"	2	-0.021	0.014
"o"	3	0.048	-0.065
"<eos>"	4	0.016	0.010

가중치 구성하기

```
In [8]:
### RNN Cell 내 가중치
w_xh = np.array([[-2.6, -1.6, -2.1],
                 [ 1.2,  0.4,  0.3],
                 [ 2.1,  1.9, -0.7],
                 [-1.4, -1.5,  2.5],
                 [-0.9,  0.4, -0.9]])

w_hh = np.array([[-0.5, -2.3,  2.9],
                 [ 1.9,  1.5,  1.7],
                 [-0.7, -1.2,  1.5]])

b_h = np.array([-0.5, -0.4, -1. ])

# Output Layer 내 가중치
w_hy = np.array([[ 0.3, -2.6,  1.2,  2.6, -1.1],
                 [-1.1, -2.4,  2.2,  1.6, -2.4],
                 [-0.4, -3.1, -3. ,  3.6,  3. ]])

b_y = np.array([-1.8, -0.5,  1.3,  0.1,  0.8])
```

Time step 1 계산

In [9]:
```
def softmax(x):
    """Compute softmax values for each sets of scores in x."""
    return np.exp(x) / np.sum(np.exp(x), axis=-1)
```

In [10]:
```
init_h = np.array([0,0,0])
x_0 = char2vec['h']

a_1 = np.dot(init_h, w_hh) + np.dot(x_0,w_xh) + b_h
h_1 = np.tanh(a_1)

y_1 = np.dot(h_1, w_hy) + b_y
o_1 = softmax(y_1)

print("1번째 Timestamp의 hidden : {}".format(h_1))
print("1번째 Timestamp의 output : {}".format(y_1))
print("1번째 Timestamp의 result : {}".format(idx2char[np.argmax(o_1)]))
```

```
1번째 Timestamp의 hidden : [-0.99594936 -0.96402758 -0.99594936]
1번째 Timestamp의 output : [-0.63997473  7.49057754  0.97184817 -7.61733016  1.22136241]
1번째 Timestamp의 result : e
```

04 Time Step별 계산

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

Time step 2 계산

In [11]:
```
x_1 = char2vec['e']

a_2 = np.dot(h_1, w_hh) + np.dot(x_1, w_xh) + b_h
h_2 = np.tanh(a_2)

y_2 = np.dot(h_2, w_hy) + b_y
o_2 = softmax(y_2)


print("2번째 Timestamp의 hidden : {}".format(h_2))
print("2번째 Timestamp의 output : {}".format(y_2))
print("2번째 Timestamp의 result : {}".format(idx2char[np.argmax(o_2)]))
```

```
2번째 Timestamp의 hidden : [ 0.06340167  0.96673299 -0.99999709]
2번째 Timestamp의 output : [-2.44438695  0.11498748  6.50288586 -1.78837242 -4.58989229]
2번째 Timestamp의 result : l
```

Time step 3 계산

In [12]:
```
x_2 = char2vec['l']

a_3 = np.dot(h_2, w_hh) + np.dot(x_2,w_xh) + b_h
h_3 = np.tanh(a_3)

y_3 = np.dot(h_3,w_hy)+b_y
o_3 = softmax(y_3)

print("3번째 Timestamp의 hidden : {}".format(h_3))
print("3번째 Timestamp의 output : {}".format(y_3))
print("3번째 Timestamp의 result : {}".format(idx2char[np.argmax(o_3)]))
```

```
3번째 Timestamp의 hidden : [ 0.9994564  0.999335  -0.8793026]
3번째 Timestamp의 output : [-2.24771054 -2.7711526   7.33579249  1.1320333  -5.33571385]
3번째 Timestamp의 result : l
```

04 Time Step별 계산

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

Time step 4 계산

In [13]:
```
x_3 = char2vec['l']

a_4 = np.dot(h_3, w_hh) + np.dot(x_3,w_xh) + b_h
h_4 = np.tanh(a_4)

y_4 = np.dot(h_4,w_hy)+b_y
o_4 = softmax(y_4)

print("4번째 Timestamp의 hidden : {}".format(h_4))
print("4번째 Timestamp의 output : {}".format(y_4))
print("4번째 Timestamp의 result : {}".format(idx2char[np.argmax(o_4)]))
```

```
4번째 Timestamp의 hidden : [0.99855062 0.9419888  0.91834213]
4번째 Timestamp의 output : [-2.90395935 -8.20386532  1.81560973  7.50944534  0.19584758]
4번째 Timestamp의 result : o
```

In [14]:

```
from tensorflow.keras.layers import Layer

class RNNCell(Layer):
    def __init__(self, n_units, **kwargs):
        self.n_units = n_units
        self.state_size = n_units
        super(RNNCell, self).__init__(**kwargs)

    def build(self, input_shape):
        self.w_xh = self.add_weight(name='weight_xh',
                                    shape=(input_shape[-1], self.n_units),
                                    initializer=tf.initializers.glorot_normal())
        self.w_hh = self.add_weight(name='weight_hh',
                                    shape=(self.n_units, self.n_units),
                                    initializer=tf.initializers.orthogonal())
        self.b_h = self.add_weight(name='bias_h',
                                   shape=(self.n_units,),
                                   initializer=tf.initializers.zeros())
        super(RNNCell, self).build(input_shape)

    def call(self, inputs, states):
        prev_states = states[0]
        h = (tf.matmul(inputs, self.w_xh)
             + tf.matmul(prev_states, self.w_hh)
             + self.b_h)
        a = tf.tanh(h)
        return a, [a]
```

05 RNN Layer 만들기

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

RNNCell layer를 사용해 model 구성하기

In [15]:

```
from tensorflow.keras.layers import Input, Dense, RNN

n_inputs = 5 # Input 차원 수
n_steps = 5 # time step의 크기
n_neurons = 3 # Hidden 차원 수
n_outputs = n_inputs # Output 차원 수

inputs = Input(shape=(n_steps,n_inputs))
hidden = RNN(RNNCell(n_neurons), return_sequences=True)(inputs)
output = Dense(n_outputs, activation='softmax')(hidden)
```

In [17]:
```
from tensorflow.keras.models import Model

n_inputs = 5 # Input 차원 수
n_steps = 5 # time step의 크기
n_neurons = 3 # Hidden 차원 수
n_outputs = n_inputs # Output 차원 수

inputs = Input(shape=(n_steps,n_inputs))
hidden = RNN(RNNCell(n_neurons), return_sequences=True)(inputs)
output = Dense(n_outputs, activation='softmax')(hidden)

model = Model(inputs, output)

# pretrained weight들로 setting
model.set_weights([w_xh, w_hh,b_h,w_hy,b_y])
```

In [18]:
```
input_values = "hello"
print("입력값 : ",list(input_values))
input_vecs = np.stack([char2vec[char]
                       for char in input_values])[np.newaxis]

results = model.predict(input_vecs)
result_indices = np.argmax(results,axis=-1)[0]
print("출력값 : ",[idx2char[idx] for idx in result_indices])
```

```
입력값 :  ['h', 'e', 'l', 'l', 'o']
출력값 :  ['e', 'l', 'l', 'o', '<EOS>']
```

06 RNN Model 만들기

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

Keras의 simpleRNN Cell 사용하기

In [19]:
```
from tensorflow.keras.layers import SimpleRNNCell

inputs = Input(shape=(n_steps,n_inputs))
hidden = RNN(SimpleRNNCell(n_neurons), return_sequences=True)(inputs)
output = Dense(n_outputs, activation='softmax')(hidden)

model = Model(inputs, output)

# pretrained weight들로 setting
model.set_weights([w_xh, w_hh,b_h,w_hy,b_y])
```

In [20]:
```
input_values = "hello"
print("입력값 : ",list(input_values))
input_vecs = np.stack([char2vec[char]
                       for char in input_values])[np.newaxis]

results = model.predict(input_vecs)
result_indices = np.argmax(results,axis=-1)[0]
print("출력값 : ",[idx2char[idx] for idx in result_indices])
```

```
입력값 :  ['h', 'e', 'l', 'l', 'o']
출력값 :  ['e', 'l', 'l', 'o', '<EOS>']
```

Keras의 simpleRNN Layer 사용하기

In [21]:
```
from tensorflow.keras.layers import SimpleRNN

inputs = Input(shape=(n_steps,n_inputs))
hidden = SimpleRNN(n_neurons, return_sequences=True)(inputs)
output = Dense(n_outputs, activation='softmax')(hidden)

model = Model(inputs, output)

# pretrained weight들로 setting
model.set_weights([w_xh, w_hh,b_h,w_hy,b_y])
```

In [22]:
```
input_values = "hello"
print("입력값 : ",list(input_values))
input_vecs = np.stack([char2vec[char]
                       for char in input_values])[np.newaxis]

results = model.predict(input_vecs)
result_indices = np.argmax(results,axis=-1)[0]
print("출력값 : ",[idx2char[idx] for idx in result_indices])
```

```
입력값 :  ['h', 'e', 'l', 'l', 'o']
출력값 :  ['e', 'l', 'l', 'o', '<EOS>']
```

APPLICATION

03 RNN 모델 학습하기

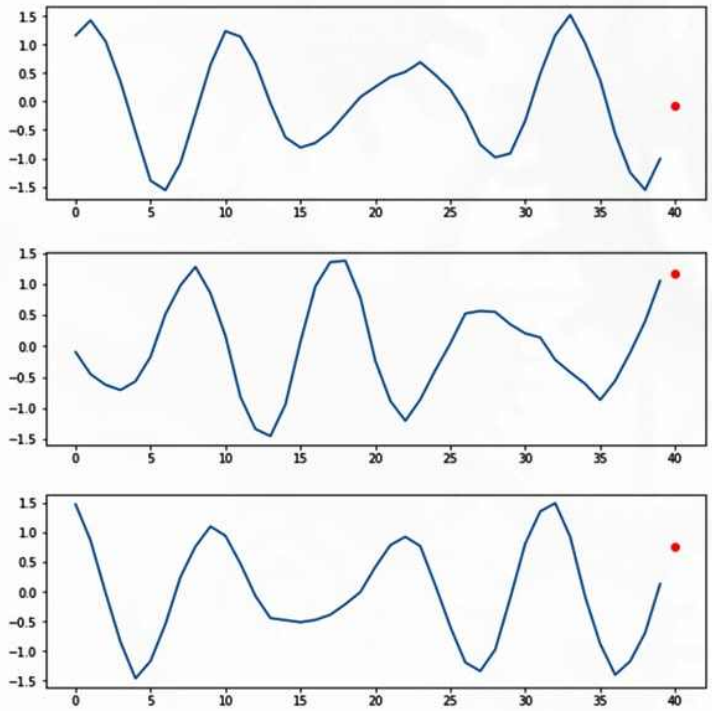
01 데이터셋 확인하기

Toy 데이터셋 생성 및 시각화 코드

In [2]:
```
def generate_timeseries(n_steps=50):
    m_x = np.random.uniform(0,10)
    xs = np.linspace(0, 5, n_steps+1)
    ys = np.array([0.5*np.sin(2*np.pi*(x+m_x))+ np.cos(3*np.pi/2*(x+m_x/4))
                   + np.random.uniform(-0.1,0.1) for x in xs])
    return ys[:-1],ys[-1]
```

In [3]:
```
for _ in range(3):
    xs, ys = generate_timeseries(n_steps=40)
    timesteps = np.arange(len(xs)+1)
    plt.figure(figsize=(10,3))
    plt.plot(timesteps[:-1],xs)
    plt.scatter(timesteps[-1],ys,c='r')
    plt.show()
```

01 데이터셋 확인하기



출처: 퍼블릭에이아이(www.publicai.co.kr)

02 RNN의 역전파

BPTT(Back Propagation Through Time)

기존의 역전파

- Feed-forward의 연산 순서를 반대로 넘어가며 계산함

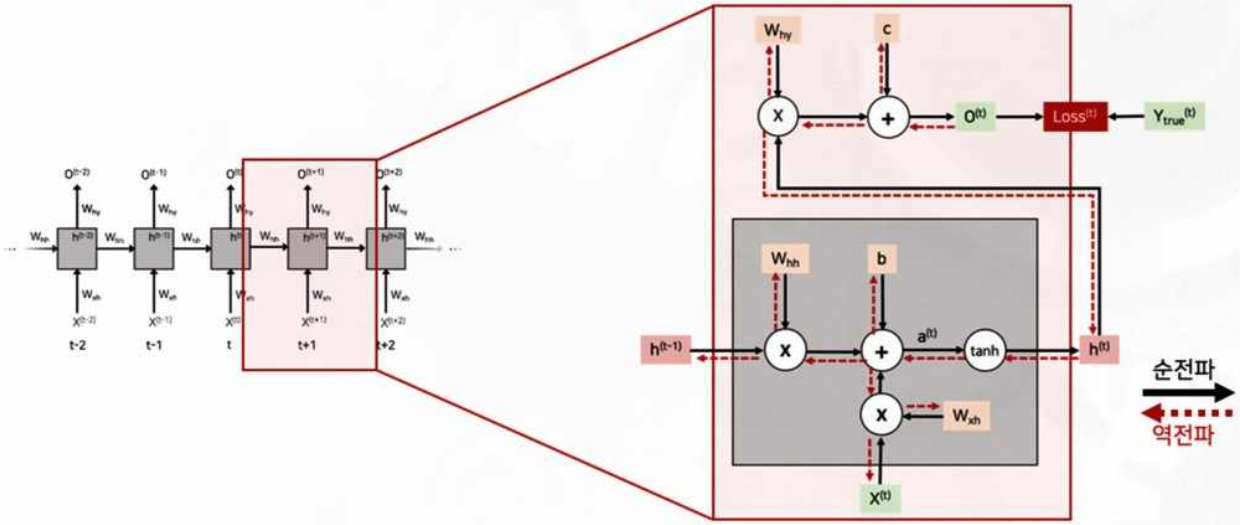
VS

RNN의 역전파

- 기존의 feed-forward와는 달리 각 time step 별로 계산이 진행됨

> **RNN에서의 역전파 역시 time step에 따라 오차가 전파됨**

각 Cell에서의 역전파



출처: 퍼블릭에이아이(www.publicai.co.kr)

RNN 모델 구성하기

In [5]:
```
K.clear_session()

n_inputs = 1
n_steps = 50
n_neurons = 200
n_outputs = n_inputs

inputs = Input(shape=(n_steps,n_inputs))
hidden = SimpleRNN(n_neurons, return_state=False)(inputs)
output = Dense(1)(hidden)

model = Model(inputs,output)

model.summary()
```

```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 50, 1)]           0
_________________________________________________________________
simple_rnn (SimpleRNN)       (None, 200)               40400
_________________________________________________________________
dense (Dense)                (None, 1)                 201
=================================================================
Total params: 40,601
Trainable params: 40,601
Non-trainable params: 0
_________________________________________________________________
```

RNN 모델 컴파일하기

In [6]:
```
model.compile(loss='mse',
              optimizer=Adam(lr=1e-4))
```

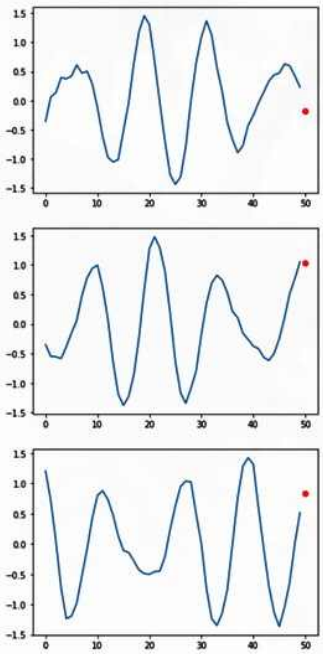
03 RNN 모델 학습

DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

RNN 모델의 input generator 구현하기

In [7]:
```
def timeseries_generator(n_steps=50, batch_size=32):
    while True:
        batch_xs, batch_ys = [], []
        for _ in range(batch_size):
            x, y = generate_timeseries(n_steps)
            batch_xs.append(x[:,np.newaxis])
            batch_ys.append(y[np.newaxis])
        yield np.stack(batch_xs), np.stack(batch_ys)
```

Generator에 의해 모델에 입력되는 데이터



출처: 퍼블릭에이아이(www.publicai.co.kr)

03 RNN 모델 학습

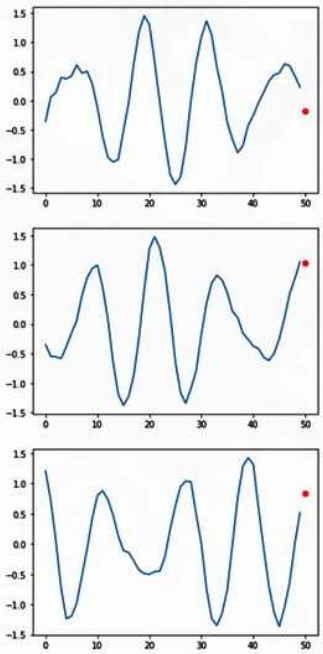
DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

In [9]:
```
batch_size = 16
train_gen = timeseries_generator(n_steps, batch_size)

hist = model.fit_generator(train_gen,
                           steps_per_epoch=20,
                           epochs=10)
```

```
WARNING:tensorflow:From <ipython-input-9-90d238ca3847>:6: Model.fit_generator (from tensorflow.python.keras.engine.training) is deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/10
20/20 [==============================] - 1s 33ms/step - loss: 0.2089
Epoch 2/10
20/20 [==============================] - 1s 32ms/step - loss: 0.0124
Epoch 3/10
20/20 [==============================] - 1s 31ms/step - loss: 0.0080
Epoch 4/10
20/20 [==============================] - 1s 31ms/step - loss: 0.0076
Epoch 5/10
20/20 [==============================] - 1s 31ms/step - loss: 0.0057
Epoch 6/10
20/20 [==============================] - 1s 32ms/step - loss: 0.0054
Epoch 7/10
20/20 [==============================] - 1s 30ms/step - loss: 0.0060
Epoch 8/10
20/20 [==============================] - 1s 32ms/step - loss: 0.0048
```

Generator에 의해 모델에 입력되는 데이터



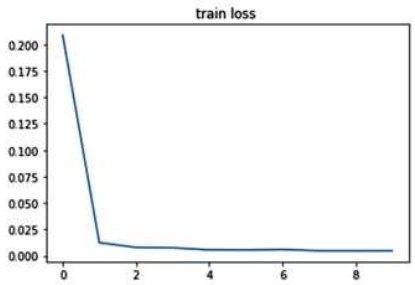
출처: 퍼블릭에이아이(www.publicai.co.kr)

03 RNN 모델 학습

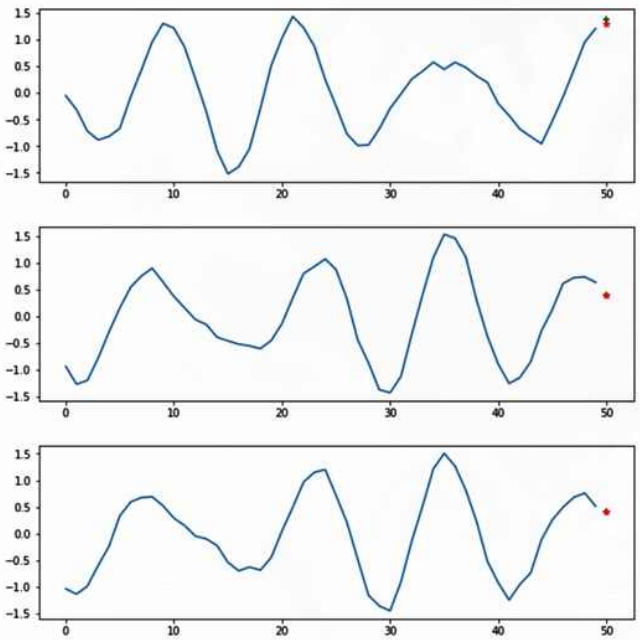
DEEP LEARNING AND NATURAL LANGUAGE PROCESSING

학습 결과 확인하기

```
In [10]: 1 plt.title("train loss ")
         2 plt.plot(hist.history['loss'])
         3 plt.show()
```



학습 결과 확인하기



출처: 퍼블릭에이아이(www.publicai.co.kr)

SUMMARY

학습정리

- 데이터 집합 내의 객체들이 어떤 순서를 갖고 있는 순차데이터
- 순환신경망, Recurrent Neural Network
- RNN에서의 Feed-forward 연산 방식
- RNN에서의 역전파, BPTT
- Keras를 활용한 RNN 학습 과정

EXPANSION

확장하기

1. 순차 데이터는 무엇이고 실생활에서 접할 수 있는 순차데이터에는 어떤 것들이 있을까요?
2. 순환 신경망은 무엇이고 작동 원리는 무엇일까요?
3. 순환신경망의 Time Step은 무엇일까요?
4. BPTT는 무엇이고 일반적인 신경망에서의 역전파와는 어떤 차이가 있을까요?

참고 문헌

REFERENCE

◆ 참고 사이트
- 용어들에 대한 정의 : https://ko.wikipedia.org/wiki.
- 퍼블릭에이아이(www.publicai.co.kr)

서체 출처 : 에스코어드림체-㈜에스코어, 나눔글꼴체-㈜네이버

THINKING

생각해보기

✓ 단어 임베딩은 어떻게 활용할까요?