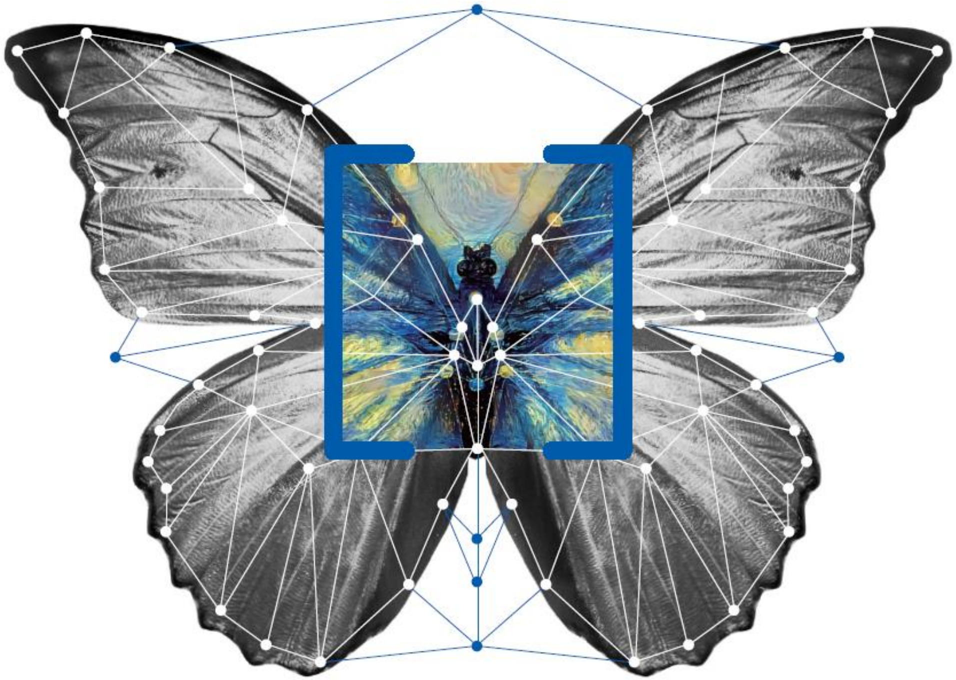
한빛아카데미
Hanbit Academy, Inc.



MACHINE LEARNING 기계 학습

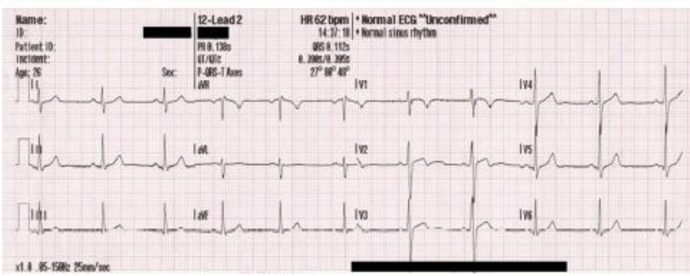
오일석 지음

8장. 순환 신경망

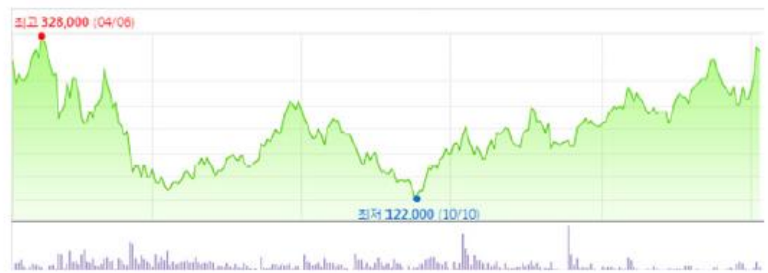
PREVIEW

시간성 데이터

- 특징이 순서를 가지므로 순차 데이터라 부름(이전 장에서 다룬 데이터는 어느 한 순간에 취득한 정적인 데이터이고 고정 길이임)
- 순차 데이터는 동적이며 보통 가변 길이임



(a) 심전도 신호



(b) 주식 시세

(c) 음성 신호

내려갈 때 보았네 올라갈 때 보지 못한 그 꽃

(d) 문장

ATGCTTCGGCAAGACTCAAAAAATA

(e) 유전자 열

그림 8-1 순차 데이터

PREVIEW

- 순환 신경망과 LSTM
  - 순환 신경망은 시간성 정보를 활용하여 순차 데이터를 처리하는 효과적인 학습 모델
  - 매우 긴 순차 데이터(예, 30단어 이상의 긴 문장)를 처리하는 데에는 장기 의존성을 잘 다루는 LSTM을 주로 사용(LSTM은 선별 기억 능력을 가짐)

- 최근에는 순환 신경망을 생성 모델로 사용
  - 예, CNN과 LSTM이 협력하여 자연 영상에 주석을 다는 문제를 풂(8.5.3절)

각 절에서 다루는 내용

8.1절_ 시간성을 지닌 순차 데이터의 성질과 표현 방법을 기술한다.

8.2절_ 순환 신경망의 구조와 동작, BPTT 학습 알고리즘을 설명한다.

8.3절_ 장기 문맥 의존성을 적절히 처리하는 일의 중요성을 소개한다.

8.4절_ 게이트를 이용하여 장기 문맥 의존성을 처리하는 LSTM 모델을 설명한다.

8.5절_ 응용 사례로서 언어 모델, 기계 번역, 영상 주석 생성을 소개한다.

8.1 순차 데이터

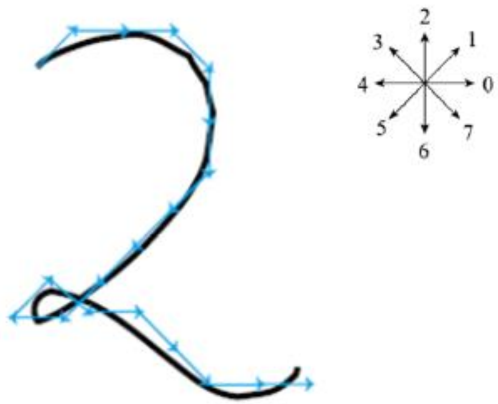
- 8.1.1 순차 데이터의 표현
- 8.1.2 순차 데이터의 특성

- 많은 응용
  - 심전도 신호를 분석하여 심장 이상 유무 판정
  - 주식 시세 분석하여 사고 파는 시점 결정
  - 음성 인식을 통한 지능적인 인터페이스 구축
  - 기계 번역기 또는 자동 응답 장치 제작
  - 유전자 열 분석을 통한 치료 계획 수립 등

8.1.1 순차 데이터의 표현

순차 데이터의 예시

- 온라인 숫자와 3채널 심전도 신호



x=100766555541707700

(a) 온라인 숫자

(b) 심전도 신호(3채널)

그림 8-2 순차 데이터의 표현

8.1.1 순차 데이터의 표현

- 순차 데이터의 일반적 표기
  - 벡터의 벡터(벡터의 요소가 벡터임)

$$\mathbf{x} = \left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(T)}\right)^{\mathrm{T}} \tag{8.1}$$

  - 온라인 숫자의 요소는 1차원, 심전도의 요소는 3차원
    - 예, 심전도 신호(초당 100번 샘플링하고 2분간 측정한다면 길이는 T=12000)

$$\mathbf{x} = \left(\begin{pmatrix}0.3\\0.1\\0.2\end{pmatrix}, \begin{pmatrix}0.5\\0.6\\0.4\end{pmatrix}, \cdots\ \cdots\right)^{\mathrm{T}}$$

- 훈련집합 $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$, $\mathbb{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n\}$
  - 각 샘플은 식 (8.2)로 표현

$$\mathbf{x} = \left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(T)}\right)^{\mathrm{T}}, \quad \mathbf{y} = \left(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \cdots, \mathbf{y}^{(L)}\right)^{\mathrm{T}} \tag{8.2}$$

8.1.1 순차 데이터의 표현

- 텍스트 순차 데이터의 표현
  - 예, 기계 번역에서 입력 $\mathbf{x}$가 "April is the cruelest month."이고 출력 $\mathbf{y}$가 "사월은 가장 잔인한 달"일 때, 식 (8.2) 표기법으로 어떻게 표현할까?

- 사전을 사용하여 표현
  - 사전 구축 방법
    - 사람이 사용하는 단어를 모아 구축 또는 주어진 말뭉치를 분석하여 단어를 자동 추출하여 구축
    - 예, 영어를 불어로 번역하는 논문 [Cho2014b]에서는 사용 빈도가 가장 높은 3만 개 단어로 사전 구축함
  - 사전을 사용한 텍스트 순차 데이터의 표현 방법
    - 단어가방BoW(bag of words)
    - 원핫 코드
    - 단어 임베딩

8.1.1 순차 데이터의 표현

단어가방

- 단어 각각의 빈도수를 세어 m차원의 벡터로 표현(m은 사전 크기)
- 한계
  - 정보 검색에 주로 사용되지만, 기계 학습에는 부적절(“April is the cruelest month”와 “The cruelest month is April”은 같은 특징 벡터로 표현되어 시간성 정보가 사라짐)

원핫 코드

- 해당 단어의 위치만 1로 표시
- 예, “April is the cruelest month”는 $\mathbf{x} = \left((0,0,1,0,0,0,\cdots)^{\mathrm{T}}, (0,0,0,0,1,0,\cdots)^{\mathrm{T}}, \cdots\right)^{\mathrm{T}}$로 표현 ← m차원 벡터를 요소로 가진 5차원 벡터
- 한계
  - 한 단어를 표현하는데 m차원 벡터를 사용하는 비효율성
  - 단어 간의 유사도를 측정하는 기능이 없음

8.1.1 순차 데이터의 표현

■ 단어 임베딩

- 단어 사이의 상호작용을 분석하여 새로운 공간으로 변환(보통 m보다 훨씬 낮은 차원으로 변환). 변환 과정은 학습이 말뭉치를 훈련집합으로 사용하여 알아냄
- 예, [Cho2014b]는 m=30000차원을 620차원으로 변환

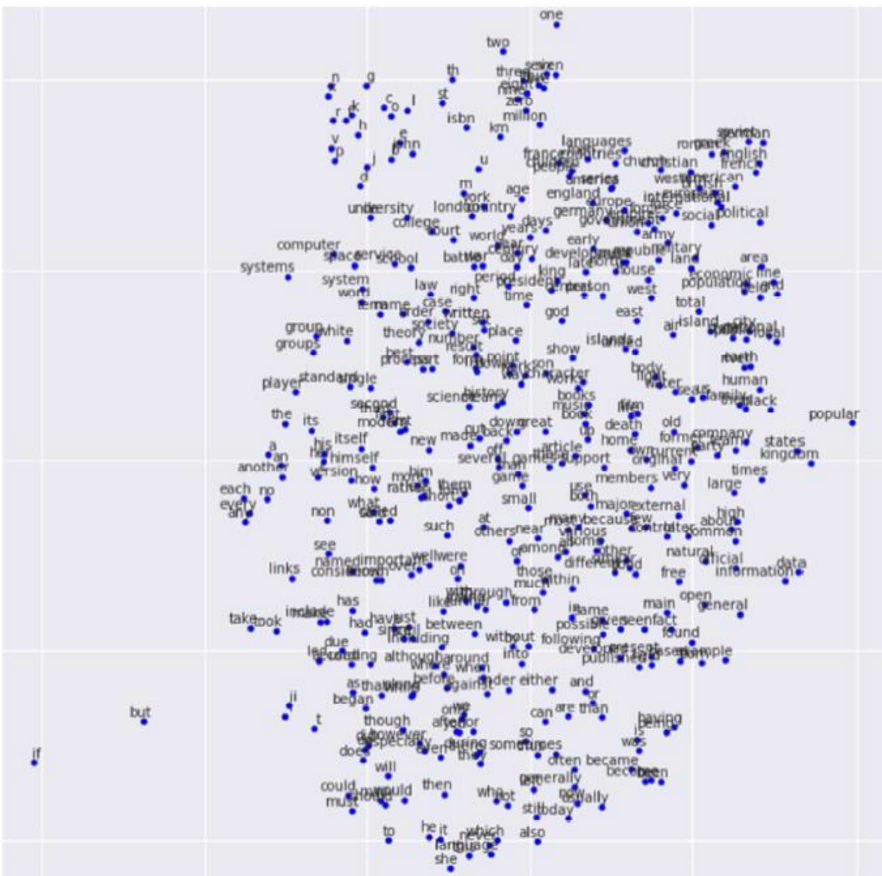


그림 8-3 단어 임베딩

word2vec 소프트웨어 사용
https://code.google.com/archive/p/word2vec/

8.1.2 순차 데이터의 특성

- 특징이 나타나는 순서가 중요
  - "내려갈 때 보았네."를 "때 내려갈 보았네."로 바꾸면 의미가 크게 훼손
  - 비순차 데이터에서는 순서를 바꾸어도 무방

- 샘플마다 길이가 다름
  - [그림 8-2]의 예제
  - 순환 신경망은 은닉층에 순환 에지를 부여하여 가변 길이 수용

- 문맥 의존성
  - 비순차 데이터는 공분산이 특징 사이의 의존성을 나타냄
  - 순차 데이터에서는 공분산은 의미가 없고, 대신 문맥 의존성이 중요함
  - 예, "그녀는 점심때가 다 되어서야 .... 아침을 먹었는데, 철수는 ..."에서 "그녀는"과 "먹었는데"는 강한 문맥 의존성
  - 특히 이 경우 둘 사이의 간격이 크므로 장기 의존성이라 부름 ← LSTM으로 처리

8.2 순환 신경망RNN(recurrent neural network)

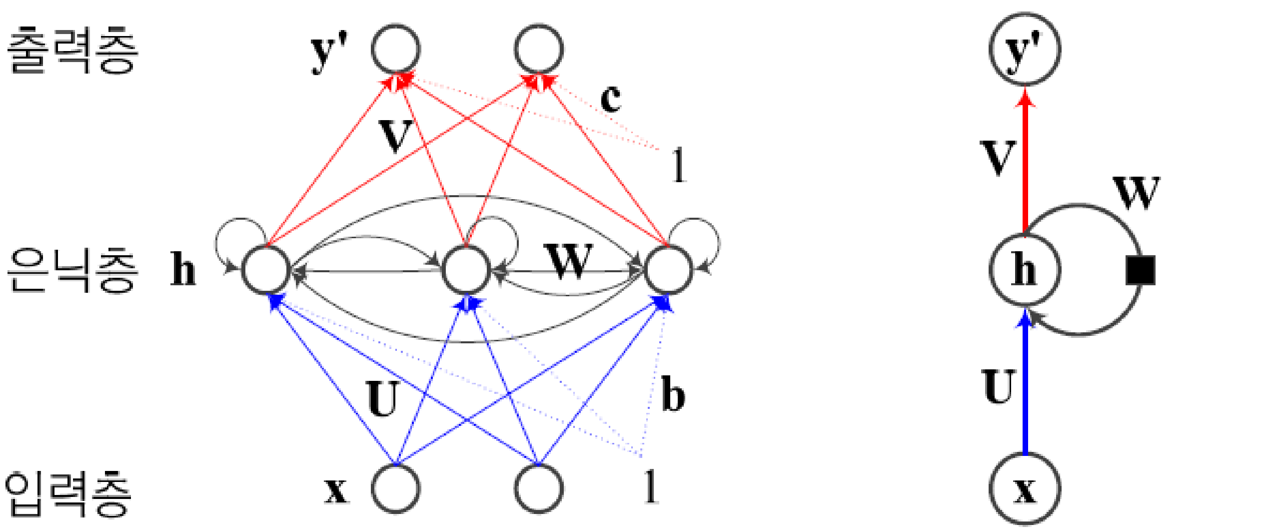
- 8.2.1 구조
- 8.2.2 동작
- 8.2.3 BPTT 학습
- 8.2.4 양방향 RNN

- 순환 신경망(RNN)이 갖추어야 할 세 가지 필수 기능
  - 시간성: 특징을 순서대로 한 번에 하나씩 입력해야 한다.
  - 가변 길이: 길이가 T인 샘플을 처리하려면 은닉층이 T번 나타나야 한다. T는 가변적이다.
  - 문맥 의존성: 이전 특징 내용을 기억하고 있다가 적절한 순간에 활용해야 한다.

8.2.1 구조

RNN의 구조

- 3장의 MLP와 비슷
  - 입력층, 은닉층, 출력층을 가짐
- 다른 점은 은닉층이 순환 에지를recurrent edge 가진다는 점
  - 시간성, 가변 길이, 문맥 의존성을 모두 처리할 수 있음
  - 순환 에지는 t-1 순간에 발생한 정보를 t 순간으로 전달하는 역할



(a) RNN (b) 축약형으로 그림

8.2.1 구조

- 수식으로 쓰면,

$$\mathbf{h}^{(t)} = f\left(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \Theta\right) \tag{8.3}$$

  - 1 순간에 계산하고, 그 결과를 가지고 2 순간에 계산하고, 그 결과를 가지고 3 순간에 계산하고, ..., T 순간까지 반복
  - t 순간에는 t-1 순간의 은닉층 값(상태) $\mathbf{h}^{(t-1)}$과 t 순간의 입력 $\mathbf{x}^{(t)}$를 받아 $\mathbf{h}^{(t)}$로 전환함
  - Θ는 순환 신경망의 매개변수

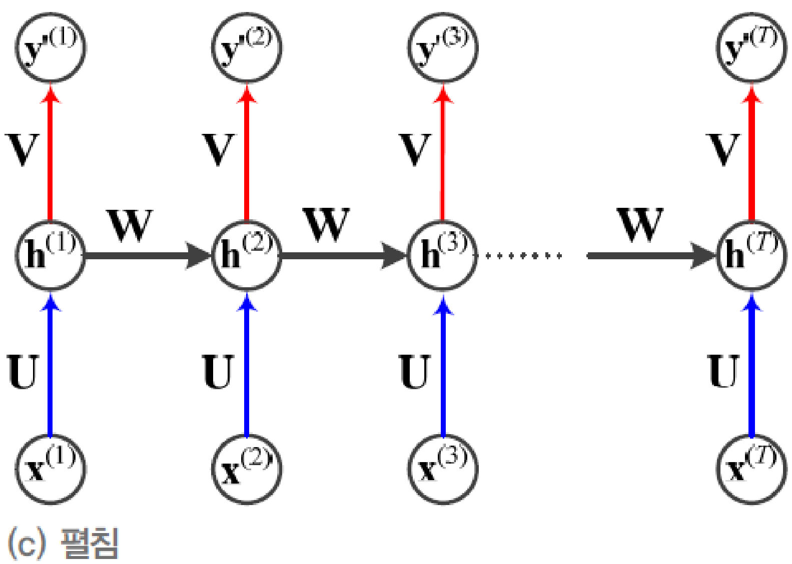


그림 8-4 RNN의 구조

8.2.1 구조

- 펼쳐서 다시 그리면,

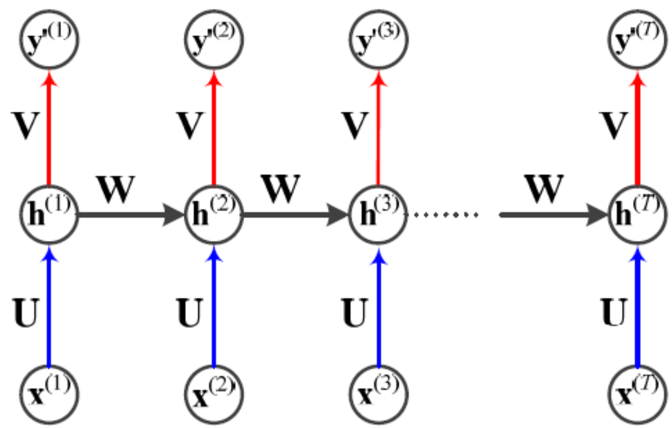


그림 8-4 RNN의 구조

- 식 (8.3)을 펼치면,

$$
\begin{aligned}
\mathbf{h}^{(T)} &= f(\mathbf{h}^{(T-1)}, \mathbf{x}^{(T)}; \Theta) \\
&= f(f(\mathbf{h}^{(T-2)}, \mathbf{x}^{(T-1)}; \Theta), \mathbf{x}^{(T)}; \Theta) \\
&\qquad \vdots \\
&= f(f(\cdots f(\mathbf{h}^{(1)}, \mathbf{x}^{(2)}; \Theta), \cdots, \mathbf{x}^{(T-1)}; \Theta), \mathbf{x}^{(T)}; \Theta) \\
&= f(f(\cdots f(f(\mathbf{h}^{(0)}, \mathbf{x}^{(1)}; \Theta), \mathbf{x}^{(2)}; \Theta), \cdots, \mathbf{x}^{(T-1)}; \Theta), \mathbf{x}^{(T)}; \Theta)
\end{aligned}
\tag{8.4}
$$

8.2.1 구조

- 순환 신경망의 매개변수(가중치 집합)는 $\Theta = \{\mathbf{U}, \mathbf{W}, \mathbf{V}, \mathbf{b}, \mathbf{c}\}$
  - $\mathbf{U}$는 입력층과 은닉층을 연결하는 $p*d$ 행렬
  - $\mathbf{W}$는 은닉층과 은닉층을 연결하는 $p*p$ 행렬
  - $\mathbf{V}$는 은닉층과 출력층을 연결하는 $q*p$ 행렬
  - $\mathbf{b}, \mathbf{c}$는 바이어스로서 각각 $p*1$과 $q*1$ 행렬
  - RNN 학습이란 훈련집합을 최적의 성능으로 예측하는 Θ 값을 찾는 일

- 매개변수 공유
  - 매 순간 다른 값을 사용하지 않고 같은 값을 공유함([그림 8-3(c)])
  - 공유의 장점
    - 추정할 매개변수 수가 획기적으로 줄어듦
    - 매개변수의 수가 특징 벡터의 길이 T에 무관
    - 특징이 나타나는 순간이 뒤바뀌어도 같거나 유사한 출력을 만들 수 있음(예, "어제 이 책을 샀다"와 "이 책을 어제 샀다"를 비슷한 영어 문장으로 번역할 수 있음)

8.2.1 구조

여러 가지 구조

- [그림 8-4]는 입력의 개수 T와 출력의 개수 L이 같은 경우
- [그림 8-5]는 $T \neq L$인 경우(왼쪽은 퀴즈풀이 응용 예($L = 1$), 입력은 "2000년 노벨 평화상을 받은 사람은?", 출력은 "김대중")

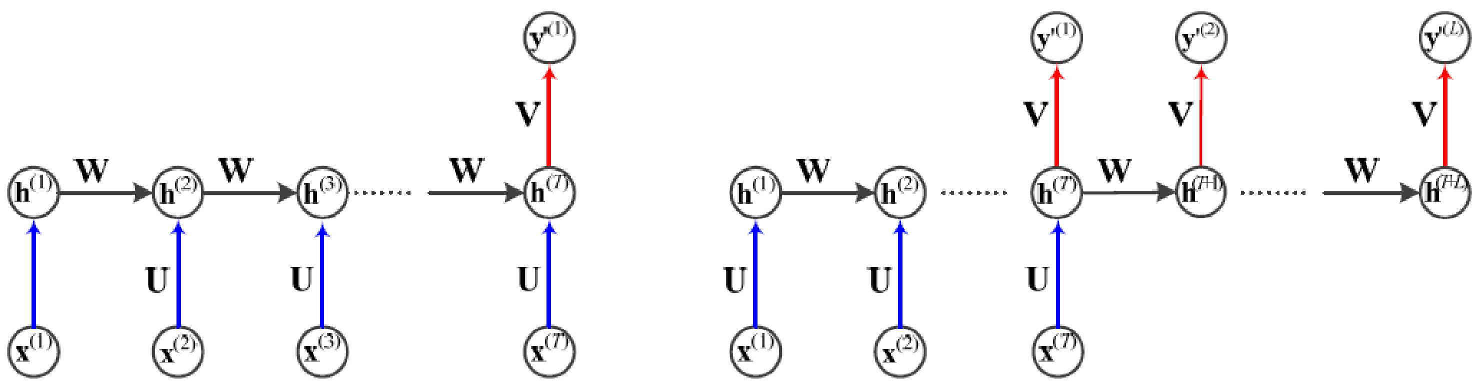


그림 8-5 RNN의 여러 가지 변형

8.2.2 동작

RNN의 가중치

- $\mathbf{u}_j = (u_{j1}, u_{j2}, \cdots, u_{jd})$는 $\mathbf{U}$ 행렬의 j번째 행(h_j에 연결된 에지의 가중치들)

$\mathbf{u}_2$

$$\mathbf{U} = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1d} \\ u_{21} & u_{22} & \cdots & u_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ u_{p1} & u_{p2} & \cdots & u_{pd} \end{pmatrix}, \mathbf{V} = \begin{pmatrix} v_{11} & v_{12} & \cdots & v_{1p} \\ v_{21} & v_{22} & \cdots & v_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ v_{q1} & v_{q2} & \cdots & v_{qp} \end{pmatrix}, \mathbf{W} = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1p} \\ w_{21} & w_{22} & \cdots & w_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ w_{p1} & w_{p2} & \cdots & w_{pp} \end{pmatrix} \quad (8.5)$$

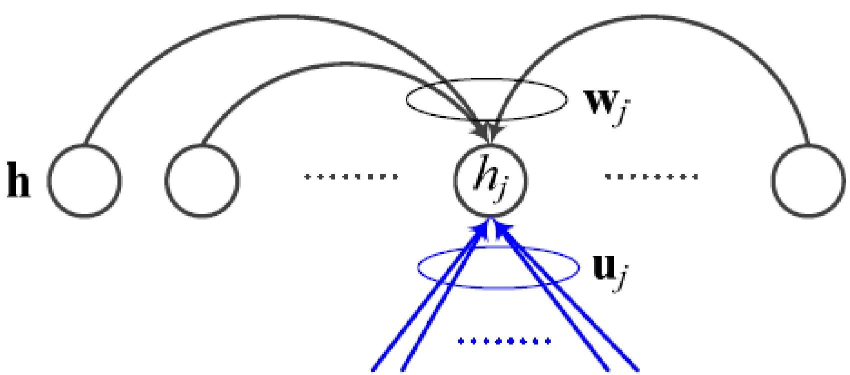


그림 8-6 은닉 노드의 가중치 표기

8.2.2 동작

- 은닉층의 계산

$$h_j^{(t)} = \tau\left(a_j^{(t)}\right), \qquad j = 1,2,\cdots,p$$
$$\text{이때, } a_j^{(t)} = \mathbf{w}_j\mathbf{h}^{(t-1)} + \mathbf{u}_j\mathbf{x}^{(t)} + b_j \tag{8.6}$$

  - MLP와 유사($\mathbf{w}_j\mathbf{h}^{(t-1)}$ 항을 제외하면 MLP와 동일함)
  - 행렬 표기로 쓰면,

$$\mathbf{h}^{(t)} = \tau(\mathbf{a}^{(t)})$$
$$\text{이때, } \mathbf{a}^{(t)} = \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)} + \mathbf{b} \tag{8.7}$$

- 은닉층 계산이 끝난 후 출력층의 계산

$$\mathbf{o}^{(t)} = \mathbf{V}\mathbf{h}^{(t)} + \mathbf{c} \tag{8.8}$$

$$\mathbf{y}'^{(t)} = \text{softmax}(\mathbf{o}^{(t)}) \tag{8.9}$$

8.2.2 동작

예제 8-1 **RNN의 동작**

[그림 8-7]은 간단한 예제 RNN이다. 그림을 간결하게 하려고 가중치가 0인 에지는 숫자를 기입하지 않았다. 식 (8.5)에 따라 이 RNN의 매개변숫값은 다음과 같다.

$$\mathbf{U}=\begin{pmatrix}0.1 & 0.1\\ 0.0 & 0.0\\ 0.0 & -0.1\end{pmatrix},\ \mathbf{W}=\begin{pmatrix}0.1 & 0.1 & 0.0\\ 0.0 & 0.0 & 0.0\\ 0.2 & -0.1 & -0.1\end{pmatrix},\ \mathbf{V}=\begin{pmatrix}0.0 & 0.1 & 0.0\\ -0.2 & 0.0 & 0.0\end{pmatrix},\ \mathbf{b}=\begin{pmatrix}0.0\\ 0.0\\ 0.2\end{pmatrix},\ \mathbf{c}=\begin{pmatrix}0.2\\ 0.1\end{pmatrix}$$

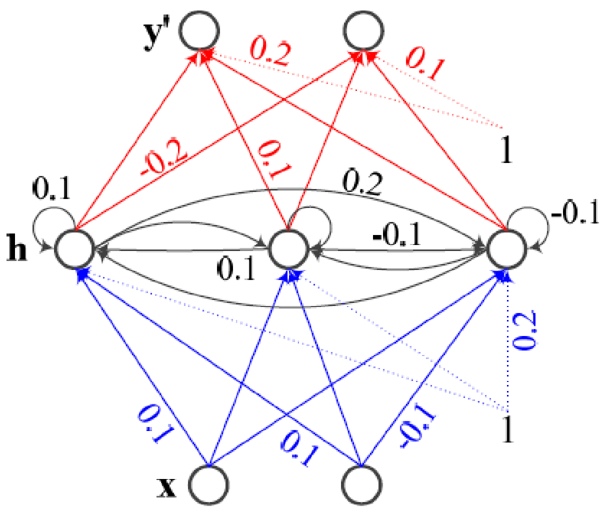


그림 8-7 예제 RNN

이 RNN에 샘플 $\mathbf{x}=\left(\begin{pmatrix}0.0\\1.0\end{pmatrix},\begin{pmatrix}0.0\\0.1\end{pmatrix},\begin{pmatrix}0.1\\-0.2\end{pmatrix},\begin{pmatrix}0.5\\0.0\end{pmatrix}\right)^{\mathrm{T}}$, $\mathbf{y}=\left(\begin{pmatrix}0.7\\0.4\end{pmatrix},\begin{pmatrix}0.3\\0.4\end{pmatrix},\begin{pmatrix}0.2\\0.4\end{pmatrix},\begin{pmatrix}0.4\\0.5\end{pmatrix}\right)^{\mathrm{T}}$가 주어졌다고 가정하면 다음과 같은 연산이 일어난다.

8.2.2 동작

t=1일 때, 식 (8.7)과 식 (8.8)에 값을 대입하면 다음과 같다. 활성함수로 tanh를 사용한다고 가정하였다. 은닉층의 초깃값 $\mathbf{h}^{(0)} = (0\ 0\ 0)^{\mathrm{T}}$라고 가정한다.

$$\mathbf{a}^{(1)} = \mathbf{W}\mathbf{h}^{(0)} + \mathbf{U}\mathbf{x}^{(1)} + \mathbf{b} = \begin{pmatrix} 0.1 & 0.1 & 0.0 \\ 0.0 & 0.0 & 0.0 \\ 0.2 & -0.1 & -0.1 \end{pmatrix}\begin{pmatrix} 0.0 \\ 0.0 \\ 0.0 \end{pmatrix} + \begin{pmatrix} 0.1 & 0.1 \\ 0.0 & 0.0 \\ 0.0 & -0.1 \end{pmatrix}\begin{pmatrix} 0.0 \\ 1.0 \end{pmatrix} + \begin{pmatrix} 0.0 \\ 0.0 \\ 0.2 \end{pmatrix} = \begin{pmatrix} 0.1 \\ 0.0 \\ 0.1 \end{pmatrix}$$

$$\mathbf{h}^{(1)} = \tau(\mathbf{a}^{(1)}) = \begin{pmatrix} 0.0997 \\ 0.0 \\ 0.0997 \end{pmatrix}$$

$$\mathbf{y}'^{(1)} = \mathrm{softmax}(\mathbf{V}\mathbf{h}^{(1)} + \mathbf{c}) = \mathrm{softmax}\left(\begin{pmatrix} 0.0 & 0.1 & 0.0 \\ -0.2 & 0.0 & 0.0 \end{pmatrix}\begin{pmatrix} 0.0997 \\ 0.0 \\ 0.0997 \end{pmatrix} + \begin{pmatrix} 0.2 \\ 0.1 \end{pmatrix}\right) = \begin{pmatrix} 0.5299 \\ 0.4701 \end{pmatrix}$$

비슷한 방식으로 t=2, 3, 4일 때 계산 결과는 다음과 같다.

$$\mathbf{y}'^{(2)} = \begin{pmatrix} 0.5260 \\ 0.4740 \end{pmatrix},\ \mathbf{y}'^{(3)} = \begin{pmatrix} 0.5246 \\ 0.4754 \end{pmatrix},\ \mathbf{y}'^{(4)} = \begin{pmatrix} 0.5274 \\ 0.4726 \end{pmatrix}$$

이 샘플의 레이블, 즉 기대 출력이 $\mathbf{y} = \left(\begin{pmatrix} 0.7 \\ 0.4 \end{pmatrix}, \begin{pmatrix} 0.3 \\ 0.4 \end{pmatrix}, \begin{pmatrix} 0.2 \\ 0.4 \end{pmatrix}, \begin{pmatrix} 0.4 \\ 0.5 \end{pmatrix}\right)^{\mathrm{T}}$ 인데, 출력이 $\mathbf{y}' = \left(\begin{pmatrix} 0.5299 \\ 0.4701 \end{pmatrix}, \begin{pmatrix} 0.5260 \\ 0.4740 \end{pmatrix}, \begin{pmatrix} 0.5246 \\ 0.4754 \end{pmatrix}, \begin{pmatrix} 0.5274 \\ 0.4726 \end{pmatrix}\right)^{\mathrm{T}}$ 이므로 현재 가중치, 즉 매개변수 Θ는 상당한 오차를 발생시켰다고 판단할 수 있다. 8.2.3 절에서는 매개변수 Θ의 값을 반복적으로 개선하여 최적해를 구하는 RNN의 학습 알고리즘을 학습한다.

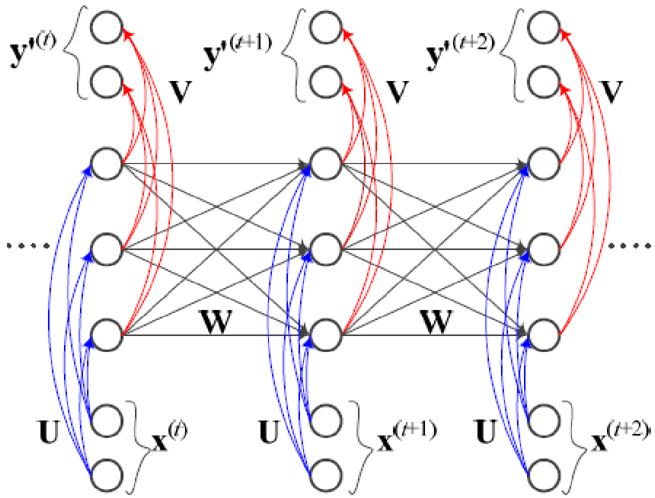
8.2.2 동작

- RNN의 기억 기능
  - $\mathbf{x}^{(1)}$이 변하면 상태 $\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}, \mathbf{h}^{(4)}$ 가 바뀌고, 그에 따라 출력 $\mathbf{y}'^{(1)}, \mathbf{y}'^{(2)}, \mathbf{y}'^{(3)}, \mathbf{y}'^{(4)}$가 바뀜 → RNN이 $\mathbf{x}^{(1)}$을 기억한다고 말할 수 있음
  - 또한 $\mathbf{x}^{(1)}$은 $\mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \mathbf{x}^{(4)}$와 상호작용을 한다고 볼 수 있음 → 문맥 의존성
  - 기억이 얼마나 지속되는지는 8.3절에서 다룸

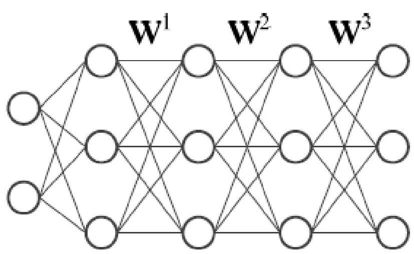
8.2.3 BPTT 학습

- 훈련집합 $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$, $\mathbb{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n\}$
  - 샘플 $\mathbf{x}_i$와 $\mathbf{y}_i$는 길이가 T_i와 L_i인 시간성 데이터

- RNN과 DMLP의 유사성
  - 둘 다 입력층, 은닉층, 출력층을 가짐([그림 8-8(a)]는 RNN의 노드를 수직으로 배치하여 DMLP와 비교하기 쉽게 함)



(a) RNN



(b) DMLP

그림 8-8 RNN과 DMLP의 비교

8.2.3 BPTT 학습

- RNN과 DMLP의 차별성
  - RNN은 샘플마다 은닉층의 수가 다름
  - DMLP는 왼쪽에 입력, 오른쪽에 출력이 있지만, RNN은 매 순간 입력과 출력이 있음
  - RNN은 가중치를 공유함 → DMLP는 가중치를 $\mathbf{W}^1, \mathbf{W}^2, \mathbf{W}^3, \cdots$로 표기하는데, RNN은 $\mathbf{W}$로 표기

8.2.3 BPTT 학습

- 목적함수의 정의
  - 출력 값을 $\mathbf{y}' = \left(y'^{(1)}, y'^{(2)}, \cdots, y'^{(T)}\right)^{\mathrm{T}}$, 목푯값을 $\mathbf{y} = \left(y^{(1)}, y^{(2)}, \cdots, y^{(T)}\right)^{\mathrm{T}}$ 로 표기
    - 평균제곱 오차, 교차 엔트로피, 로그우도 중에 선택하여 사용

$$J(\mathbf{\Theta}) = \sum_{t=1}^{T} J^{(t)}(\mathbf{\Theta}) \tag{8.10}$$

$$\text{평균제곱 오차: } J^{(t)}(\mathbf{\Theta}) = \sum_{j=1}^{q} \left(y_j^{(t)} - y_j'^{(t)}\right)^2 \tag{8.11}$$

$$\text{교차 엔트로피: } J^{(t)}(\mathbf{\Theta}) = -\mathbf{y}^{(t)} \log \mathbf{y}'^{(t)} = -\sum_{j=1}^{q} y_j^{(t)} \log y_j'^{(t)} \tag{8.12}$$

$$\text{로그우도: } J^{(t)}(\mathbf{\Theta}) = -\log y'^{(t)} \tag{8.13}$$

- 학습이 할 일 $$\widehat{\mathbf{\Theta}} = \underset{\mathbf{\Theta}}{\operatorname{argmin}}\, J(\mathbf{\Theta}) = \underset{\mathbf{\Theta}}{\operatorname{argmin}} \sum_{t=1}^{T} J^{(t)}(\mathbf{\Theta}) \tag{8.14}$$

8.2.3 BPTT 학습

그레이디언트 계산

- $\frac{\partial J}{\partial \Theta}$를 구하려면, $\Theta = \{\mathbf{U}, \mathbf{W}, \mathbf{V}, \mathbf{b}, \mathbf{c}\}$이므로 $\frac{\partial J}{\partial \mathbf{U}}, \frac{\partial J}{\partial \mathbf{W}}, \frac{\partial J}{\partial \mathbf{V}}, \frac{\partial J}{\partial \mathbf{b}}, \frac{\partial J}{\partial \mathbf{c}}$를 계산해야 함
- 그 중 [그림 8-9]에서처럼 $\mathbf{V}$는 출력에만 영향을 미치므로 $\frac{\partial J}{\partial \mathbf{V}}$ 계산이 가장 쉬움 → $\frac{\partial J}{\partial \mathbf{V}}$를 먼저 유도해 봄

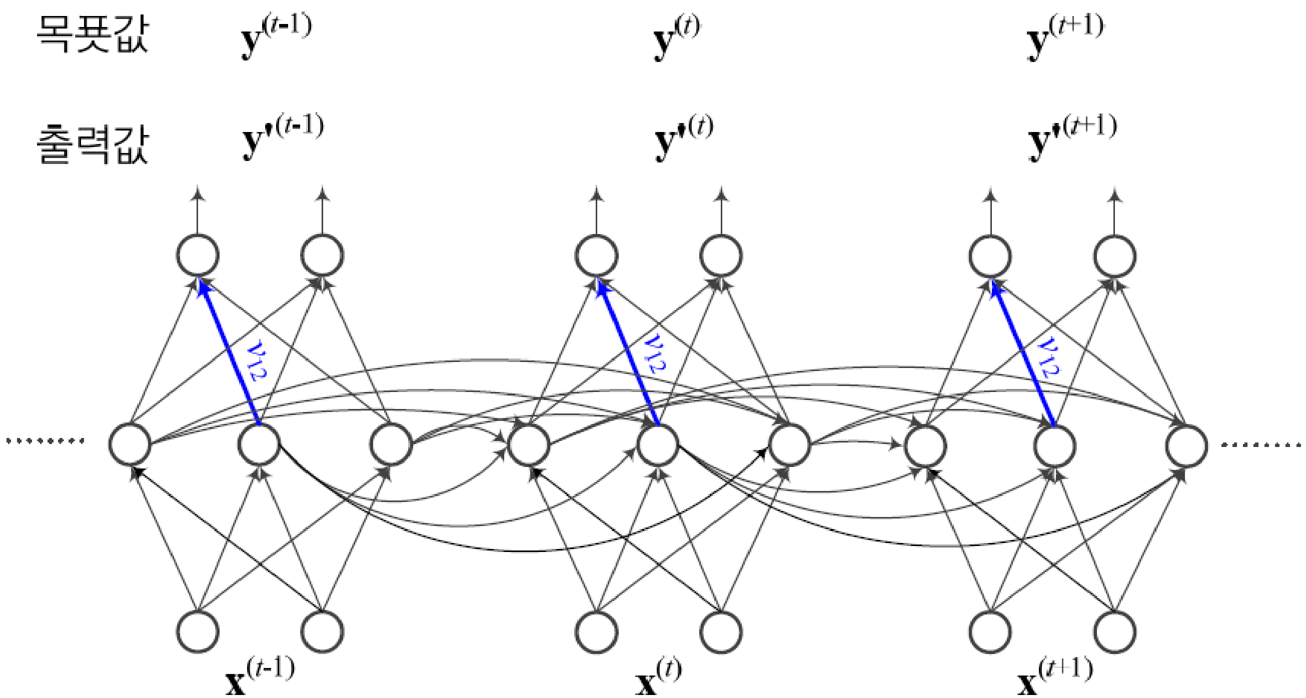


그림 8-9 BPTT 유도를 위한 그레이디언트 계산 예시

8.2.3 BPTT 학습

- $\frac{\partial J}{\partial \mathbf{V}}$는 $q*p$ 행렬

$$\frac{\partial J}{\partial \mathbf{V}} = \begin{pmatrix} \frac{\partial J}{\partial v_{11}} & \frac{\partial J}{\partial v_{12}} & \cdots & \frac{\partial J}{\partial v_{1p}} \\ \frac{\partial J}{\partial v_{21}} & \frac{\partial J}{\partial v_{22}} & \cdots & \frac{\partial J}{\partial v_{2p}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial J}{\partial v_{q1}} & \frac{\partial J}{\partial v_{q2}} & \cdots & \frac{\partial J}{\partial v_{qp}} \end{pmatrix} \tag{8.15}$$

- [그림 8-9]는 t 순간에 v_{ji}의 영향을 보여줌($j = 1, i = 2$)
- 로그우도를 사용하기로 하고 연쇄법칙을 적용하여 식 (8.13)을 v_{12}로 미분하면,

$$\frac{\partial J^{(t)}}{\partial v_{12}} = \frac{\partial J^{(t)}}{\partial y'^{(t)}} \frac{\partial y'^{(t)}}{\partial o_1^{(t)}} \frac{\partial o_1^{(t)}}{\partial v_{12}}$$

- 맨 오른쪽 항은 $o_1^{(t)} = \mathbf{v}_1 \mathbf{h}^{(t)} = v_{11} h_1^{(t)} + v_{12} h_2^{(t)} + v_{13} h_3^{(t)}$ 이므로 $\frac{\partial o_1^{(t)}}{\partial v_{12}} = h_2^{(t)}$

8.2.3 BPTT 학습

- 앞의 2개 항의 계산은 $\mathbf{y}^{(t)} = (1,0)^{\mathrm{T}}$인 경우와 $\mathbf{y}^{(t)} = (0,1)^{\mathrm{T}}$인 경우로 나누어 생각해야 함
  - $\mathbf{y}^{(t)} = (1,0)^{\mathrm{T}}$인 경우를 계산하면,

$$\begin{aligned}\frac{\partial J^{(t)}}{\partial o_1^{(t)}} = \frac{\partial J^{(t)}}{\partial y'^{(t)}}\frac{\partial y'^{(t)}}{\partial o_1^{(t)}} &= \frac{\partial\left(-\log\frac{\exp\left(o_1^{(t)}\right)}{\exp\left(o_1^{(t)}\right)+\exp\left(o_2^{(t)}\right)}\right)}{\partial o_1^{(t)}} \\ &= \frac{\partial\left(-o_1^{(t)} + \log\left(\exp\left(o_1^{(t)}\right) + \exp\left(o_2^{(t)}\right)\right)\right)}{\partial o_1^{(t)}} \\ &= -1 + \frac{\exp\left(o_1^{(t)}\right)}{\exp\left(o_1^{(t)}\right) + \exp\left(o_2^{(t)}\right)} \\ &= -1 + y_1'^{(t)}\end{aligned} \qquad (8.16)$$

  - $\mathbf{y}^{(t)} = (0,1)^{\mathrm{T}}$인 경우도 유도한 다음 두 경우를 같이 쓰면,

$$\left.\begin{aligned}\frac{\partial J^{(t)}}{\partial v_{12}} &= \left(y_1'^{(t)} - 1\right)h_2^{(t)},\ \mathbf{y}^{(t)} = (1,0)^{\mathrm{T}}\text{일 때} \\ \frac{\partial J^{(t)}}{\partial v_{12}} &= y_1'^{(t)}h_2^{(t)},\ \mathbf{y}^{(t)} = (0,1)^{\mathrm{T}}\text{일 때}\end{aligned}\right\}$$

8.2.3 BPTT 학습

- v_{12}를 v_{ji}로 일반화하고, 2부류를 q개 부류로 일반화하면,

$$\left.\begin{array}{l} \dfrac{\partial J^{(t)}}{\partial v_{ji}} = \left(y_j'^{(t)} - 1\right)h_i^{(t)},\ \mathbf{y}^{(t)}\text{의 } j\text{번째 요소가 1일 때} \\ \dfrac{\partial J^{(t)}}{\partial v_{ji}} = y_j'^{(t)} h_i^{(t)},\ \mathbf{y}^{(t)}\text{의 } j\text{번째 요소가 0일 때} \end{array}\right\}$$

- 좀 더 간결하게 표현하면,

$$\frac{\partial J^{(t)}}{\partial v_{ji}} = \left(y_j'^{(t)} - y_j^{(t)}\right)h_i^{(t)} \qquad (8.17)$$

- $1,2,\cdots,T$ 순간을 모두 고려하면,

$$\frac{\partial J}{\partial v_{ji}} = \sum_{t=1}^{T}\left(y_j'^{(t)} - y_j^{(t)}\right)h_i^{(t)} \qquad (8.18)$$

8.2.3 BPTT 학습

- BPTT(back-propagation through time) 알고리즘
  - v_{ji}로 미분하는 식 (8.18)을 행렬 전체를 위한 식 $\frac{\partial J}{\partial \mathbf{V}}$로 확장하고, $\frac{\partial J}{\partial \mathbf{U}}, \frac{\partial J}{\partial \mathbf{W}}, \frac{\partial J}{\partial \mathbf{b}}, \frac{\partial J}{\partial \mathbf{c}}$까지 유도하면 BPTT가 완성됨
  - 이 확장 작업에 필요한 식 (8.16)을 벡터 형태로 일반화하면,

$$\frac{\partial J^{(t)}}{\partial \mathbf{o}^{(t)}} = \mathbf{y}'^{(t)} - \mathbf{y}^{(t)} \tag{8.19}$$

- 은닉층에서의 미분
  - 순간 t의 은닉층값 $\mathbf{h}^{(t)}$는 그 이후의 은닉층과 출력층에 영향을 주므로 $\mathbf{V}$로 미분하는 것보다 복잡
  - 우선 이후가 없는 마지막 순간 T에 대해 미분식을 유도하면,

$$\frac{\partial J^{(T)}}{\partial \mathbf{h}^{(T)}} = \frac{\partial J^{(T)}}{\partial \mathbf{o}^{(T)}} \frac{\partial \mathbf{o}^{(T)}}{\partial \mathbf{h}^{(T)}} = \mathbf{V}^{\mathrm{T}} \frac{\partial J^{(T)}}{\partial \mathbf{o}^{(T)}} \tag{8.20}$$

8.2.3 BPTT 학습

- T-1 순간의 그레이디언트를 유도하면,
  - $\mathbf{D}\left(1-\left(\mathbf{h}^{(T)}\right)^2\right)$는 i번 열의 대각선이 $1-\left(h_i^{(T)}\right)^2$을 가진 대각 행렬

$$\frac{\partial\left(J^{(T-1)}+J^{(T)}\right)}{\partial \mathbf{h}^{(T-1)}}=\frac{\partial J^{(T-1)}}{\partial \mathbf{o}^{(T-1)}}\frac{\partial \mathbf{o}^{(T-1)}}{\partial \mathbf{h}^{(T-1)}}+\frac{\partial \mathbf{h}^{(T)}}{\partial \mathbf{h}^{(T-1)}}\frac{\partial J^{(T)}}{\partial \mathbf{h}^{(T)}}$$

$$=\mathbf{V}^{\mathrm{T}}\frac{\partial J^{(T-1)}}{\partial \mathbf{o}^{(T-1)}}+\mathbf{W}^{\mathrm{T}}\frac{\partial J^{(T)}}{\partial \mathbf{h}^{(T)}}\mathbf{D}\left(1-\left(\mathbf{h}^{(T)}\right)^2\right)$$

- t 순간으로 일반화하면, 그레이디언트를 역전파하는 순환식인 식 (8.21)을 얻음
  - $J^{(\tilde{t})}$는 t를 포함하여 이후의 목적함숫값을 모두 더한 값, 즉 $J^{(\tilde{t})}=J^{(t)}+J^{(t+1)}+\cdots+J^{(T)}$

$$\frac{\partial J^{(\tilde{t})}}{\partial \mathbf{h}^{(t)}}=\mathbf{V}^{\mathrm{T}}\frac{\partial J^{(t)}}{\partial \mathbf{o}^{(t)}}+\mathbf{W}^{\mathrm{T}}\frac{\partial J^{(\widetilde{t+1})}}{\partial \mathbf{h}^{(t+1)}}\mathbf{D}\left(1-\left(\mathbf{h}^{(t+1)}\right)^2\right) \tag{8.21}$$

8.2.3 BPTT 학습

- [그림 8-10]은 식 (8.21)을 설명

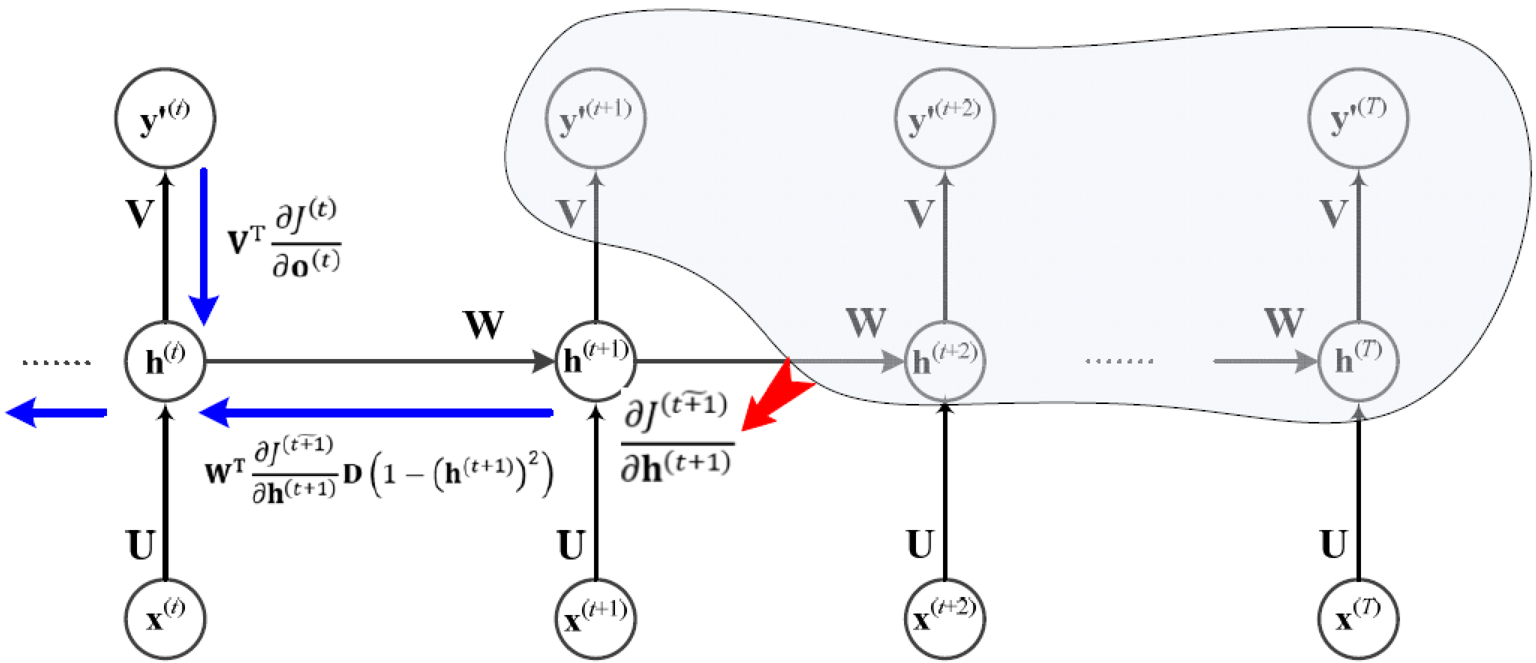


그림 8-10 역전파 순환식으로서 식 (8.21)의 동작

8.2.3 BPTT 학습

■ BPTT 알고리즘

$$
\begin{cases}
\dfrac{\partial J}{\partial \mathbf{V}} = \displaystyle\sum_{t=1}^{T} \dfrac{\partial J^{(t)}}{\partial \mathbf{o}^{(t)}} \mathbf{h}^{(t)\mathrm{T}} & (8.22) \\[2ex]
\dfrac{\partial J}{\partial \mathbf{W}} = \displaystyle\sum_{t=1}^{T} \mathbf{D}\left(1-\left(\mathbf{h}^{(t)}\right)^2\right) \dfrac{\partial J^{(\tilde{t})}}{\partial \mathbf{h}^{(t)}} \mathbf{h}^{(t-1)\mathrm{T}} & (8.23) \\[2ex]
\dfrac{\partial J}{\partial \mathbf{U}} = \displaystyle\sum_{t=1}^{T} \mathbf{D}\left(1-\left(\mathbf{h}^{(t)}\right)^2\right) \dfrac{\partial J^{(\tilde{t})}}{\partial \mathbf{h}^{(t)}} \mathbf{x}^{(t)\mathrm{T}} & (8.24) \\[2ex]
\dfrac{\partial J}{\partial \mathbf{c}} = \displaystyle\sum_{t=1}^{T} \dfrac{\partial J^{(t)}}{\partial \mathbf{o}^{(t)}} & (8.25) \\[2ex]
\dfrac{\partial J}{\partial \mathbf{b}} = \displaystyle\sum_{t=1}^{T} \mathbf{D}\left(1-\left(\mathbf{h}^{(t)}\right)^2\right) \dfrac{\partial J^{(\tilde{t})}}{\partial \mathbf{h}^{(t)}} & (8.26)
\end{cases}
$$

8.2.4 양방향 RNN

- 양방향 문맥 의존성
  - 왼쪽에서 오른쪽으로만 정보가 흐르는 단방향 RNN은 한계
  - 예, [그림 8-11]에서 '거지'와 '지지'를 구별하기 어려움

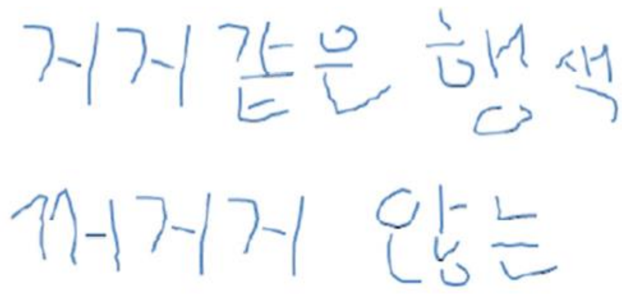


그림 8-11 양방향 문맥 의존성

- 양방향 RNN(BRNN)
  - t 순간의 단어는 앞쪽 단어와 뒤쪽 단어 정보를 모두 보고 처리됨
  - 기계 번역에서도 BRNN을 활용함 ← 8.5.2절

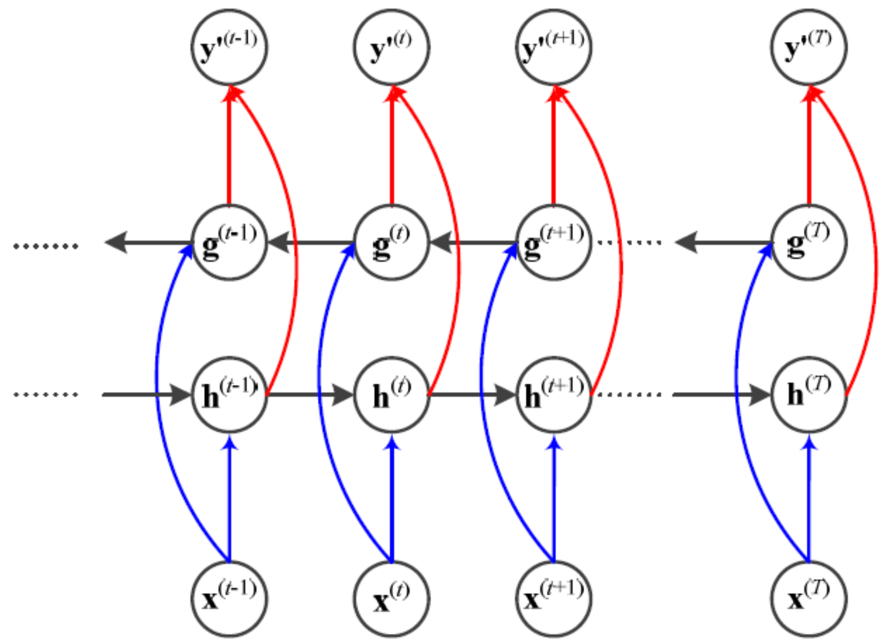


그림 8-12 양방향 RNN(BRNN)의 구조와 동작

8.3 장기 문맥 의존성

장기 문맥 의존성

- 관련된 요소가 멀리 떨어진 상황
- 예, 아래 문장에서 순간 1의 '길동은'과 순간 32의 '쉬기로'는 아주 밀접한 관련

"길동은, 어제는 친구와 소풍을 다녀왔고, 글피는 엄마를 따라 시장에 가서 반찬거리를 사 오고, 그글피는 여자 친구와 함께 비가 옴에도 불구하고 놀이동산에서 재미있게 놀고 왔기 때문에 오늘은 집에서 푹 쉬기로 작정하였다."

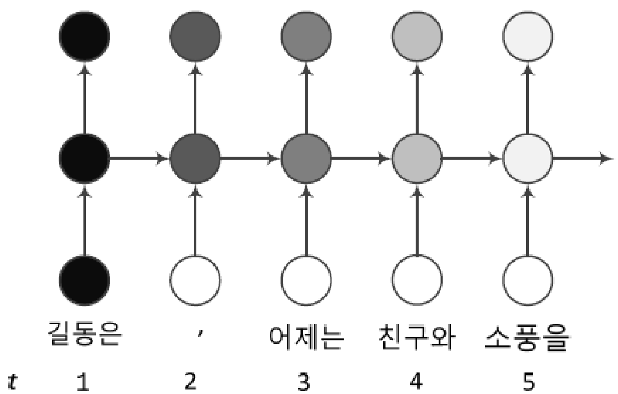


그림 8-13 긴 순차 데이터에서 영향력 감쇠 현상

8.3 장기 문맥 의존성

- 문제점
  - 그레이디언트 소멸($\mathbf{W}$ 요소가 1보다 작을 때) 또는 그레이디언트 폭발($\mathbf{W}$ 요소가 1보다 클 때)
  - RNN은 DMLP나 CNN보다 심각
    - 긴 입력 샘플이 자주 발생하기 때문
    - 가중치 공유 때문에 같은 값을 계속 곱함
- LSTM은 가장 널리 사용되는 해결책

8.4 LSTM(long short term memory)

- 8.4.1 게이트를 이용한 영향력 범위 확장
- 8.4.2 LSTM의 동작
- 8.4.3 망각 게이트와 핍홀

8.4.1 게이트를 이용한 영향력 범위 확장

입력 게이트와 출력 게이트

- 게이트를 열면( ) 신호가 흐르고, 닫으면( ) 차단됨
- 예, [그림 8-14]에서 t=1에서는 입력만 열렸고, 32와 33에서는 입력과 출력이 모두 열림
- 실제로는 [0,1] 사이의 실숫값으로 개폐 정도를 조절
- 이 값은 학습으로 알아냄

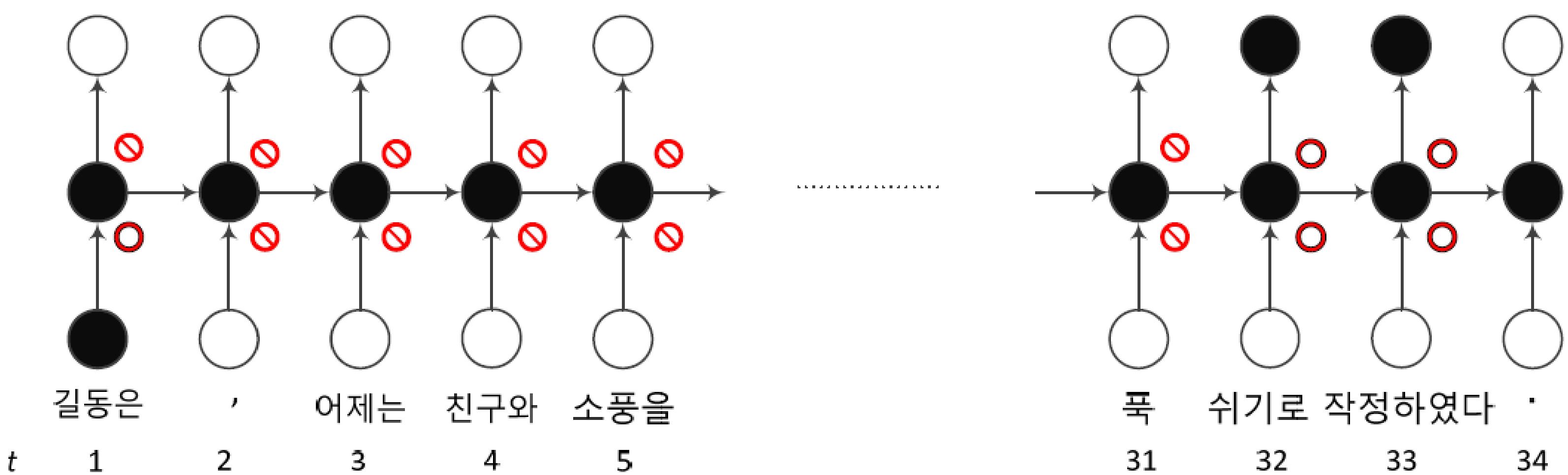
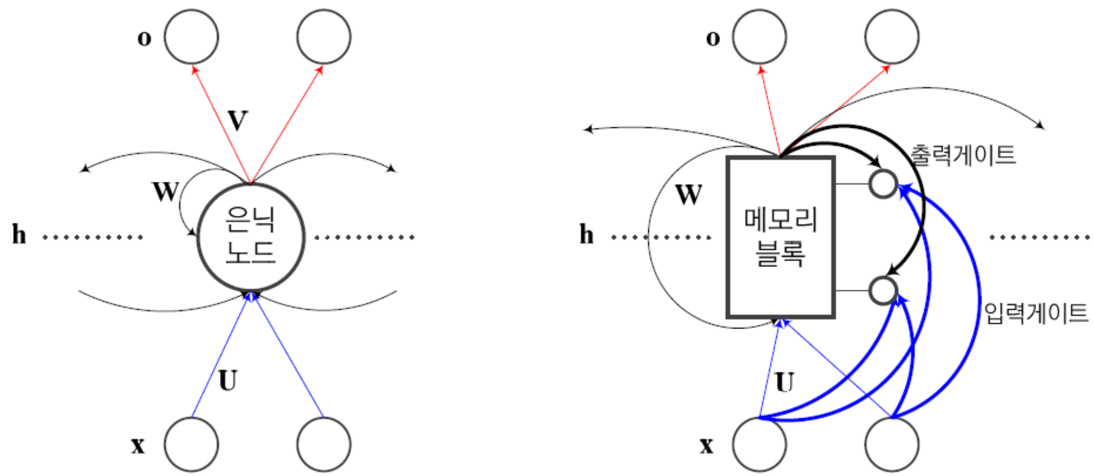


그림 8-14 입력 게이트와 출력 게이트를 이용한 입출력 제어

8.4.1 게이트를 이용한 영향력 범위 확장

RNN과 LSTM의 비교

- [그림 8-15(a)]의 RNN은 [그림 8-4(a)]를 다른 형태로 그린 것 (LSTM과 비교 목적)
  - 얇은 선분: 입력→은닉(파랑), 은닉→은닉(검정), 은닉→출력(빨강)



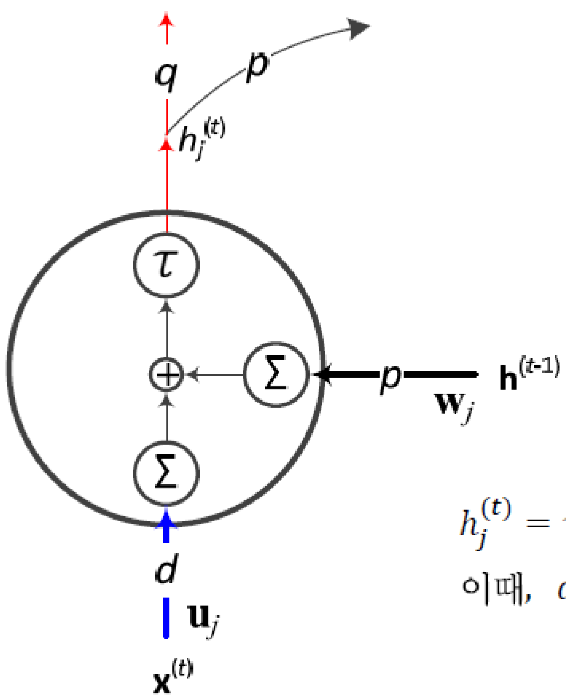
(a) RNN의 은닉 노드 (b) LSTM의 은닉 노드

그림 8-15 RNN과 LSTM의 비교

- [그림 8-15(b)]의 LSTM은 메모리 블록을 가짐
  - 얇은 선분: 입력→은닉(파랑), 은닉→은닉(검정), 은닉→출력(빨강) ← RNN과 동일
  - 추가로,
    - 메모리 블록의 출력→출력 게이트, 입력 게이트(굵은 검정)
    - 입력 벡터 **x**→출력 게이트, 입력 게이트(굵은 파랑)

8.4.2 LSTM의 동작

- RNN의 은닉 노드를 확대하여 다시 살펴보면,
  - [그림 8-16]은 LSTM과 같은 표기법을 쓰기 위해 다시 그린 것
  - 굵은 선은 가중치 벡터
  - 식 (8.6)은 RNN의 동작을 나타냄



$$h_j^{(t)} = \tau\left(a_j^{(t)}\right), \qquad j = 1,2,\cdots,p$$
$$\text{이때, } a_j^{(t)} = \mathbf{w}_j \mathbf{h}^{(t-1)} + \mathbf{u}_j \mathbf{x}^{(t)} + b_j \tag{8.6}$$

그림 8-16 RNN 은닉 노드의 구조와 동작 다시 보기(j번째 은닉 노드)

8.4.2 LSTM의 동작

LSTM의 동작

- [그림 8-17]의 LSTM에서 출력 게이트와 입력 게이트의 값이 1.0으로 고정되면 RNN 동작과 동일함
- 하지만 이들 값은 가중치와 신호 값에 따라 정해지며 개폐 정도를 조절함 ← RNN과 차별성

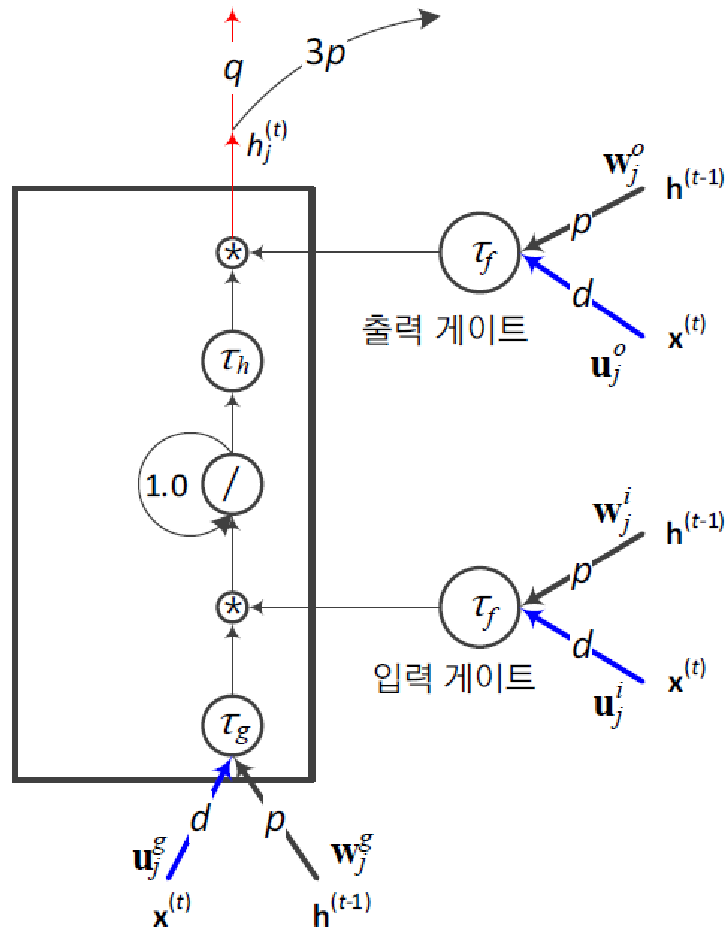


그림 8-17 LSTM 메모리 블록의 구조와 동작(j 번째 메모리 블록)

8.4.2 LSTM의 동작

- LSTM의 가중치
  - 은닉층과 은닉층을 잇는 순환 에지는 세 종류의 가중치를 가짐(굵은 검정 선)
    - 입력단과 연결하는 $\mathbf{W}^g$, 입력 게이트와 연결하는 $\mathbf{W}^i$, 출력 게이트와 연결하는 $\mathbf{W}^o$
  - 입력층과 은닉층을 연결하는 가중치 $\mathbf{U}$도 마찬가지(굵은 파란 선)

- 가중치를 행렬로 표현하면,

$$\mathbf{W}^g = \begin{pmatrix} w_{11}^g & w_{12}^g & \cdots & w_{1p}^g \\ w_{21}^g & w_{22}^g & \cdots & w_{2p}^g \\ \vdots & \vdots & \ddots & \vdots \\ w_{p1}^g & w_{p2}^g & \cdots & w_{pp}^g \end{pmatrix}, \mathbf{W}^i = \begin{pmatrix} w_{11}^i & w_{12}^i & \cdots & w_{1p}^i \\ w_{21}^i & w_{22}^i & \cdots & w_{2p}^i \\ \vdots & \vdots & \ddots & \vdots \\ w_{p1}^i & w_{p2}^i & \cdots & w_{pp}^i \end{pmatrix}, \mathbf{W}^o = \begin{pmatrix} w_{11}^o & w_{12}^o & \cdots & w_{1p}^o \\ w_{21}^o & w_{22}^o & \cdots & w_{2p}^o \\ \vdots & \vdots & \ddots & \vdots \\ w_{p1}^o & w_{p2}^o & \cdots & w_{pp}^o \end{pmatrix} \quad (8.27)$$

$$\mathbf{U}^g = \begin{pmatrix} u_{11}^g & u_{12}^g & \cdots & u_{1d}^g \\ u_{21}^g & u_{22}^g & \cdots & u_{2d}^g \\ \vdots & \vdots & \ddots & \vdots \\ u_{p1}^g & u_{p2}^g & \cdots & u_{pd}^g \end{pmatrix}, \mathbf{U}^i = \begin{pmatrix} u_{11}^i & u_{12}^i & \cdots & u_{1d}^i \\ u_{21}^i & u_{22}^i & \cdots & u_{2d}^i \\ \vdots & \vdots & \ddots & \vdots \\ u_{p1}^i & u_{p2}^i & \cdots & u_{pd}^i \end{pmatrix}, \mathbf{U}^o = \begin{pmatrix} u_{11}^o & u_{12}^o & \cdots & u_{1d}^o \\ u_{21}^o & u_{22}^o & \cdots & u_{2d}^o \\ \vdots & \vdots & \ddots & \vdots \\ u_{p1}^o & u_{p2}^o & \cdots & u_{pd}^o \end{pmatrix} \quad (8.28)$$

8.4.2 LSTM의 동작

- 세 곳(입력단, 입력 게이트, 출력 게이트)에서의 계산

$$\text{입력단: } g = \tau_g\left(\mathbf{u}_j^g\mathbf{x}^{(t)} + \mathbf{w}_j^g\mathbf{h}^{(t-1)} + b_j^g\right) \tag{8.29}$$

$$\text{입력 게이트: } i = \tau_f\left(\mathbf{u}_j^i\mathbf{x}^{(t)} + \mathbf{w}_j^i\mathbf{h}^{(t-1)} + b_j^i\right) \tag{8.30}$$

$$\text{출력 게이트: } o = \tau_f\left(\mathbf{u}_j^o\mathbf{x}^{(t)} + \mathbf{w}_j^o\mathbf{h}^{(t-1)} + b_j^o\right) \tag{8.31}$$

  - g, i, o 값은 가중치 $\mathbf{u}, \mathbf{w}$, 현재 순간의 입력벡터 $\mathbf{x}^{(t)}$, 이전 순간의 상태 $\mathbf{h}^{(t-1)}$에 따라 정해짐 → 이들 값에 따라 개폐 정도가 정해짐
  - τ_g는 tanh, τ_f는 로지스틱 시그모이드를 주로 사용

- 아래쪽 곱 기호 *는 개폐를 조절하는 역할
  - 입력 게이트의 값 i가 0.0에 가깝다면 $g * i$는 0.0에 가깝게 되어 입력단을 차단, 1.0에 가깝다면 그대로 전달하는 효과

8.4.2 LSTM의 동작

- 기호 /가 붙어 있는 원은 메모리 블록의 상태
  - 메모리 블록이 기억하는 내용으로 시간에 따라 변하므로 $s^{(t)}$로 표기

$$s^{(t)} = s^{(t-1)} + g * i \qquad (8.32)$$

  - 해석해 보면,
    - 입력 게이트(즉 i)가 0.0이면 $g * i$는 0이 되어 이전 상태와 같게 됨(입력 게이트가 차단되어 이전 내용을 그대로 기억) → 이전 입력의 영향력을 더 멀리 확장하는 효과

- 위쪽 곱 기호 *는 개폐를 조절하는 역할
  - 출력 게이트의 값 o가 개폐 정도를 조절

$$h_j^{(t)} = \tau_h(s^{(t)}) * o \qquad (8.33)$$

- 식 (8.33)의 계산 결과인 $h_j^{(t)}$는
  - q개의 출력 노드로 전달되어 출력단 계산에 사용(즉 식 (8.8)의 벡터 $\mathbf{h}^{(t)}$의 j번째 요소임)
  - 입력단, 입력 게이트, 출력 게이트에 있는 노드로 전달되어 t+1 순간의 계산에 사용됨

8.4.2 LSTM의 동작

- 지금까지 수식을 정리하면,

$$\text{입력단: } \mathbf{g} = \tau_g\left(\mathbf{U}^g\mathbf{x}^{(t)} + \mathbf{W}^g\mathbf{h}^{(t-1)} + \mathbf{b}^g\right) \tag{8.34}$$

$$\text{입력 게이트: } \mathbf{i} = \tau_f\left(\mathbf{U}^i\mathbf{x}^{(t)} + \mathbf{W}^i\mathbf{h}^{(t-1)} + \mathbf{b}^i\right) \tag{8.35}$$

$$\text{출력 게이트: } \mathbf{o} = \tau_f\left(\mathbf{U}^o\mathbf{x}^{(t)} + \mathbf{W}^o\mathbf{h}^{(t-1)} + \mathbf{b}^o\right) \tag{8.36}$$

$$\mathbf{s}^{(t)} = \mathbf{s}^{(t-1)} + \mathbf{g} \odot \mathbf{i} \tag{8.37}$$

$$\mathbf{h}^{(t)} = \tau_h\left(\mathbf{s}^{(t)}\right) \odot \mathbf{o} \tag{8.38}$$

$$\mathbf{y}'^{(t)} = \text{softmax}(\mathbf{V}\mathbf{h}^{(t)} + \mathbf{c}) \tag{8.39}$$

8.4.3 망각 게이트와 핍홀

■ 망각 게이트에 의한 LSTM의 확장

- 이전 순간의 상태 $h^{(t-1)}$(즉 메모리 블록의 기억)을 지우는 효과

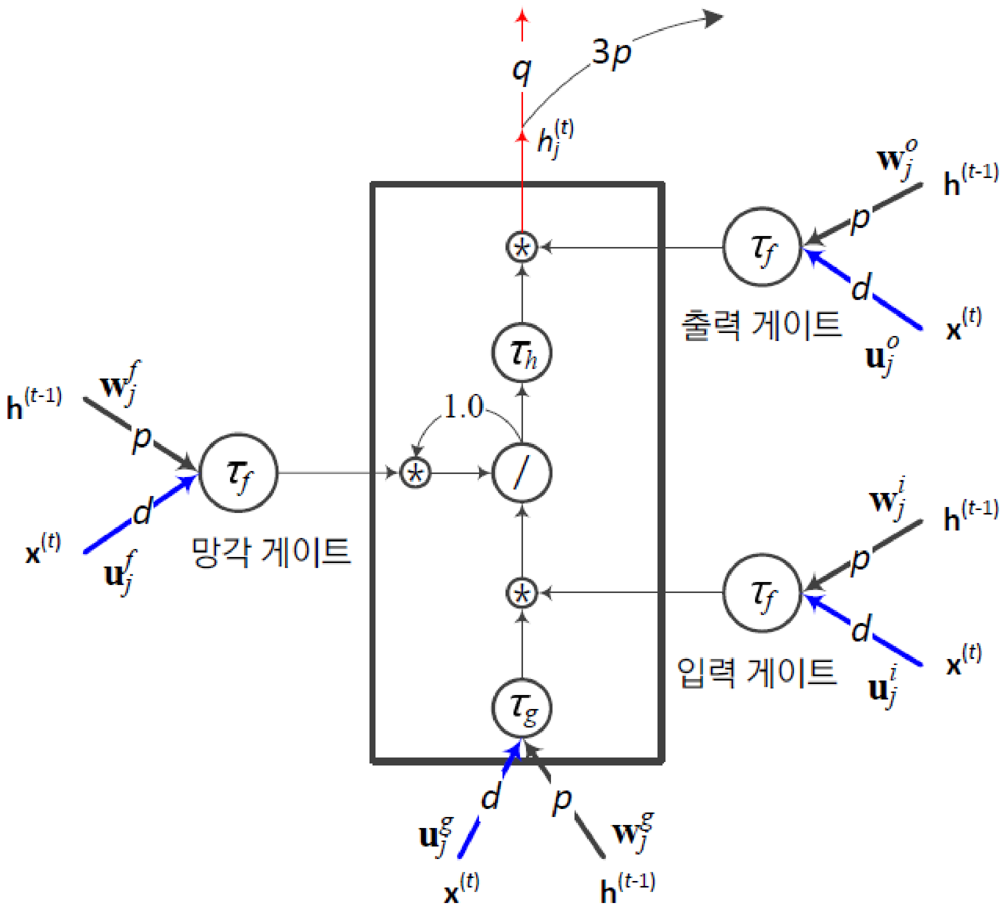


그림 8-18 망각 게이트가 추가된 LSTM 메모리 블록

8.4.3 망각 게이트와 핍홀

- 망각 게이트를 가진 LSTM의 동작(파란 박스는 이전과 다른 곳)

$$\text{입력단: } \mathbf{g} = \tau_g\left(\mathbf{U}^g\mathbf{x}^{(t)} + \mathbf{W}^g\mathbf{h}^{(t-1)} + \mathbf{b}^g\right) \tag{8.40}$$

$$\text{입력 게이트: } \mathbf{i} = \tau_f\left(\mathbf{U}^i\mathbf{x}^{(t)} + \mathbf{W}^i\mathbf{h}^{(t-1)} + \mathbf{b}^i\right) \tag{8.41}$$

$$\text{출력 게이트: } \mathbf{o} = \tau_f\left(\mathbf{U}^o\mathbf{x}^{(t)} + \mathbf{W}^o\mathbf{h}^{(t-1)} + \mathbf{b}^o\right) \tag{8.42}$$

$$\text{망각 게이트: } \boxed{\mathbf{f} = \tau_f\left(\mathbf{U}^f\mathbf{x}^{(t)} + \mathbf{W}^f\mathbf{h}^{(t-1)} + \mathbf{b}^f\right)} \tag{8.43}$$

$$\boxed{\mathbf{s}^{(t)} = \mathbf{f} \odot \mathbf{s}^{(t-1)} + \mathbf{g} \odot \mathbf{i}} \tag{8.44}$$

$$\mathbf{h}^{(t)} = \tau_h\left(\mathbf{s}^{(t)}\right) \odot \mathbf{o} \tag{8.45}$$

$$\mathbf{y}'^{(t)} = \mathrm{softmax}(\mathbf{h}^{(t)}) \tag{8.46}$$

8.4.3 망각 게이트와 핍홀

핍홀 기능으로 LSTM 확장

- 핍홀(노란색 에지)은 블록의 내부 상태를 3개의 게이트에 알려주는 역할을 함
- 순차 데이터를 처리하다가 어떤 조건에 따라 특별한 조치를 취해야 하는 응용에 효과적
  - 예, 음성 인식을 수행하다가 특정 단어가 발견되면 지정된 행위를 수행

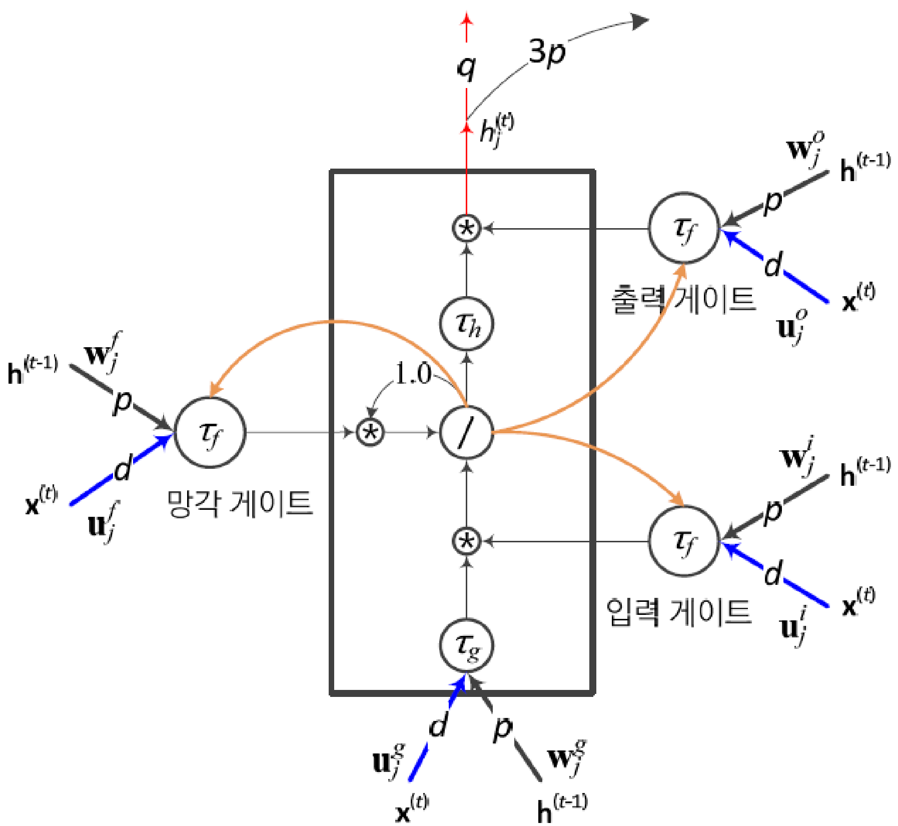


그림 8-19 핍홀이 추가된 LSTM 메모리 블록

8.5 응용 사례

- 8.5.1 언어 모델
- 8.5.2 기계 번역
- 8.5.3 영상 주석 생성

- 순환 신경망은 분별 모델뿐 아니라 생성 모델로도 활용됨
- 장기 문맥을 처리하는 데 유리한 LSTM이 주로 사용됨

8.5.1 언어 모델

언어 모델이란

- 문장, 즉 단어 열의 확률분포
- 예, P(자세히, 보아야,예쁘다)>P(예쁘다,보아야, 자세히)
- 활용
  - 음성 인식기 또는 언어 번역기가 후보로 출력한 문장이 여럿 있을 때, 언어 모델로 확률을 계산한 다음 확률이 가장 높은 것을 선택하여 성능을 높임
- 확률분포를 추정하는 방법
  - n-그램
  - 다층 퍼셉트론
  - 순환 신경망

8.5.1 언어 모델

n-그램을 이용한 언어 모델

- 기계 학습 이전에 사용하던 고전적인 방법
- 문장을 $\mathbf{x} = (z_1, z_2, \cdots, z_T)^{\mathrm{T}}$라 하면, $\mathbf{x}$가 발생할 확률을 식 (8.47)로 추정

$$P(z_1, z_2, \cdots, z_T) = \prod_{t=1}^{T} P(z_t | z_1, z_2, \cdots, z_{t-1}) \tag{8.47}$$

- *n*-그램은 *n*-1개의 단어만 고려하는데, 이때 식 (8.48)이 성립

$$P(z_1, z_2, \cdots, z_T) \approx \prod_{t=1}^{T} P\left(z_t | z_{t-(n-1)}, \cdots, z_{t-1}\right) \tag{8.48}$$

  - 알아야 할 확률의 개수는 m^n → 차원의 저주 때문에 n을 1~3 정도로 작게 해야만 함
  - 확률 추정은 말뭉치를corpus 사용
- 단어가 원핫 코드로 표현되므로 단어 간의 의미 있는 거리를 반영하지 못하는 한계

8.5.1 언어 모델

순환 신경망을 이용한 언어 모델

- 현재까지 본 단어 열을 기반으로 다음 단어를 예측하는 방식으로 학습 → 확률분포 추정뿐만 아니라 문장 생성 기능까지 갖춤
- 비지도 학습에 해당하여 말뭉치로부터 쉽게 훈련집합 구축 가능
- 예, "자세히 보아야 예쁘다"라는 문장은 다음과 같은 샘플이 됨(왼쪽으로 한 칸씩 이동)

$$\mathbf{x} = (<\text{시작}>, \text{자세히}, \text{보아야}, \text{예쁘다})^{\mathrm{T}}, \quad \mathbf{y} = (\text{자세히}, \text{보아야}, \text{예쁘다}, <\text{끝}>)^{\mathrm{T}}$$

- 일반화하면,

$$\mathbf{x} = (<\text{시작}>, \mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_T)^{\mathrm{T}}, \quad \mathbf{y} = (\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_T, <\text{끝}>)^{\mathrm{T}} \tag{8.49}$$

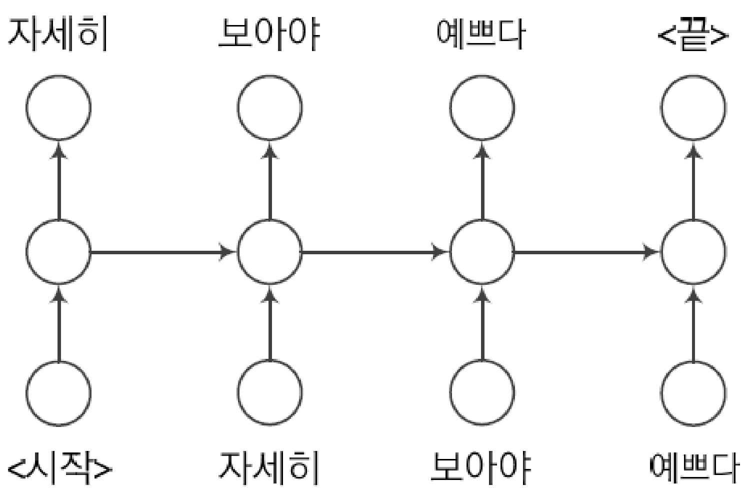


그림 8-20 순환 신경망의 예측 능력을 학습하는 데 사용하는 훈련 샘플

8.5.1 언어 모델

- 순환 신경망의 학습
  - 말뭉치에 있는 문장을 식 (8.49)처럼 변환하여 훈련집합을 만든 다음, 8.2.3절의 BPTT학습 알고리즘을 적용
- 학습을 마친 순환 신경망(언어 모델)의 활용
  - 기계 번역기나 음성 인식기의 성능을 향상하는 데 활용
  - 예, 음성 인식기가 $\tilde{\mathbf{x}}_1 = (자세히, 보아야, 예쁘다)^{\mathrm{T}}$와 $\tilde{\mathbf{x}}_2 = (자세를, 모아야, 예쁘다)^{\mathrm{T}}$라는 2개 후보를 출력했을 때 언어 모델로 $P(\tilde{\mathbf{x}}_i)$를 계산한 후, 높은 확률의 후보를 선택

8.5.1 언어 모델

생성 모델로 활용

- 문장 생성한 예[Karpathy2015]
  - "anyway, to the city scene you're an idiot teenager.", "What ?!!! Ignore!"
- 문장 생성 알고리즘

알고리즘 8-1 순환 신경망으로 문장 생성

입력: 학습된 RNN 언어 모델
출력: 문장 $\mathbf{s}$

1 $\mathbf{s} = (<\text{시작}>)^{\mathrm{T}}$
2 while ($\mathbf{s}$의 마지막 요소 $\neq$ 〈끝〉)
3   $\mathbf{s}$를 RNN에 입력하여 출력 $\tilde{\mathbf{y}} = (\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_{|s|})^{\mathrm{T}}$를 구한다. // 다음 단어 예측
4   $\mathbf{w}_{|s|}$의 확률에 따라 m개 단어 중 하나를 샘플링한다.
5   $\mathbf{s}$의 끝에 라인 4에서 샘플링한 단어를 추가한다.

8.5.2 기계 번역

기계 번역

- 훈련 샘플 예

$$\mathbf{x} = (<\text{시작}>, \text{자세히}, \text{보아야}, \text{예쁘다})^{\mathrm{T}}, \quad \mathbf{y} = (\text{It}, \text{is}, \text{beautiful}, \text{to}, \text{see}, \text{more}, \text{closely}, <\text{끝}>)^{\mathrm{T}}$$

- 언어 모델보다 어려움
  - 언어 모델은 입력 문장과 출력 문장의 길이가 같은데, 기계 번역은 길이가 서로 다른 열대 열sequence to sequence 문제
  - 어순이 다른 문제
- 예전의 통계적 기계 번역 방법은 한계
- 현재는 딥러닝에 기반한 신경망 기계 번역 방법이 주류

8.5.2 기계 번역

- LSTM을 사용하여 번역 과정 전체를 통째로 학습
  - LSTM 2개를 사용(앞쪽은 인코더, 뒤쪽은 디코더)
  - 인코더는 원시 언어 문장 $\mathbf{x}$를 $\mathbf{h}_{Ts}$라는 특징 벡터로 변환
  - 디코더는 $\mathbf{h}_{Ts}$를 가지고 목적 언어 문장 $\mathbf{y}$를 생성함

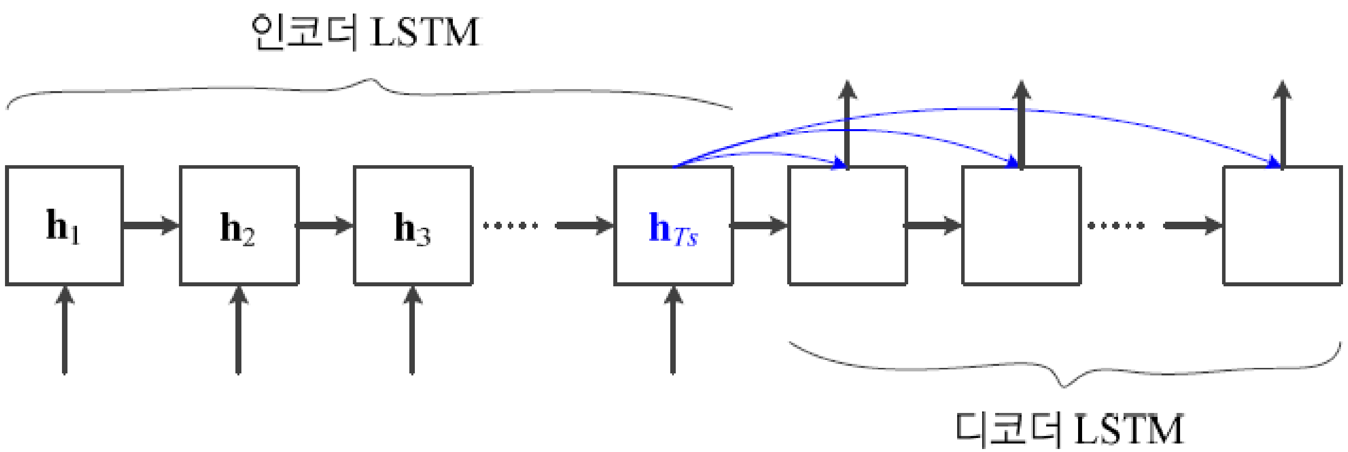


그림 8-21 인코더와 디코더가 특징 벡터를 하나만 사용하는 방식

  - 가변 길이의 문장을 고정 길이의 특징 벡터로 변환한 후, 고정 길이에서 가변 길이 문장을 생성 → 문장이 길이가 크게 다를 때는 성능 저하

8.5.2 기계 번역

- 모든 순간의 상태 변수를 사용하는 방식
  - 인코더의 계산 결과인 $\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_{TS}$를 모두 디코더에 넘겨 줌
  - 양방향 구조를 채택하여 어순이 다른 문제를 해결

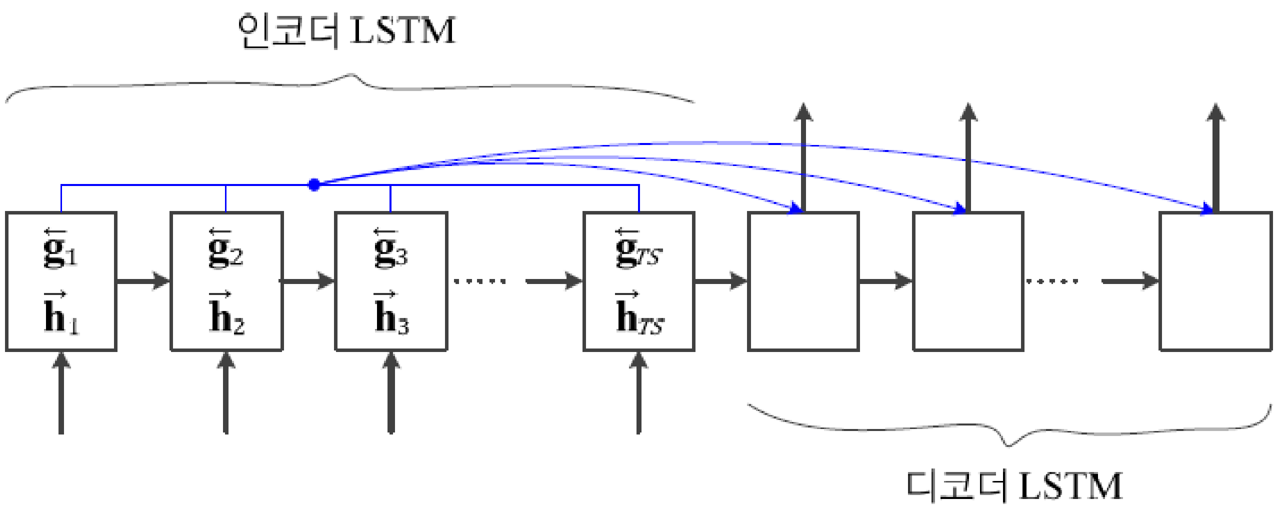


그림 8-22 인코더와 디코더가 여러 특징 벡터를 사용하는 방식

8.5.3 영상 주석 생성

- 영상 주석 생성 데모 사이트(http://deeplearning.cs.toronto.edu/i2t)

a man wearing a blue shirt with his arms on the grass.
a man holding a frisbee bat in front of a green field.
a man throwing a frisbee in a green field.
a boy playing ball with a disc in a field.
a young man playing in the grass with a green ball.

a red car on the side of the road in the small race.
a truck driving uphill on the side of the road.
a person driving a truck on the road.
a small car driving down a dirt and water.
a truck in a field of car is pulled up to the back.

a group of birds standing next to each other.
a group of ducks that are standing in a row.
a group of ducks that are standing on each other.
a group of sheep next to each other on sand.
a group of small birds is standing in the grass.

a kite flying over the ocean on a sunny day.
a person flying over the ocean on a sunny day.
a person flying over the ocean on a cloudy day.
a kite on the beach on the water in the sky.
a large flying over the water and rocks.

그림 8-24 영상 주석 자동 생성 데모 사이트[5]

8.5.3 영상 주석 생성

영상 주석 생성 응용

- 영상 속 물체를 검출하고 인식, 물체의 속성과 행위, 물체 간의 상호 작용을 알아내는 일 + 의미를 요약하는 문장 생성하는 일 ← 매우 도전적인 문제
- 예전에는 물체 분할, 인식, 단어 생성과 조립 단계를 따로 구현한 후 연결하는 접근방법
- 현재는 딥러닝 기술을 사용하여 통째 학습

딥러닝 접근방법

- CNN은 영상을 분석하고 인식 + LSTM은 문장을 생성

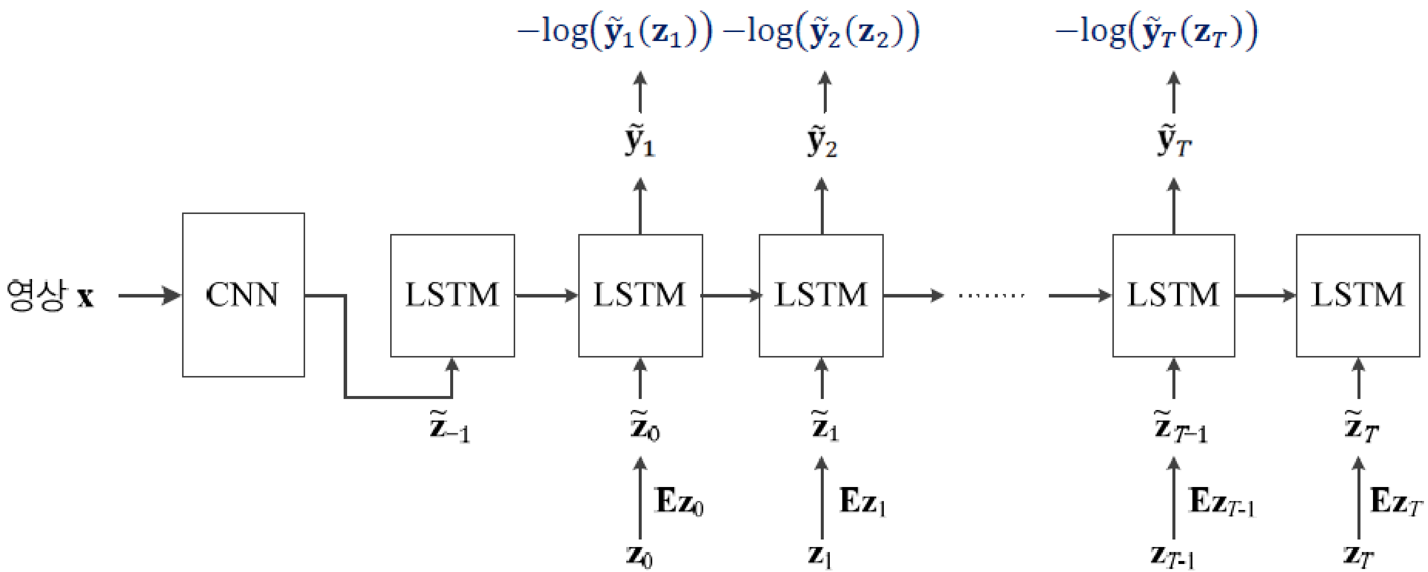


그림 8-23 자연 영상에서 자연어 문장을 생성하는 시스템 구조

8.5.3 영상 주석 생성

- 훈련집합
  - $\mathbf{x}$는 영상, $\mathbf{y}$는 영상을 기술하는 문장($\mathbf{y} = (<\text{시작}>, \mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_T, <\text{끝}>)^{\mathrm{T}}$로 표현됨)
- CNN
  - 입력 영상 $\mathbf{x}$를 단어 임베딩 공간의 특징 벡터 $\tilde{\mathbf{z}}_{-1}$로 변환(식 (8.50)의 첫 번째 줄)

$$\left.\begin{aligned} \tilde{\mathbf{z}}_{-1} &= cnn(\mathbf{x}) \\ \tilde{\mathbf{z}}_t &= \mathbf{E}\mathbf{z}_t, \quad t = 0,1,\cdots,T \end{aligned}\right\} \tag{8.50}$$

- 훈련 샘플 $\mathbf{y}$의 단어 $\mathbf{z}_t$는 단어 임베딩 공간의 특징 벡터 $\tilde{\mathbf{z}}_t$로 변환됨
  - 식 (8.50)의 두 번째 줄에서 행렬 $\mathbf{E}$를 이용하여 변환
  - $\mathbf{E}$는 통째 학습 과정에서 CNN, LSTM과 동시에 최적화됨

8.5.3 영상 주석 생성

- 학습 과정의 입력
  - 영상 $\mathbf{x}$를 CNN에 입력함
  - 문장 $\mathbf{z}_0, \mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_T$를 임베딩 공간의 점 $\tilde{\mathbf{z}}_0, \tilde{\mathbf{z}}_1, \tilde{\mathbf{z}}_2, \cdots, \tilde{\mathbf{z}}_T$로 변환하여 LSTM에 입력함

- 목적함수
  - LSTM의 출력 $\tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_2, \cdots, \tilde{\mathbf{y}}_T$와 $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_T$가 일치할수록 예측을 잘한다고 평가
  - 식 (8.51)의 로그우도로 일치 정도를 평가

$$J(\Theta) = -\sum_{t=1}^{T} \log\left(\tilde{\mathbf{y}}_t(\mathbf{z}_t)\right) \tag{8.51}$$

- 학습이 최적화해야 할 매개변수 집합
  - $\Theta = \{\text{CNN 매개변수}, \text{LSTM 매개변수}, \text{단어 임베딩 매개변수}\}$
    - 전이 학습을 사용하므로 CNN 매개변수는 완전연결층의 가중치
    - LSTM 매개변수는 식 (8.40)~(8.46)에 있는 $\mathbf{U}^g, \mathbf{W}^g, \mathbf{U}^i, \mathbf{W}^i, \mathbf{U}^o, \mathbf{W}^o, \mathbf{U}^f, \mathbf{W}^f$
    - 단어 임베딩 매개변수는 식 (8.50)의 $\mathbf{E}$

- Θ는 통째 학습으로 한꺼번에 최적화됨