Teacher Training Presentation

MLDL

인공지능(artifical intelligence)

INTRODUCE

schedule

01 퍼셉트론, 인공신경망

02 RNN,CNN

03 강화학습,생성형 AI

04 지도,비지도학습+앙상블

05 LLM

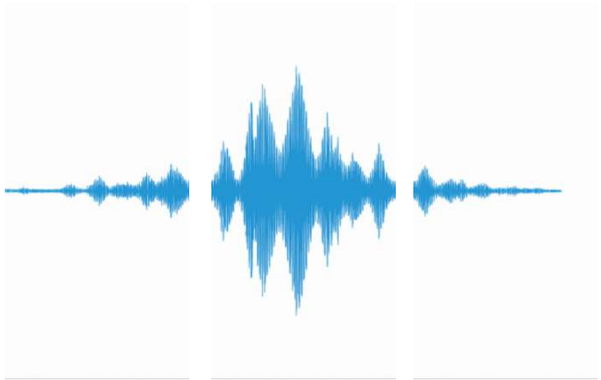
CONTENTS

RNN 01

시퀀스(sequences)

시퀀스: 시간적인 순서를 가진 데이터의 나열

예시		시퀀스(시간축 데이터)
텍스트 + time flow	=	텍스트 시퀀스
음성 + time flow	=	소리 파형 시퀀스
이미지 + time flow	=	이미지 시퀀스 -> 영상



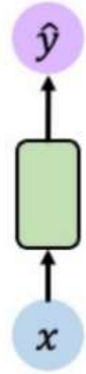
토큰이란?
텍스트를 쪼개는 단위

1. 띄어쓰기 -> ['this', 'morning', 'i', 'took', 'my', 'cat', 'for', 'a', 'walk']

2. 서브워드 -> ["오늘","Ġ아침","-에","ĠLł","-는","Ġ고양이","-를"....] 형태소 단위(Ġ는 띄어쓰기)

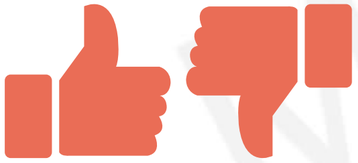
3. 문자 단위(character) -> ['t', 'h', 'i', 's', ' ', 'm', 'o', 'r', 'n', 'i', 'n', 'g', ' '....] 한 글자씩 쪼갬

시퀀스 적용

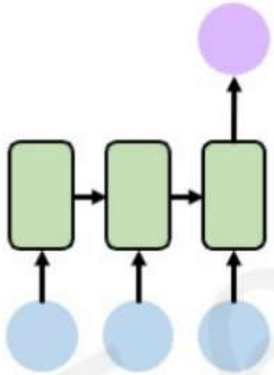


one to one

이진 분류

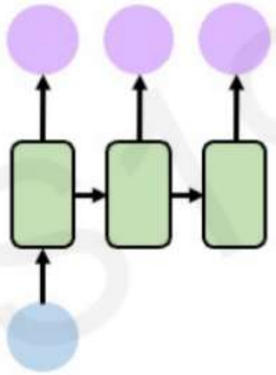


"시험 통과 할 수 있을까?"



many to one

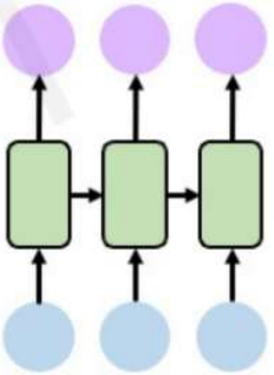
감정 분류



one to many

이미지 자막 캡션

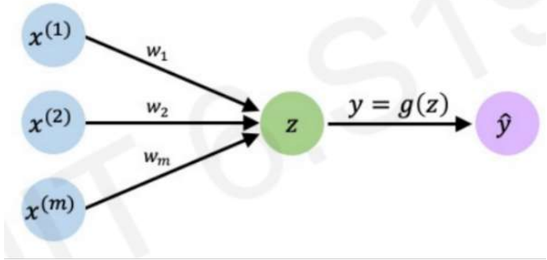
"야구선수가 공을 던진다"

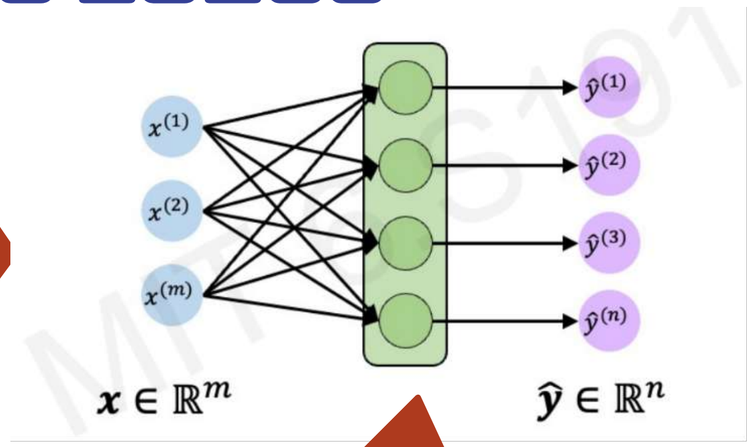


many to many

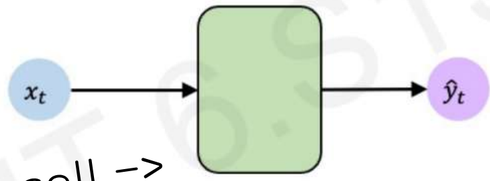
기계 번역

순방향 인공신경망





단순화



cell ->

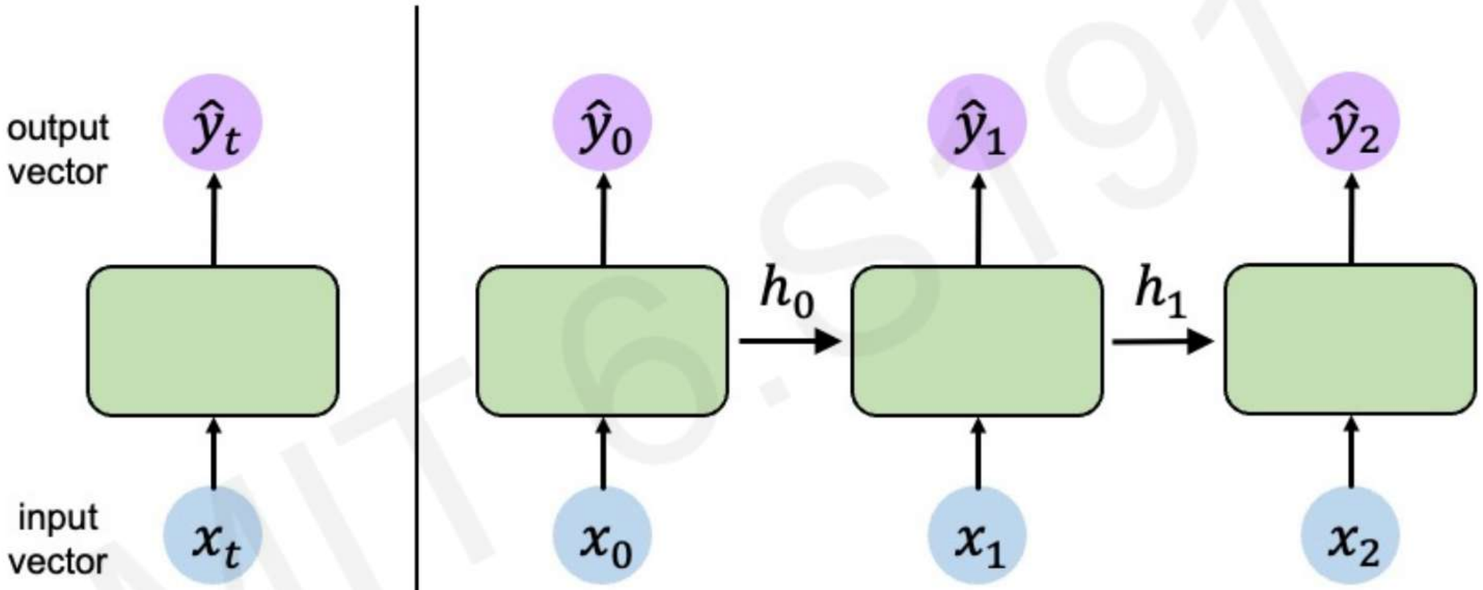
$\boldsymbol{x_t} \in \mathbb{R}^m$

$\hat{\boldsymbol{y}}_t \in \mathbb{R}^n$

퍼셉트론-> 인공신경망 = 순방향 인공신경망 -> 단순화

RNN 01

RNN



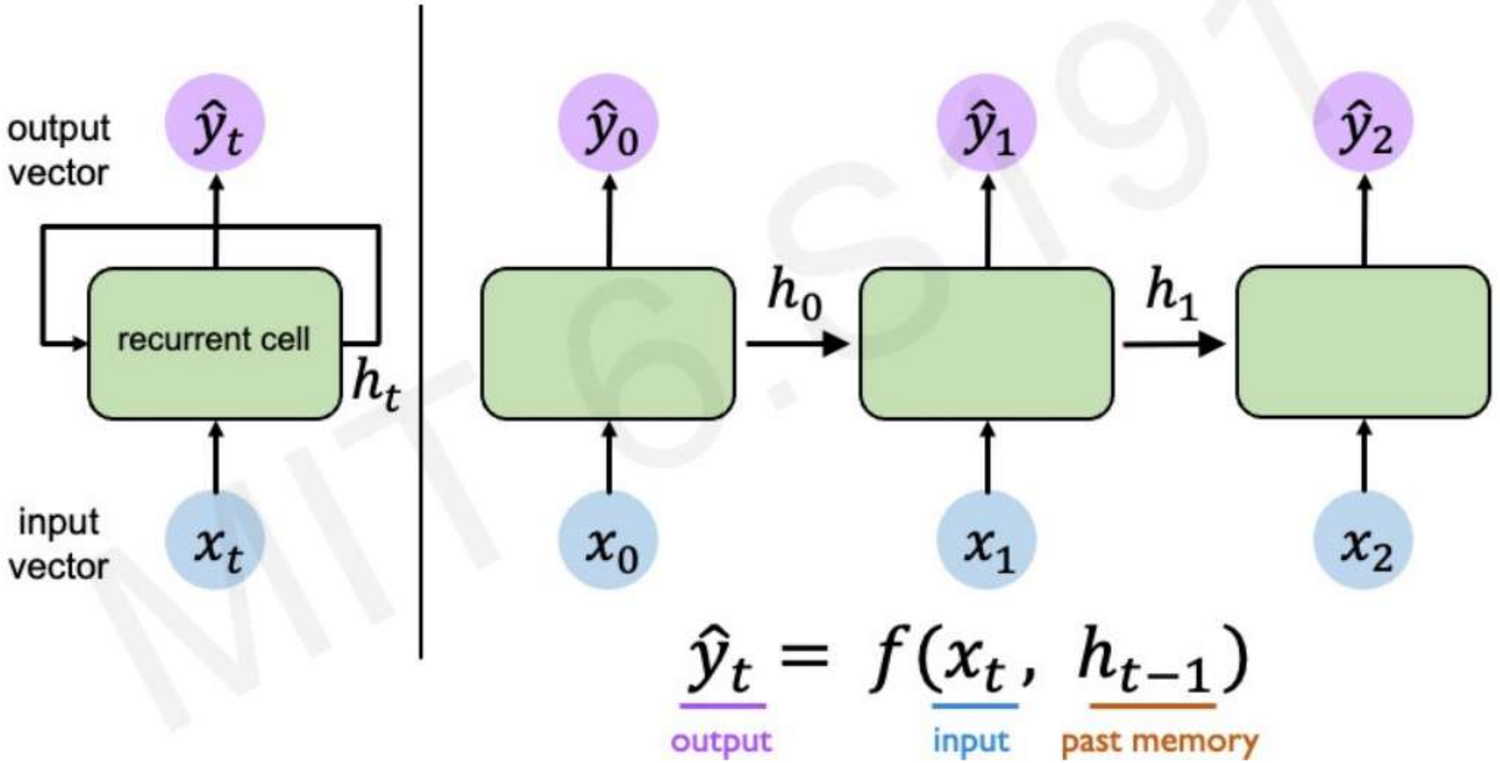
cell의 출력 h = 은닉상태

$$\underbrace{\hat{y}_t}_{\text{output}} = f(\underbrace{x_t}_{\text{input}}, \underbrace{h_{t-1}}_{\text{past memory}})$$

시퀀스 데이터는 시간에 따라 변화함
->RNN은 시간 흐름에 따라 데이터를 하나씩 순서대로 처리함

RNN 01

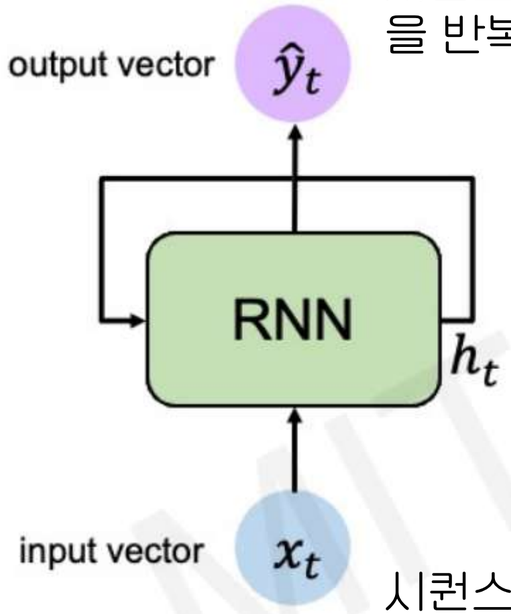
RNN



$$\underbrace{\hat{y}_t}_{\text{output}} = f(\underbrace{x_t}_{\text{input}}, \underbrace{h_{t-1}}_{\text{past memory}})$$

RNN은 과거 입력에 의존함
-> 시퀀스 데이터에서는 과거 정보가 현재의 예측값에 매우 중요한 영향을 주기 때문임

RNN 01



시퀀스를 처리하기 위해 각 시점에 이전 결과를 참고해서 계산을 반복한다.

변수에 의미가 다 있음

$$h_t = f_W(x_t, h_{t-1})$$

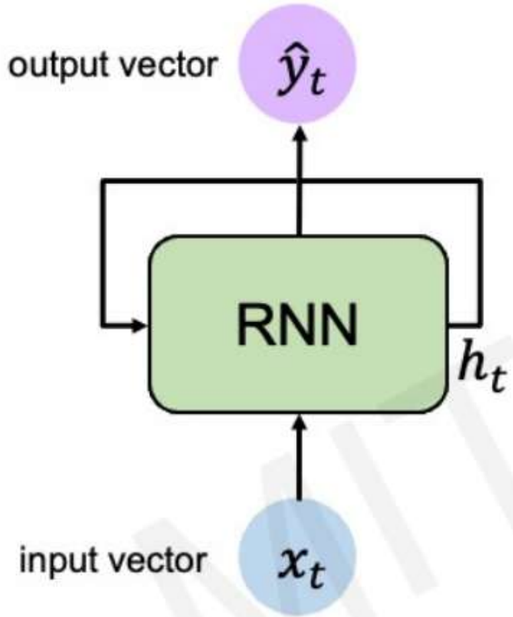
cell state / function with weights W / input / old state

현재 시점의 은닉 상태 =
학습 가능한 가중치를 가진 함수(입력 시퀀스, 과거의 은닉 상태)

시퀀스의 각 시점마다 같은 함수와 같은 매개변수를 사용하여 상태를 반복적으로 업데이트함

RNN 01

RNN



출력 벡터 = 출력에 대한 은닉 가중치 · 현재 은닉 상태

Output Vector

$$\hat{y}_t = \boldsymbol{W}_{\boldsymbol{hy}}^{\boldsymbol{T}} h_t$$

Update Hidden State

$$h_t = \tanh(\boldsymbol{W}_{\boldsymbol{hh}}^{\boldsymbol{T}} h_{t-1} + \boldsymbol{W}_{\boldsymbol{xh}}^{\boldsymbol{T}} x_t)$$

Input Vector

$$x_t$$

생략 되었지만 사실 이게 맞음 ->

$$\hat{y}_t = g\left(W_{hy}^T h_t + b_y\right)$$

$$h_t = \tanh(W_{hh}^T h_{t-1} + W_{xh}^T x_t + b_h)$$

시퀀스 모델링:다음 단어 예측

단어 시퀀스가 주어졌을 때 다음 순서에 올 단어를 예측
-> LLM의 핵심 원리

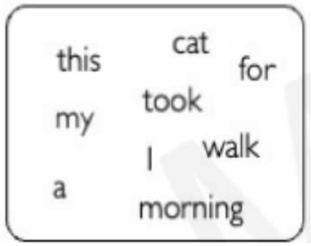
"나는 오늘 아침에 고양이를 산책시켰다."

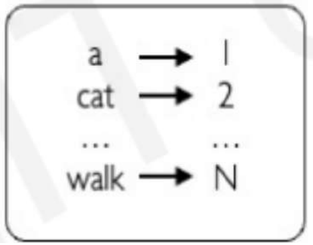
주어진 단어들 뒤의 단어를 예측

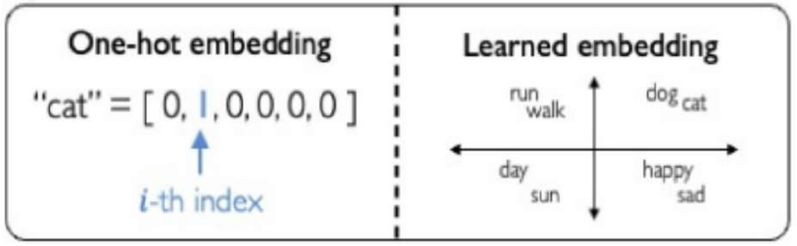
단어 예측이 중요한 이유는 단어의 배열이(순서가) 문장 전체의 뜻을 좌우 할 수도 있기 때문

단어 예측: 인코딩

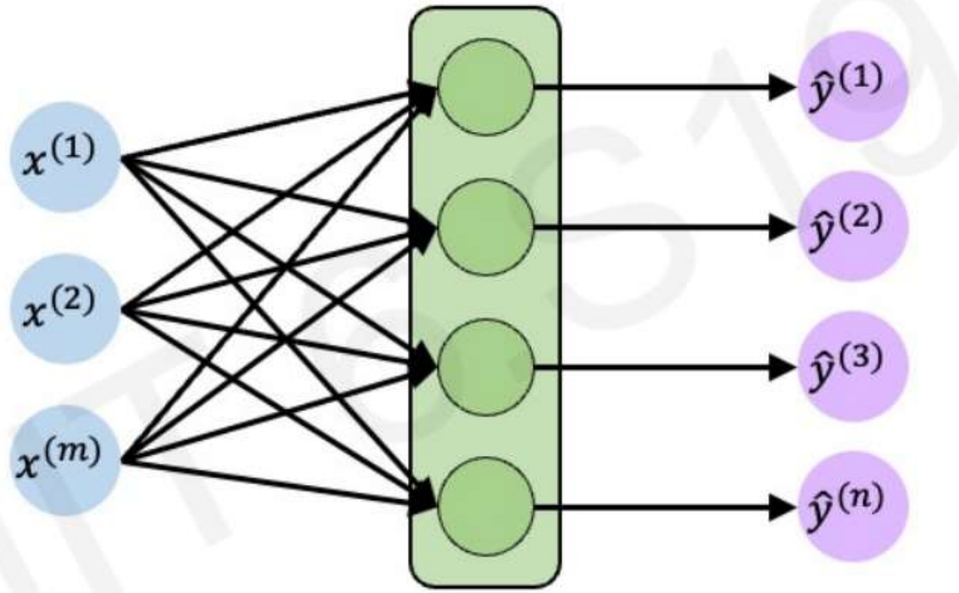
인코딩: 데이터를 모델이 이해 할 수 있게 변환하는 과정 전체
토큰화: 문장을 쪼갬(형태소, 단어등등을 기준으로)
벡터화: 토큰을 모두 숫자로 바꿈
임베딩: 고차원 벡터 공간을 저차원으로 변형하는 것
-> 연속 벡터로 변형시켜 의미를 부여하여 모델이 이해 할 수 있도록 만들어줌
∈
벡터를 구성하는 데이터의 개수 n개가 n차원 벡터
차원: 수학적 추상공간으로 데이터의 특징과 의미를 표현함
ex) 방향, 크기, 거리등
연속 벡터: 실수로 구성 된 벡터
ex) 연속 벡터로 변환: "강아지" → [0.12, -1.44, 0.87, ...]







순전파



인공신경망에서 입력신호를 출력 신호로 변환하는 과정
-> 손실 함수에 들어갈 값을 구함

손실 함수

$$\mathcal{L}\left(f\left(x^{(i)}; \boldsymbol{W}\right), y^{(i)}\right)$$

Predicted Actual

손실함수 L (모델의 예측값(입력값에 대해 가중치 W를 사용한 출력값) – 실제 값)

손실함수는 평가지표의 역할도 하지만 모델의 예측을 할 것인지 분류를 할 것인지 학습 목표를 정해줌

손실 함수의 종류

$$J(W) = -\frac{1}{n}\sum_{i=1}^{n} \underbrace{y^{(i)}}_{\text{Actual}} \log\left(\underbrace{f(x^{(i)}; W)}_{\text{Predicted}}\right) + \underbrace{(1 - y^{(i)})}_{\text{Actual}} \log\left(1 - \underbrace{f(x^{(i)}; W)}_{\text{Predicted}}\right)$$

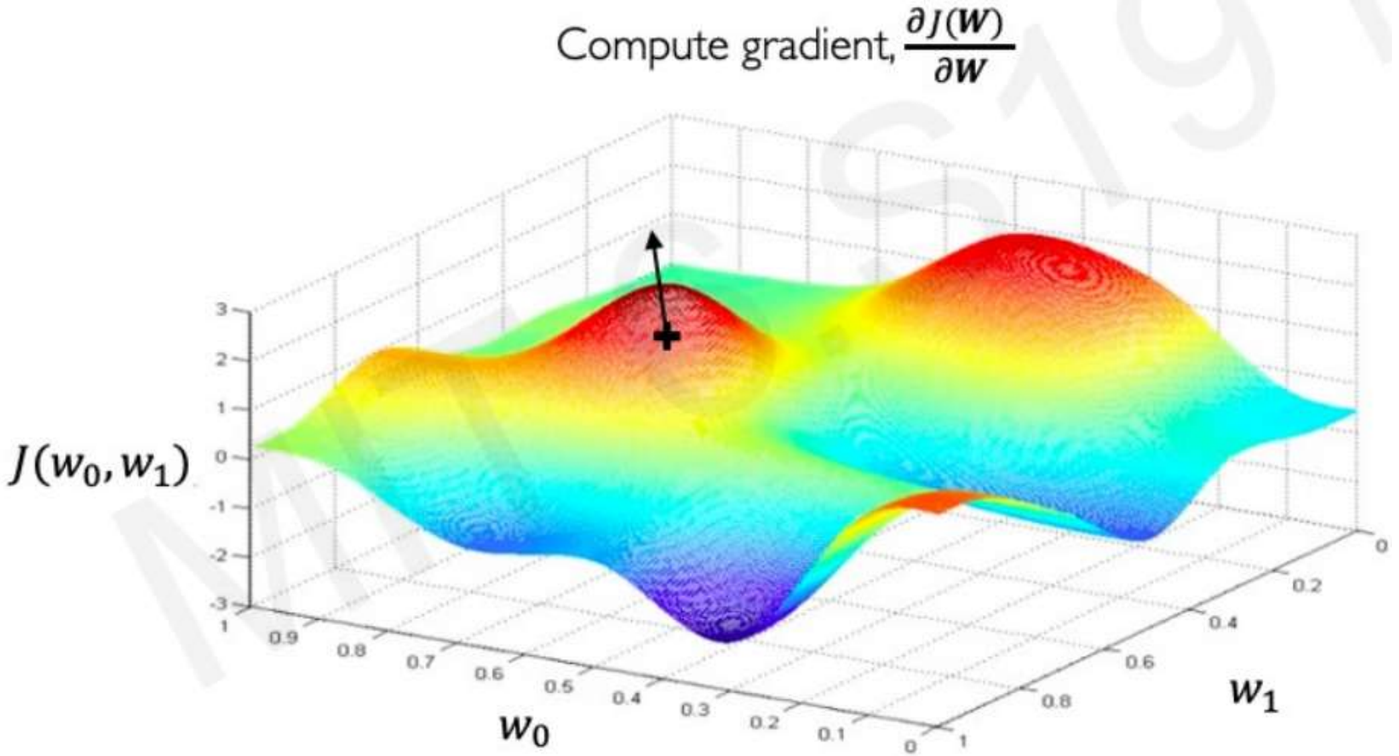
이진 교차 엔트로피 (분류)
n = 데이터 샘플의 수
log = 자연 로그인데 그냥 -> $\log_{10}$ 이라고 생각하고 넘어가면 된다.
실제 값이 1일 대 모델의 예측 확률이 클수록 손실이 작아짐
실제 값이 0일 때 모델의 예측 확률이 작을 수록 손실이 작아짐

$$J(W) = \frac{1}{n}\sum_{i=1}^{n} \left(\underbrace{y^{(i)}}_{\text{Actual}} - \underbrace{f(x^{(i)}; W)}_{\text{Predicted}}\right)^2$$

CCE, MAE, RMSE등 많음

평균 제곱 오차(MSE)(예측)
n = 데이터 샘플의 수
예측 값과 실제값의 차의 제곱의 평균

경사하강법



손실함수의 기울기를 따라가면서 손실 함수의 값을 줄이며 업데이트 하는 방식
기울기는 가중치가 손실에 어떤 영향을 주는지 나타냄

미분,편미분

미분: 모든 변수의 영향을 한꺼번에 고려한 변화율
편미분: 특정 변수 하나만 바꿨을 때 오차가 어떻게 변하는지

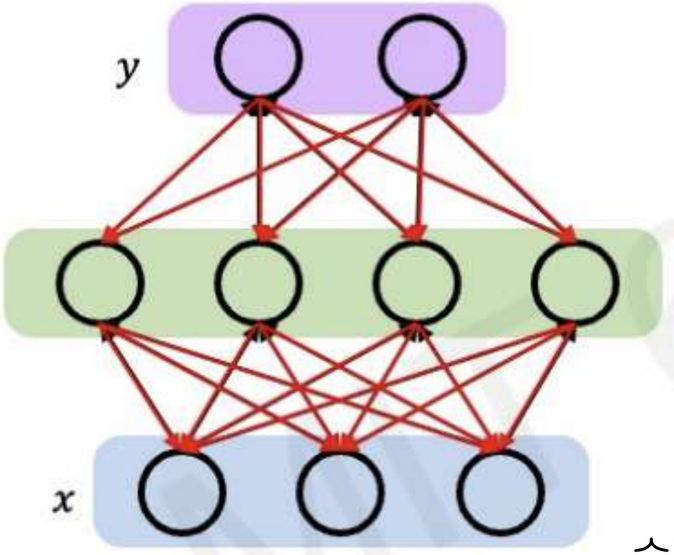
제곱 오차

$$L = (W_1 \cdot x_1 + W_2 \cdot x_2 - y)^2$$

예측값 실제값

$\frac{\partial J(W)}{\partial w_1}$ W_1이 바뀔 때 오차 L이 어떻게 변하는지와 W_2가 바뀔 때 오차 L이 어떻게 변하는지를 구해야하기 때문에 편미분을 써야한다 -> 기울기 구할 때 편미분을 사용하는 이유

역전파



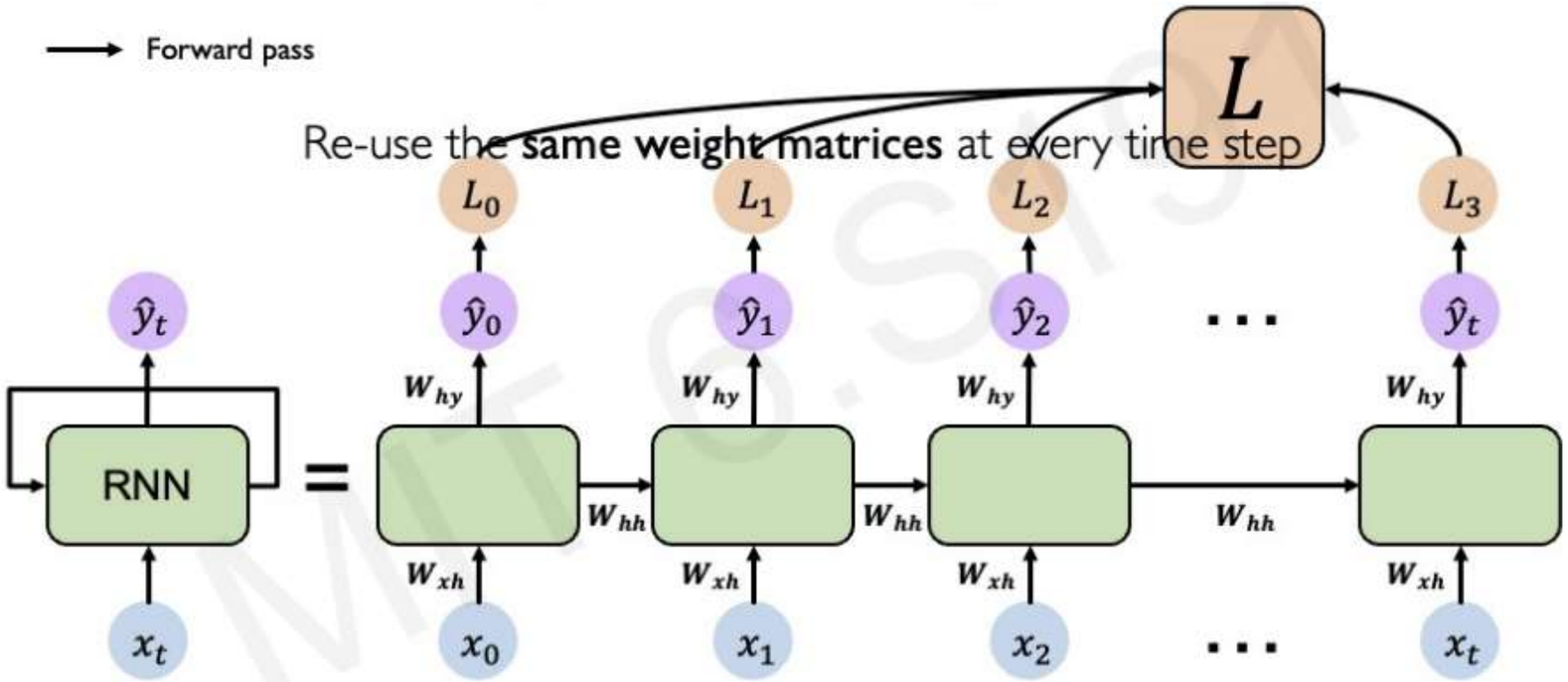
순전파의 출력으로 손실을 구하고
역전파는 그 손실로 최적의 가중치를 구함
-> 학습

학습:더 정확한 출력을 내도록
가중치를 조정하는 과정

순전파 역전파

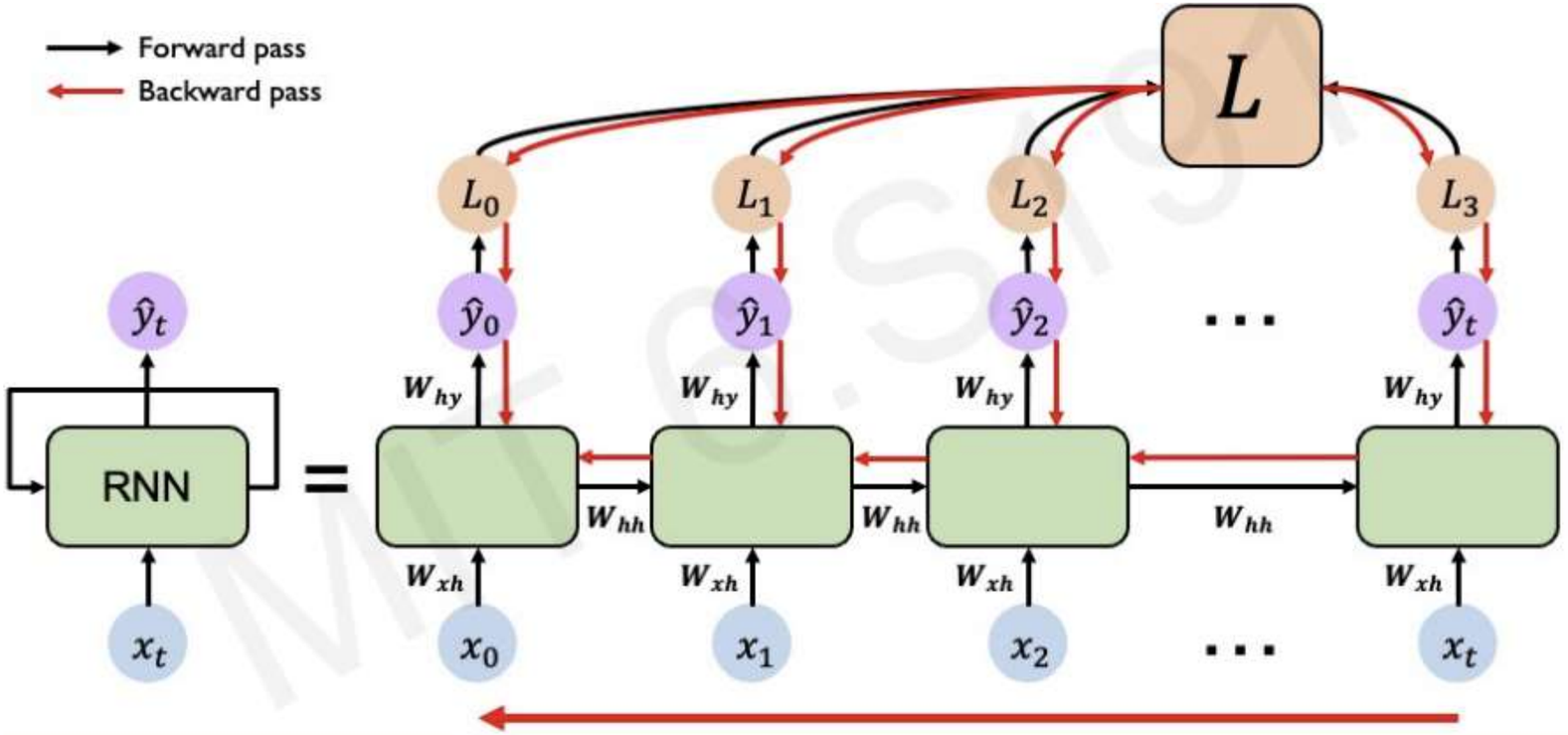
손실을 출력부터 입력까지 기울기를 계산 하는 것 -> 손실을 줄임

RNN의 순전파



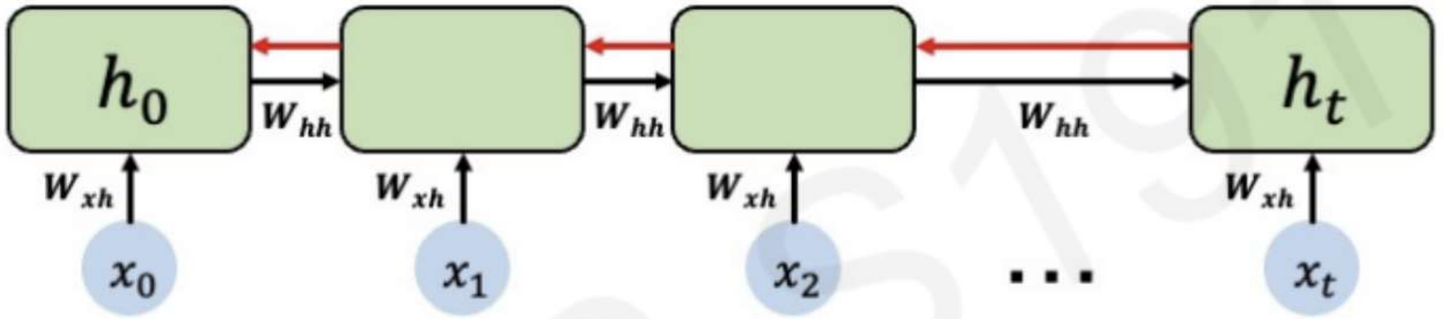
RNN의 손실은 모든 시간 단계에서의 손실을 합산하여 최종 손실을 얻음

BPTT(시간을 통한 역전파)



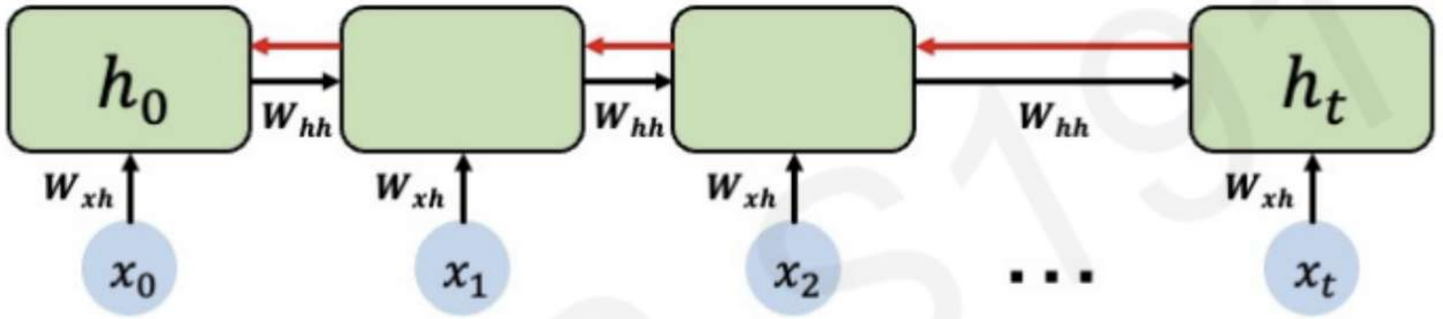
RNN은 각 시간대 개별로 역전파를 진행하고 이를 통해 과거 시간대의 손실을 처음부터 끝까지 모두 전파 할 수 있다

기울기 소실



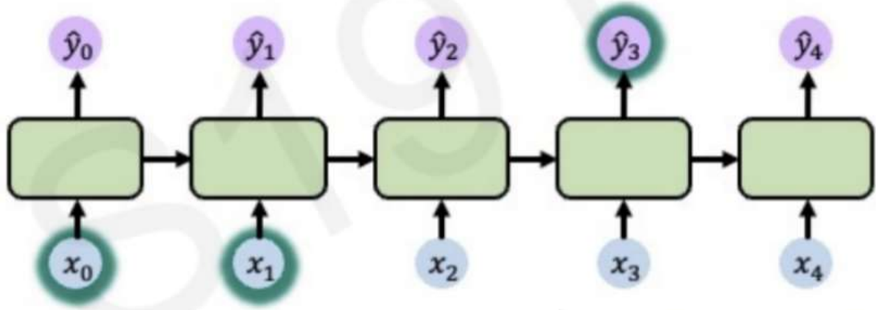
역전파 과정에서 활성화 함수는 대부분 1부터 0까지로 출력 범위가 좁고, 미분을 반복 했을 때 점점 0에 가까워지는 경우가 많아짐 -> 기울기가 0에 수렴하며 그 이전 층들은 거의 학습되지 않음

기울기 폭주



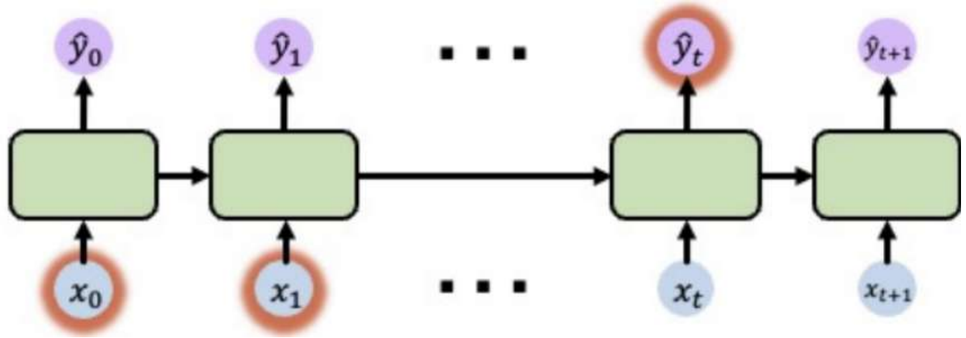
입력값에 가까워 질수록 각 층의 기울기가 1보다 큰 값을 가지게 되면 미분이 반복되는 연쇄법칙에 의해 그 기울기들이 곱해지고 기울기가 폭발적으로 커지며 가중치가 발산하여 학습이 망가짐

기울기 문제의 원인



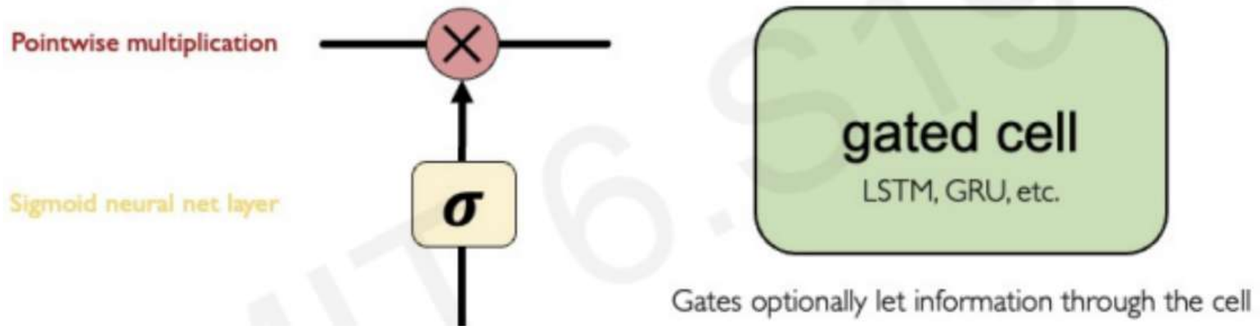
<- 단기 의존성

장기 의존성 ->



RNN은 멀리 떨어진 입력 간의 관계를 이해하고 반영하는 능력이 부족하여 기울기 문제를 야기하게 된다. -> RNN은 장기 의존성이 부족함

기울기 문제의 해결



RNN에 게이트(문)을 달아서 어떤 정보를 기억할지 선택 할 수 있도록 하는 것
보통 게이트 메커니즘의 구조는 보통 시그모이드함수를 거쳐 ⊙(아다마르 곱셈)을 함
게이트 셀의 상태를 업데이트함

아다마르 곱셈 : 두 벡터 a 와 b에 동일한 위치의 요소를 곱하여 새로운 벡터 생성

RNN은 단순한 반복 구조라 장기적인 기억이 취약함 이를 해결 하기 위해 RNN에 게이트 메커니즘을 추가한다.
EX) LSTM GRU 등등

LSTM

LSTM(Long Short-Term Memory)

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t \odot \tanh(c_t)$$

망각 (forget) 게이트
입력 게이트
셀 상태 후보
셀 상태 업데이트
출력 게이트
은닉 상태 업데이트

W? : ?가 의미하는 gate의 가중치 행렬
b : ?가 의미하는 gate의 편향벡터
σ : 시그모이드 함수
tanh: 하이퍼볼릭 탄젠트 함수
c: 셀 상태 벡터, c~ : 셀 상태 후보 벡터
h: 은닉 상태

GRU

GRU(Gated Recurrent Unit)

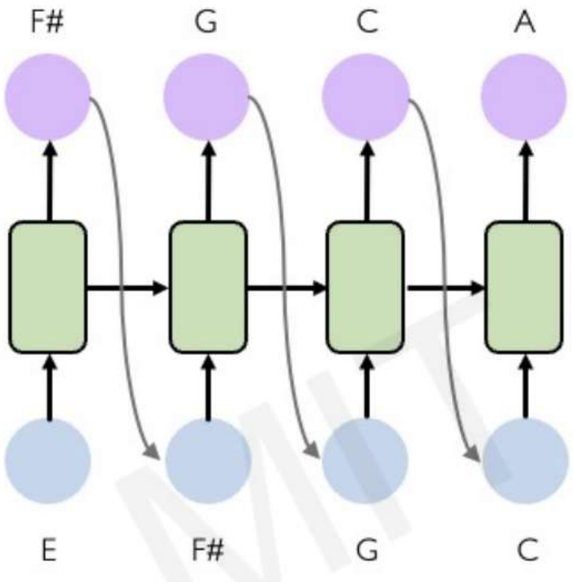
$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$$
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$
$$\tilde{h}_t = \tanh(W \cdot x_t + U \cdot (r_t \odot h_{t-1}) + b)$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

업데이트 게이트
리셋 게이트
은닉 상태 후보
은닉 상태 업데이트

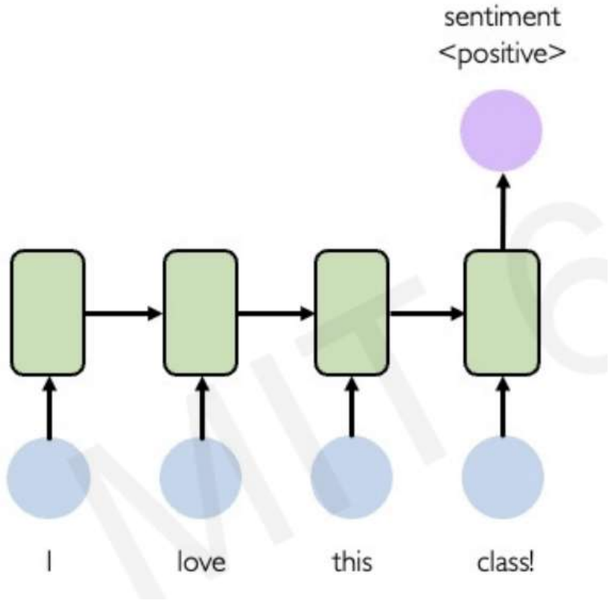
W? : ?가 의미하는 gate의 가중치 행렬
b : ?가 의미하는 gate의 편향벡터
σ : 시그모이드 함수
tanh: 하이퍼볼릭 탄젠트 함수
c: 셀 상태 벡터, c~ : 셀 상태 후보 벡터
h: 은닉 상태, h~ : 은닉 상태 후보 벡터
U : h_{t-1}

RNN 적용

악보 생성



감정 분류



RNN 한계

RNN의 상태(state)가
보유할 수 있는 정보
양에 대한 병목 현상

속도 문제, 병렬(동시 처리) 불가
-> 시간 단위로 정보 처리

단기 기억

Transformer 02

Transformer

Attention Is All You Need

Ashish Vaswani[*] Google Brain avaswani@google.com
Noam Shazeer[*] Google Brain noam@google.com
Niki Parmar[*] Google Research nikip@google.com
Jakob Uszkoreit[*] Google Research usz@google.com
Llion Jones[*] Google Research llion@google.com
Aidan N. Gomez[*] [†] University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser[*] Google Brain lukaszkaiser@google.com
Illia Polosukhin[*] [‡] illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

[*]Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

[†]Work performed while at Google Brain.

[‡]Work performed while at Google Research.

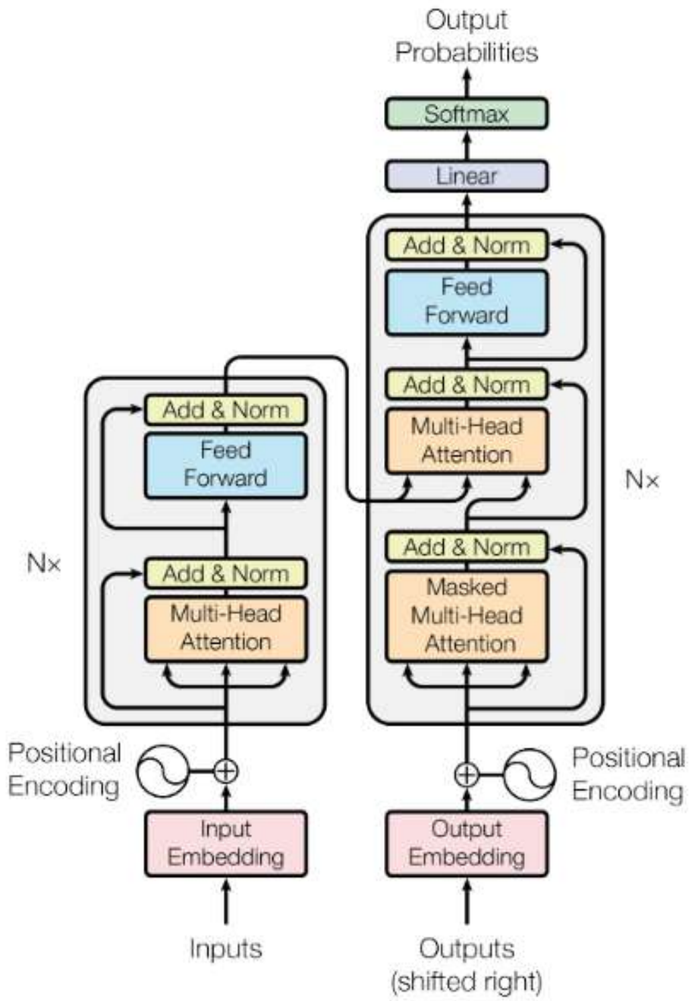
31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Tranformer: Attention is All you Need 논문에서 처음 소개 된
self attention 기반 모델

병렬 처리, 장기 기억, 다양한 활용성
-> RNN 한계 극복

Transformer 02

Transformer



Encoder: 입력 처리
Decoder: 출력 생성

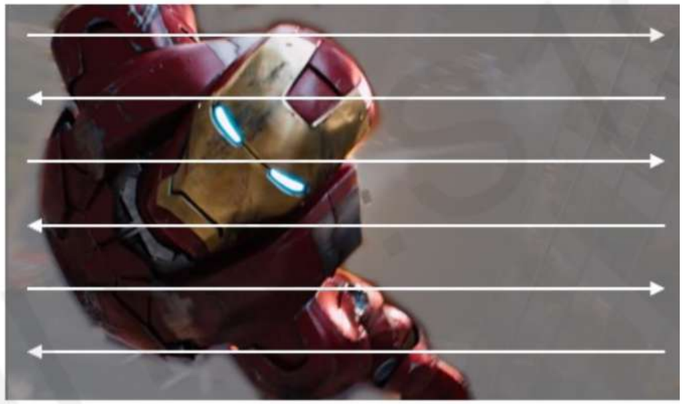
Encoder, Decoder에는 각각 attention이 있음

attention: 입력 중 어떤 부분이 더 중요한지 계산하고 그 부분에 더 집중할 수 있도록 도와주는 메커니즘

Attention 03

Attention is all you need

인간의 뇌는 정보가 입력값으로 들어오면
이에 대해 생각하고 자동으로 중요한 특징에 맞춰
집중을 할 수 있음



인간과 달리 AI는 이미지에서 특징을
이미지의 왼쪽에서 오른쪽으로 픽셀 단위로
이동하여 이미지를 스캔하고 각각 픽셀에 중요도에
대한 값을 계산하여 중요한 부분을 구별할 수 있음

attention의 처리 방식

Query: 집중하고 싶은 정보가 어떤 것인지 나타내는 벡터
Key: 입력의 특징을 요약한 벡터, Query와 유사성을 비교하는 기준
Value: 실제 내용, Query가 집중한 결과로 얻는 실제 내용(정보)

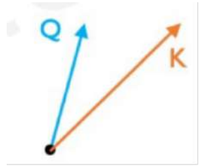
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

1. 입력 벡터 정의

$Q \in \mathbb{R}^{n \times d_k}$: n개의 Query 벡터
$K \in \mathbb{R}^{n \times d_k}$: n개의 Key 벡터
$V \in \mathbb{R}^{n \times d_v}$: n개의 value 벡터

d_k, d_v = 벡터 차원

2. Q와 K 사이의 유사도 계산



$QK^T \in \mathbb{R}^{n \times n}$: n x n의 matrix(행렬)
$QK^T[i][j]$: i 번 번째 Q 벡터 qi, j번째 k 벡터 kj의 내적(dot product) -> 유사도

	K_1	K_2	...	K_n
Q_1	$Q_1 \cdot K_1$	$Q_1 \cdot K_2$	...	$Q_1 \cdot K_n$
Q_2	$Q_2 \cdot K_1$	$Q_2 \cdot K_2$	...	$Q_2 \cdot K_n$
...	...	...	...	...
Q_n	$Q_n \cdot K_1$	$Q_n \cdot K_2$	...	$Q_n \cdot K_n$

attention의 처리 방식

3. 스케일 안정화

$$\frac{QK^T}{\sqrt{d_k}}$$

dot product(내적)의 값은
차원이 클 수록 커짐
-> softmax에서 큰 수가 나와
기울기가 소실 될 수 있음

4. 중요도 정규화

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \in \mathbb{R}^{n \times n}$$

Q에 대한 K와의 유사도를 softmax 함수로 정규화
->Q가 K에 얼마나 집중할지 확률로 나타냄
이렇게 나온 값을 attention weight라고 함

5. 가중합

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

중요도(attention weight)와 정보의 행렬곱
-> 유사도 높은 K에 연결된 V를 가중합(합쳐서 평균을 냄)
정보를 가중합을 통해 얻는다.

attention의 처리 방식

attention weight(중요도) x value = 출력

끝