

1. RNN

Decoder - Encoder를 이용해 예측하는 Seq2Seq 공부

인코더 디코더로 구성되어 있음.

인코더 : 입력 문장의 모든 단어를 순서대로 입력받은 뒤에 마지막 그 모든 단어 정보들을 출력하는 것을 **context vector**.

디코더 : context vector를 받아서 번역된 단어로 한계로 순차적으로 출력.

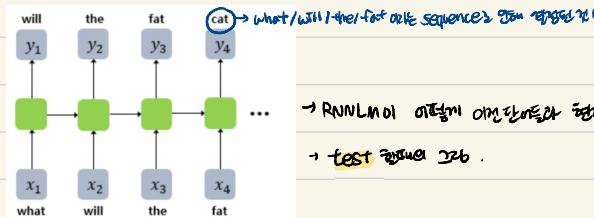
인코더/디코더 구조는 RNN 형태임.

[RNN architecture]

time-step(시그널)이라는 개념이 생김 \rightarrow 그 시그널을 전달하는 방식은 Timestep

RNN은 인출 모델이 training하는 방법

주어진 단어 sequence를 통해 다음 단어를 예측하는 문제



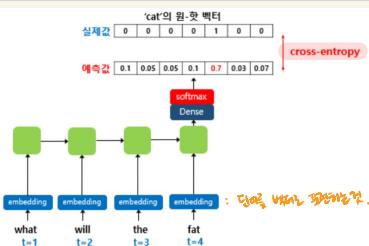
\rightarrow RNN들이 어떤 이미지나 텍스트와 같은 데이터를 받아서 그 데이터를 다음 단어를 예측하는지 보여주는 그림.
 \rightarrow test case example.

training 예제. Ex "What will the fat cat sit on"라는 training samples의 예제

"What will the fat cat sit" \rightarrow 이 sequence를 모델이 입력으로 넣으면 "will the fat cat sit on"을 예측하도록 훈련.

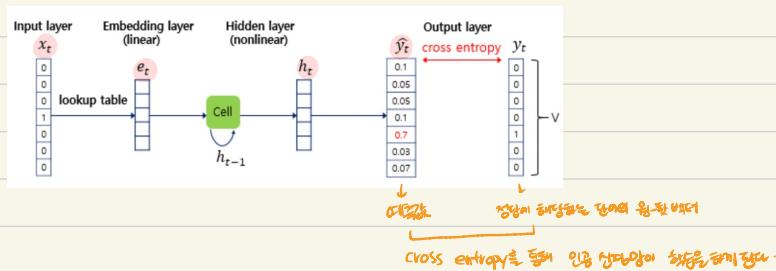
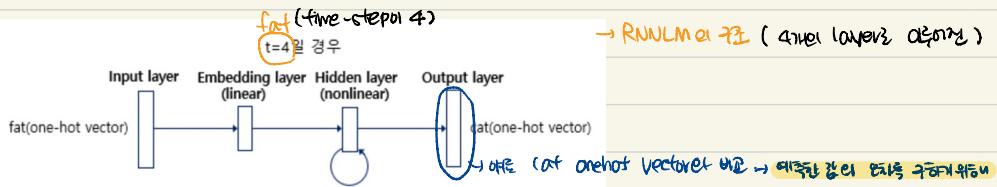
모든 RNN 훈련기법은 **teacher forcing**이라고 함

\rightarrow test diagram \rightarrow 예상된 출력과 t+1 시점의 입력으로 들어가는 RNN 단계를 훈련시킬 때 사용되는 훈련기법
(예측 X, 실제 정답 (label))



time step 예측
softmax

model이 예측한 결과 쉽게 table과 같이 흐지를 계산하기 위한 공식함수 : cross-entropy



① 현 time-step의 원-핫 벡터 자리를 임베딩 벡터로 embedding layer!

임베딩 벡터를 갖는 projection layer

원-핫 벡터의 차원.

② 단어 간의 크기가 V일 때, 임베딩 벡터의 크기를 M으로 설정하면 → 각 단어 word는 embedding layer에서 $V \times M$ 크기의 임베딩 벡터로 표현됨

$$(그림) V = 7, N = 5 \Rightarrow \text{임베딩 행렬} = 7 \times 5 \text{ 행렬}$$

↳ 예전에 대화하는 다른 기관에서도 행렬 사용함.

$$\text{임베딩} = \text{lookup}(x_t)$$

Classification 문제의 cross entropy는 cost function / loss function의 일종.

[cross entropy의 정의와 의미]

→ Loss function의 한 종류 (cost function의 일종)

→ 툴박수도 있는 짚은 행렬을 고려한 조건으로 인증성을 두 가지 형태로 정의함

→ softmax를 거친 결과값은 확률을 가진 vector라!

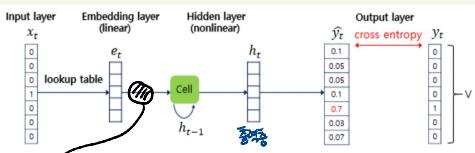
ex) 0:개, 1:고양이 \rightarrow softmax를 거친 결과값이 $[0.8, 0.2]$ 이면, 여기서 one-hot encoding은 확률 $[1, 0] \rightarrow$ 가능하다.

$$\rightarrow \text{Cross entropy} = \frac{\sum_i p(x)_i \log(p(x)_i)}{\text{실제값}} - \frac{\sum_i p(x)_i \log(Q(x)_i)}{\text{예상값}}$$

이 output이 실제 label값인 경우
줄여서 loss function = cross entropy

→ 실체값이 예상값이 다를 때 00, 같을 때 0

→ 예상값이 실제값과 같을 때 cross entropy는 loss function으로 두는 것을 추천하는 방향으로 학습.



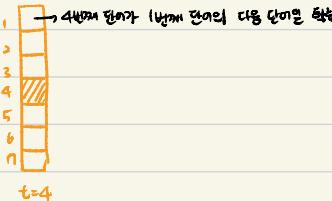
③ 임베딩 벡터는 hidden layer에서 input으로 온전히 부여된 h_{t-1} 과 함께 사용된다 \rightarrow 현재 온전히 부여된 h_t 를 계산.

$$\text{원형}: h_t = \tanh(Wxe_t + Whh_{t-1} + b)$$

④ 출력층에서는 템플릿으로 softmax 사용

$$V\text{차원의 vector} \xrightarrow{\text{Softmax}} V\text{차원의 벡터} (\text{0이 1사이의 확률. 합계}=1) \\ t\text{시간의 예측값} = \hat{y}_t$$

$$\text{원형}: \hat{y}_t = \text{softmax}(Wyh_t + b)$$



⑤ \hat{y}_t 같은 7개 차원의 확장하는 단어인 원-핫 벡터의 값은 가중치로 쓰이다.

날짜값을 확장하는 단어는 y_t 로 칭한다. \hat{y}_t 와 y_t 가 가중치로 쓰여 확장된다 \Rightarrow **연산학습**을 **cross-entropy**로 4종.

\Rightarrow 예전과는 아주 다르면서 \rightarrow 가중치 허용률도 확장됨 \rightarrow 이 과정에서 **임베딩 벡터값도 확장**

* 임베딩 행렬 E

: look up table의 대상이 되는 table