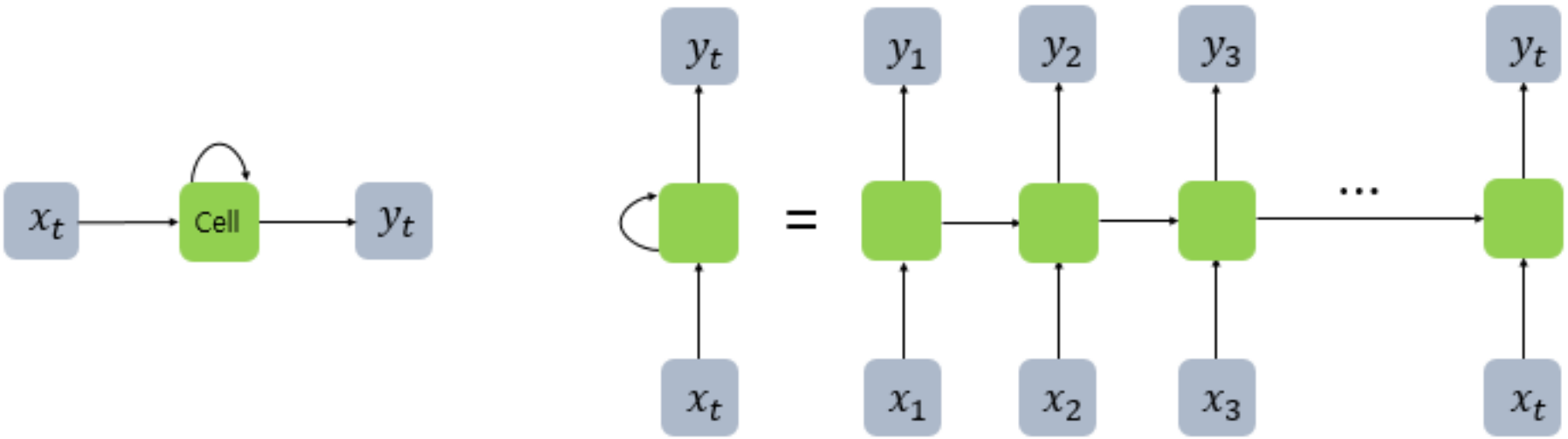
8장 순환 신경망

Recurrent Neural Network

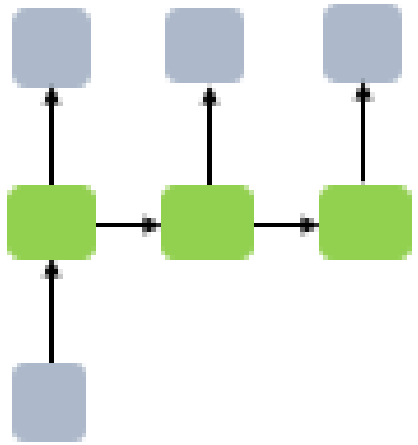
엄호용

목차 Table of Contents

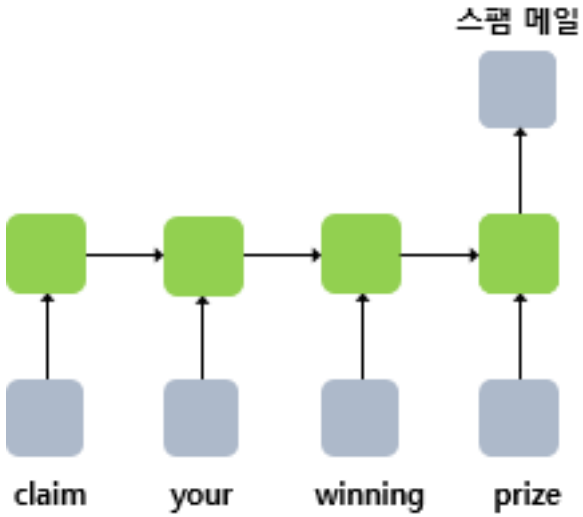
01 순환 신경망 Recurrent Neural Network



01 순환 신경망 Recurrent Neural Network

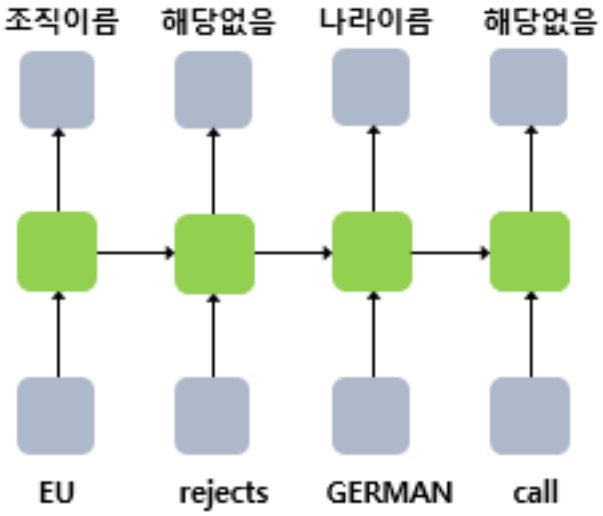


일 대 다(one-to-many)



스팸 메일 분류

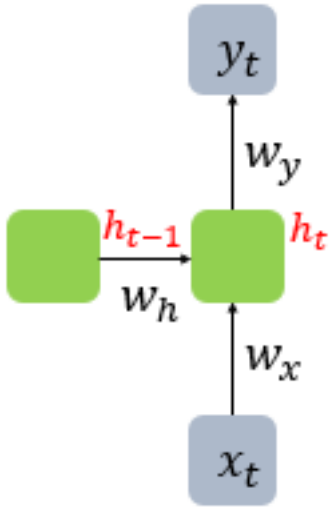
다 대 일(many-to-one)



개체명 인식

다 대 다(many-to-many)

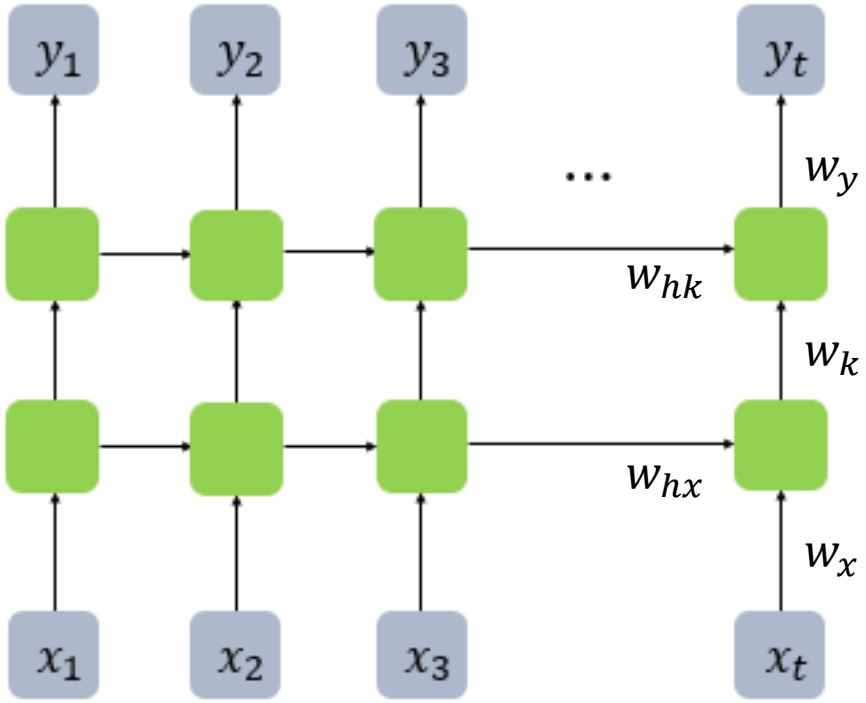
01 순환 신경망 Recurrent Neural Network



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

- 은닉층 : $h_t = tanh(W_x x_t + W_h h_{t-1} + b)$
- 출력층 : $y_t = f(W_y h_t + b)$
  단, f는 비선형 활성화 함수 중 하나.

01 순환 신경망 Recurrent Neural Network



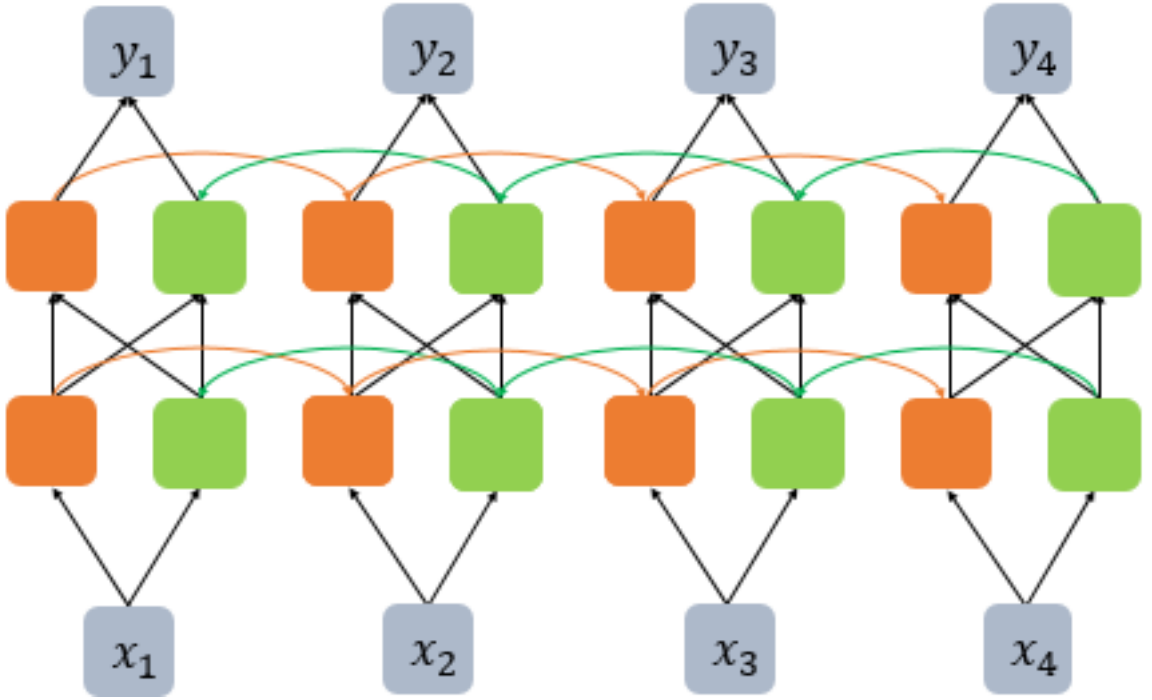
깊은 순환 신경망
Deep Recurrent Neural Network

01 순환 신경망 Recurrent Neural Network

양방향 순환 신경망 (Bidirectional Recurrent Neural Network)

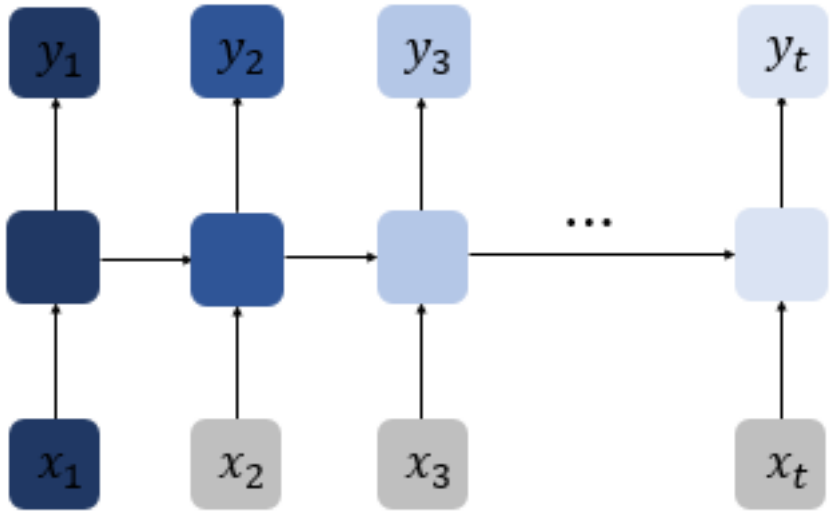
운동을 열심히 하는 것은 [ ]을 늘리는데 효과적이다.

1) 근육
2) 지방
3) 스트레스



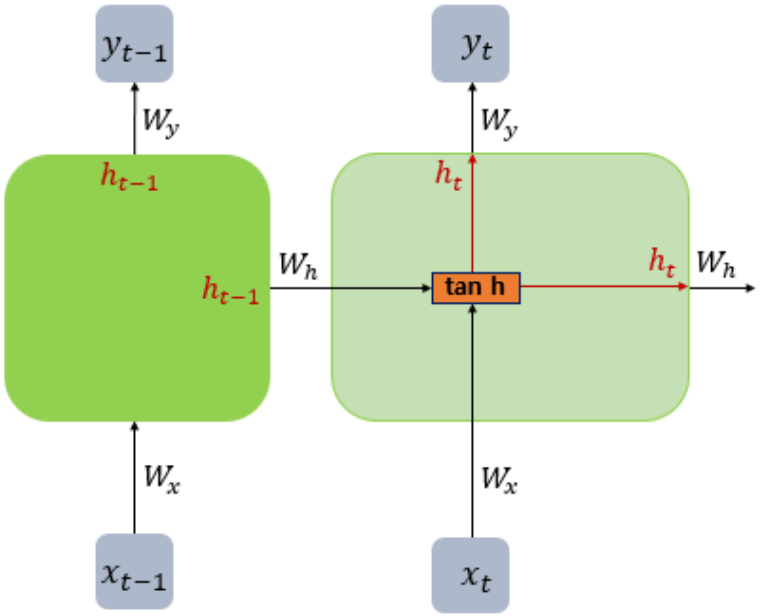
01 순환 신경망 Recurrent Neural Network

한계점

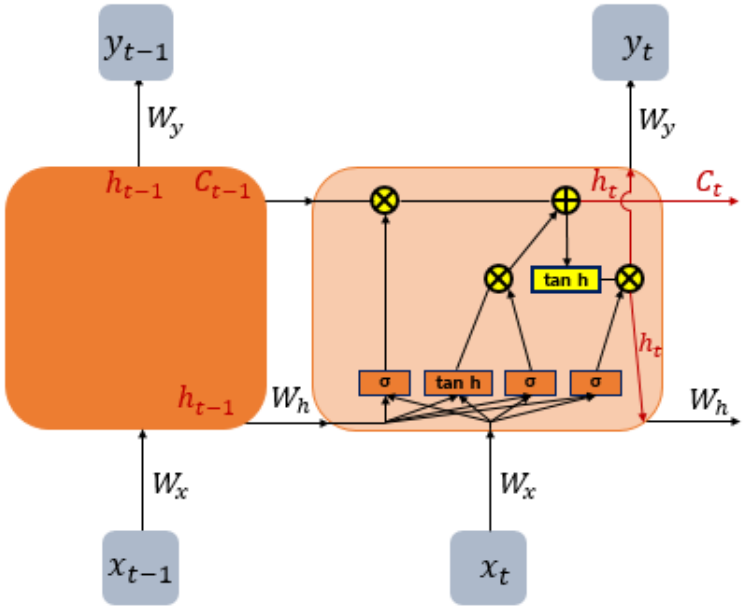


ex) 모스크바에 여행을 왔는데 건물도 예쁘고 먹을 것도 맛있었어. 그런데 글쎄 직장 상사한테 전화가 왔어.
어디냐고 묻더라구 그래서 나는 말했지. 저 여행왔는데요.
여기는 ___

02 장단기 메모리 Long Short-Term Memory

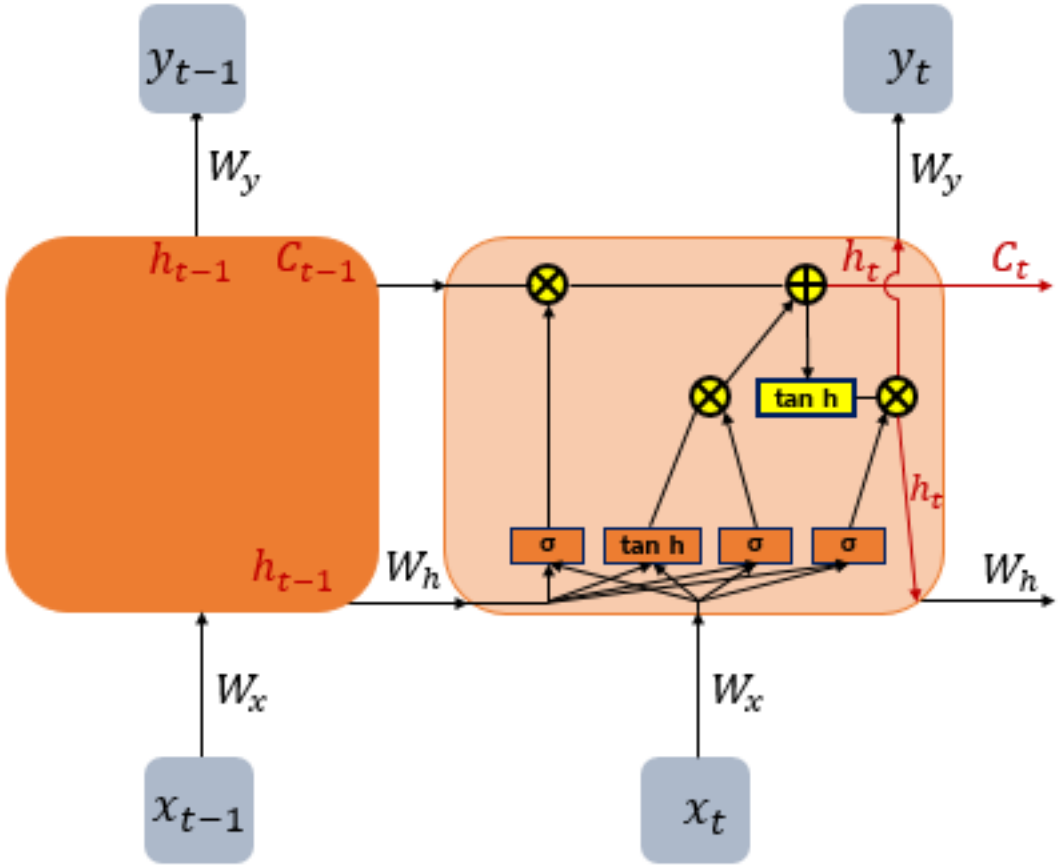


RNN의 구조

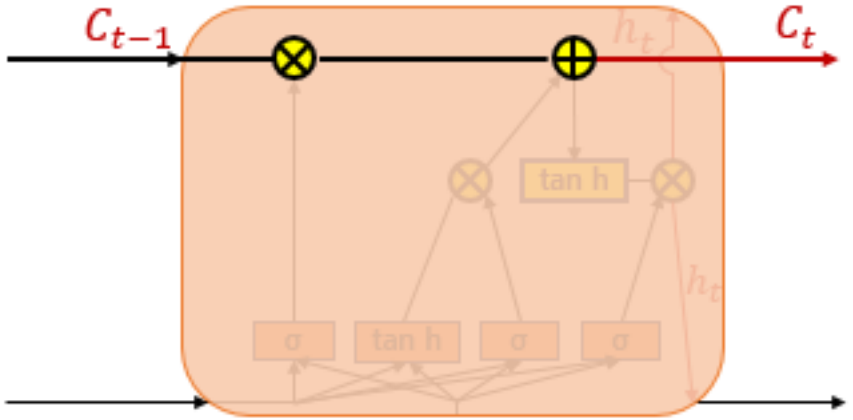


LSTM의 구조

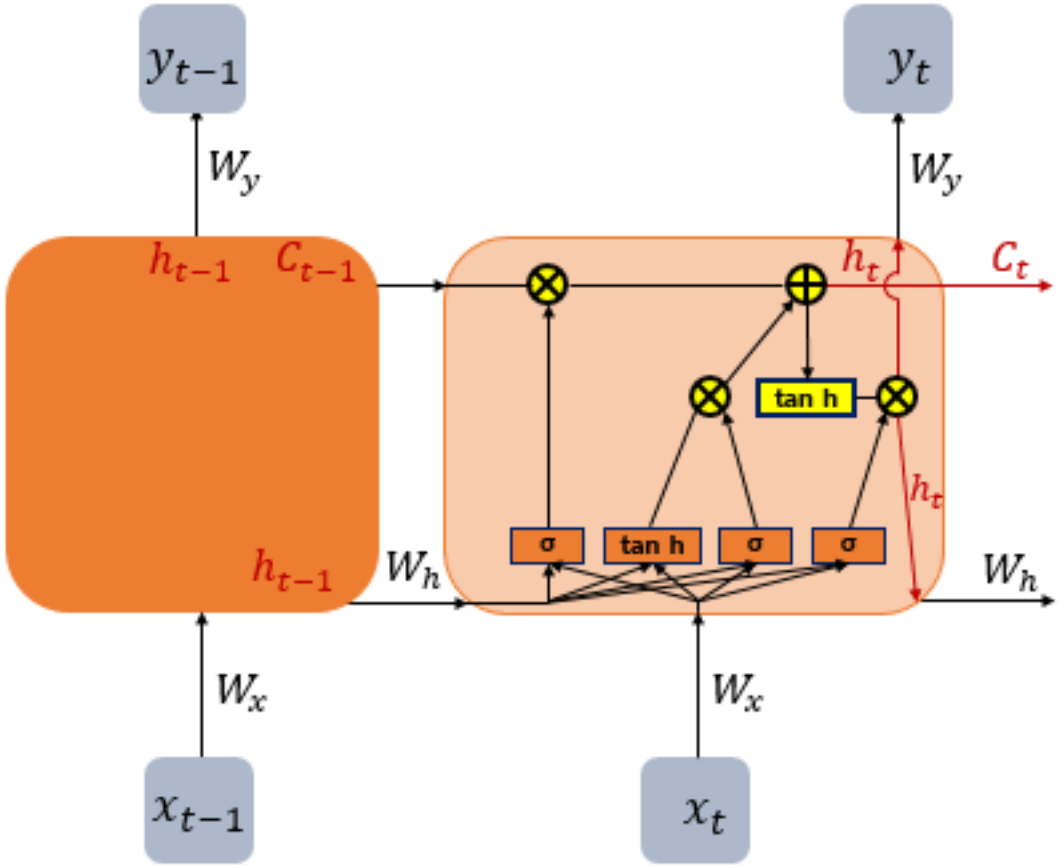
02 장단기 메모리 Long Short-Term Memory



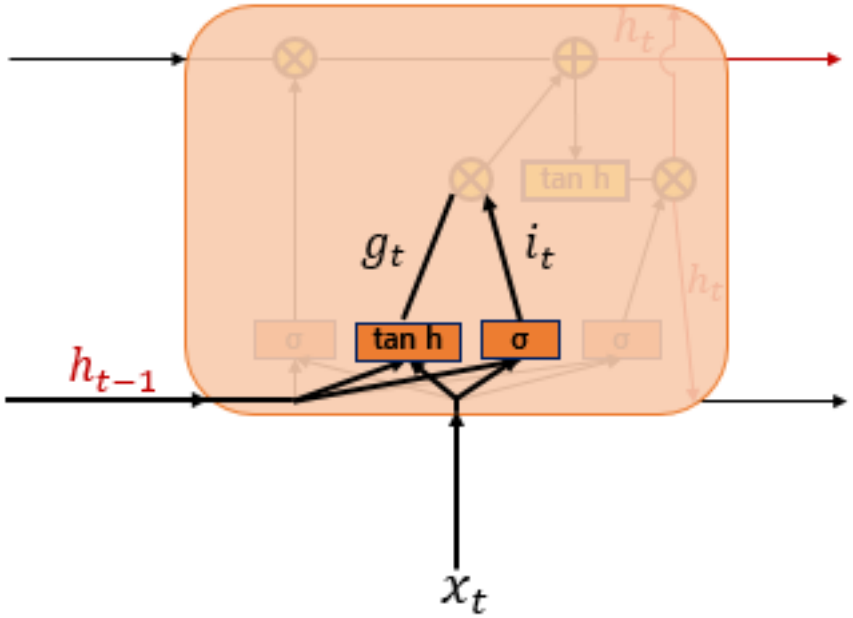
셀 상태(Cell State)



02 장단기 메모리 Long Short-Term Memory

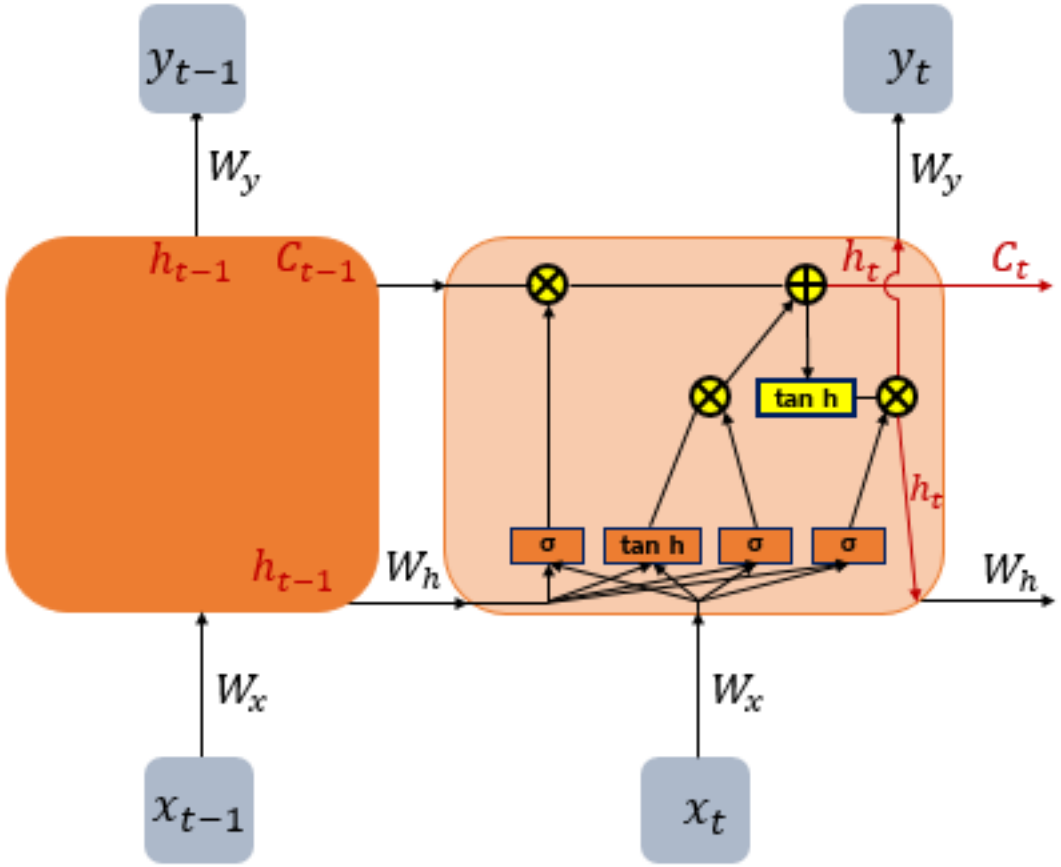


입력 게이트(Input Gate)

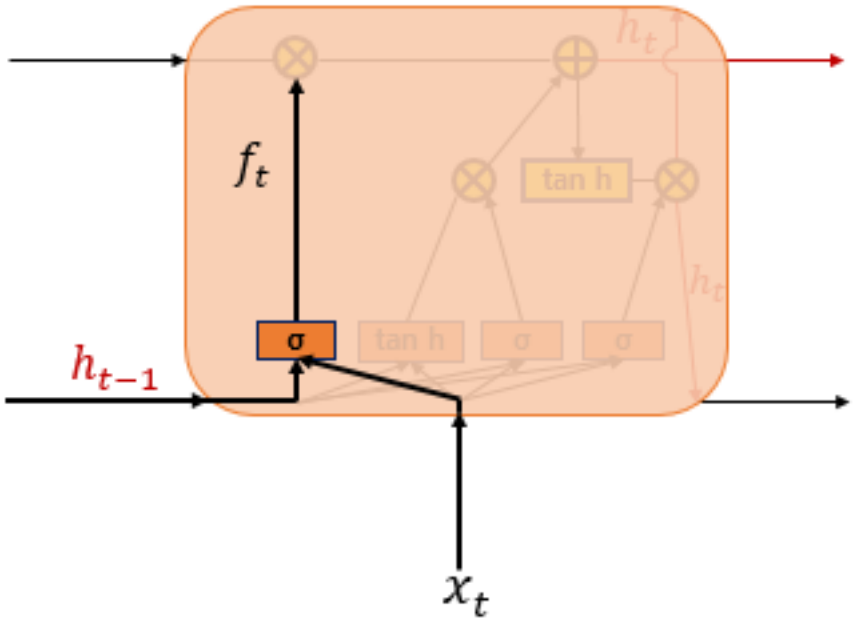


$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$
$$g_t = tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g)$$

02 장단기 메모리 Long Short-Term Memory

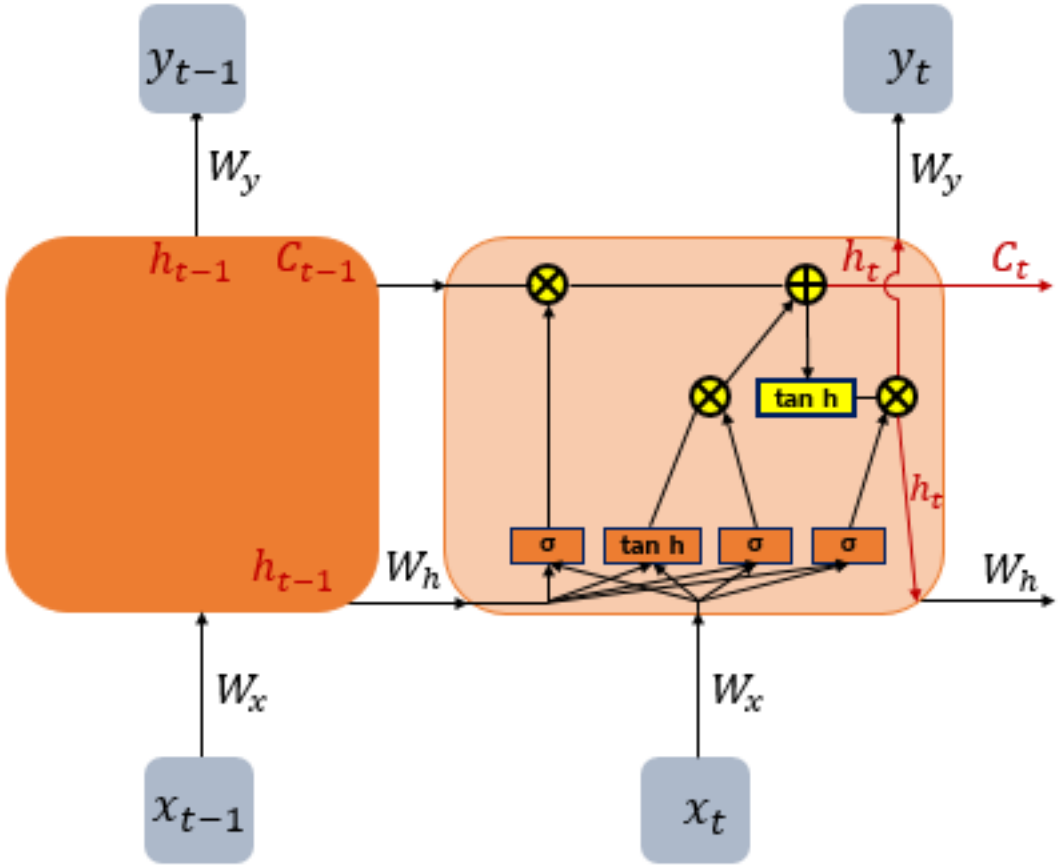


망각 게이트(Forget Gate)

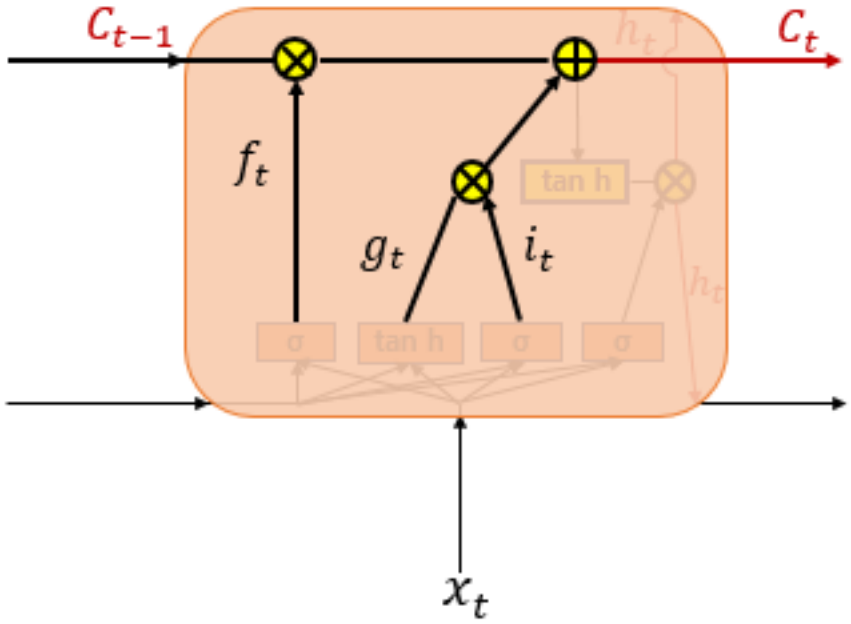


$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

02 장단기 메모리 Long Short-Term Memory

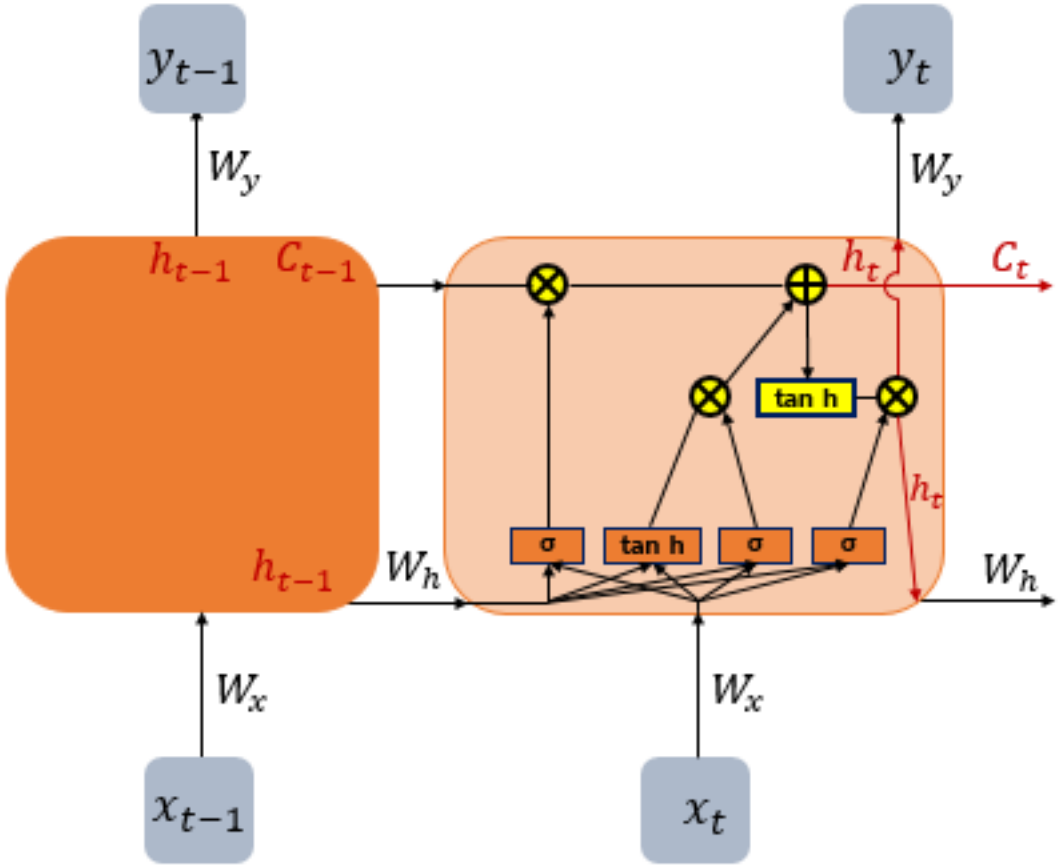


셀 상태(Cell State)

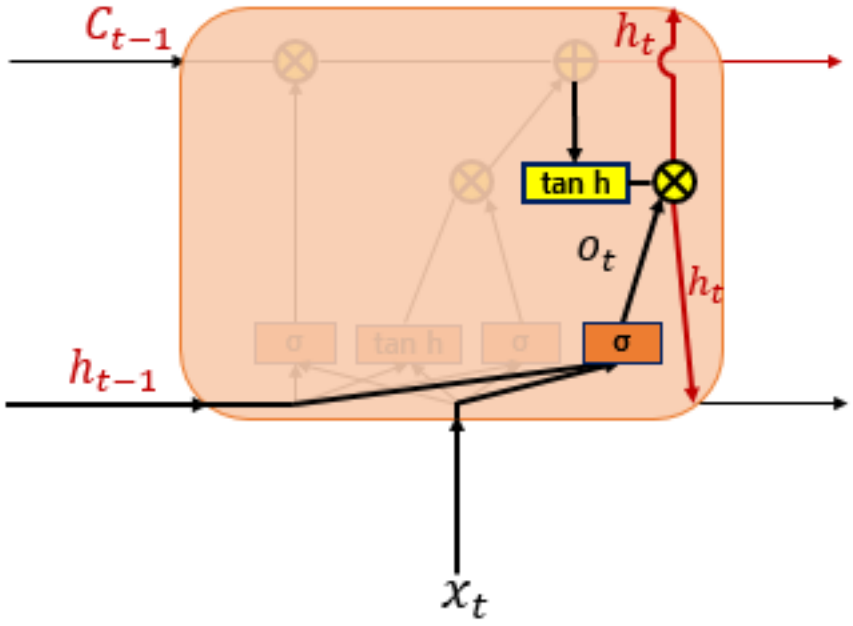


$$C_t = f_t \circ C_{t-1} + i_t \circ g_t$$

02 장단기 메모리 Long Short-Term Memory

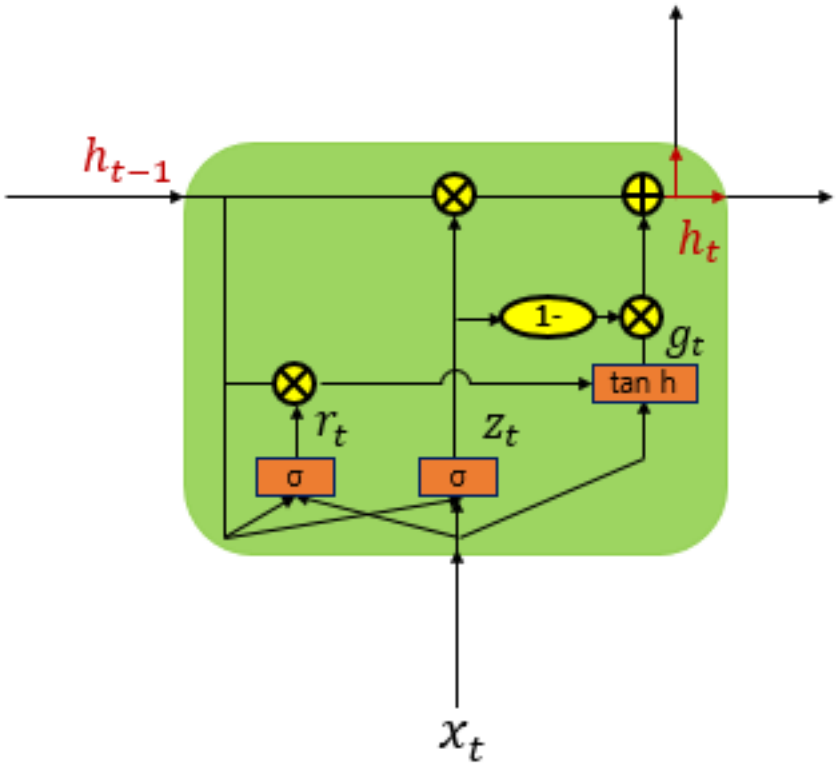


출력 게이트(Output Gate)



$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$
$$h_t = o_t \circ tanh(c_t)$$

03 게이트 순환 유닛 Gate Recurrent Unit



$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r)$$
$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z)$$
$$g_t = tanh(W_{hg}(r_t \circ h_{t-1}) + W_{xg}x_t + b_g)$$
$$h_t = (1 - z_t) \circ g_t + z_t \circ h_{t-1}$$

04 케라스의 SimpleRNN과 LSTM 이해하기

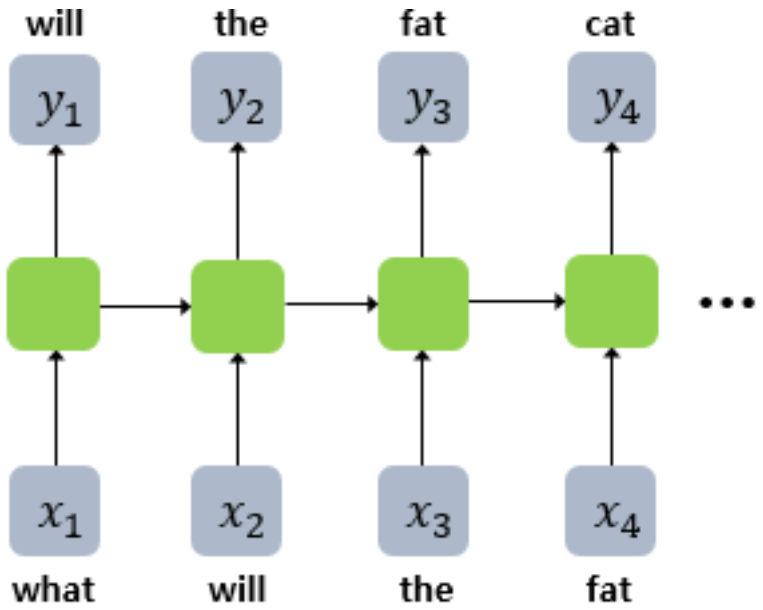
```
rnn = SimpleRNN(3)
# rnn = SimpleRNN(3, return_sequences=False, return_state=False)와 동일.
hidden_state = rnn(train_X)

print('hidden state : {}, shape: {}'.format(hidden_state, hidden_state.shape))
```

```
lstm = LSTM(3, return_sequences=False, return_state=True)
hidden_state, last_state, last_cell_state = lstm(train_X)

print('hidden state : {}, shape: {}'.format(hidden_state, hidden_state.shape))
print('last hidden state : {}, shape: {}'.format(last_state, last_state.shape))
print('last cell state : {}, shape: {}'.format(last_cell_state, last_cell_state.shape))
```

Copy

```
bilstm = Bidirectional(LSTM(3, return_sequences=False, return_state=True, \
                            kernel_initializer=k_init, bias_initializer=b_init, recurrent_initialize
r=r_init))
hidden_states, forward_h, forward_c, backward_h, backward_c = bilstm(train_X)

print('hidden states : {}, shape: {}'.format(hidden_states, hidden_states.shape))
print('forward state : {}, shape: {}'.format(forward_h, forward_h.shape))
print('backward state : {}, shape: {}'.format(backward_h, backward_h.shape))
```

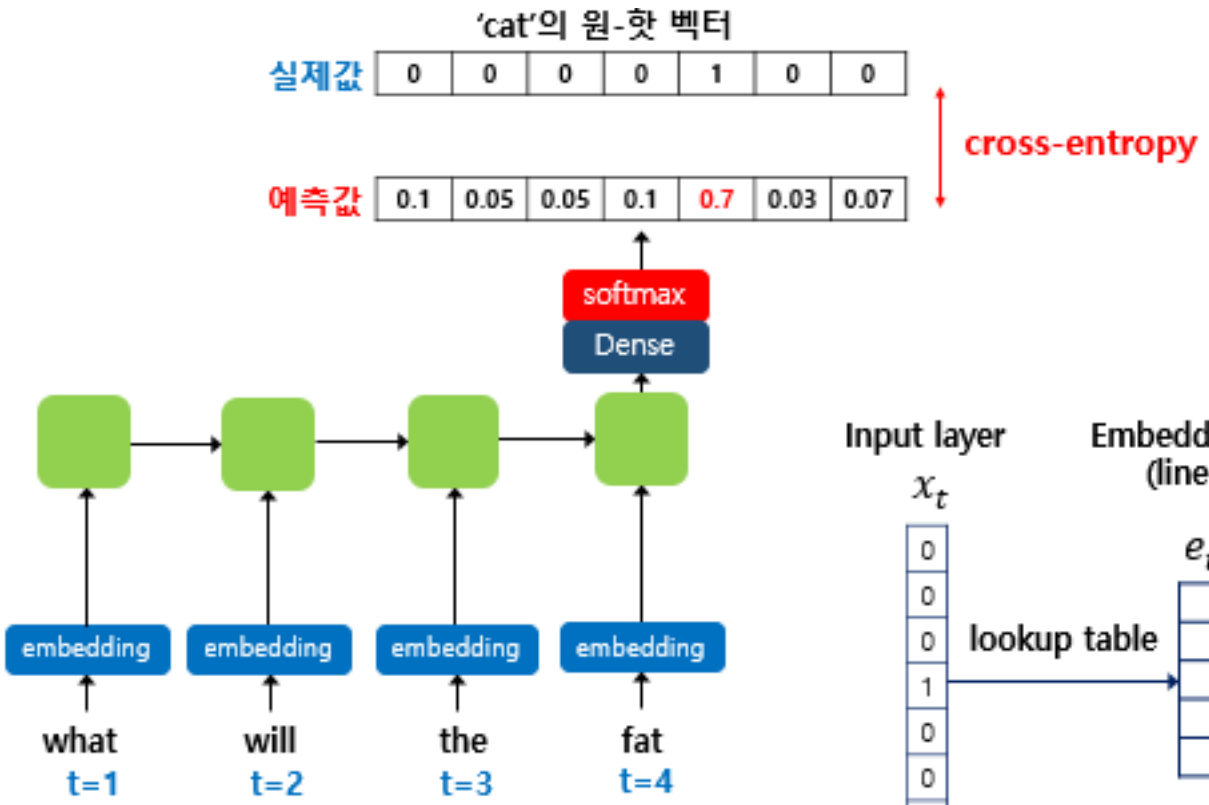
05 RNN 언어 모델 Recurrent Neural Network Language Model



RNNLM의 예측

05 RNN 언어 모델 Recurrent Neural Network Language Model

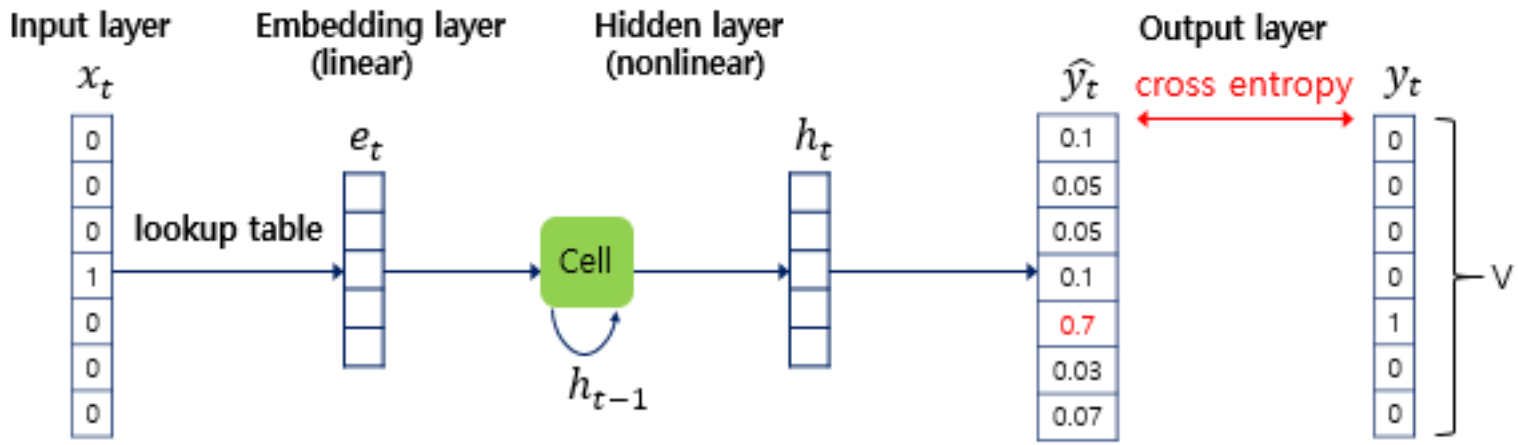
RNNLM의 학습



$$e_t = lookup(x_t)$$

$$h_t = tanh(W_x e_t + W_h h_{t-1} + b)$$

$$\hat{y}_t = softmax(W_y h_t + b)$$



06 RNN을 이용한 텍스트 생성 Text Generation using RNN

VanillaRNN

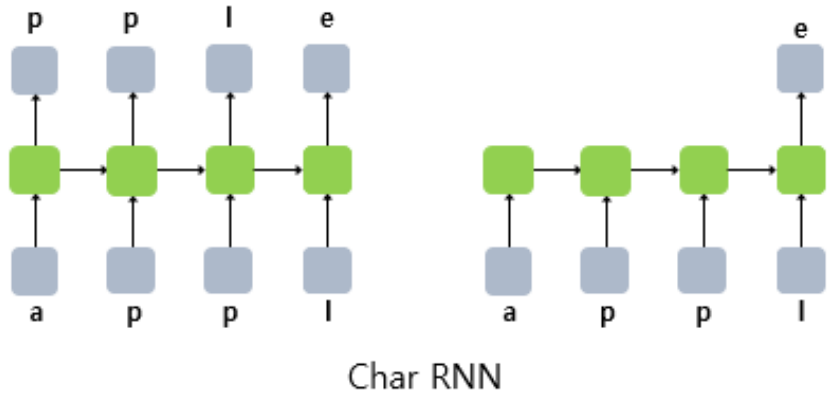
```
embedding_dim = 10
hidden_units = 32

model = Sequential()
model.add(Embedding(vocab_size, embedding_dim))
model.add(SimpleRNN(hidden_units))
model.add(Dense(vocab_size, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X, y, epochs=200, verbose=2)
```

LSTM

```
embedding_dim = 10
hidden_units = 128

model = Sequential()
model.add(Embedding(vocab_size, embedding_dim))
model.add(LSTM(hidden_units))
model.add(Dense(vocab_size, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X, y, epochs=200, verbose=2)
```

```
def sentence_generation(model, tokenizer, current_word, n): # 모델, 토크나이저, 현재 단어, 반복할 횟수
    init_word = current_word
    sentence = ''

    # n번 반복
    for _ in range(n):
        # 현재 단어에 대한 정수 인코딩과 패딩
        encoded = tokenizer.texts_to_sequences([current_word])[0]
        encoded = pad_sequences([encoded], maxlen=5, padding='pre')
        # 입력한 X(현재 단어)에 대해서 Y를 예측하고 Y(예측한 단어)를 result에 저장.
        result = model.predict(encoded, verbose=0)
        result = np.argmax(result, axis=1)

        for word, index in tokenizer.word_index.items():
            # 만약 예측한 단어와 인덱스와 동일한 단어가 있다면 break
            if index == result:
                break

        # 현재 단어 + ' ' + 예측 단어를 현재 단어로 변경
        current_word = current_word + ' '  + word

        # 예측 단어를 문장에 저장
        sentence = sentence + ' ' + word

    sentence = init_word + sentence
    return sentence
```

07 문자 단위 RNN Char RNN



Char RNN

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, TimeDistributed

hidden_units = 256

model = Sequ
model.add(LS
model.add(LS
model.add(T

model.compi
model.fit(tr
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
from tensorflow.keras.preprocessing.sequence import pad_sequences

hidden_units = 64

model = Sequential()
model.add(LSTM(hidden_units, input_shape=(X_data_one_hot.shape[1], X_data_one_hot.shape[2])))
model.add(Dense(vocab_size, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_data_one_hot, y_data_one_hot, epochs=100, verbose=2)
```

```
def sentence_generation(model, char_to_index, seq_length, seed_text, n):

    # 초기 시퀀스
    init_text = seed_text
    sentence = ''

    # 다음 문자 예측은 총 n번만 반복.
    for _ in range(n):
        encoded = [char_to_index[char] for char in seed_text] # 현재 시퀀스에 대한 정수 인코딩
        encoded = pad_sequences([encoded], maxlen=seq_length, padding='pre') # 데이터에 대한 패딩
        encoded = to_categorical(encoded, num_classes=len(char_to_index))

        # 입력한 X(현재 시퀀스)에 대해서 y를 예측하고 y(예측한 문자)를 result에 저장.
        result = model.predict(encoded, verbose=0)
        result = np.argmax(result, axis=1)

        for char, index in char_to_index.items():
            if index == result:
                break

        # 현재 시퀀스 + 예측 문자를 현재 시퀀스로 변경
        seed_text = seed_text + char

        # 예측 문자를 문장에 저장
        sentence = sentence + char

    # n번의 다음 문자 예측이 끝나면 최종 완성된 문장을 리턴.
    sentence = init_text + sentence
    return sentence
```

감사합니다!
Q&A