과학기술정보통신부
Ministry of Science and ICT

정보통신기획평가원

medici

AiDA Lab.
Artificial Intelligence & Data Analysis Laboratory

2021년 혁신성장 청년인재 집중양성 추경 사업

빅데이터 분야

자연어 처리

RNN (순환신경망)

Seokhwan Yang

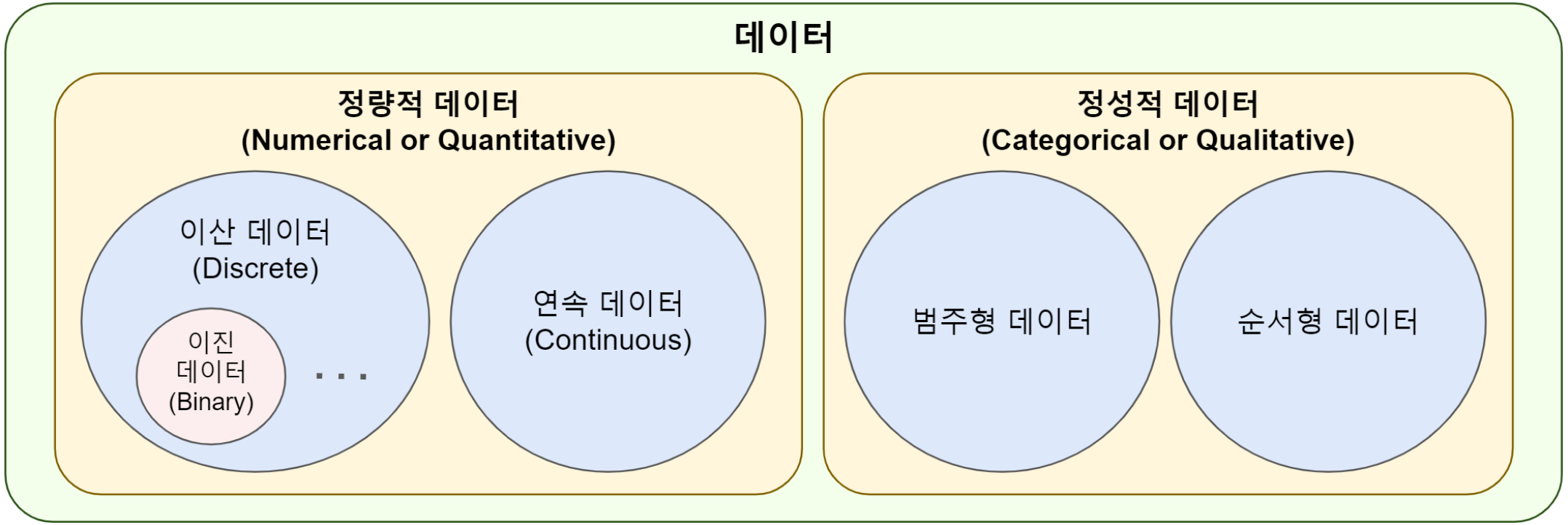
순환 신경망 모델
(RNN, Recurrent Neural Network)

데이터

- **데이터의 정의**
  - **추론과 추정의 근거를 이루는 사실 (옥스퍼드 대사전)**
  - **데이터는**
    - 객관적 사실
    - 추론, 예측, 전망, 추정을 위한 근거로 작용하는 것
    - 각각의 개별 데이터는 그 자체로 큰 의미가 없음 → 정보로 변환 필요
  - **데이터가 특정 기준에 따라 가공, 처리 및 분류되고 정리되어 데이터 간 연관 관계 속에서 의미를 가지며, 유용한 효과를 가지도록 한 것 → 정보**

데이터의 종류: 속성에 따른 분류

- 데이터의 속성에 따른 분류



이 외에도 수치 데이터(Numerical)를 범위형(Interval), 비율형(Ratio) 등 데이터의 내용을 기준으로 분류하는 경우도 있음

데이터의 종류: 형태에 따른 분류

- 테이블 데이터 (=레코드 데이터)

Tid	Refund	Marital Status	Taxable Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

(a) Record data.

TID	ITEMS
1	Bread, Soda, Milk
2	Beer, Bread
3	Beer, Soda, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Soda, Diaper, Milk

(b) Transaction data.

Projection of x Load	Projection of y Load	Distance	Load	Thickness
10.23	5.27	15.22	27	1.2
12.65	6.25	16.22	22	1.1
13.54	7.23	17.34	23	1.2
14.27	8.43	18.45	25	0.9

(c) Data matrix.

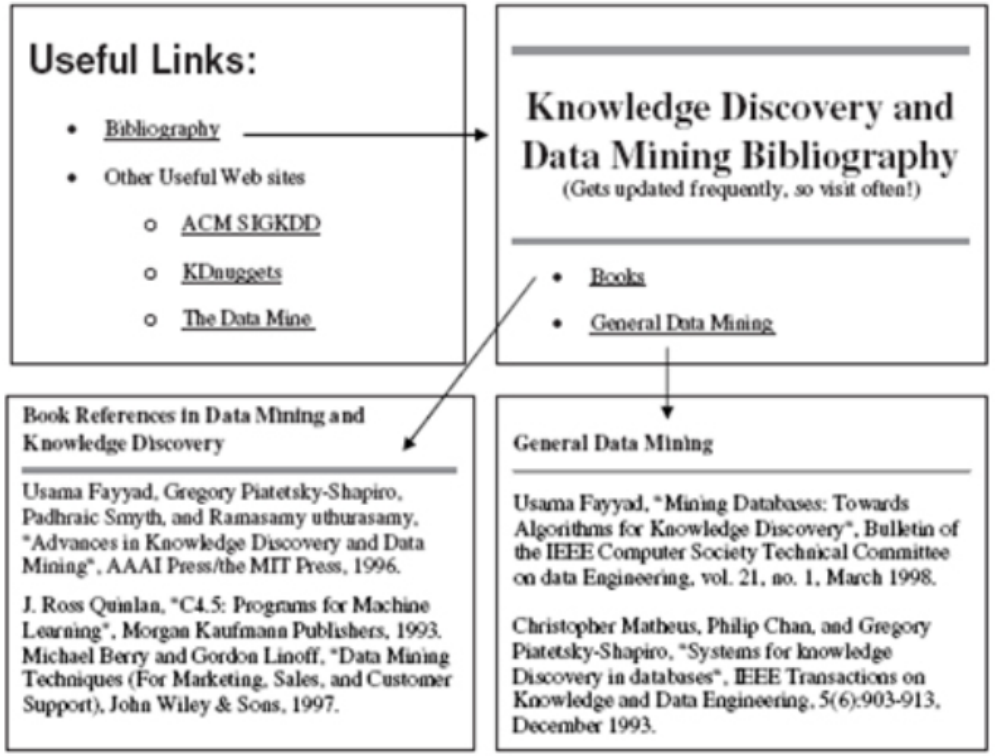
	team	coach	play	ball	score	game	win	lost	timeout	season
Document 1	3	0	5	0	2	6	0	2	0	2
Document 2	0	7	0	2	1	0	0	3	0	0
Document 3	0	1	0	0	1	2	2	0	3	0

(d) Document-term matrix.

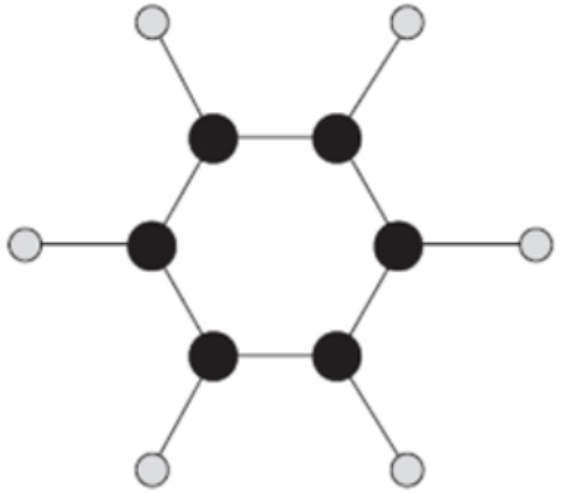
(그림 출처: Pang-Ning Tan et al, Introduction to Data Mining, Addison-Wesely, 2005)

데이터의 종류: 형태에 따른 분류

- 그래프 기반 데이터



(a) Linked Web pages.



(b) Benzene molecule.

(그림 출처: Pang-Ning Tan et al, Introduction to Data Mining, Addison-Wesely, 2005)

데이터의 종류: 형태에 따른 분류

- **순서형 데이터**
  - **Sequential Data (순차열 데이터)**
    - Transaction Data + 시간
  - **Sequence Data**
    - 데이터 개체 간 순서가 존재함
  - **Time Series Data (시계열 데이터)**
    - 시간에 따라 속성이 변화하는 데이터 집합
    - Sequential Data의 특수한 형태
  - **Spatial Data**
    - 데이터 개체가 공간 상의 위치정보와 연관되는 데이터 집합

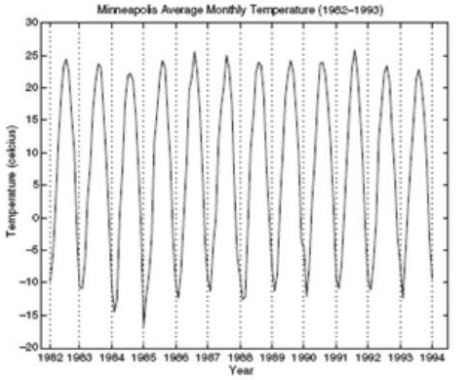
Time	Customer	Items Purchased
t1	C1	A, B
t2	C3	A, C
t2	C1	C, D
t3	C2	A, D
t4	C2	E
t5	C1	A, E

Customer	Time and Items Purchased
C1	(t1: A,B) (t2:C,D) (t5:A,E)
C2	(t3: A, D) (t4: E)
C3	(t2: A, C)

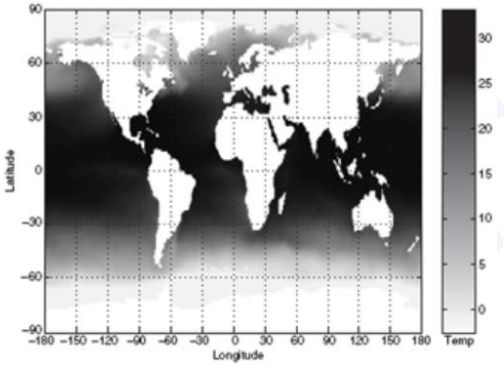
(a) Sequential transaction data.

```
GGTTCCGCCTTCAGCCCCGCGCC
CGCAGGGCCCGCCCCGCGCCGTC
GAGAAGGGCCCGCCTGGCGGGCG
GGGGGAGGCGGGGCCGCCCGAGC
CCAACCGAGTCCGACCAGGTGCC
CCCTCTGCTCGGCCTAGACCTGA
GCTCATTAGGCGGCAGCGGACAG
GCCAAGTAGAACACGCGAAGCGC
TGGGCTGCCTGCTGCGACCAGGG
```

(b) Genomic sequence data.



(c) Temperature time series.



(d) Spatial temperature data.

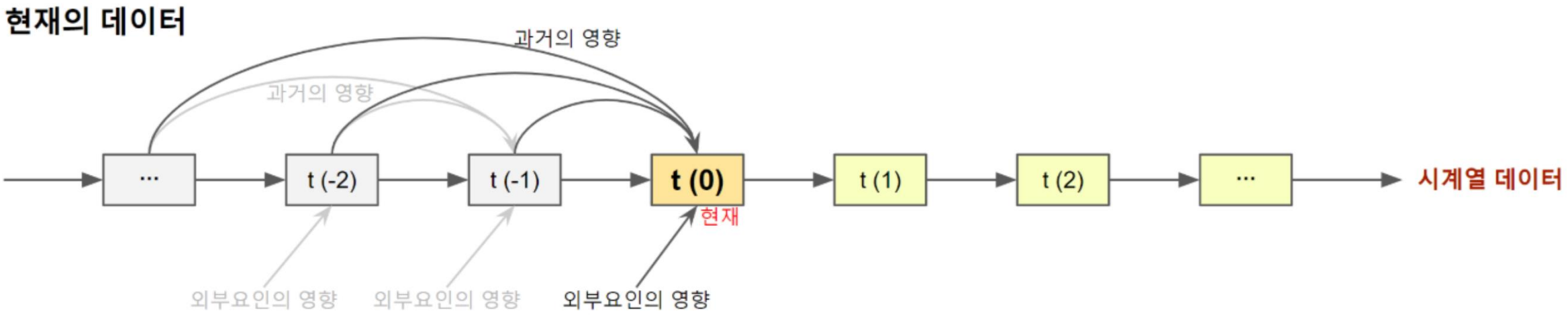
(그림 출처: Pang-Ning Tan et al, Introduction to Data Mining, Addison-Wesely, 2005)

시계열 데이터(Time Series Data)

- 시간의 순서대로 일정한 주기에 따라 측정, 저장된 데이터
- 활용 분야
  - 데이터를 기반으로 한 예측 분야에 가장 많이 활용됨
  - 간혹 장애 검출에도 활용할 수 있다고 이야기하고 있으나 결국 데이터 기반 예측임
    - 입력 데이터를 순서대로 살펴보면 조금씩 어긋나는 데이터 발견 가능
    - 어긋나는 데이터의 범위가 특정 기준을 벗어나는 시점이 장애 발생 시점
    - 데이터의 변동을 기반으로 분석, 예측한 결과를 장애 검출, 장애 예측으로 표현하는 것뿐
  - 시계열 데이터는 **시간에 따른 변화를 보고 패턴을 분석하여 이후에 어떤 데이터를 얻을 것인지 예측하는 것이 가장 큰 목적이자 활용분야**

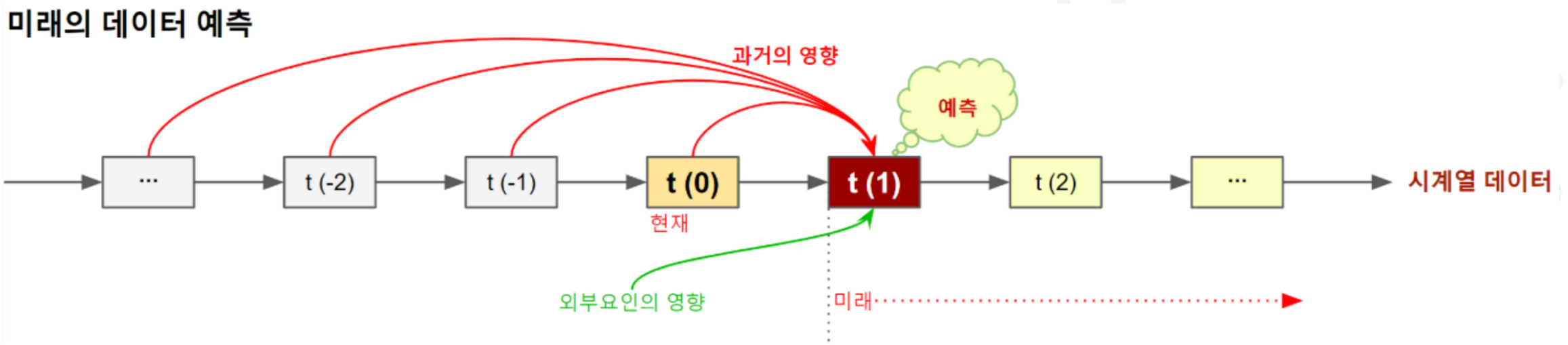
시계열 데이터(Time Series Data)

- 시계열 데이터는 어떤 영향을 주고 받으면서 구성되는가?
  - 시간의 흐름에 따른 데이터는 주변 환경과 특정 이벤트로부터 많은 영향을 받음
  - 따라서 단순히 데이터의 변화 패턴을 분석하는 것만으로는 예측이 어려움
  - 또한 어떤 값과 어떤 변수를 중심으로 분석하는가에 따라서도 큰 성능변화, 결과의 차이 발생



시계열 데이터(Time Series Data)

- 시계열 데이터를 이용한 예측은 어떤 개념인가?
  - 현재(t(0))의 데이터는 과거(t(-m) ~ t(-1))의 데이터가 누적된 결과
  - 가장 가까운 미래(t(1))의 데이터는 과거와 현재(t(-m) ~ t(-1) + t(0))의 데이터가 누적된 결과
  - 미래(t(n))의 데이터는 (t(-m) ~ t(0) ~ t(n-1)) 의 데이터가 누적된 결과
  - 여기에 각 시점에서 적용되는 특정이벤트(외부요인)의 반영으로 예측



시계열 데이터를 활용하기 위한 모델의 제안

- 모델 제안의 1단계
  - 시계열 데이터란 시간의 순서대로 일정한 주기에 따라 측정, 저장된 데이터

    → 과거의 데이터가 현재, 미래의 데이터에 영향을 미침

    → 즉, 과거의 데이터를 가지고 와서 현재, 미래의 데이터에 적용시켜야 함

    → 이를 위해서는 과거의 데이터를 기억하고 있어야 함

    → 과거 데이터를 저장, 참조할 수 있는 메모리 효과를 가진 모델이 필요함

시계열 데이터를 활용하기 위한 모델의 제안

- 모델 제안의 2단계
  - 현재의 데이터를 확인하기 위하여 과거의 데이터를 순차적으로 살펴보고

    → t(1) 미래의 데이터를 확인하기 위하여 과거+현재의 데이터를 순차적으로..

    → t(n) 미래의 데이터를 확인하기 위하여...

    → 알고자 하는 시점의 데이터를 기준으로 과거의 데이터를 순서대로 확인 반복

    → 즉, 동일한 과정, 모델이 계속적으로 반복되는 순환 과정을 가진 모델 필요

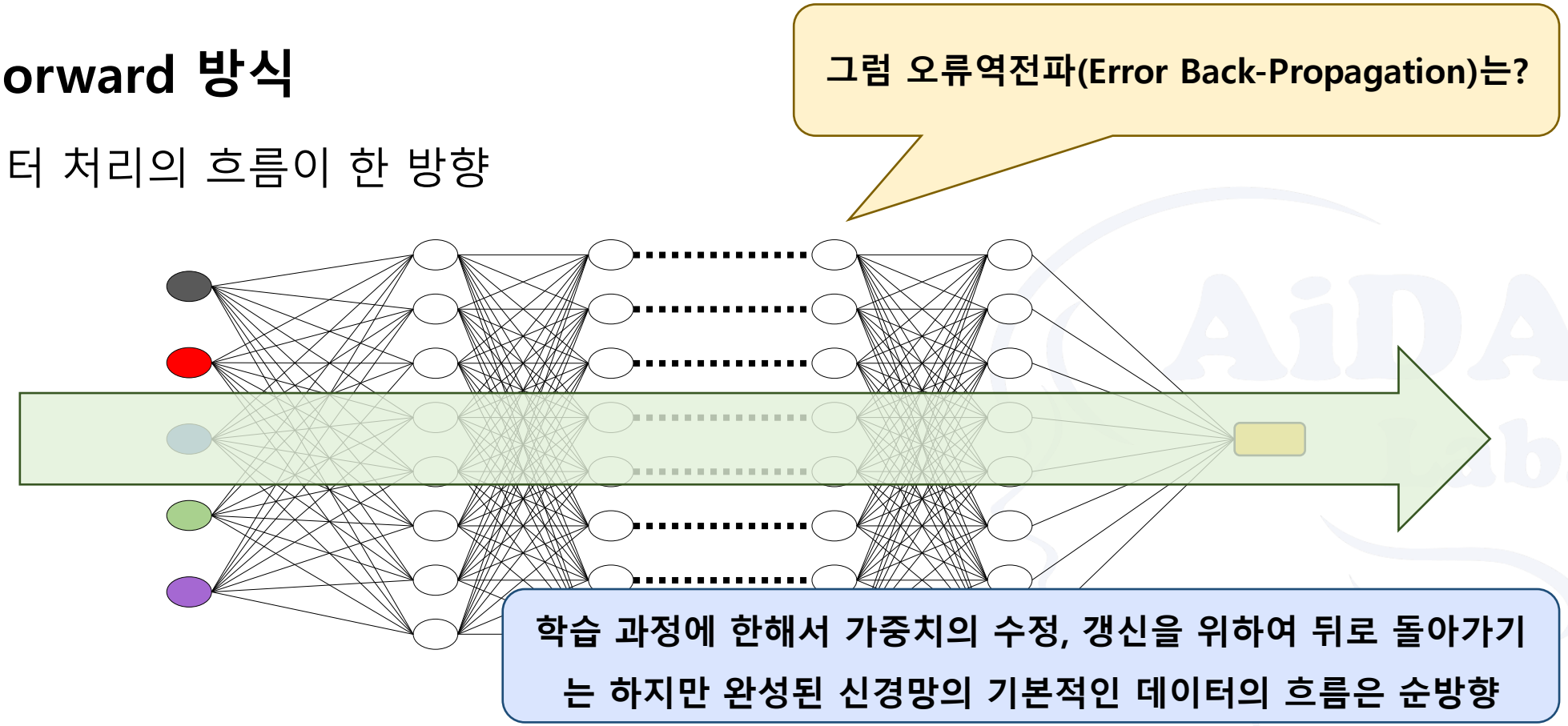
    → 순환신경망 (RNN, Recurrent Neural Network) 제안됨

시계열 데이터와 RNN의 관계

- RNN (Recurrent Neural Network, 순환 신경망)
  - 전체 네트워크 안에서 순환적으로 데이터를 처리하는 신경망 모델
  - 과거의 데이터를 끊임없이 참조하여 현재의 데이터를 학습하는 모델
  - 시간의 흐름에 따라 과거의 데이터의 특징과 패턴을 반영하여 현재의 데이터를 학습하는 모델
  - 순차열, 즉 순서가 있는 일련의 값을 처리하는 것에 특화된 모델

RNN 모델

- 기존의 신경망
  - **Feed Forward 방식**
    - 데이터 처리의 흐름이 한 방향



RNN 모델

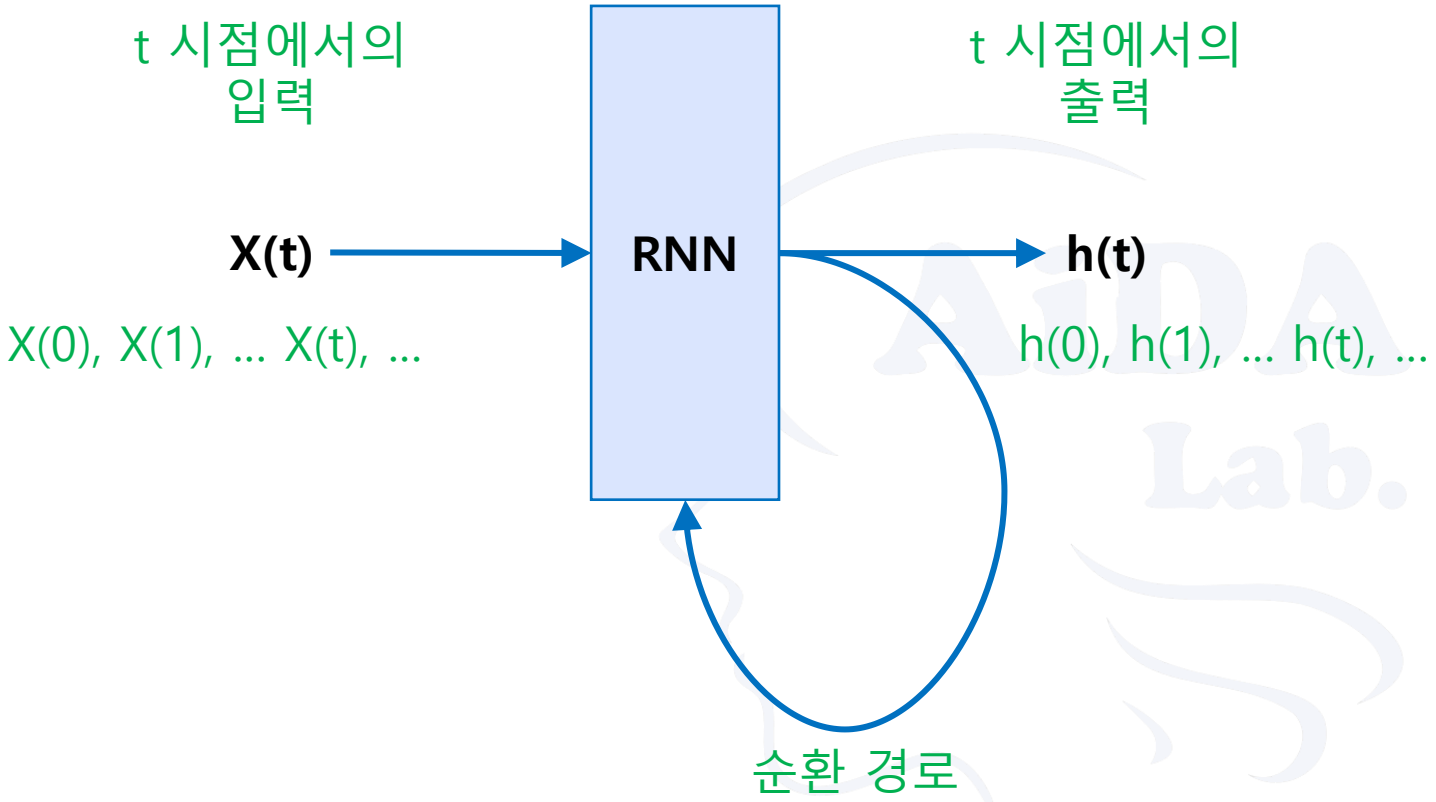
- 기존의 신경망의 단점
  - 한 방향으로만 데이터를 처리하므로 시계열 데이터의 처리가 어려움
  - 시계열 데이터는 신경망의 학습이 완료된 후에도 실제 사용 도중에 지속적으로 과거의 데이터를 참조하고 활용하여야 함
  - 학습 과정에서도 시계열 데이터의 성질, 패턴을 제대로 학습할 수 없음
  - 순환 신경망(Recurrent Neural Network, RNN) 등장의 이유
    (앞에서의 설명과 연관됨)

RNN 모델

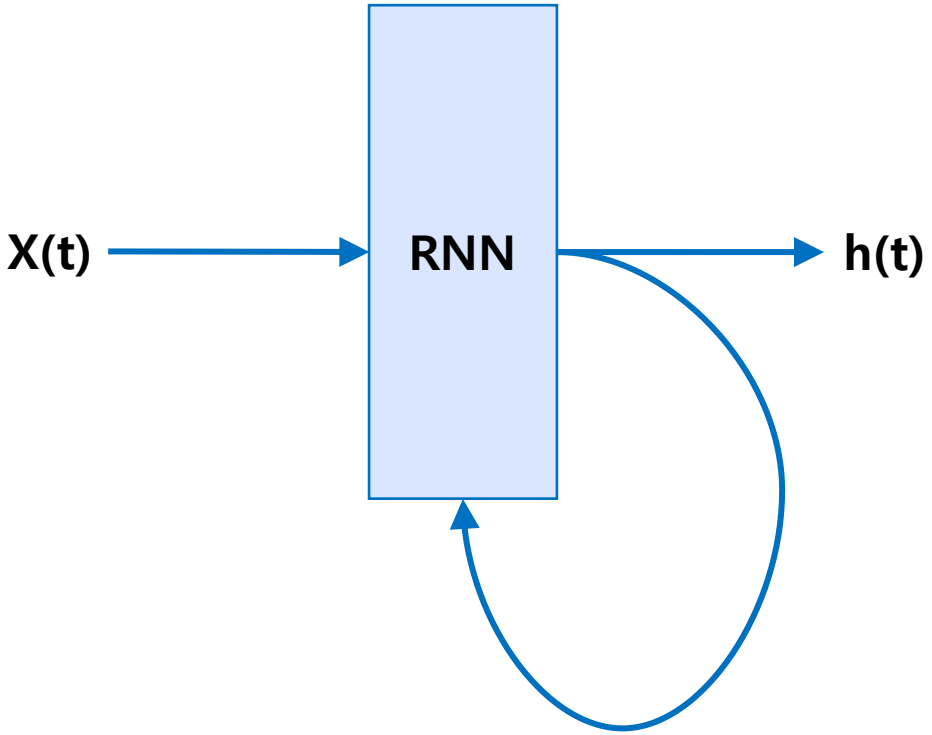
- RNN (Recurrent Neural Network, 순환 신경망)
  - 전체 네트워크 안에서 순환적으로 데이터를 처리하는 신경망 모델
  - 순차열, 즉 순서가 있는 일련의 값을 처리하는 것에 특화된 모델
  - 순환 처리를 하기 위해서는 닫힌 경로(=순환하는 경로)가 필수
  - 경로가 닫혀 있기 때문에 → 데이터가 순환하고, 데이터의 저장이 가능함
  - 순환하기 때문에 → 끊임없이 데이터가 갱신됨

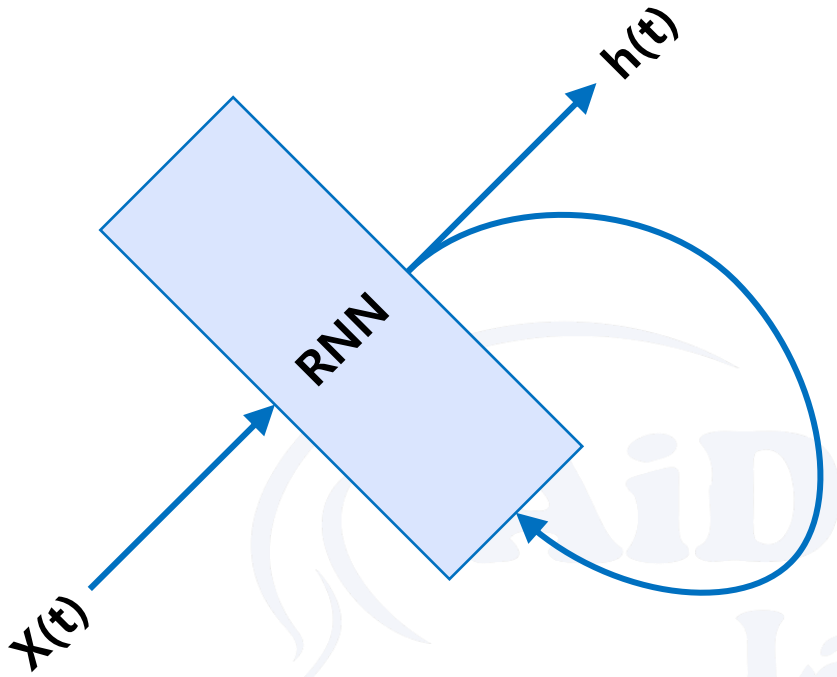
RNN 모델의 기본 원리

- **RNN의 기본 단위: RNN 계층**
  - **순환경로를 포함한다.**
  - **계속적인 참조가 반영된다.**
  - **과거의 정보를 기억한다.**



RNN 모델의 기본 원리



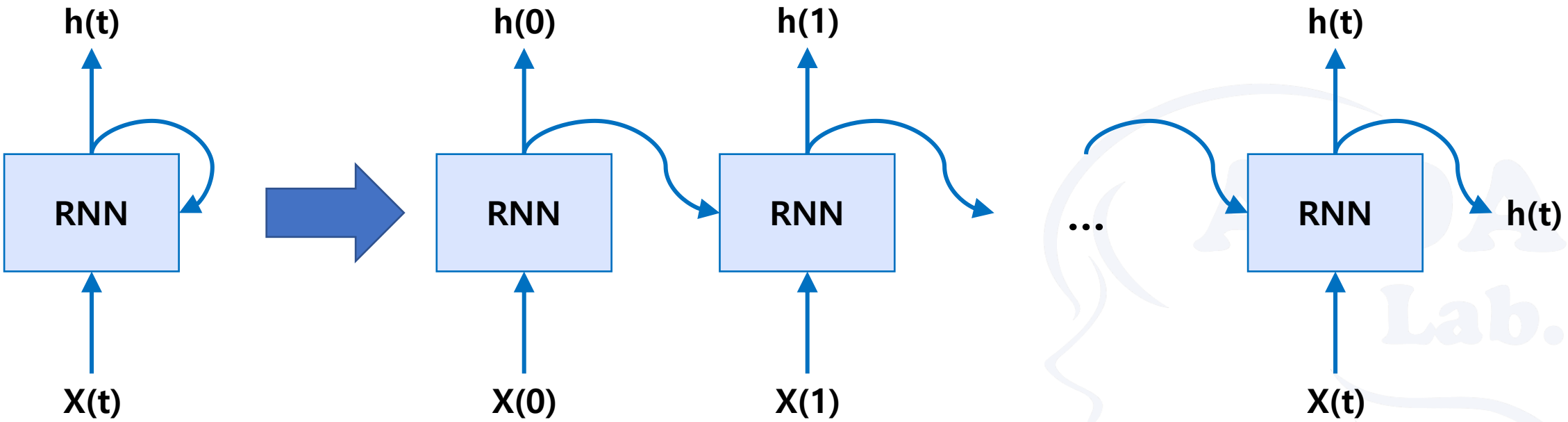


시간 흐름에 따라 계속 이어지므로
세로로 길어짐 !!!

표현하기 편하게 돌려서 씁시다.

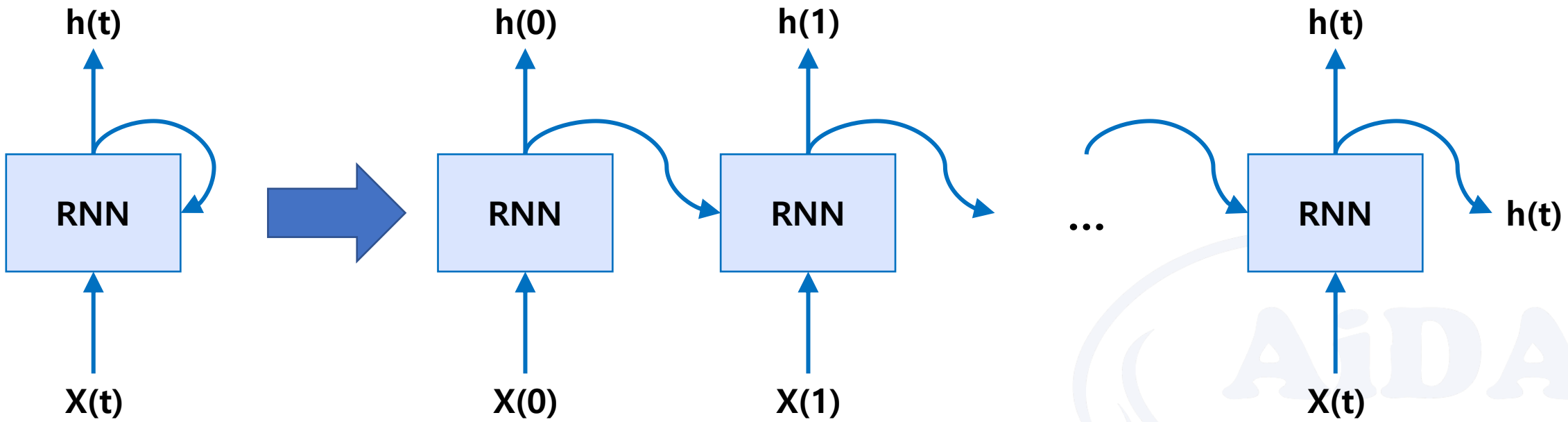
RNN 모델의 기본 원리

- 순환 구조를 알기 쉽게 펼쳐보면



- 데이터는 시간이 흐르는 방향으로 나열됨 (인덱스 t는 시간이 아니라 시각)

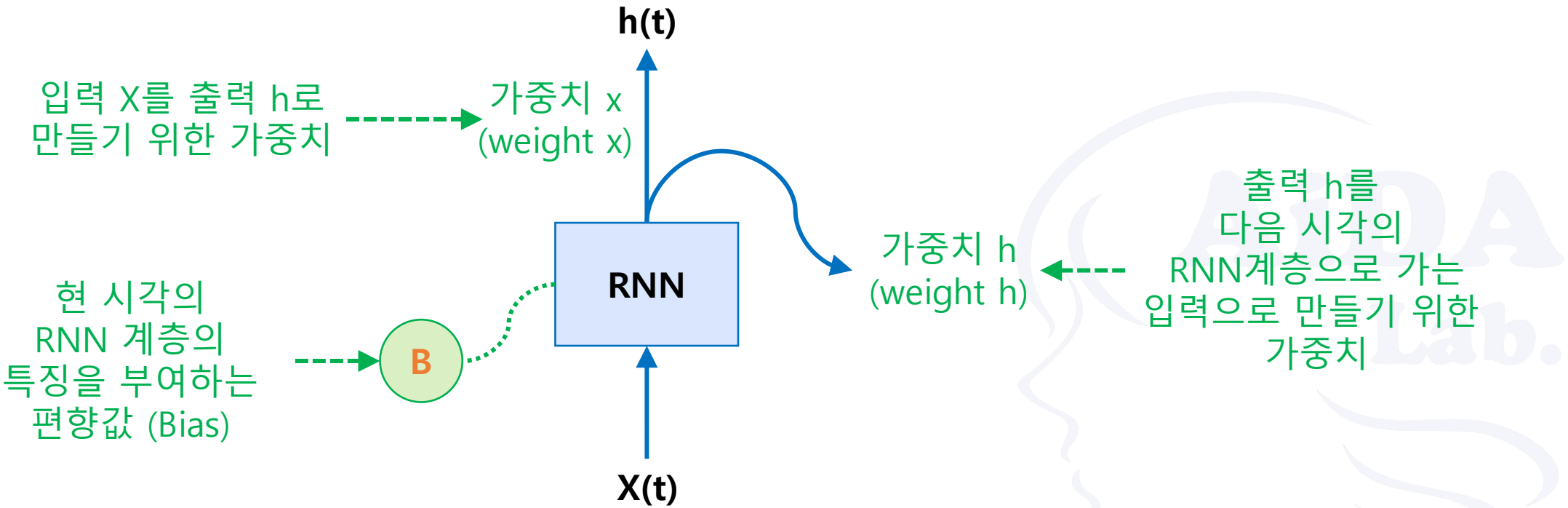
RNN 모델의 기본 원리



- 각 시각의 RNN 계층은 그 계층으로의 입력과 1개 전의 RNN 계층으로부터의 출력을 입력으로 받음
- 두 개의 입력을 기반으로 현 시각의 출력을 계산함
- 각 RNN 계층에서의 출력 계산에는 기존 신경망과 동일하게 경로 별 가중치, 편향 값이 포함됨

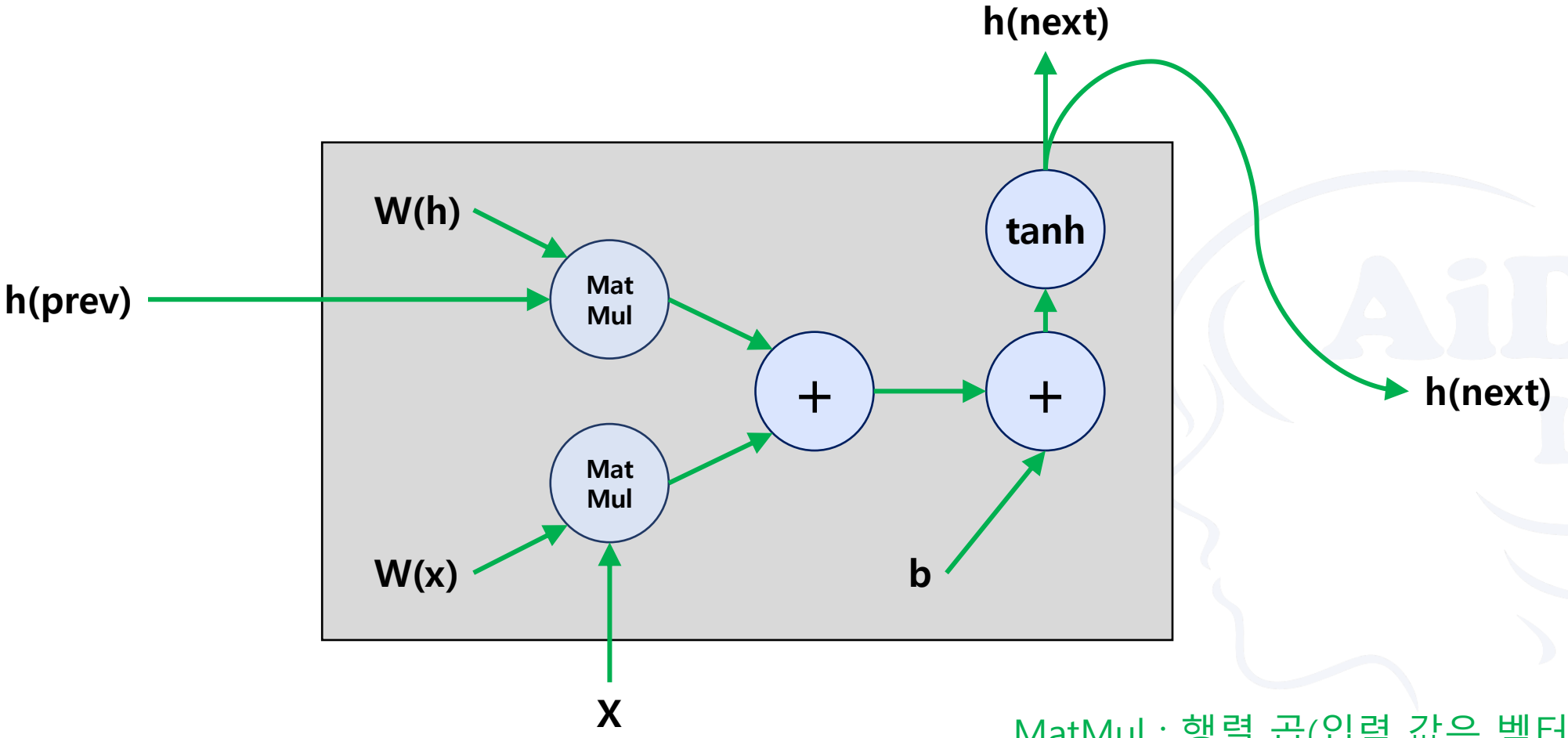
RNN 모델의 기본 원리

- 경로별 가중치와 편향치의 적용



RNN 모델의 기본 원리

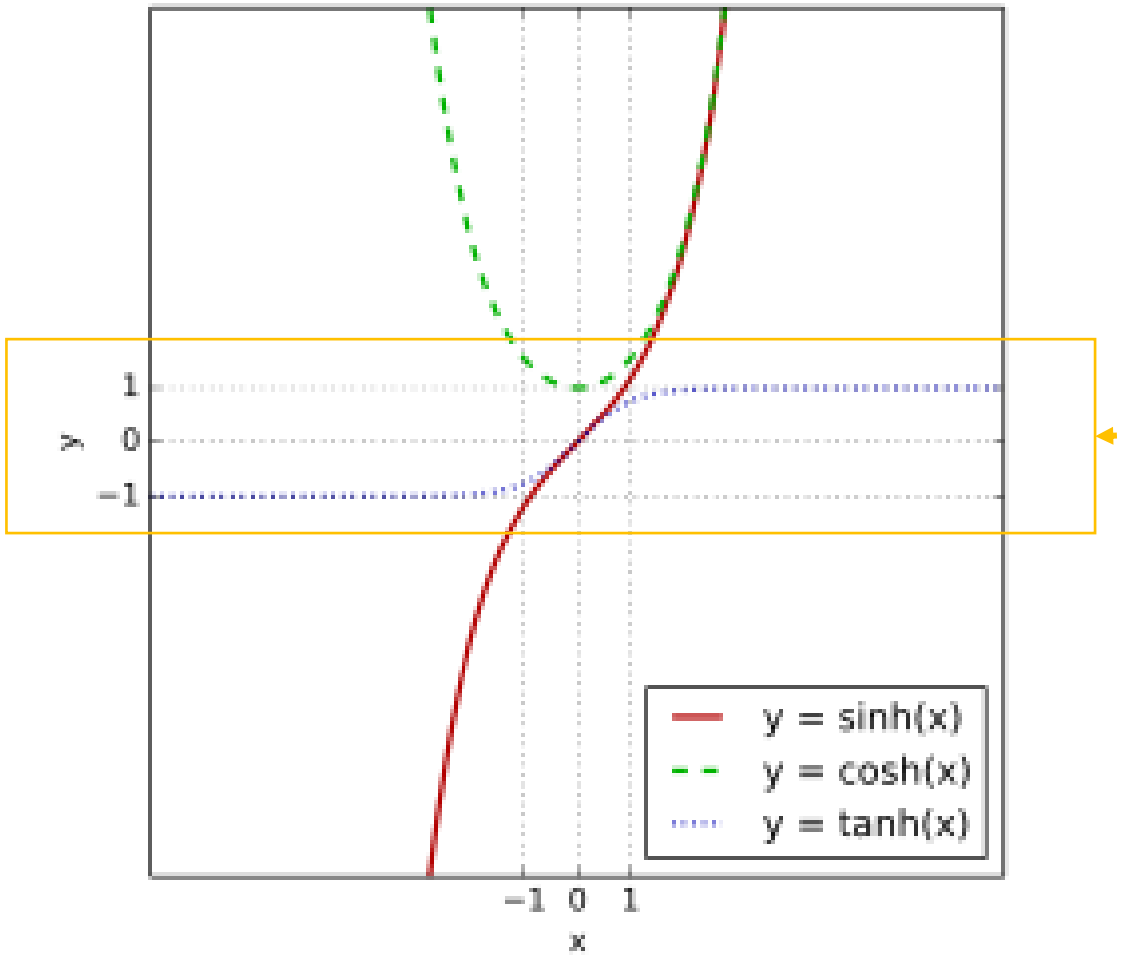
- RNN 계층은 기존 신경망과 동일하다 → 어떻게 처리되나?



MatMul : 행렬 곱(입력 값은 벡터니까)

RNN 모델의 기본 원리

- 참고



tanh : 쌍곡 탄젠트 (Hyperbolic Tangent)

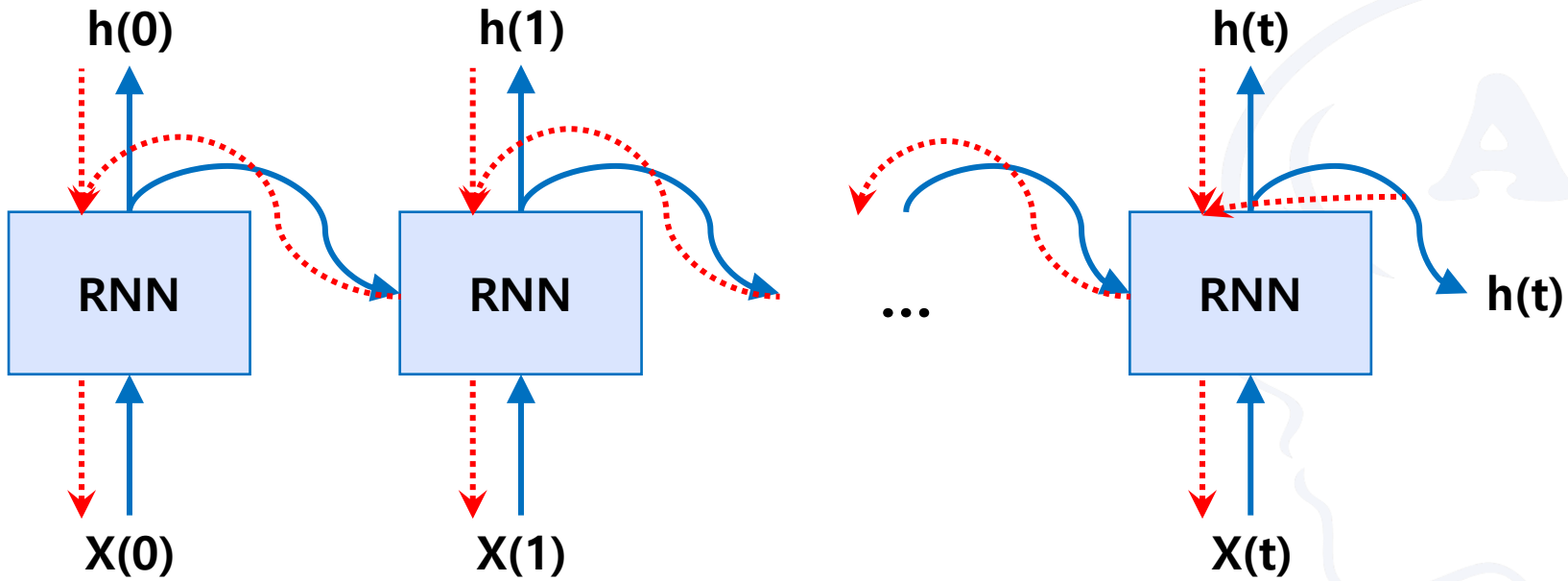
이렇게 생긴 함수다

(활성화 함수의 역할)

RNN 모델의 기본 원리

- RNN 모델의 학습 방법
  - RNN 계층은 가로로 펼쳐 놓은 신경망과 동일하다고 가정할 수 있음

    →학습 방법도 동일하게 적용할 수 있음(오류 역 전파 방식 등) → BPTT 방법



BPTT: Back Propagatopn Through Time

RNN 모델의 기본 원리

- **BPTT (Back Propagation Through Time)의 문제점**
  - **긴 시계열 데이터를 학습할 때**
    - 시계열 데이터의 시간 크기가 커질수록 BPTT가 소비하는 컴퓨팅 자원도 비례하여 증가함
    - 시간의 크기가 커질수록 역 전파 시의 비율 조정을 위한 기울기가 불안정해짐

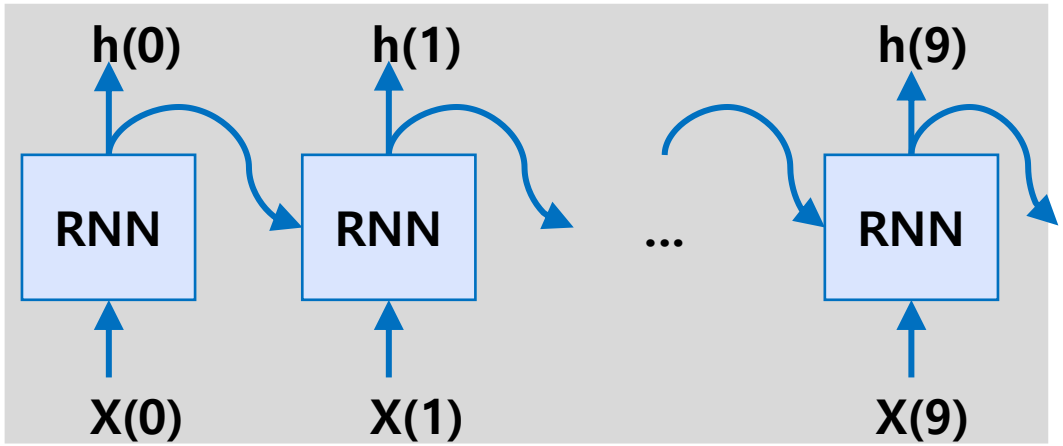
BPTT를 이용하여 기울기를 구할 때, 매 시각의 RNN 계층의 중간 데이터를 메모리에 유지해 두어야 함
→ 시계열 데이터가 길어질 수록 계산 량 및 메모리의 사용량이 증가함

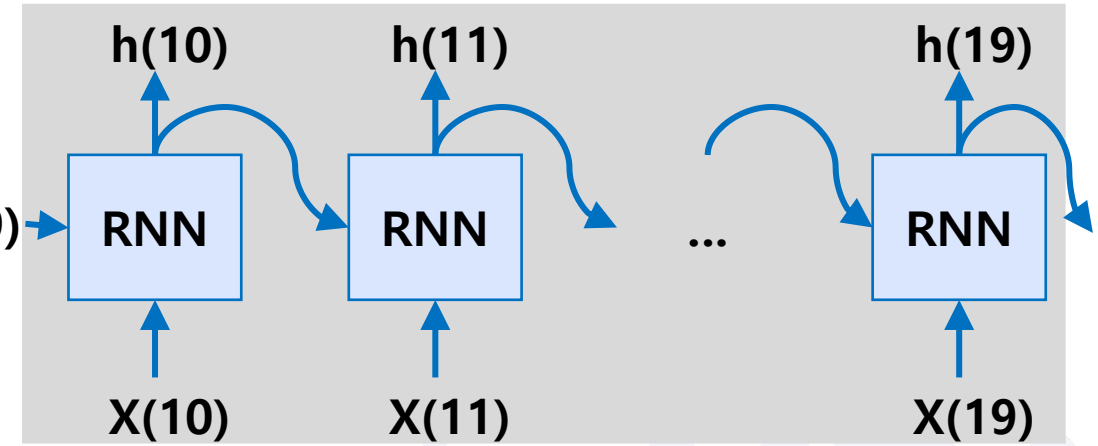
- **개선을 위하여 Truncated BPTT 기법 제안**

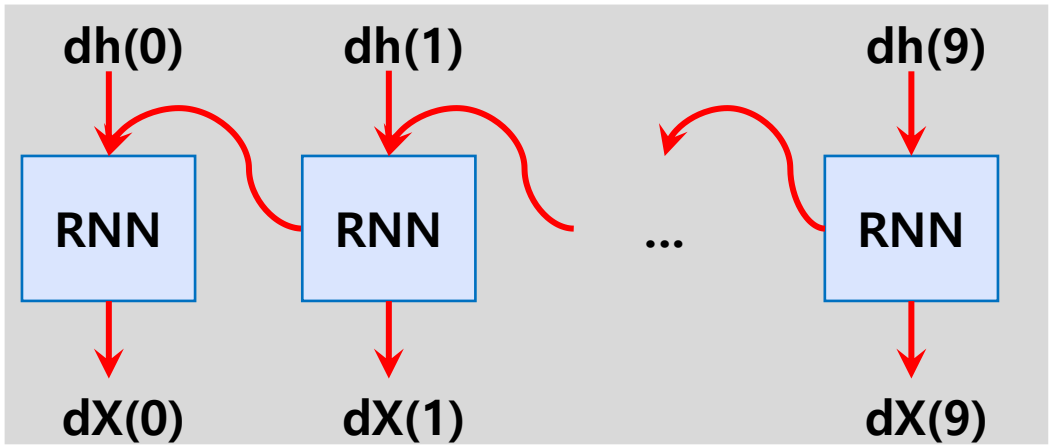
RNN 모델의 기본 원리

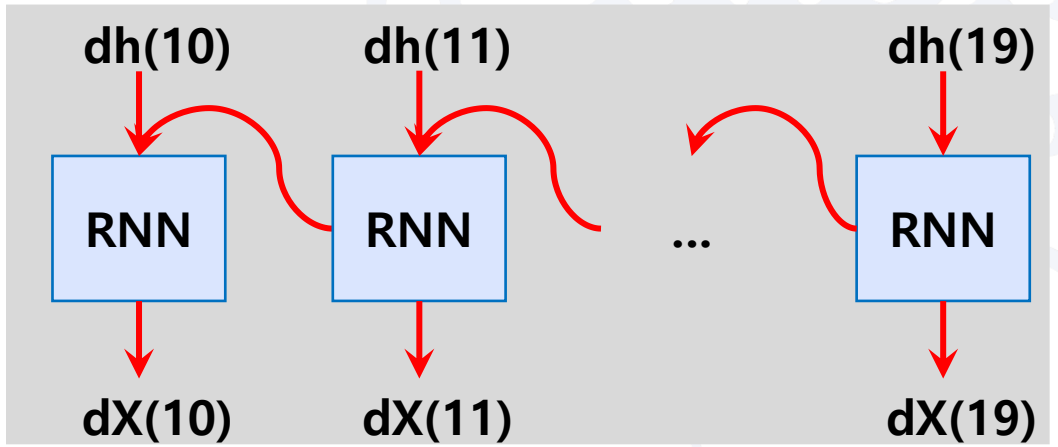
- **Truncated BPTT**
  - **시간축의 방향으로 길어진 신경망을 적당한 지점에서 잘라내어 여러 개의 작은 신경망으로 만듦**
  - **잘라낸 작은 신경망에서 BPTT를 수행함**
  - **주의점**
    - 신경망을 잘라낼 때 역전파의 연결만 절단해야 함 (순전파의 연결은 반드시 유지)
    - 순전파의 연결이 사라지면 네트워크 자체가 성립되지 않음
  - **역전파가 연결된 RNN 계층의 모임을 블록이라고 하여 다른 블록과 구분하여 처리함**

RNN 모델의 기본 원리









첫 번째 블록의 순 전파와 역 전파

두 번째 블록의 순 전파와 역 전파

RNN의 문제점

실습

RNN의 문제점

- **시계열 데이터의 학습에서 장기 의존 관계의 학습이 어렵다**
  - **BPTT에서 기울기 소실 또는 기울기 폭발 발생**
    - 기울기 소실: 역 전파 처리 시 기울기 값이 점점 작아지다가 사라지는 현상
    - 기울기 폭발: 역 전파 처리시 기울기 값이 점점 커지다가 수직에 가까워지는 현상

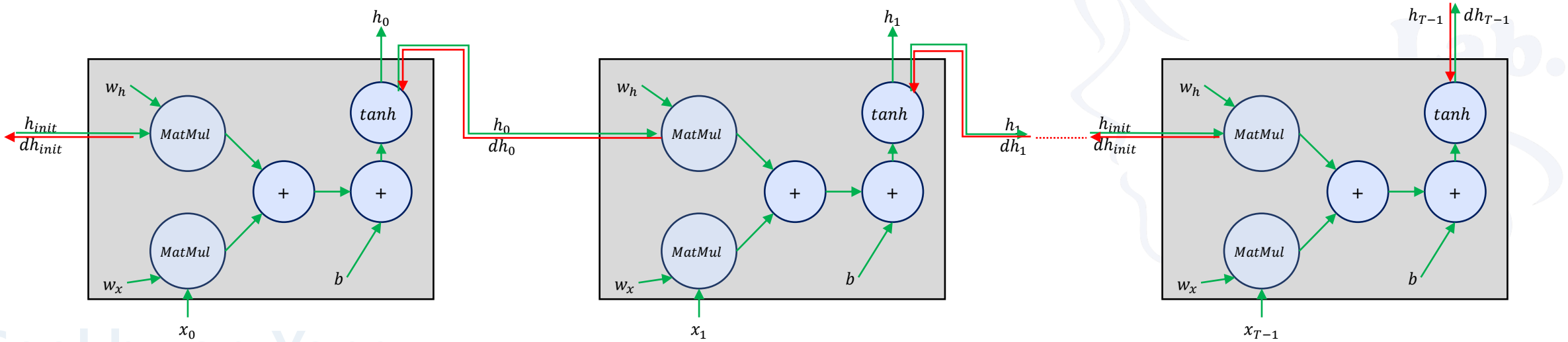
→ 게이트 구조를 적용하여 개선 가능

→ 또는 RNN 모델에서 활성화 함수로 tanh 대신 ReLU를 사용하면 기울기 소실 현상을 줄일 수 있음

RNN의 문제점

- **기울기 소실, 폭발의 원인**
  - RNN 모델 학습 시 RNN 계층은 과거의 정보를 기억하고 있어야 하며, 정답 레이블이 주어지면 과거 방향으로 **의미 있는 기울기**를 전달하여 시간 방향의 의존 관계를 학습함
  - RNN은 활성화 함수로 tanh를 주로 사용함
  - 역전파 시, 시간 방향의 기울기를 살펴 보면 tanh, +, 행렬곱의 순서대로 연산을 수행함

    (+는 큰 영향이 없음)



RNN의 문제점

- tanh의 미분 값은 1.0 이하이고 x가 0에서 멀어질수록 작아짐

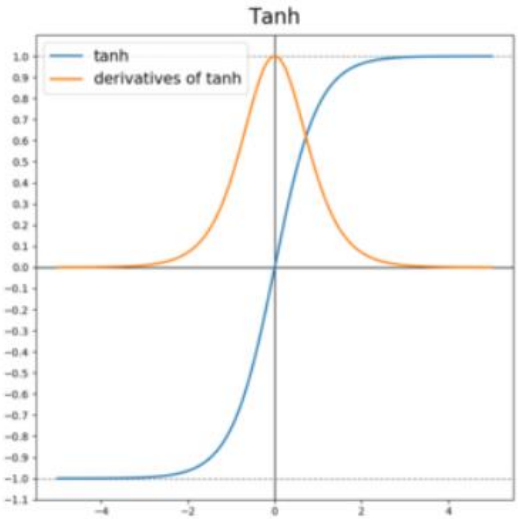
  → 역전파 시, 기울기가 tanh를 지날 때마다 계속 작아짐

  → 행렬곱의 결과가 큰 영향을 미침

- 행렬곱에는 계속 동일한 가중치가 적용되고, L2거리를 계산하여 미분 시 적용

  → 기울기의 크기가 시간에 비례하여 지수증가 / 감소

  → 폭발 또는 소실



tanh 그래프(파란색), tanh의 미분 그래프(주황색)

RNN의 문제점

- **기울기 소실, 폭발 대책**
  - 가중치 초기화
  - 수렴하지 않는 활성화 함수 사용
  - 배치 정규화
  - 기울기 클리핑

RNN의 문제점

- 가중치 초기화
  - 기울기 소실, 폭발을 막으려면 각 층의 출력에 대한 분산이 입력에 대한 분산과 같아야 함

    → 입력과 출력의 연결 개수가 같아야 함
  - 가중치의 초기화를 통해 기울기의 폭주를 막을 수 있음
    - 세이비어 초기화(Xavier Initialization)(또는 글로럿 초기화(Glorot Initialization))
      - 세이비어 추기값은 활성화 함수가 선형이라고 가정함
      - 여러 층의 기울기 분산의 균형을 유지해 줌
      - sigmoid, tanh 활성화 함수에 좋은 성능 보임, ReLU와 사용될 때는 성능이 좋지 않음
    - He 초기화
      - ReLU 활성화 함수에 대한 초기화 전략
      - ReLU는 입력이 음수일때 출력이 모두 0 → 이를 고르게 분포시켜서 폭주 예방
    - 르쿤(LeCun) 초기화
      - SeLU를 사용하는 겨우 초기화 전략.
      - 입력 개수=출력 개수 이면 세이비어 초기화와 동일

RNN의 문제점

- 수렴하지 않는 활성화 함수 (ReLU의 변형들) 사용
  - LeakyReLU
    - $LeakyReLU_{\alpha}(x) = \max(ax, x)$, α: 새는 정도, $z < 0$일 때 함수의 기울기는 보통 0.01
  - RReLU (Randomized Leaky ReLU)
    - α를 무작위로 선택, 테스트 시에는 평균을 사용
  - PReLU (Parametric Leaky ReLU)
    - 훈련하는 동안 α가 학습되는 활성화 함수.
    - 대규모 데이터 셋에서는 성능이 좋으나 소규모 데이터 셋에서는 과적합 가능

RNN의 문제점

- ELU (Exponential Linear Unit)
  - $ELU(z) = \begin{cases} \alpha(\exp(x) - 1), if\ x < 0 \\ x \qquad\qquad\qquad\quad , if\ x \geq 0 \end{cases}$
  - $x < 0$ 이어도 기울기가 0이 아니므로 소실되지 않음
  - 계산 속도가 느림
- SELU (Scaled ELU)
  - 모든 은닉층의 가중치를 르쿤 정규분포 초기화로 초기화 하여야 하며
  - 완전 연결 층만 쌓아서 신경망을 만들고
  - 모든 은닉층이 SELU 활성화를 사용한다면 네트워크가 자기 정규화됨

- **활성화 함수 성능 순위: SELU > ELU > LeakyReLU > ReLU > tanh > Ligistics**

RNN의 문제점

- 배치 정규화
  - ReLU 계열 활성화함수, He 초기화 등을 사용하면 훈련 초기의 기울기 폭주 예방 가능
  - 그러나 훈련 동안의 폭주에 대한 보장은 없음
  - 배치 정규화: 학습 과정에서 각 배치 단위마다 데이터가 다른 분포를 보이더라도 배치마다 평균과 분산을 이용하여 정규화 함
  - 딥러닝 모델 학습 동안 각 층별로 입력 데이터 분포가 변하는 "내부 공변량 변화"가 발생함
    → 기울기의 불안전성은 입력 데이터 분포의 변화로 발생함
    → 배치 정규화로 인하여 모델의 안정적인 학습이 가능해 짐
  - 학습 속도 개선, 초기화 값에 의존도 감소, 과적합 방지
  - 은닉층 → 배치 정규화 → 활성화함수 순서로 적용됨

RNN의 문제점

- 기울기 클리핑
  - 기울기 폭주를 방지하기 위해 임계값을 넘지 않도록 기울기 값을 잘라줌
  - 기울기의 모든 값을 -1.0 ~1.0 사이로 클리핑 함
  - Tensorflow 에서 사용

```
optimizer = keras.optimazers.SGD(clipvalue=1.0)
model.compile(loss="mse", optimizer=optimizer)
```

  - PyTorch 에서 사용

```
import torch.optim as optim
import torch.nn.utils as torch_utils

learning_rate = 1.
max_grad_norm = 5.

optimizer = optim.SGD(model.parameters(), lr=learning_rate)
torch_utils.clip_grad_norm_(model.parameters(), max_grad_norm)
optimizer.step()
```

RNN의 문제점

실습

RNN 모델의 개선 및 응용

- **기본적인 RNN 모델을 개선한 다양한 RNN 모델이 존재함**
  - **게이트가 추가된 RNN 모델:**
    - 게이트라는 구조를 활용하여 시계열 데이터의 장기 의존 관계를 학습할 수 있는 모델
    - 게이트를 통하여 입력 및 출력의 상태(범위)를 제어함
    - 데이터 기억을 위한 메모리 셀을 추가한 형태이며 데이터의 기억 여부, 상태도 제어함
    - LSTM (Long-Short Term Memory networks)
      - RNN의 장기 의존성 문제를 해결할 뿐만 아니라 학습 또한 빠르게 수렴함
    - GRU (Gated Recurrent Unit)
      - LSTM의 간소화된 버전이라고 볼 수 있음

RNN 모델의 적용에 대하여

- 활용분야 간의 연관성
  - 우리가 글을 읽거나 대화를 할 때, 지금 사용한 하나의 단어만이 의미가 있는가?
  - 우리는 글을 읽거나 대화를 할 때, 이전의 맥락을 이해하면서 현재 사용된 단어를 이해함
  - 같은 단어인데 다른 의미를 가지는 경우도 이런 맥락을 이용하여 문제없이 정확하게 이해함
  - 수많은 단어로 이루어진 언어의 표현은 시간의 흐름에 따라 각 단어들이 배치, 연결됨
  - 따라서 시간과 순서(순차) 정보를 잘 활용할 수 있는 RNN이 많이 사용됨

RNN 모델의 적용에 대하여

- **RNN 모델의 주요 사용 분야**
  - 기상데이터 분석 및 예측, 주가 정보 예측 (시계열 데이터)
  - 자연어 처리, 번역, 언어 모델링 등 (순차 열 데이터)

- **RNN 모델의 사용 분야에 CNN 등의 모델을 적용한다면?**
  - 시간의 흐름, 순서에 따른 맥락과 각 요소(단어) 사이의 인과관계를 처리 가능한가?
  - 가능하기는 하겠지만 매우 많은 노력과 자원이 소요됨

RNN(Recurrent NN)과 RNN(Recursive NN)?

- Recurrent Neural Network(순환 신경망)과 Recursive Neural Network(재귀 신경망)을 같은 것이라고 설명하는 경우가 많음

- 순환 신경망과 재귀 신경망은 서로 다른 모델임
  - 영문 번역 시 Recurrent와 Recursive를 동일하게 순환의 의미로 번역하여 발생한 문제
  - 일본 도서의 번역서에서 자주 발견됨(국내 도서에도 존재함)

RNN(Recurrent NN)과 RNN(Recursive NN)?

- 차이점
  - 재귀(Recursive) 신경망은 순환(Recurrent) 신경망의 또 다른 일반화 버전이라고 보면 됨
  - 순환(Recurrent) 신경망은 체인 형태의 계산 그래프를 사용
  - 재귀(Recursive) 신경망은 트리 형태의 계산 그래프를 사용

- 일반적으로 말하는 RNN은 순환 신경망을 의미함

LSTM

- **LSTM(Long-Short Term Memory)**
  - **기존의 RNN 모델**
    - 가변 길이의 시퀀셜 데이터 형태 입력에 잘 동작함
    - 데이터가 길어지면 앞에서 입력된 데이터를 잊어버림
  - **LSTM:** 기존의 RNN 모델에
    - 별도의 셀 상태(Cell State)변수 추가 → 기억 능력 증가
    - 다양한 게이트 추가 → 기억, 삭제, 데이터의 출력 등 다양한 상태를 제어
      → 긴 길이의 데이터에 대해서 효율적인 대처 가능

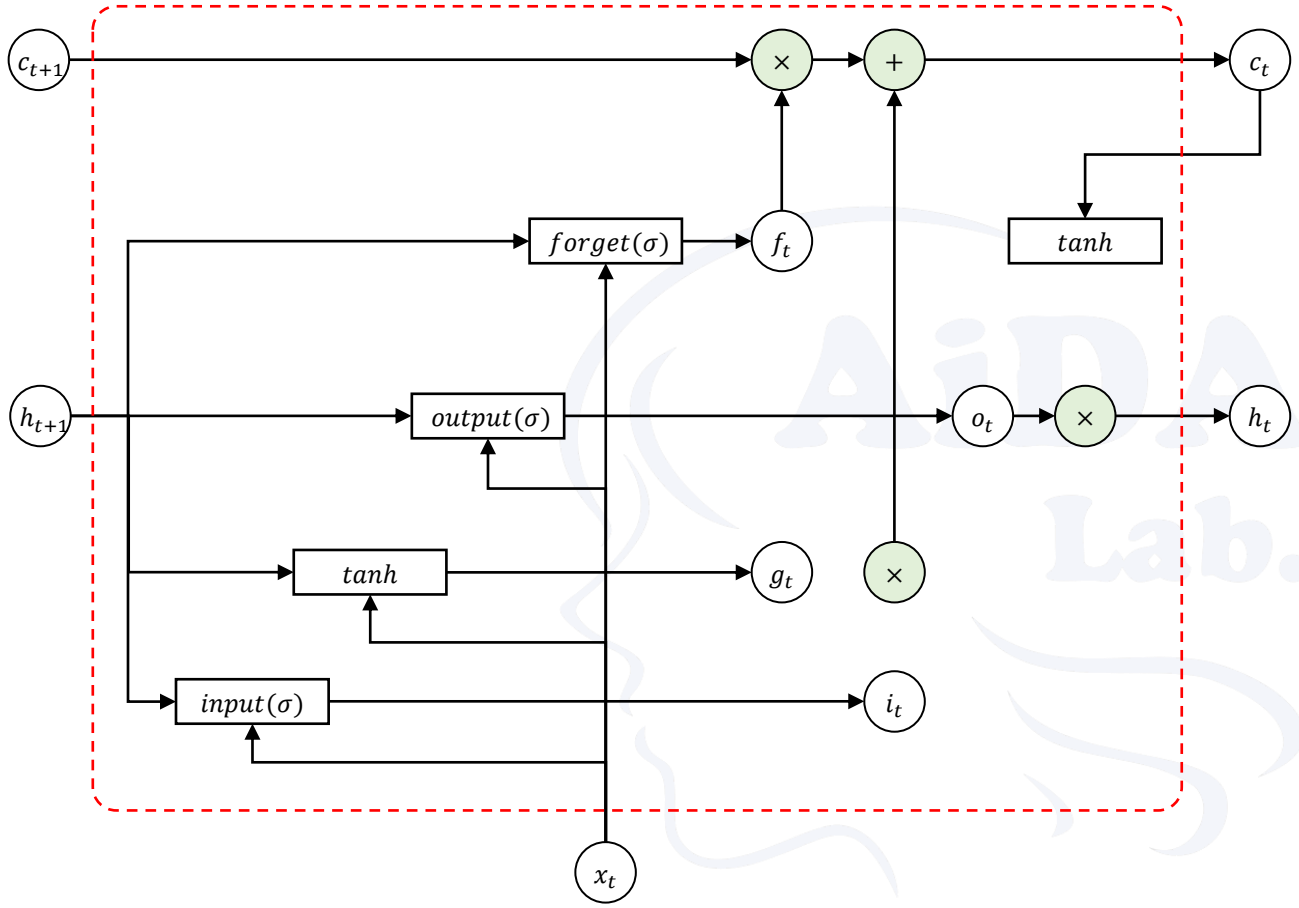
LSTM

- **LSTM 수식**
  - $i_t = \sigma\left(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi}\right)$
  - $f_t = \sigma\left(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf}\right)$
  - $g_t = tanh\left(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg}\right)$
  - $o_t = \sigma\left(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}\right)$
  - $c_t = f_t c_{(t-1)} + i_t g_t$
  - $h_t = o_t \tanh(c_t)$

LSTM

- **LSTM 구조**
  - 각 게이트 앞의 σ (0~1 값)

    → 얼마나 게이트를 열고 닫을 것인가 결정
  - 결정된 σ에 따라 셀 상태 c_{t-1}, g_t, c_t가 새롭게 인코딩 됨

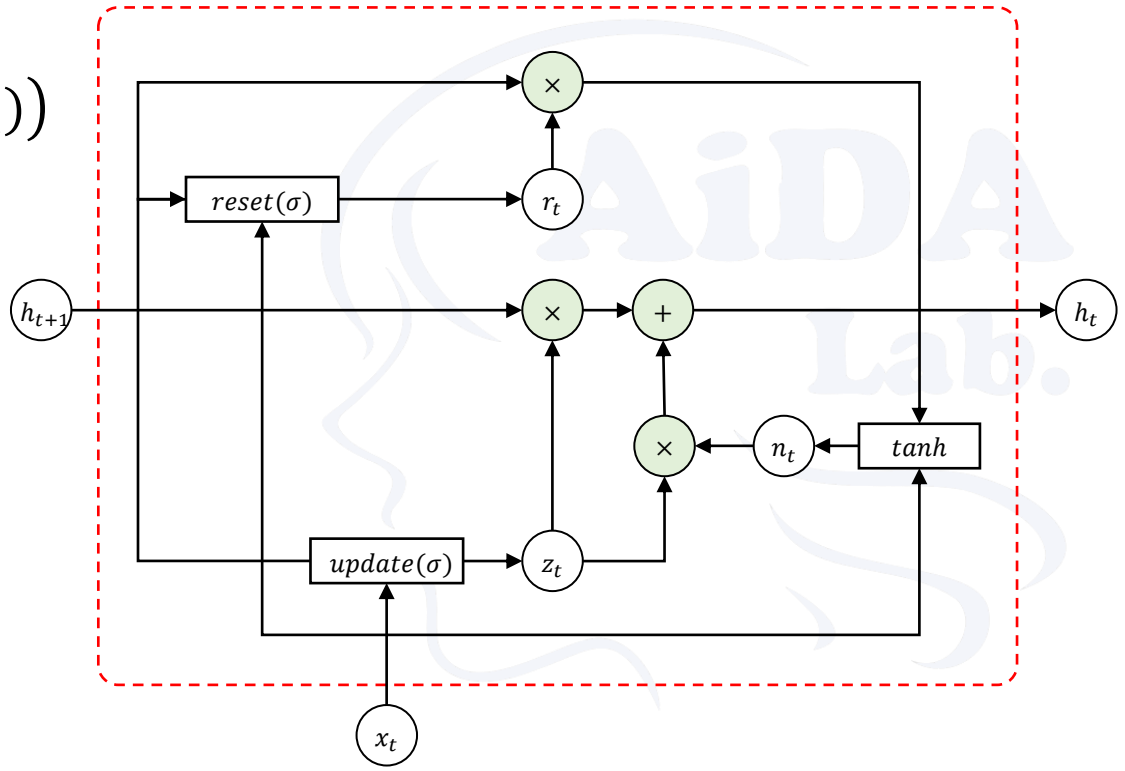


GRU

- **GRU(General Recurrent Unit)**
  - **LSTM의 간소화 버전. 기존의 LSTM보다 간단하지만 성능은 비슷함**
  - **LSTM과 마찬가지로**
    - 시그모이드 σ로 구성된 리셋 게이트 r_t와 업데이트 게이트 z_t 보유
    - 게이트의 출력 값은 시그모이드 함수로 인해 0~1
    - 데이터의 흐름을 게이트를 열고 닫아서 제어 가능
  - **기존 LSTM보다 게이트 숫자가 줄어들고 게이트에 딸린 파라미터도 줄어듦**

GRU

- **GRU 수식**
  - $r_t = \sigma\left(W_{ir}x_t + b_{ir} + W_{hr}h_{(t-1)} + b_{hr}\right)$
  - $z_t = \sigma\left(W_{iz}x_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz}\right)$
  - $n_t = tanh\left(W_{in}x_t + b_{in} + r_t(W_{hn}h_{(t-1)} + b_{hn})\right)$
  - $h_t = (1 - z_t)n_t + z_t h_{(t-1)}$



GRU

- **활용 동향**
  - GRU는 LSTM보다 가볍지만 아직까지는 LSTM을 많이 사용하고 있음
  - LSTM, GRU가 사용하는 학습률, 은닉상태 크기 등의 하이퍼 파라미터가 다름

    → 사용 모델에 따라 파라미터 설정 등을 다시 찾아야 함

    → 귀찮아서 그냥 LSTM을 쓰는 경우가 많음.
  - 성능의 차이 등의 문제가 아니라 사용자의 성향에 따라 좌우되는 편