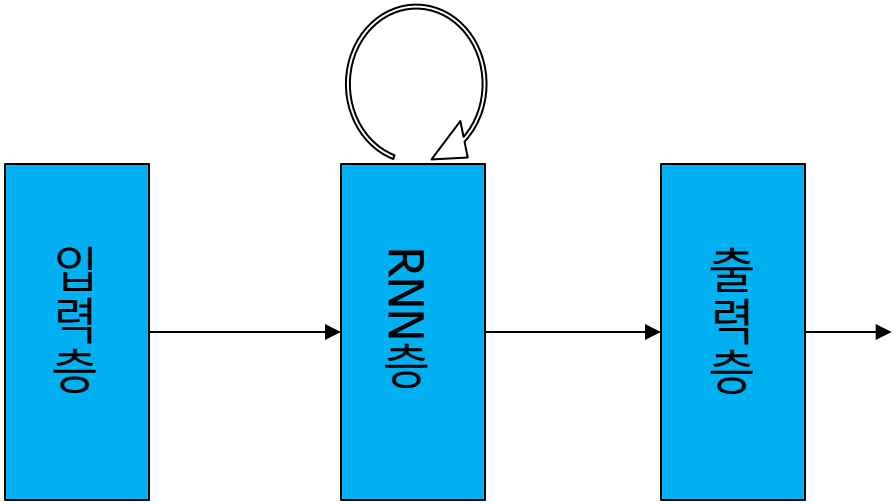
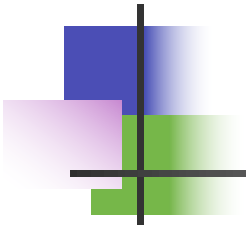
순환신경망 (Recurrent Neural Network)

Sang Yup Lee

RNN

- RNN (Recurrent Neural Networks)
  - 하나의 전체적인 신경망 모형이라기 보다는 하나의 층
  - 기본 신경망에서의 하나의 은닉층이라고 생각할 수 있음
  - 하나의 은닉층 (RNN 층이라고 함)이 순차적으로 여러 번 반복해서 사용
  - RNN층이 사용된 모형을 보통 RNN 모형이라고 함





RNN

- RNN (Recurrent Neural Networks)
  - Sequence data를 다루기에 적합
    - Sequence data: 어떤 것들이 순서를 가지고 연속적으로 나열되어 있는 데이터
    - 예)
      - 텍스트: 단어들의 sequence
      - 비디오: 이미지들의 sequence
  - 텍스트 데이터 분석에 적합
    - 단어들의 문맥적 의미 추출에 용이
    - 단어들의 간의 관계 정보 추출에 용이
    - 어떠한 단어들이 어떠한 순서로 언제 사용되었는지에 대한 정보 추출 용이
      ⇒ 텍스트가 가지고 있는 정보를 추출하기에 적합

RNN

- RNN을 사용한 대표적 텍스트 분석
  - 언어모형 (language model), 분류(예, 감성분석 등), 기계 번역 등
  - 언어 모형
    - 여러개의 단어들이 동시에 출현할 확률이나 특정한 단어들이 주어질 때 그 다음 나올 단어가 무엇인지를 예측하는데 사용되는 모형
    - Example
      - "I like the movie. The movie is ____" : is 다음에 나오는 단어를 맞히고자 할 때 RNN 사용 가능
      - is 다음에 나오는 단어를 정확하게 예측하기 위해서는 이전에 어떠한 단어들이 어떠한 순서를 가지고 사용되었는지에 대한 정보를 사용하는 것이 필요
      - The movie is 만을 사용하는 것 보다 그 앞 문장인 I like the movie라는 정보까지 같이 사용하는 것이 더 효과적

RNN

문서란? 하나 이상의 문장으로 구성된 텍스트의 단위
예) 하나의 신문기사, 하나의 영화평 등

- RNN을 사용한 대표적 텍스트 분석 (cont'd)
  - 감성분석
    - 문서 (혹은 문장)의 감성을 분석할 때 단순히 단어들의 출현 빈도만을 고려하는 것 보다 단어들이 다른 단어들과 어떠한 관계를 갖고 출현하였는지에 대한 정보를 사용하는 것이 필요 => 정답을 맞히는데 필요한 문서가 가지고 있는 정보를 더 잘 추출할 수 있다!
    - 예) "The movie was not fun."
    - fun 과 not이라는 단어들의 빈도 혹은 출현 여부 보다는 앞에 있는 not 이 fun 앞에 사용되었다라는 것을 고려한 경우에 보다 정확하게 전체 문장의 감성 파악 가능

RNN

전처리 과정이 끝난 후의 단어들로 구성된 문서라고 간주

- RNN 작동 방식
  - 텍스트 데이터를 이용해서 설명
  - 예제 텍스트
    - “the movie is fun”
  - RNN 작동 방식을 이해하기 위해서는 텍스트 데이터가 RNN에 어떠한 식으로 입력되는지를 이해하는 것이 필요
  - RNN에 텍스트 데이터를 입력하기 위해서는 각 단어을 저차원의 벡터로 표현하는 것이 필요
    - 이러한 저차원 벡터를 embedding vector라고 함
    - 여기서는 설명을 위해 각 단어가 100차원의 벡터로 표현된다고 가정

RNN

- RNN 작동 원리 (cont')
  - "the movie is fun"

100 차원

the	0.13	-2.31	...	1.69
movie	-1.11	3.59	...	-2.22
is	3.93	0.01	...	0.97
fun	-2.77	2.67	...	1.09

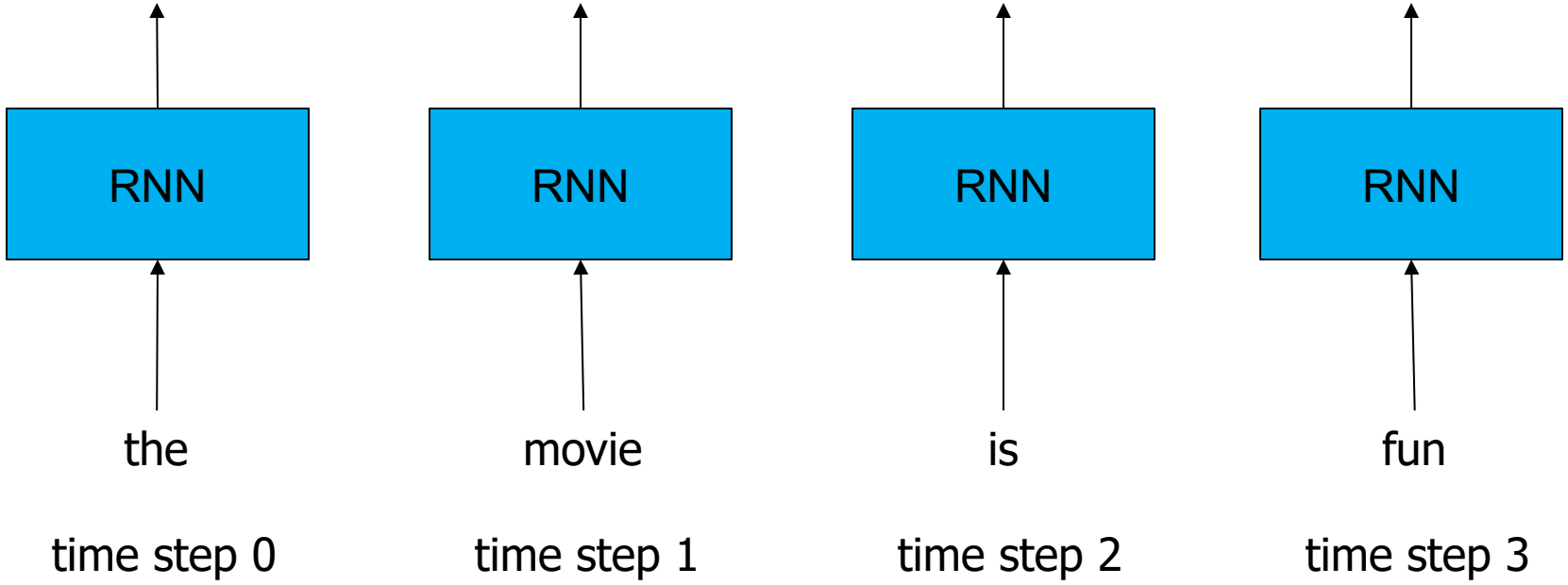
각 원소의 값도 학습을 통해서 그 값이 결정되는 파라미터임

초기에는 랜덤하게 설정되며, 학습을 통해서 업데이트 됨

  - RNN에는 각 단어의 벡터 정보가 순차적으로 입력
  - 즉, 처음에는 the, 두 번째는 movie, 세 번째는 is, 네번째는 fun에 대한 벡터 정보가 입력됨

RNN

동일한 RNN 층(즉, 은닉층)이 여러번 순차적으로 반복 사용되는 것임

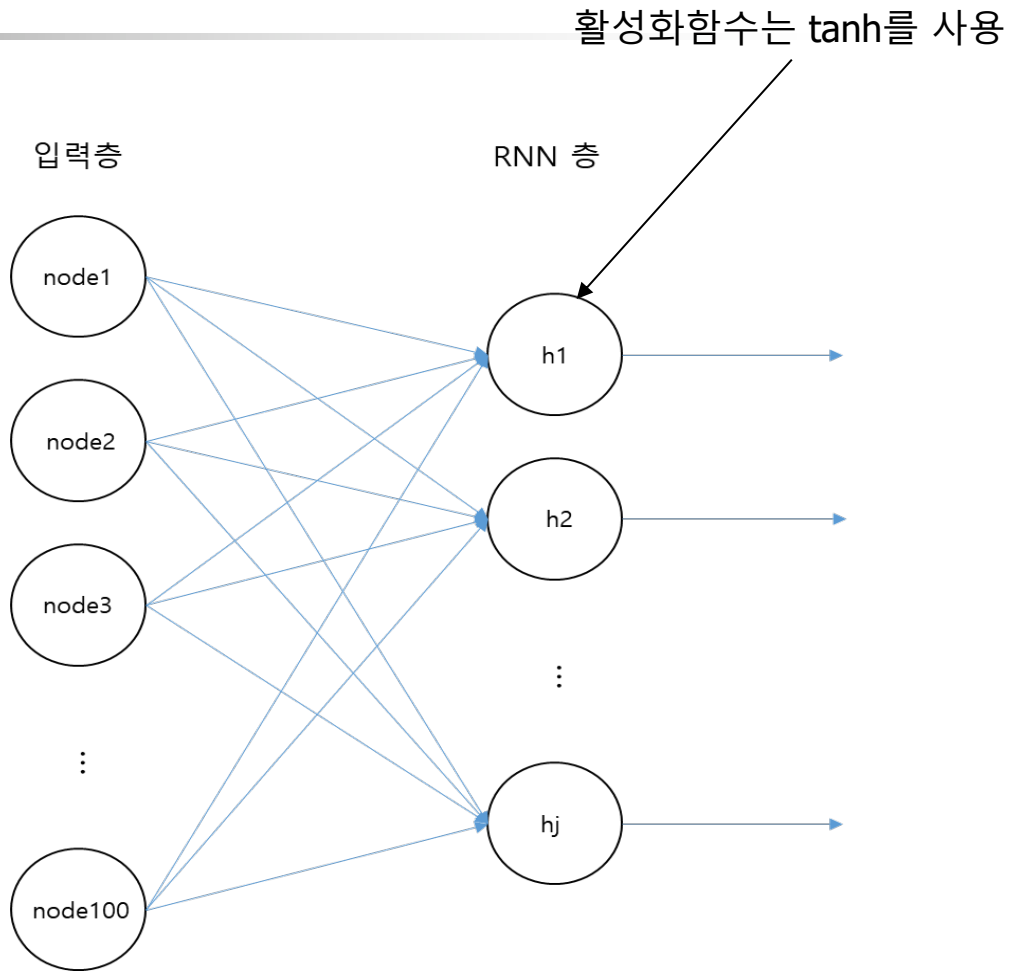


입력되는 순서를 나타내기 위해서 RNN에서는 보통 time step이라는 표현을 사용

RNN

- 입력층에는 각 단어의 정보가 입력
  - 입력노드의 수 = 임베딩 벡터의 원소의 수
  - 여기서 N = 100
  - 각 노드가 각 원소의 값을 입력 받음
- 설명을 위해 편향 노드는 생략

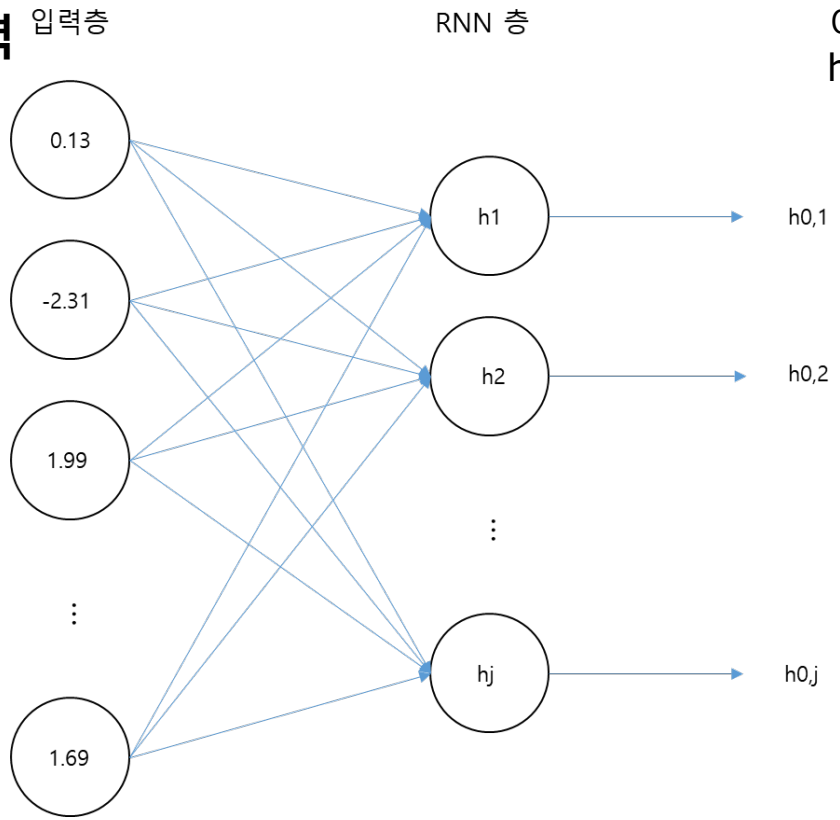
- RNN 층은 은닉층과 유사
  - 즉, 여러개의 노드로 구성되어 있음
  - 노드의 수는 사용자가 결정
  - RNN 층이 출력하는 값을 hidden state라고 함



RNN

- **첫번째 단어 정보 입력**

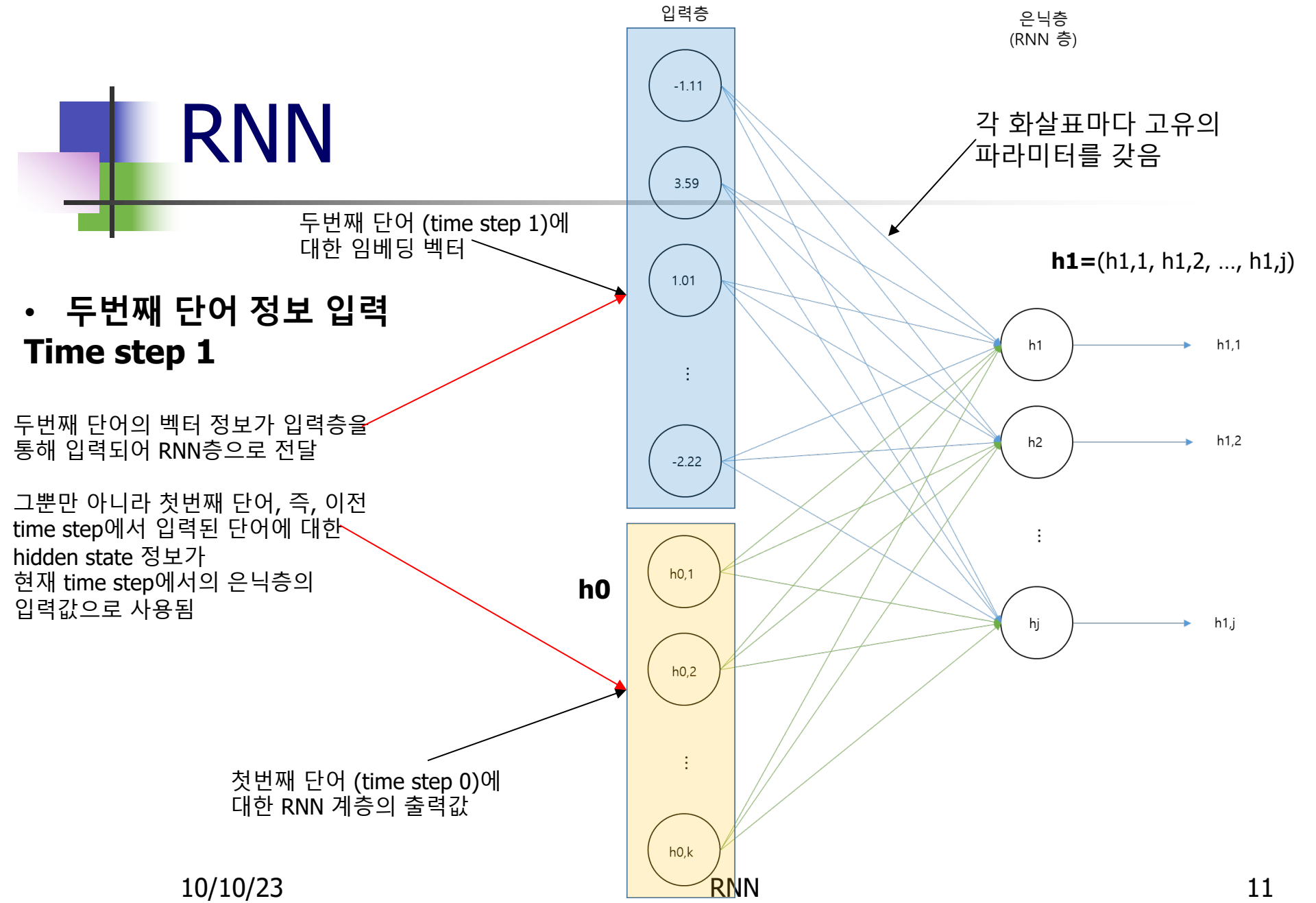
Time step 0



RNN 층에서 출력하는 이러한 값을 hidden state 라고 함

h0=(h0,1, h0,2, …, h0,j)

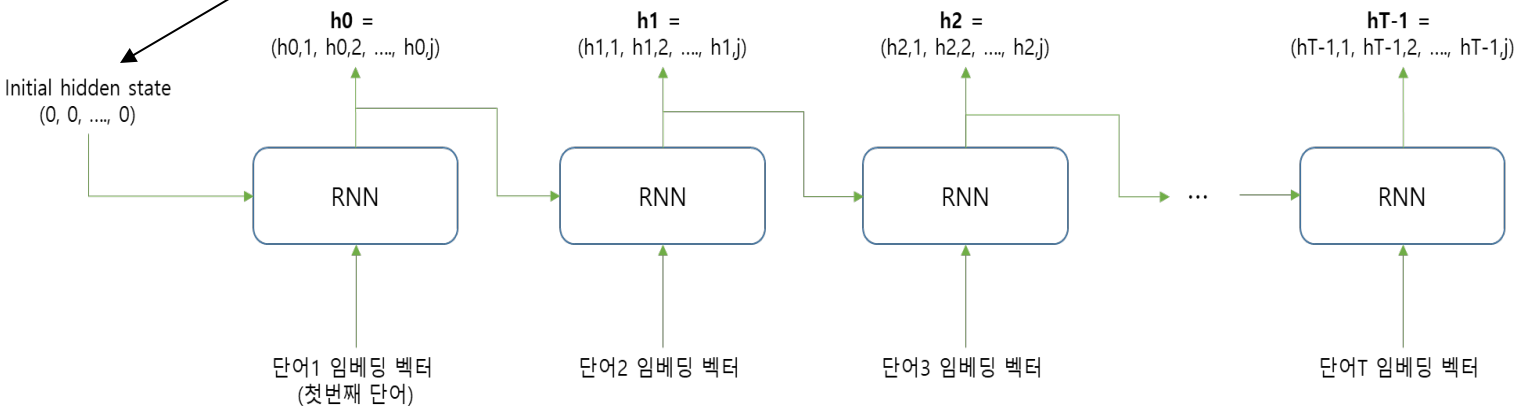
RNN


입력층
은닉층
(RNN 층)
-1.11
3.59
1.01
-2.22
h0,1
h0,2
h0,k
h1
h2
hj
h1,1
h1,2
h1,j
각 화살표마다 고유의
파라미터를 갖음
h1=(h1,1, h1,2, …, h1,j)
두번째 단어 (time step 1)에
대한 임베딩 벡터
• 두번째 단어 정보 입력
Time step 1
두번째 단어의 벡터 정보가 입력층을
통해 입력되어 RNN층으로 전달
그뿐만 아니라 첫번째 단어, 즉, 이전
time step에서 입력된 단어에 대한
hidden state 정보가
현재 time step에서의 은닉층의
입력값으로 사용됨
h0
첫번째 단어 (time step 0)에
대한 RNN 계층의 출력값

RNN

초기 hidden state의 값은 보통 0으로 초기화 됩니다.
(이후부터는 이부분은 명시적으로 필요하지 않는 이상 생략하고 설명)

RNN의 구조



보통 **hT-1** 이 RNN 층이 출력하는 (입력된) 텍스트 데이터에 대한 최종값 ⇒ 이 값이 다음 계층으로 전달
hT-1 에는 이전에 입력된 단어들의 정보가 저장되어 있음
이전 단어들에 대한 hidden state 정보도 다음 계층으로 전달할 수 있음

RNN

- 행렬과 벡터로 표현하기

$\mathbf{h}_t$: 특정 time step 에서 RNN층을 통해 출력되는 값

$$\mathbf{h}_t = \tanh(\mathbf{x}_t \cdot \mathbf{W}_\mathbf{x} + \mathbf{h}_{t-1} \cdot \mathbf{W}_h + \mathbf{b})$$

$\mathbf{x}_t\mathbf{W}_\mathbf{x} + \mathbf{h}_{t-1}\mathbf{W}_h + \mathbf{b}$의 차원수와 $\mathbf{h}_t$의 차원수가 같게 됩니다!

$\mathbf{x}_t$: 해당 time step에서 입력되는 단어의 임베딩 벡터

$\mathbf{h}_{t-1}$: 이전 time step에서 전달되는 hidden state

$\mathbf{W}_h$ → **time step과 상관없이 가중치가 동일하다. 즉, h는 가중치를 공유(parameter sharing) 한다.**

$\mathbf{W}_\mathbf{x}$ → **time step과 상관없이 가중치가 동일하다. 즉, x는 가중치를 공유한다.**

RNN

- 행렬과 벡터로 표현하기 (cont'd)

$$\mathbf{W_x} = \begin{bmatrix} W_{0,0}^{(x)} & W_{0,1}^{(x)} & \cdots & W_{0,j-1}^{(x)} \\ W_{1,0}^{(x)} & W_{1,1}^{(x)} & \cdots & W_{1,j-1}^{(x)} \\ W_{2,0}^{(x)} & W_{2,1}^{(x)} & \cdots & W_{2,j-1}^{(x)} \\ \vdots & \vdots & \ddots & \vdots \\ W_{k-1,0}^{(x)} & W_{k-1,1}^{(x)} & \cdots & W_{k-1,j-1}^{(x)} \end{bmatrix} \quad \mathbf{W_h} = \begin{bmatrix} W_{0,0}^{(h)} & W_{0,1}^{(h)} & \cdots & W_{0,j-1}^{(h)} \\ W_{1,0}^{(h)} & W_{1,1}^{(h)} & \cdots & W_{1,j-1}^{(h)} \\ W_{2,0}^{(h)} & W_{2,1}^{(h)} & \cdots & W_{2,j-1}^{(h)} \\ \vdots & \vdots & \ddots & \vdots \\ W_{j-1,0}^{(h)} & W_{j-1,1}^{(h)} & \cdots & W_{j-1,j-1}^{(h)} \end{bmatrix}$$

$$\mathbf{x_t} = (x_{t,0}, x_{t,1}, \ldots, x_{t,k-1})$$

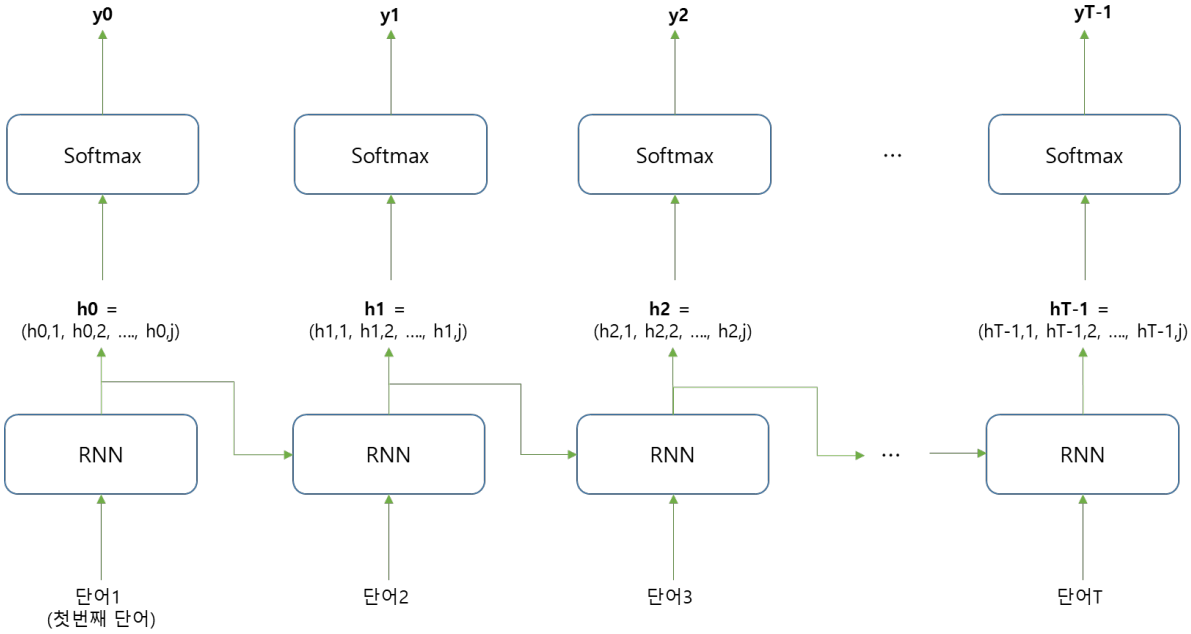
$$\mathbf{h_{t-1}} = (h_{t-1,0}, h_{t-1,1}, \ldots, h_{t-1,j-1})$$

$$\mathbf{b} = (b_0, b_1, \ldots, b_{j-1})$$

RNN

- RNN 활용의 예 1: 언어모형
  - 언어 모형에 RNN이 적용되는 경우에는 각 단어에 대한 RNN의 출력값 (즉, ht)은 다음 단어를 예측하는 목적으로 사용

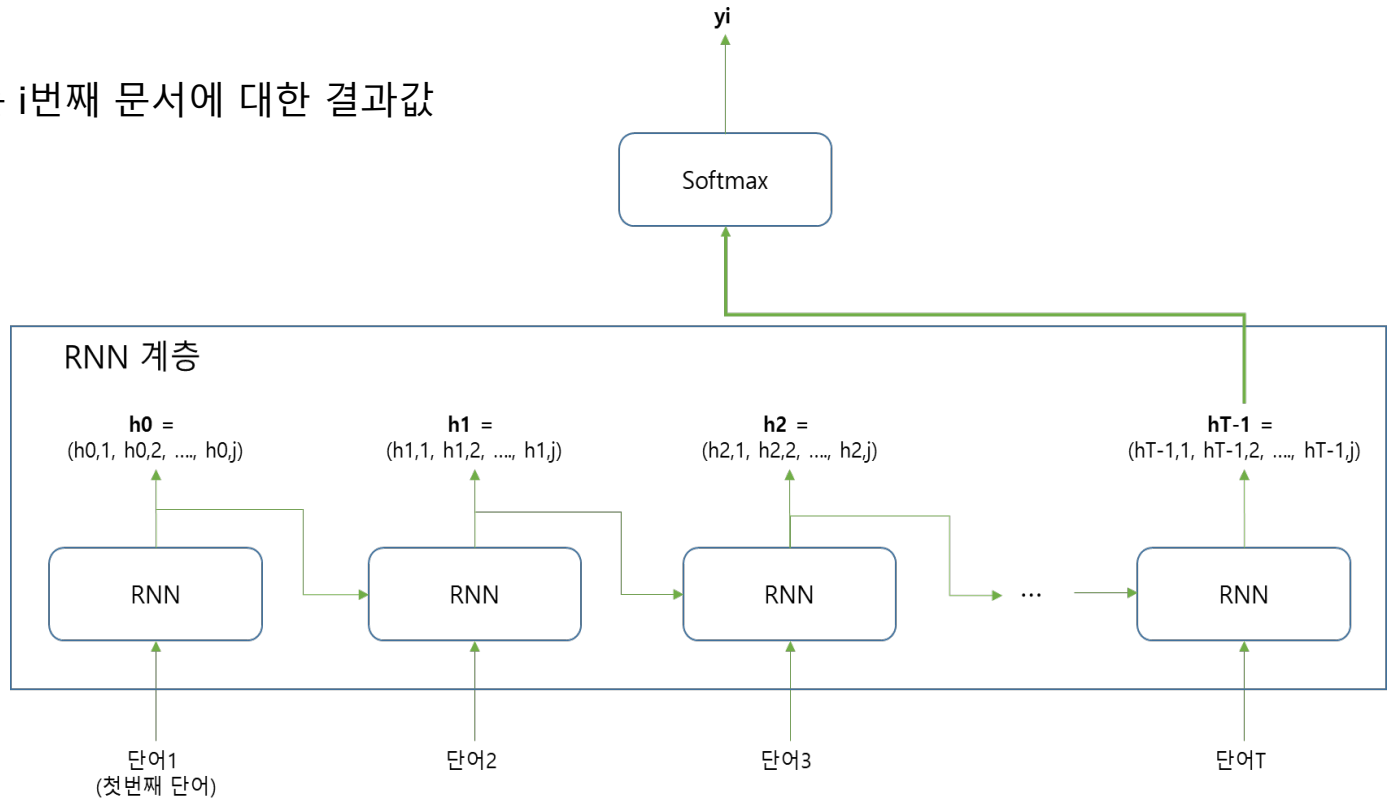
- Vocabulary size=n 인 경우, softmax 층에 존재하는 노드의 수 = n
- 각 노드가 특정 단어의 확률 의미



RNN

- RNN 활용의 예 1: 감성분석 (sentiment analysis)

yi벡터는 i번째 문서에 대한 결과값

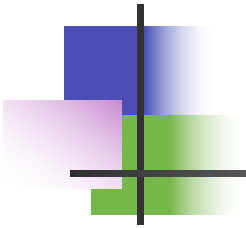


Python coding

- IMDb 영화평 감성분석
  - Keras에서 제공하는 imdb 영화평 데이터셋을 사용
    - https://www.imdb.com/
  - 주의해야할 점
    - 전체 데이터(말뭉치라고함)에 존재하는 전체 단어들 중에서 빈도수를 기준으로 상위의 몇개 단어들만 사용
    - 단어들을 인덱스 번호를 사용해서 표현
    - 각 문서를 동일한 길이로 표현
      - 하나의 문서가 T개의 단어로 구성되었다라고 가정할 수 있다.
      - 만약, T개 보다 단어가 많은 경우에는 나머지 단어를 제거 (truncate)하게 되고, 만약 T 개 보다 단어가 적다면 패딩 (보통 Zero padding) 방법을 사용.

Python coding

- IMDb 영화평 감성분석 (cont'd)
  - 파이썬 코드: RNN_imdb_example.ipynb
  - Keras
    - SimpleRNN 사용
    - https://keras.io/api/layers/recurrent_layers/simple_rnn/

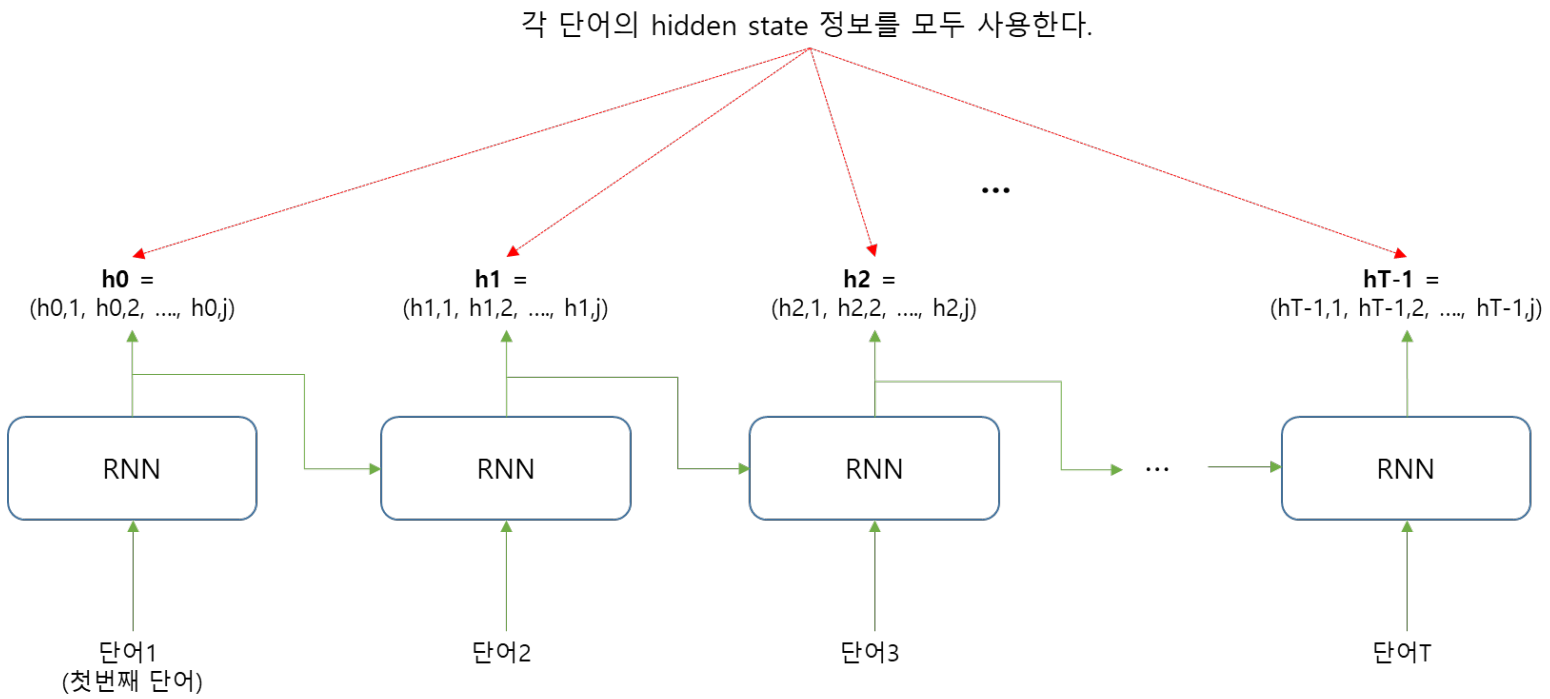


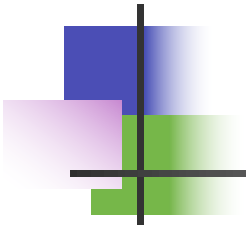
RNN

- 한글 텍스트에 대해서
  - 데이터
    - “2016_filtered_review.txt” 파일
  - See “RNN_sentiment_Korean.ipynb”
  - 순서
    - 기본적인 전처리 수행
    - 전체의 단어 중 상위 몇개의 단어를 사용할 것인지를 결정
    - 선택된 단어에 index 번호를 부여
    - 각 문서를 선택된 단어들의 index를 이용해서 표현
    - 문서의 길이를 동일하게 맞춰줌 (zero padding)
    - 그 이후의 과정은 영어 텍스트와 동일

RNN

- 각 단어의 hidden state를 모두 사용하기



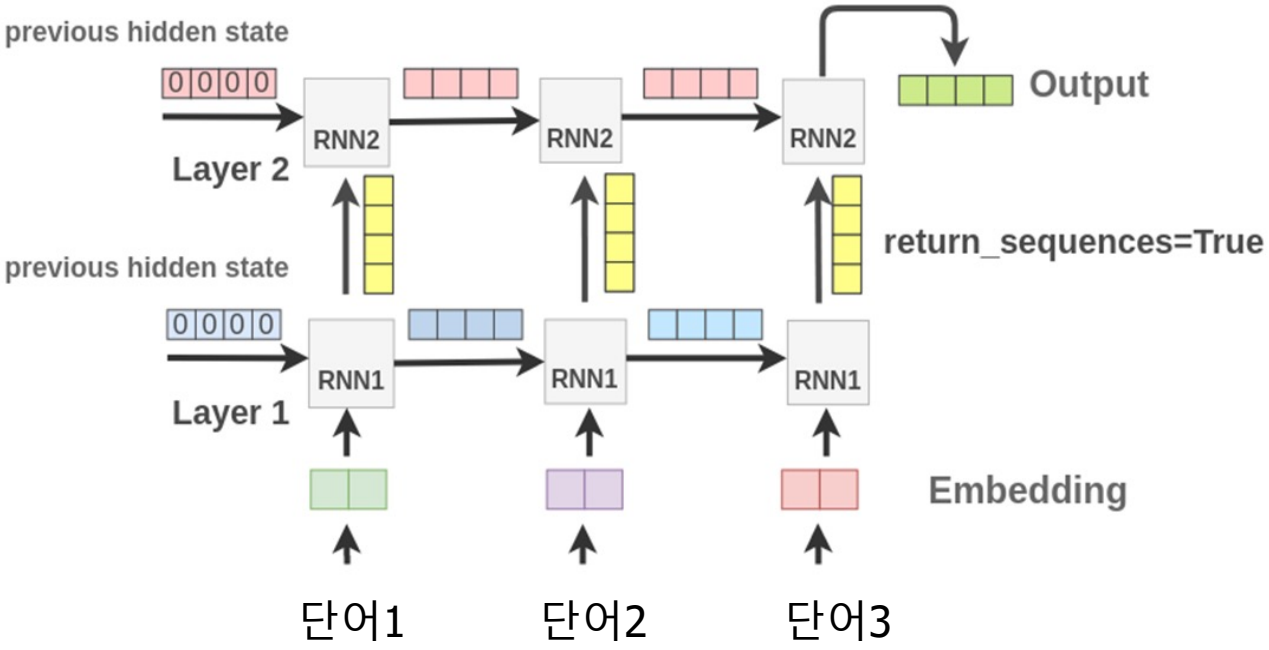


RNN

- 각 단어의 hidden state를 모두 사용하기 (cont'd)
  - 크기 두가지 방법
    - 1) 이어붙이기 (concatenation)
      - See "IMDb_RNN_return_seq_true_concat.ipynb"
    - 2) 평균 (mean)
      - See "IMDb_RNN_return_seq_true_mean.ipynb "

RNN

- 여러개의 RNN층 사용하기 (Stacked RNN)



Source: https://www.researchgate.net/figure/Illustrations-of-normal-RNN-stacked-RNN-and-bidirectional-RNN_fig7_311839720

RNN

- Stacked RNN (cont'd)
  - See "IMDb_stacked_RNN_example.ipynb"

```
model1 = models.Sequential()
model1.add(layers.Embedding(100, 128))
model1.add(layers.SimpleRNN(64, return_sequences = True))
model1.add(layers.SimpleRNN(32))
model1.add(layers.Dense(2, activation = 'softmax'))
model1.summary()
```

위와 같이 하면 두번째 SimpleRNN에 입력되는 벡터의 차원은 64가 된다. 즉, 각 단어에 대해서 64차원의 임베딩 벡터 정보가 입력된다고 생각할수도 있다.