Chapter 9. 뉴럴튜링머신

<기계학습 개론> 강의
서울대학교 컴퓨터공학부
장 병 탁

교재: 장교수의 딥러닝, 홍릉과학출판사, 2017.

Slides Prepared by
장병탁, 한동식

Biointelligence Laboratory
School of Computer Science and Engineering
Seoul National University

BI
BIOINTELLIGENCE

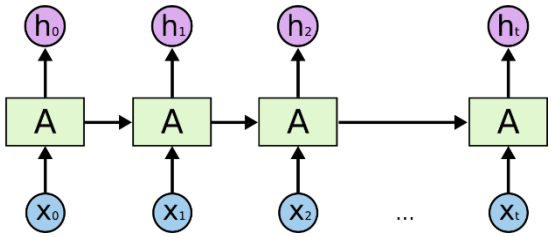
목차

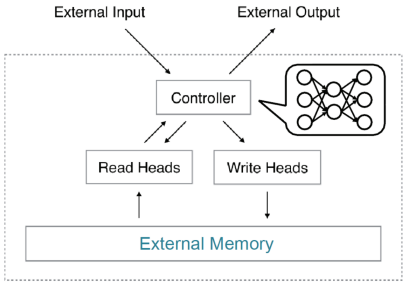
들어가는 질문

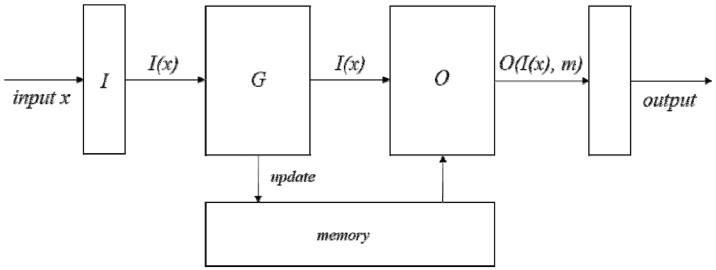
- 순환신경망(RNN) 모델에서 망각 문제는 왜 발생하는가?
- 학습시 망각 문제를 해결하기 위한 여러가지 방법들을 제안하시오.
- 뉴럴튜링머신(NTM)의 구조를 기술하시오. 특징은 무엇인가?
- NTM에서 읽기, 쓰기, 삭제 연산 메카니즘을 설명하시오.
- NTM에서 두 가지 주소지정 메커니즘을 설명하시오.
- NTM은 어떻게 학습하는가?
- 메모리넷(MemNet)의 구조를 기술하시오.
- 메모리넷에서 입력, 일반화, 출력, 응답 모듈의 기능을 설명하시오.
- 메모리넷의 학습 방법을 설명하시오.

Overview

- 순환신경망(Recurrent Neural Networks: RNN)
  - 순차적 정보를 하나의 고정된 은닉 뉴런에 인코딩
  - 메모리 능력의 부족에 따른 망각 문제
- 뉴럴튜링머신(Neural Turing Machine, NTM)
  - 순환신경망을 통한 제어기와 외부 메모리를 결합하는 딥러닝 모델
  - 벡터 연산으로 동작하는 미분 가능한 기계학습 기반의 튜링머신
- 메모리넷 (Memory Networks, MemNet)
  - 순환신경망의 기억 능력을 향상시키기 위한 방법
  - 입력이 들어올 때 은닉 유닛을 만들어 메모리 장치에 장기 메모리로 저장







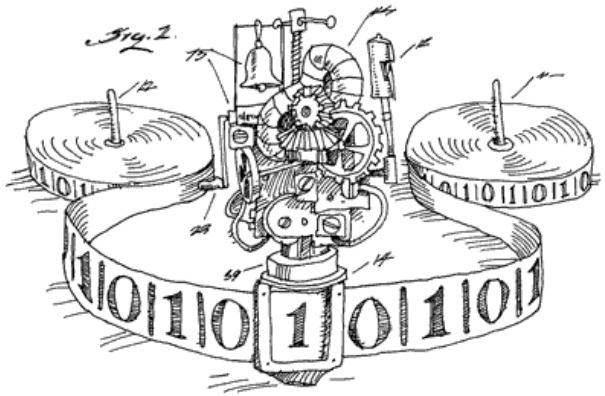
9.1 뉴럴튜링머신(NTM)의 구조 (1/5)

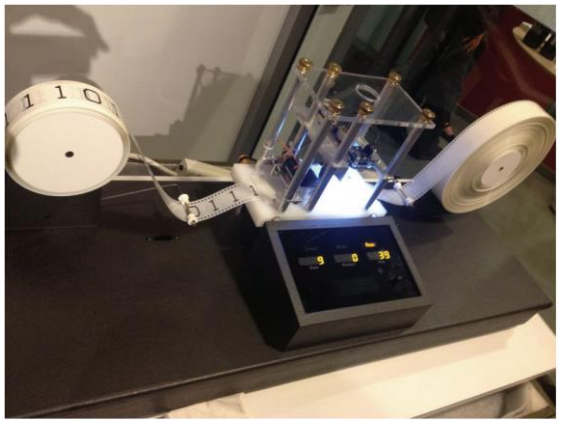
- 컴퓨터 프로그램의 대표적 연산들
  - 산술 연산: 일반적인 계산, 예: 1+3
  - 논리 흐름 제어 연산: 조건부 분기를 담당
  - 메모리 연산: 데이터를 읽고 쓰는 외부 저장장치에 대한 접근을 다룸

- 튜링 완전(Turing Complete): 위의 연산들이 가능한 시스템은 모든 프로그램을 실행 가능
  - 기계학습에서는 추상화된 표현에 사상하는 “산술연산”을 중시한 방법을 주로 사용
  - 프로그램을 완벽히 수행하기 위한 분기 및 외부 저장장치에 대한 고려 부족

- David Hilbert (1928), *Entscheidungsproblem* (decision problem, 결정 문제): 공리에 대한 1차 논리의 보편적 유효성을 판별하는 알고리듬이 있는가?
  - 기계적 절차(알고리듬)의 엄밀한 정의와 이를 해결할 수 있는 시스템의 정의가 필요

9.1 뉴럴튜링머신(NTM)의 구조 (2/5)





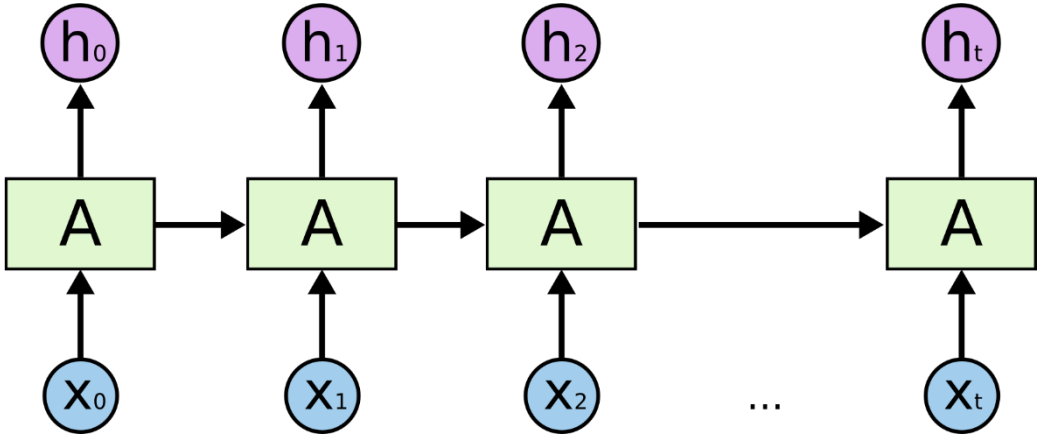
- 튜링머신: 결정 문제를 증명하기 위해 고안한 계산기계
- 정해진 절차에 따라 특정 계산을 수행하는 기계가 충분한 기억 장소와 정확한 알고리듬만 주어진다면 어떠한 계산이라도 가능함을 보여줌
- 튜링머신의 요약
  - “셀”이라 불리는 칸으로 나누어진 무한한 길이의 연속된 테이프가 있다고 가정
  - “헤드”는 기호를 읽거나 테이프에 쓸 수 있으며 한 번에 한 칸의 셀로 이동 가능
  - “상태 저장소” 존재. 상태 들은 인간이 계산을 수행할 때 인간 마음의 상태에 비유할 수 있음
  - 튜링머신이 특정 작업을 완료하기 위해 필요한 유한 개의 명령어 테이블 존재

9.1 뉴럴튜링머신(NTM)의 구조 (3/5)

- 순환신경망
  - 순환신경망은 연속된 입력 패턴들의 시간적인 순서를 고려할 수 있음
  - 이론적으로, 모든 시간 단위를 고려한 연산을 모델링 할 수 있기 때문에 순환신경망 튜링완전기계의 조건을 충족

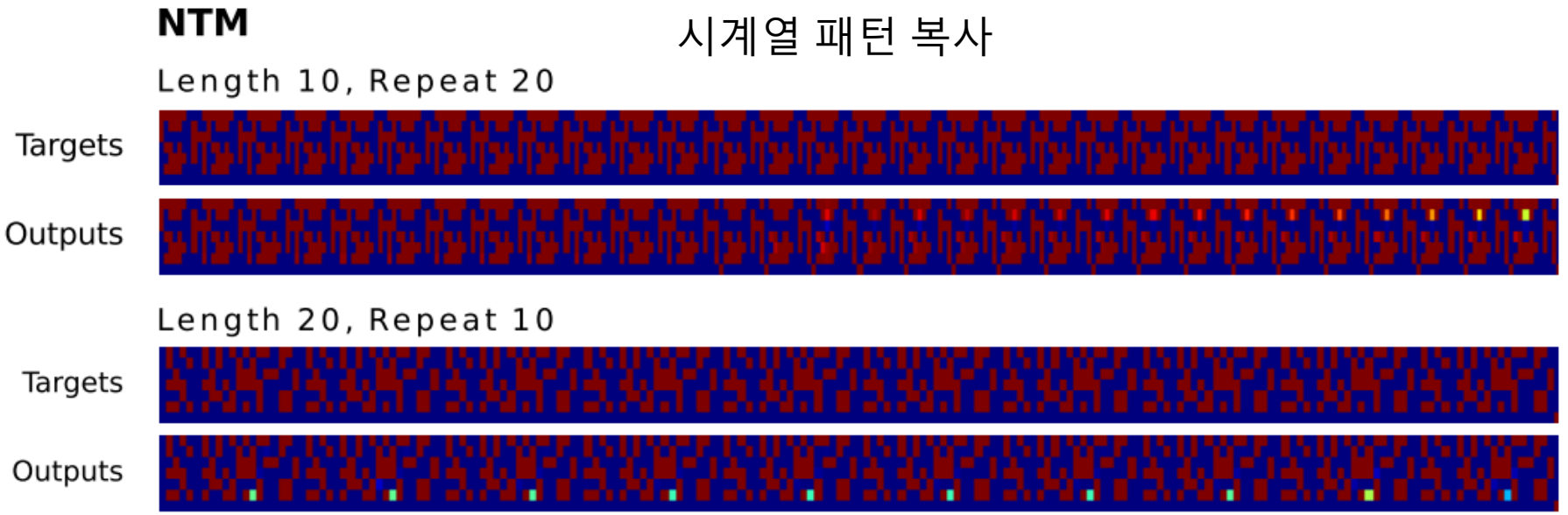
    <=> RNN을 학습하여 튜링 완전 언어를 시뮬레이션 할 수 있음

    <=> RNN을 학습하여 튜링머신을 만들 수 있음
  - 실제로는, 모델의 크기와 학습시간 등의 조건이 제한된 상황에서는 단순 순환신경망은 강건하지 못하고, 기억에 많은 오류가 있음을 실험적으로 보여줌



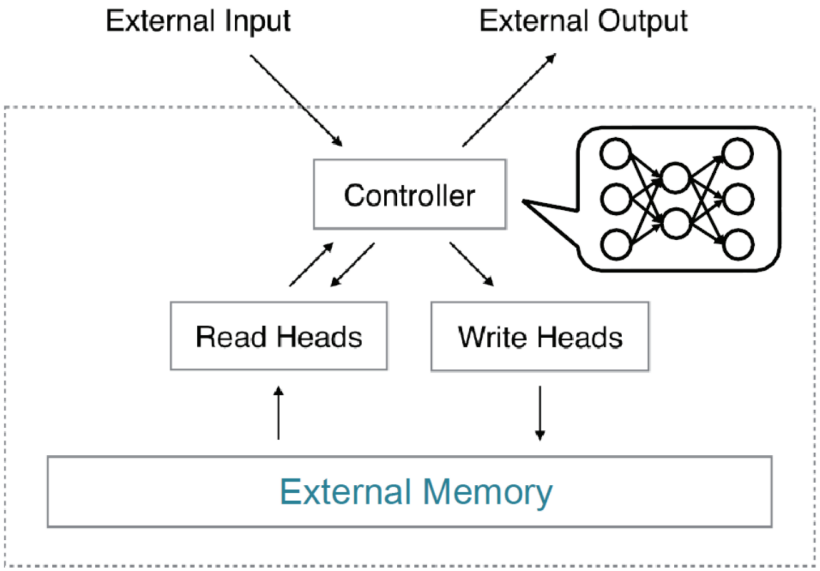
9.1 뉴럴튜링머신(NTM)의 구조 (4/5)

- 뉴럴튜링머신
  - 튜링머신의 개념에서 영감을 받아 구현된 실제적인 시스템
  - 순환신경망을 통한 제어기 + 외부 메모리
  - 벡터 연산으로 동작하는 미분 가능한 튜링머신
- 뉴럴튜링머신을 이용하여 복사, 정렬, 연관기억 연상 등의 간단한 알고리듬이 수행될 수 있음을 실험을 통해 증명함



9.1 뉴럴튜링머신(NTM)의 구조 (5/5)

- 실수 벡터 형태의 입력과 출력
- 제어기(Controller)
  - 매개변수화된 제어기의 출력 연산들을 튜링 기계의 헤드에 비유
  - 실수 벡터의 흐름만으로 읽기, 쓰기 연산을 사용하여 메모리와 직접 상호작용
- 외부 기억장치(External Memory)
- 선택적 집중 매커니즘(attentional process)을 도입한 구성요소 학습
  - 읽기와 쓰기 연산에 일종의 가중치를 도입한 것
  - 선택적 집중 연산의 도입을 통해 전체 메모리의 일부분만을 접근



9.2 NTM의 읽기 및 쓰기 연산 (1/2)

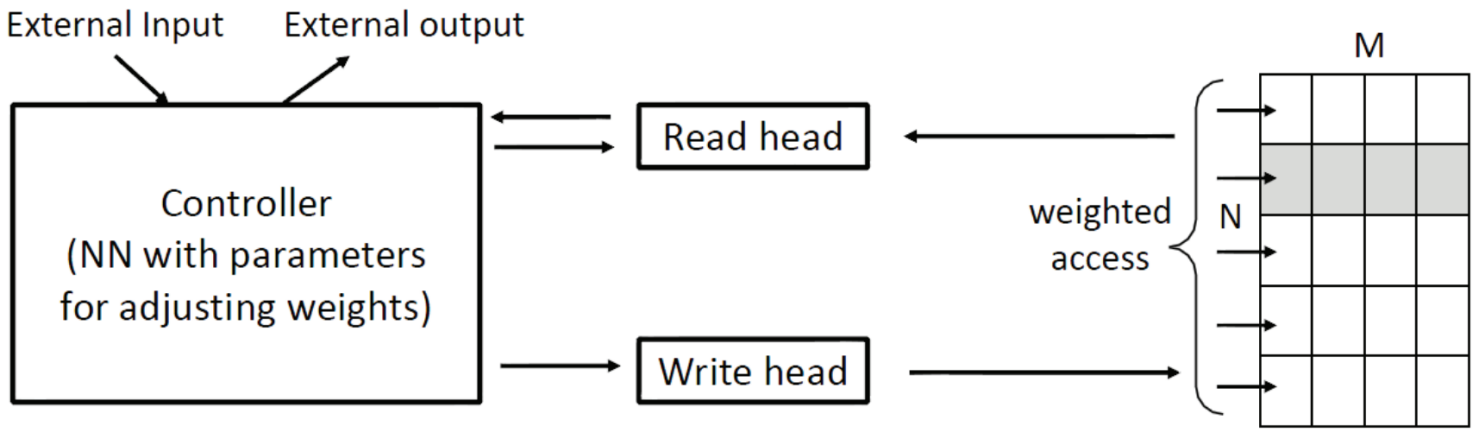
- 제어기를 거쳐 메모리 슬롯에 대한 읽기와 쓰기 헤드 가중치를 표출
  - 읽기 가중치에 따라 현재의 출력 벡터 생성
  - 쓰기 가중치에 따라 메모리를 새로운 상태로 변환
- M_t: 모델이 주로 다루게 되는 각 시간 단계 t에서의 $M \times N$ 크기의 메모리 행렬 (N: 메모리 슬롯의 갯수, M: 데이터 벡터의 크기)
- w_t: 시각 t때 읽기 헤드로부터 나온 모든 위치 N에 대한 가중치 벡터

$$\sum_i w_t(i) = 1, \qquad 0 \le w_t(i) \le 1, \forall i.$$

- **읽기 작업**

  r_t : 읽기 헤드에서 나오는 길이 M의 읽기 벡터. 메모리의 행 벡터 $M_t(i)$들의 convex 조합

$$r_t \leftarrow \sum_i w_t(i) M_t(i)$$



9.2 NTM의 읽기 및 쓰기 연산 (2/2)

- **삭제 작업**
  - e_t : 모든 요소가 0~1 사이의 값을 가지는 길이 M 의 삭제 벡터

$$\tilde{M}_t(i) \leftarrow M_{t-1}(i)[1 - w_t(i)e_t]$$

  - 1은 모든 요소 값이 1인 행 벡터를 의미하고, 메모리에 적용되는 곱셈은 요소 단위의 스칼라 곱셈을 의미
- **쓰기 작업**
  - a_t: 길이 M 의 추가 벡터

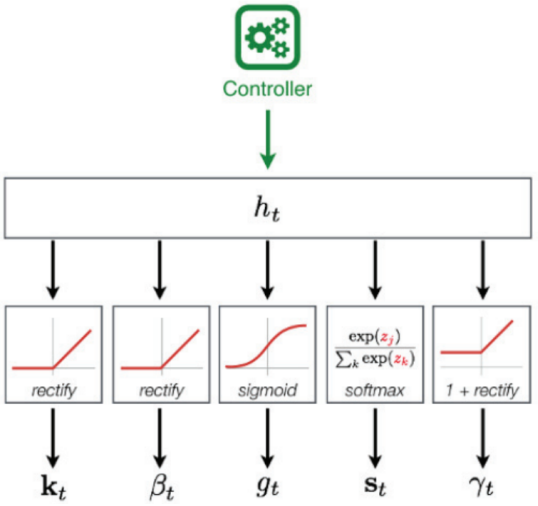
$$M_t(i) \leftarrow \tilde{M}_t(i) + w_t(i)a_t$$

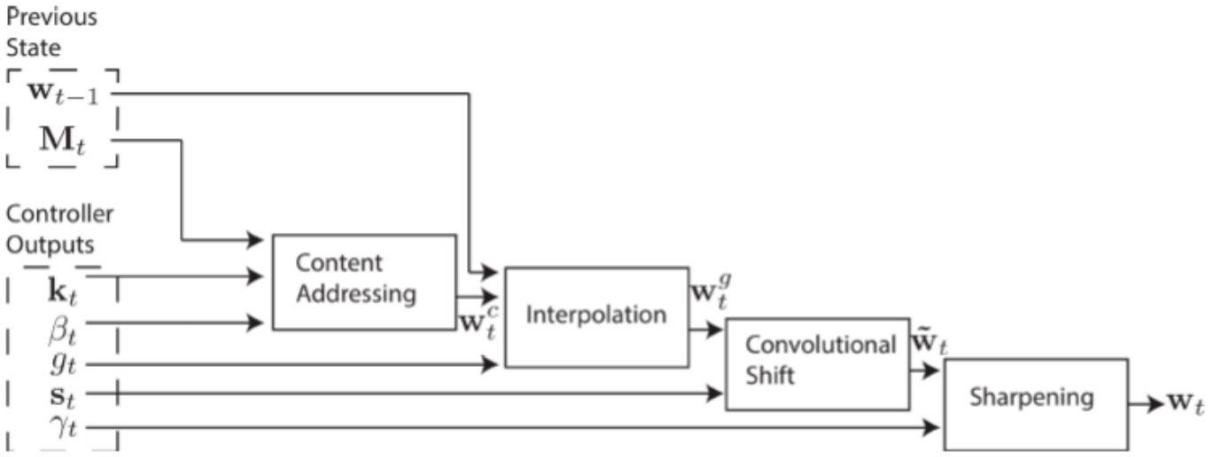
  - 삭제 및 추가연산을 모든 쓰기 헤드에 대해 수행하면 최종적으로 시각 t에서의 메모리 M_t를 계산 가능
- 이러한 일련의 과정들은 모두 학습이 가능하다

9.3 NTM의 주소지정 메커니즘 (1/4)

두 개의 주소지정 매커니즘을 적절히 조합해서 가중치 갱신에 사용

- w_t: 가중치 벡터
  - 메모리 내 벡터들 중 적절한 위치를 특정하여 사상
  - 두 가지 주소지정 매커니즘을 고려하여 그 값이 계산됨
- 내용기반 주소지정 매커니즘
  - 메모리 내의 실수 벡터 중 제어기의 값과 유사한 벡터의 위치에 헤드에 집중
  - 메모리에 저장된 원하는 내용을 그대로 불러올 수 있다는 장점
- 위치기반 주소지정 메커니즘
  - 메모리 내의 유사도를 비교하는 등의 번거로움 없이 단순히 값이 저장된 위치에 직접 접근하는 것이 효율적일 경우 사용





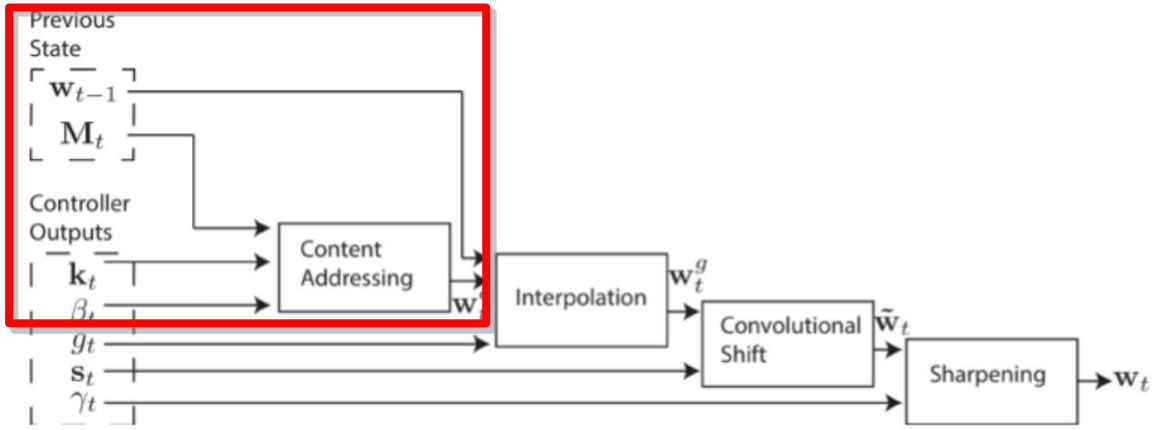
9.3 NTM의 주소지정 메커니즘 (2/4)

- **내용기반 주소지정**
  - 읽기 또는 쓰기 헤더는 먼저 길이 M 의 키 벡터 k_t를 생성

$$K[u,v] = \frac{u \cdot v}{\|u\| \cdot \|v\|}$$

  - 집중도를 증폭시키기 위한 키 강도 β_t와 함께 현재의 M_t내에서 k_t와 가장 비슷한 벡터를 찾아 곱한 후 정규화

$$w_t^{\mathrm{i}}(i) \leftarrow \frac{\exp(\beta_t K[k_t, M_t(i)])}{\sum_j \exp(\beta_t K[k_t, M_t(j)])}$$



9.3 NTM의 주소지정 메커니즘 (3/4)

- **위치기반 주소지정**
  - 보간(interpolation): 이전 가중치 w_{t-1}와 현재 내용기반 가중치 w^c_{t-1}를 0~1 사이의 값을 가지는 보간 게이트 g_t를 사용하여 혼합

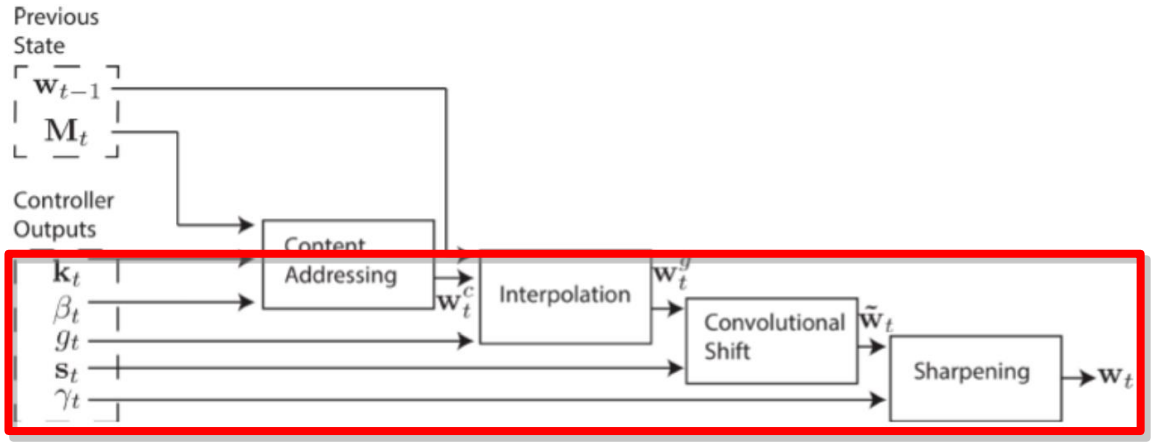
$$w_t^g \leftarrow g_t w_t^c + (1-g_t) w_{t-1}$$

  - 회선(convolutional): 이동 가중치에 의해 회선이동(convolutional shift)

$$w_t(i) \leftarrow \sum_{j=0}^{N-1} w_t^g(j) s(i-j)$$

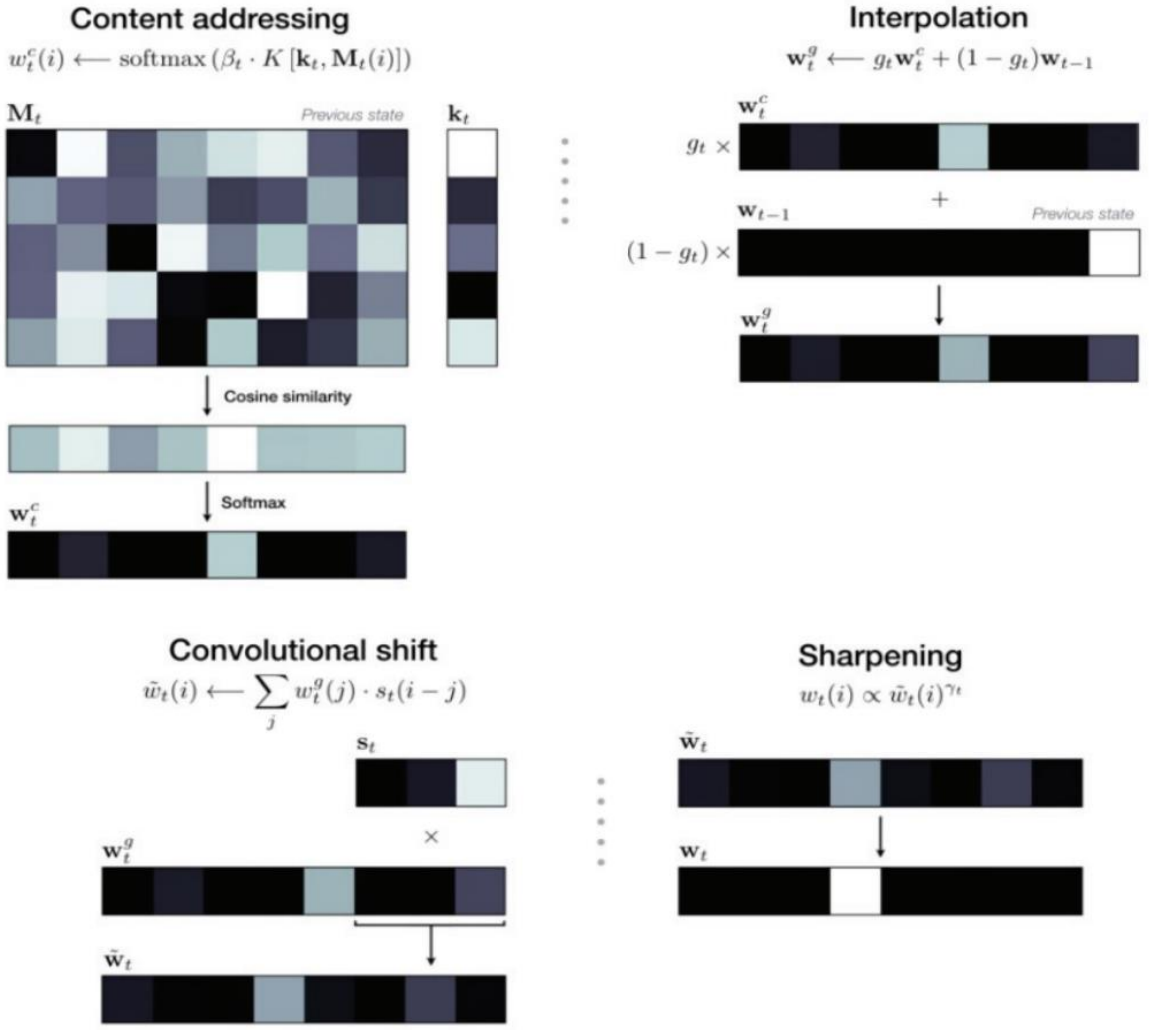
  - 샤프닝(sharpening): 샤프닝 가중치 $\gamma_t \geq 1$에 의해 최종적으로 샤프닝되어 주소지정이 완료

$$w_t(i) \leftarrow \frac{w_t(i)^{\gamma_t}}{\sum_j w_t(j)^{\gamma_t}}$$



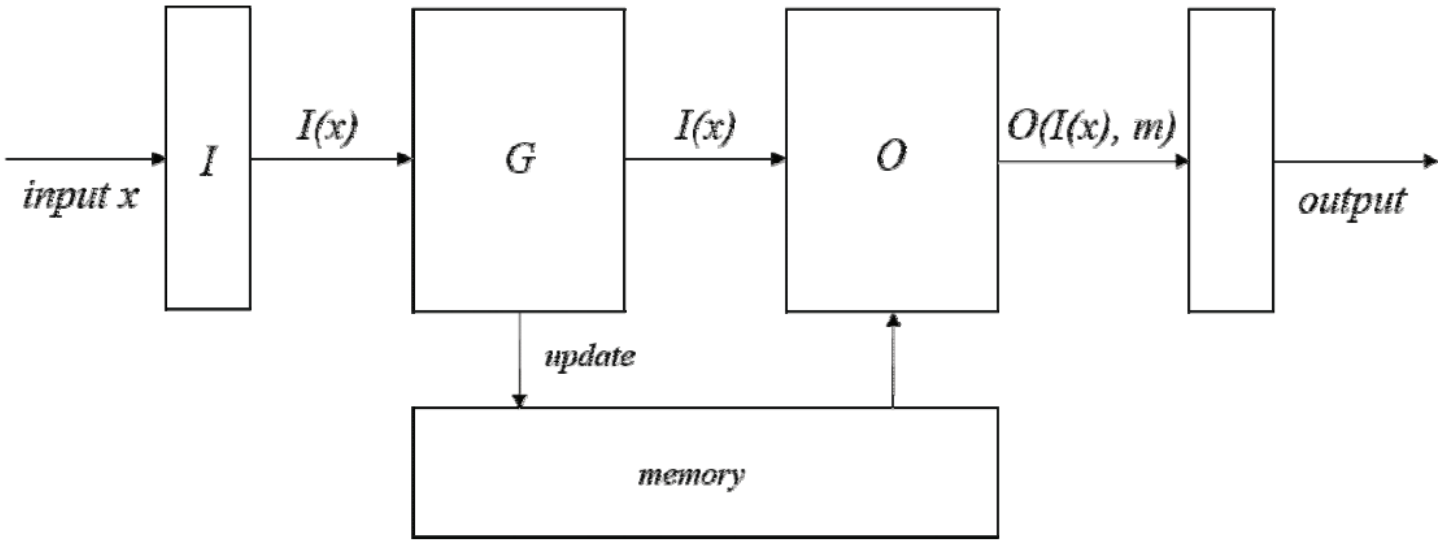
9.3 NTM의 주소지정 메커니즘 (4/4)

- 주소지정 메커니즘의 시각화



9.4 메모리넷(MemNet) (1/7)

- 기억 기제를 모사하기 위해 메모리를 장기 메모리로 활용
- 기존 딥러닝 모델의 기억 용량 한계를 극복하려는 시도
  - 각 시간 순서마다 은닉 벡터를 새로 만들어서 장기 메모리에 저장함
- 순환신경망의 고정된 크기의 은닉 유닛의 문제
  - 입력의 길이가 길어질수록 은닉 유닛의 크기도 커져야 함
  - 은닉 유닛의 업데이트가 이루어질수록 이전 정보가 손실됨



9.4 메모리넷(MemNet) (2/7)

- *I*: (입력 특징 모듈) - 들어오는 입력 *x*를 특징 벡터 *I*(*x*)로 변환시킨다.
  - 일반적인 전처리 과정 포함 (파싱, 엔티티 분석 등)
  - 임베딩 행렬을 통한 분산 정보 표현 가능
- *G*: (일반화 모듈) - 오래된 메모리를 업데이트하거나 새로 추가
  - 예) *I*(*x*)를 메모리의 한 슬롯에 저장
  - $m_{H(x)} = I(x)$ 이 때, *H*는 슬롯을 선택하는 함수
  - *G*는 메모리 *m*의 인덱스 *H*(*x*)를 업데이트
- *O*: (출력 특징 모듈) - 입력과 현재 메모리 상태를 바탕으로 새로운 출력을 생성
- *R*: (응답 모듈) - 벡터 공간 상에서 표현된 출력을 적절한 포맷으로 변환

9.4 메모리넷(MemNet) (3/7)

- 모델의 흐름
  - x를 특징 벡터 $I(x)$로 변환한다.
  - 새로운 입력에 따라서 메모리 m_i를 업데이트 시킨다.
    $$m_i = G(m_i, I(x), m)$$
  - 입력과 메모리 상황에 따라 출력 특징 o를 계산한다.
    $$o = (I(x), m)$$
  - 마지막으로 o를 최종 응답 r로 디코딩한다.
    $$r = R(o)$$
- 입력
  - 입력들은 메모리 상에서 다음으로 가능한 슬롯에 저장이 된다
  - 예) $H(x)$는 다음으로 비어 있는 메모리 슬롯 N을 반환하고 $m_N = x$, $N = N+1$이 된다.
- 일반화
  - G모듈은 새로운 입력을 메모리 에 저장하고 다른 저장되어 있는 메모리는 변경하지 않는다
- 추론
  - O와 R 모듈에서 일어난다
  - O 모듈은 입력 x가 주어졌을 때 k개의 관련 메모리들을 찾아서 출력을 생성한다

9.4 메모리넷(MemNet) (4/7)

메모리넷의 구현

- *k*가 1일 때 가장 높은 매칭 점수를 얻는 관련 메모리가 선택된다

$$o_1 = O_1(x, m) = \arg\max_{i=1,\ldots,N} s_O(x, m_i)$$

  - 이 때 s_O는 문장 *x*와 메모리 m_i의 쌍을 비교하고 점수를 매기는 함수
- *k*가 2인 경우, 선택한 관련 메모리와 입력을 바탕으로 두 번째 관련 메모리를 찾는다

$$o_2 = O_2(x, m) = \arg\max_{i=1,\ldots,N} s_O([x, m_{o1}], m_i)$$

  - 관련 후보 메모리 m_i는 입력과 첫번째 관련 메모리에 기반하여 점수가 매겨짐
- 출력 *o*는 [*x*, m_{o1}, m_{o2}]고 모듈 *R*의 입력이 된다.

9.4 메모리넷(MemNet) (5/7)

- *R* 모듈은 텍스트 응답 *r*을 생성한다.
  - 가장 간단한 응답은 m_{ok}를 반환하는 것
  - 문장생성을 하려면 이 모듈에서 RNN을 사용
  - 출력이 한 단어일 경우 후보 단어들에 점수를 매겨서 선택

$$r = \arg\max_{w \in W} s_R([x, m_{o1}, m_{o2}], w)$$

  - *W*는 가능한 단어의 집합이고, s_R은 매치에 점수를 매기는 함수이다.
- 신경망을 사용하는 함수 s_O와 s_R은 다음과 같이 모델링될 수 있다.

$$s(x, y) = \phi_x(x)^T U^T U \phi_y(\mathrm{y})$$

  - *U*: $n \times D$ 크기의 행렬
    - *D*: 특징 개수
    - *n*: 임베딩 크기
  - ϕ_x, ϕ_y 는 입력 문장을 크기 *D*의 특징 공간에 매핑시켜주는 함수이다. 간단한 특징 공간은 단어 주머니(bag-of-words)를 사용하는 것이다.

9.4 메모리넷(MemNet) (6/7)

메모리넷 학습

- 훈련데이터에 주어진 입력과 최종 응답 및 관련 문장 정보를 통해 감독학습
- 훈련은 마진기반 랭킹 함수와 확률적 경사 하강법 (stochastic gradient descent, SGD)를 사용
- 주어진 입력 x와 출력 r과 관련 메모리 m_{o1}과 m_{o2}가 있을 때, 모델 파라미터 U_O와 U_R을 최소화시키는 방향으로 학습이 이루어짐

$$\sum_{\bar{f} \neq m_{o1}} \max(0, \gamma - s_O(x, m_{o1})) + s_O(x, \bar{f})) +$$

$$+ \sum_{\bar{f}' \neq m_{o2}} \max(0, \gamma - s_O([x, m_{o1}], m_{o2})) + s_O([x, m_{o1}], \bar{f}'))$$

$$+ \sum_{\bar{r} \neq r} \max(0, \gamma - s_R([x, m_{o1}, m_{o2}], r)) + s_R([x, m_{o1}, m_{o2}], \bar{r}))$$

  - $\bar{f}, \bar{f}', \bar{r}$: 정답 레이블이 아닌 다른 선택 값들이고 γ는 마진
  - R 모듈에서 RNN을 사용하여 문장을 생성하는 경우, 위의 식을 일반적인 언어모델에서 사용하는 일반적인 로그우도 함수를 사용할 수 있다

9.4 메모리넷(MemNet) (7/7)

Table 1: Sample statements and questions from tasks 1 to 10.

Task 1: Single Supporting Fact
Mary went to the bathroom.
John moved to the hallway.
Mary travelled to the office.
Where is Mary? A:office

Task 2: Two Supporting Facts
John is in the playground.
John picked up the football.
Bob went to the kitchen.
Where is the football? A:playground

Task 3: Three Supporting Facts
John picked up the apple.
John went to the office.
John went to the kitchen.
John dropped the apple.
Where was the apple before the kitchen? A:office

Task 4: Two Argument Relations
The office is north of the bedroom.
The bedroom is north of the bathroom.
The kitchen is west of the garden.
What is north of the bedroom? A: office
What is the bedroom north of? A: bathroom

Task 5: Three Argument Relations
Mary gave the cake to Fred.
Fred gave the cake to Bill.
Jeff was given the milk by Bill.
Who gave the cake to Fred? A: Mary
Who did Fred give the cake to? A: Bill

Task 6: Yes/No Questions
John moved to the playground.
Daniel went to the bathroom.
John went back to the hallway.
Is John in the playground? A:no
Is Daniel in the bathroom? A:yes

Task 7: Counting
Daniel picked up the football.
Daniel dropped the football.
Daniel got the milk.
Daniel took the apple.
How many objects is Daniel holding? A: two

Task 8: Lists/Sets
Daniel picks up the football.
Daniel drops the newspaper.
Daniel picks up the milk.
John took the apple.
What is Daniel holding? milk, football

Task 9: Simple Negation
Sandra travelled to the office.
Fred is no longer in the office.
Is Fred in the office? A:no
Is Sandra in the office? A:yes

Task 10: Indefinite Knowledge
John is either in the classroom or the playground.
Sandra is in the garden.
Is John in the classroom? A:maybe
Is John in the office? A:no

1. A가 사과를 집었다.
2. **A**가 **사무실**로 갔다.
3. **A**가 **부엌**으로 갔다.
4. **A**가 **사과**를 놓았다.

Q. **사과**는 **부엌** 전에 어디 있었는가? **사무실**

추론 과정: 4 (사과,부엌) ➔ 3 (A,부엌) ➔ 2 (A)

13 - Compound Coref.	26	94	99	100	100	100	100	100	250 ex.	100
14 - Time Reasoning	19	27	99	99	100	99	100	99	500 ex.	99
15 - Basic Deduction	20	21	96	74	73	**100**	77	**100**	100 ex.	**100**
16 - Basic Induction	43	23	24	27	**100**	**100**	**100**	**100**	100 ex.	94
17 - Positional Reasoning	46	51	61	54	46	49	57	65	FAIL	72
18 - Size Reasoning	52	52	62	57	50	74	54	**95**	1000 ex.	93
19 - Path Finding	0	8	49	0	9	3	15	36	FAIL	19
20 - Agent's Motivations	76	91	95	100	100	100	100	100	250 ex.	100
Mean Performance	34	49	79	75	79	83	87	93		92

요약

- NTM의 구조
  - 신경망으로 이루어진 제어기와 메모리, 그리고 메모리를 조작할 수 있는 벡터들의 연산 방식
- NTM의 읽기 및 쓰기 연산
  - 메모리 슬롯에 대한 읽기와 쓰기 헤드에 대한 가중치를 표출
  - 선택적 집중 메커니즘을 도입한 미분 가능한 연산
- NTM의 주소지정 메커니즘
  - 내용기반 주소지정
  - 위치기반 주소지정
- 메모리넷
  - 입력, 일반화, 출력, 응답 모듈
  - 1회 이상의 메모리 검색을 통한 추론