


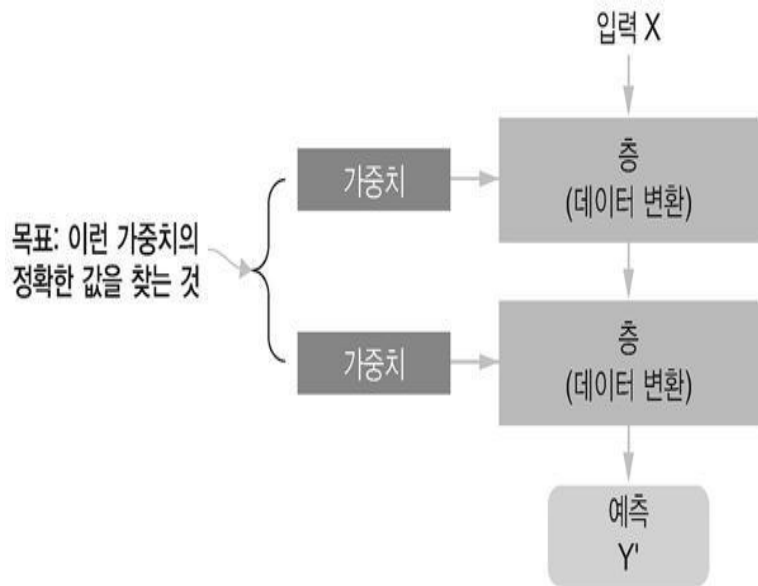


Deep Learning week 2

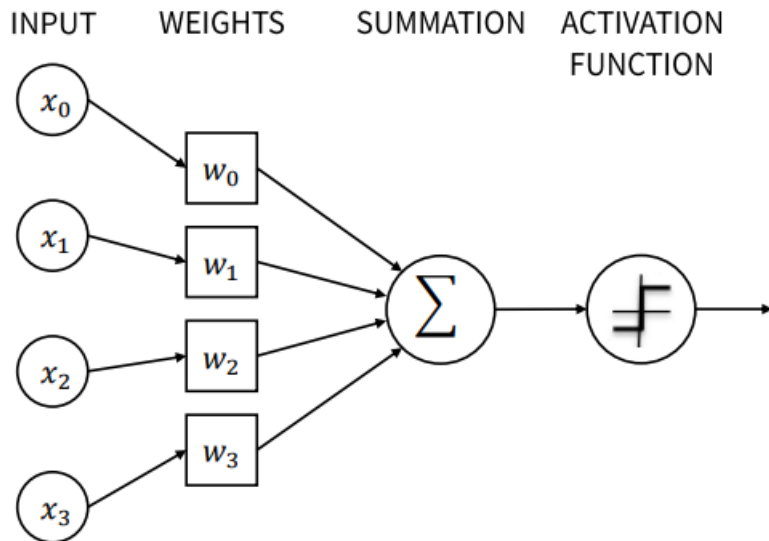
문구영



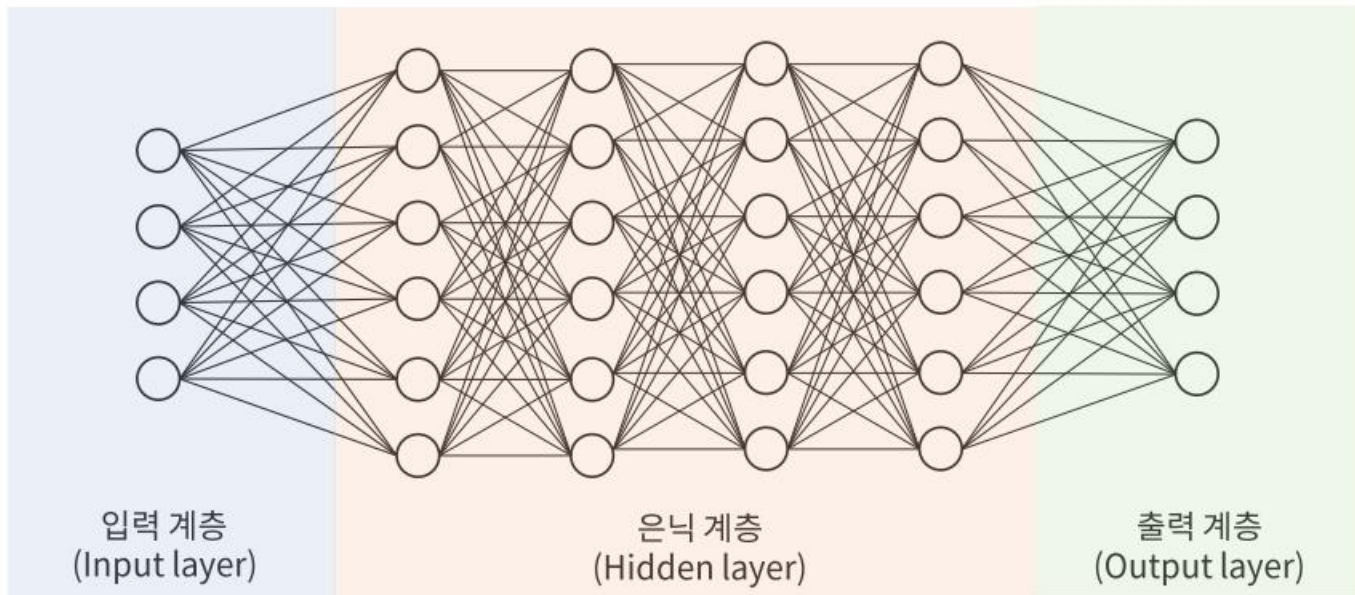
요약, 복습



Copyright © Gilbut, Inc. All rights reserved.

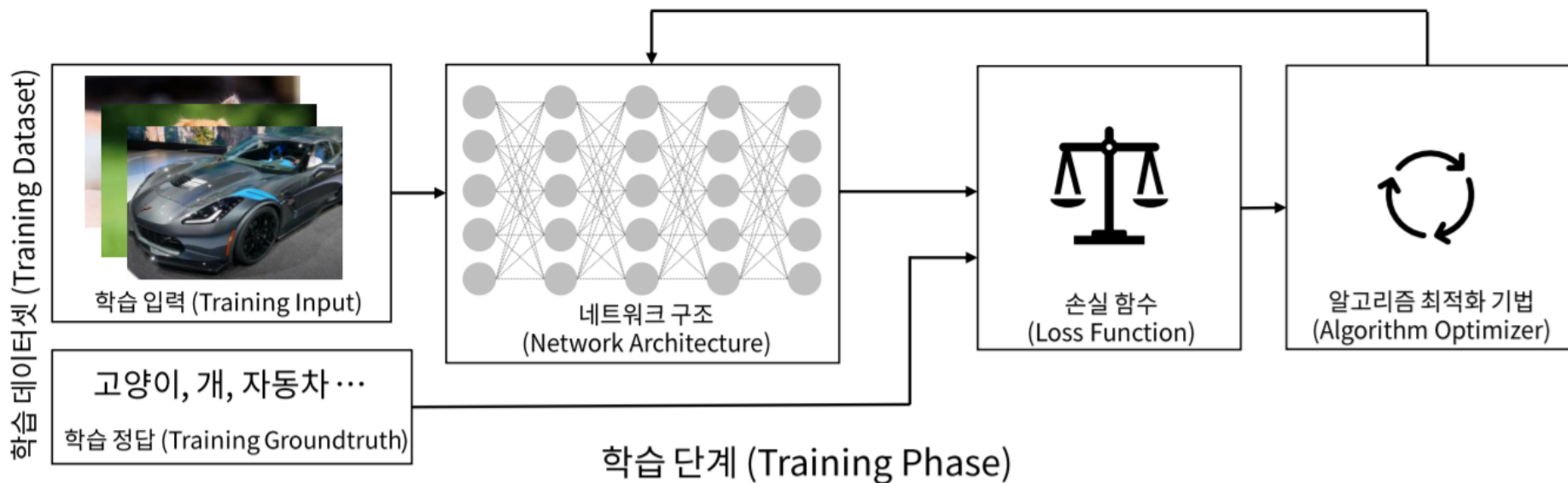


요약, 복습 - 심층 신경망



- 은닉층의 복잡한 절차를 통해 데이터의 유의미한 표현 학습

요약, 복습 - 딥러닝의 학습



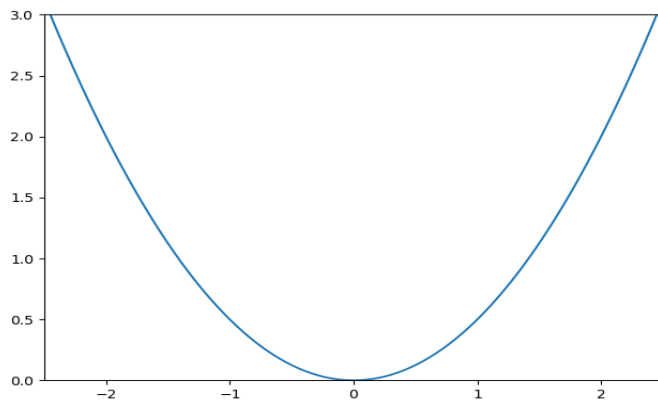
- 네트워크 구조는 형태를 예측하기 힘든 복잡한 함수
- 손실함수의 값 (손실점수)를 피드백 신호로 사용하여 가중치 업데이트
- 손실점수를 최소화하는 가중치(parameter)를 구하는 최적화 문제

손실 함수

MSE : 가장 기본적인 손실 함수

회귀에 많이 쓰임

오차가 커질수록 손실함수가 빠르게 증가

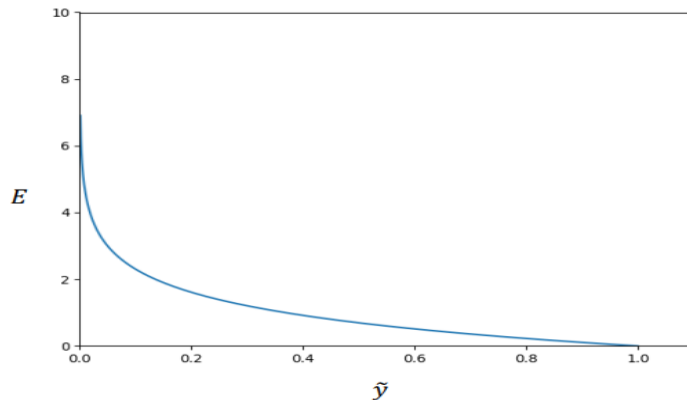


Cross Entropy (엔트로피 오차)

$$E = -\sum_i y_i \log \tilde{y}_i$$

y_i : 정답 (원 핫 인코딩)
 \tilde{y}_i : 추정치

정답인 클래스에 대해서만 오차 계산



최적화 이론 (Optimization Theory)

$\text{minimize}_x f(x)$

subject to $g_i(x) \leq 0, h_j(x) = 0$

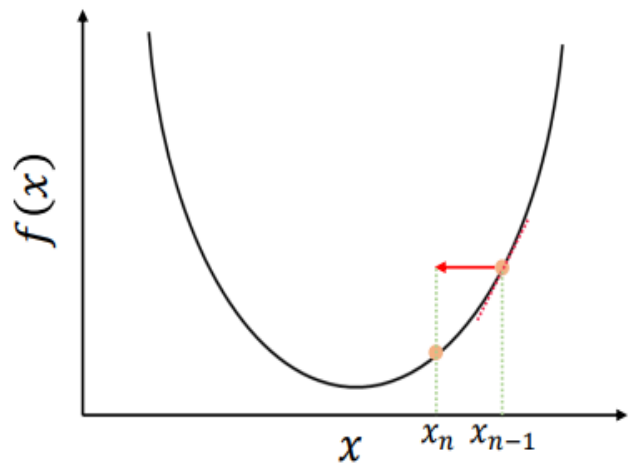
$f(x)$: 목적함수 (손실함수)

$g_i(x)$: 부등식 제약 조건

$h_j(x)$: 등식 제약 조건

- 가능한 모든 해 중 **최적의 해**를 찾는 문제를 해결
- 부등식, 등식 제약 조건을 지키면서 **목적 함수를 최소화**
- 대부분의 딥러닝에서 제약조건은 미사용
- 목적함수의 형태에 따라 다양한 문제 해결 알고리즘 존재

경사 하강법 (Gradient Descent)

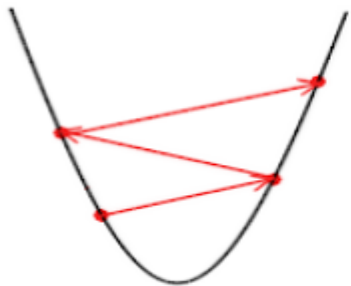


- $x_n = x_{n-1} - \alpha \frac{df(x_{n-1})}{dx}$
- $x_n = x_{n-1} - \alpha \nabla f(x_{n-1})$
- α : 학습률

$f(x)$ 의 값이 변하지 않을 때 까지 위 과정을 반복

적절한 학습률의 선택

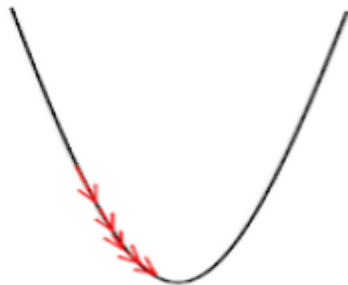
Big Learning Rate



Just right

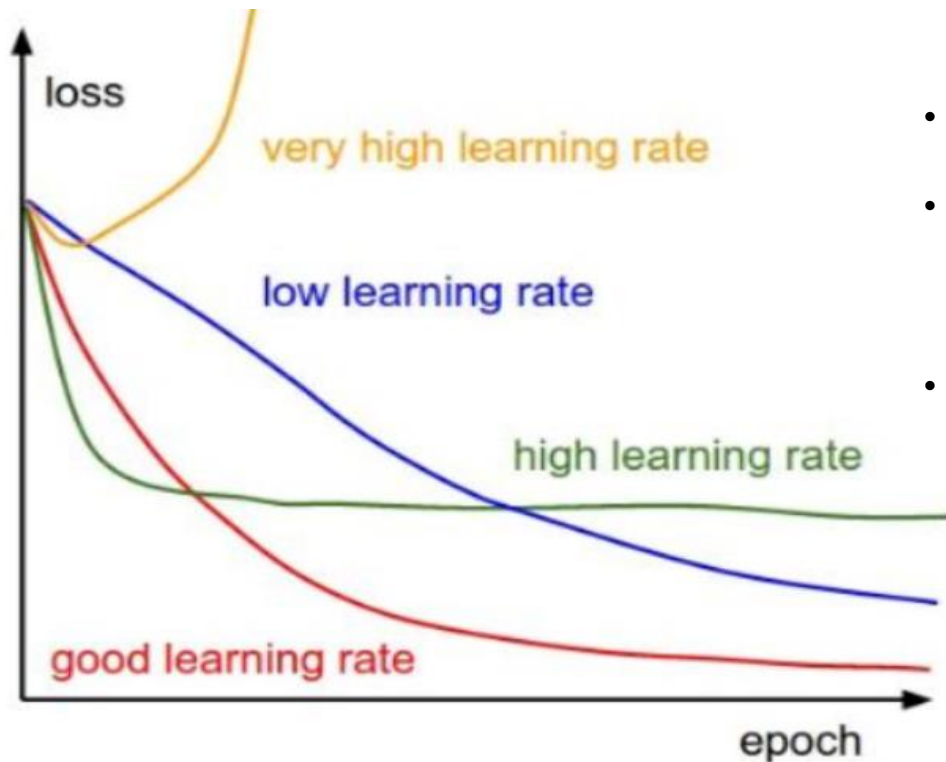


Too small



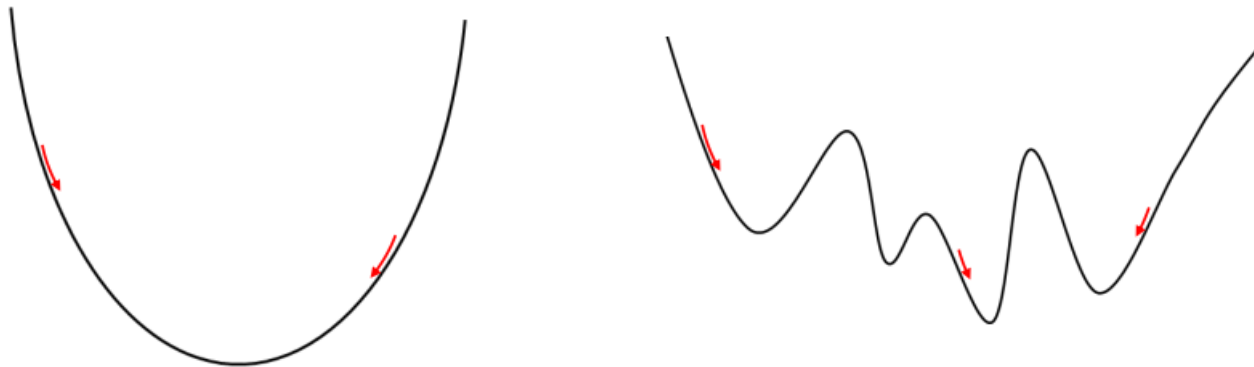
- 학습률이 크면 데이터가 무질서하게 이탈
- 학습률이 작으면 학습시간이 오래 걸림
- 두 경우 모두 최저점에 수렴하지 못함

학습률과 손실



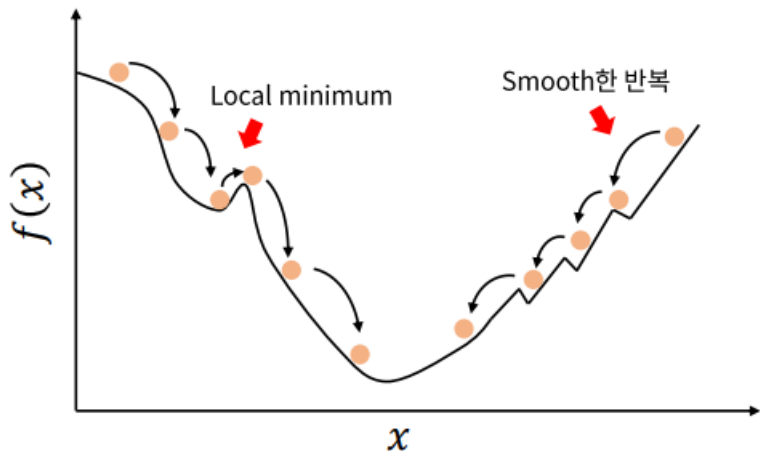
- **Low** : 손실의 감소가 선형의 형태, 천천히 학습
- **high** : 손실의 감소가 지수적인 형태, 구간에 따라 빠른 학습 or 정체
- **Very high** : 손실 점수 발산

경사 하강법의 문제



- 볼록 함수(convex)는 어디에서 시작하더라도 경사 하강법으로 최적값에 도달
- 비볼록 함수(non-convex)는 초깃값에 따라 다른 최적값 (global min이 아닌 **local min에 빠질 위험**)
- Saddle point (안장점) : 기울기가 0이 되지만 극값이 아닌 지점
- 경사 하강법은 안장점에서 벗어나지 못한다

Momentum



- **Local minimum에 대처** 가능
- 경사 하강법 대비 2배의 메모리

- **이동 벡터**를 이용해 이전 기울기에 영향을 받는 알고리즘

- $v_t = rv_{t-1} + n\nabla f(x_{t-1})$

- $x_t = x_{t-1} - v_t$

r : 관성 계수 (momentum term) ≈ 0.9

n : 학습률 (learning rate)

v_t : t 번째 스텝에서의 x 의 이동벡터

AdaGrad

- 적응적 기울기 (Adaptive Gradient) : 변수 별로 학습율이 달라지게 조절

- $g_t = g_{t-1} + (\nabla f(x_{t-1}))^2$

- $x_t = x_{t-1} - \frac{n}{\sqrt{g_t + \epsilon}} \cdot \nabla f(x_{t-1})$

g_t : t 번째 스텝까지의 기울기의 누적 크기

기울기가 커서 학습이 많이 된 변수는 학습율을 감소 (다른 변수들이 잘 학습되도록 한다)

g_t 는 증가함수 꼴, 학습이 오래 진행되면 (값이 계속 커지면) 더 이상 학습이 이루어지지 않음

RMS Prop

RMS Prop : 합 대신 지수평균을 사용

- $g_t = r g_{t-1} + (1 - r)(\nabla f(x_{t-1}))^2$

- $x_t = x_{t-1} - \frac{n}{\sqrt{g_t + \epsilon}} \cdot \nabla f(x_{t-1})$

g_t 가 무한정 커져 학습이 이루어지지 않는 것을 방지 (오랜 학습이 가능)

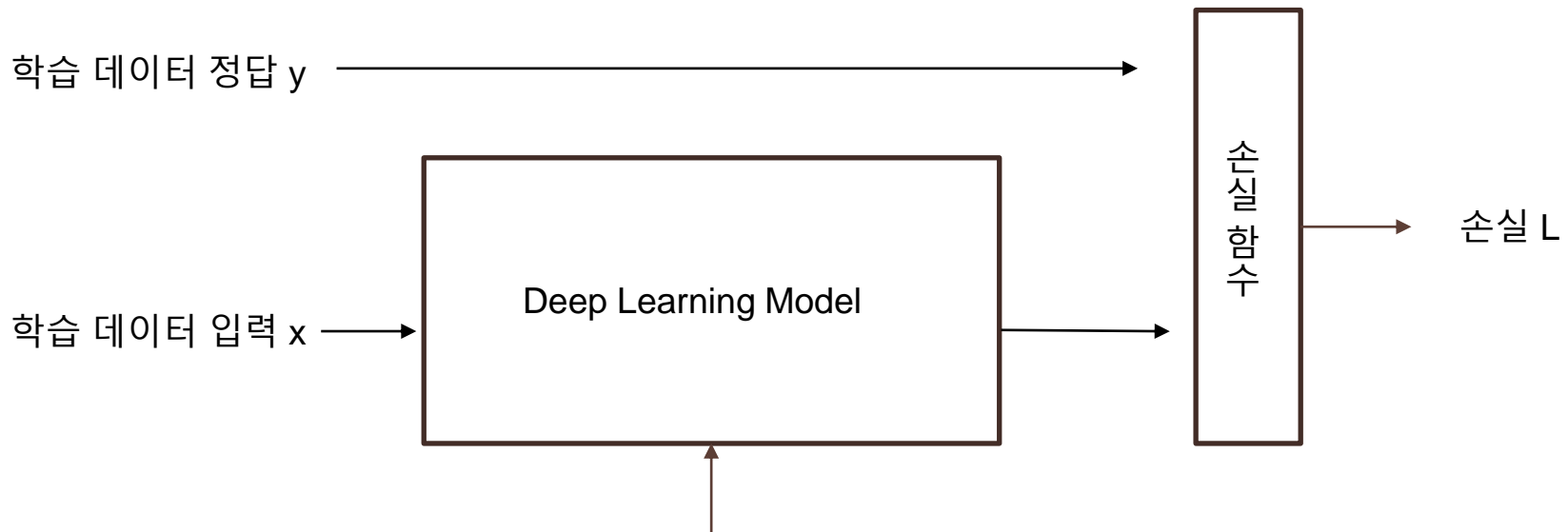
r : 지수 평균의 업데이트 계수

Adam

ADaptive Moment Estimation : RMSProp과 Momentum의 장점 결합

- $m_t = B_1 m_{t-1} + (1 - B_1) \nabla f(x_{t-1})$
- $g_t = B_2 g_{t-1} + (1 - B_2) (\nabla f(x_{t-1}))^2$
- $\widehat{m}_t = \frac{m_t}{1 - B_1^t} \quad \widehat{g}_t = \frac{g_t}{1 - B_2^t}$
- $x_t = x_{t-1} - \frac{n}{\sqrt{\widehat{g}_t + \epsilon}} \widehat{m}_t$
- 가장 최신의 기술, 가장 많이 사용

딥러닝 가중치 업데이트



모델을 묘사하는 매개 변수
(가중치 : a, b, c, d)

- 매개 변수에 의해 동작이 달라짐

딥러닝 모델의 학습

$$\theta_{n+1} = \theta_n - n\nabla L(\theta_n)$$

$$a_{n+1} = a_n - n\nabla \frac{\partial L(\theta_n)}{\partial a}$$

$$b_{n+1} = b_n - n\nabla \frac{\partial L(\theta_n)}{\partial b}$$

$$c_{n+1} = c_n - n\nabla \frac{\partial L(\theta_n)}{\partial c}$$

$$d_{n+1} = d_n - n\nabla \frac{\partial L(\theta_n)}{\partial d}$$

- 각 변수를 각각 조금씩 바꾸어 대입
- (4+1)번의 손실 함수 평가

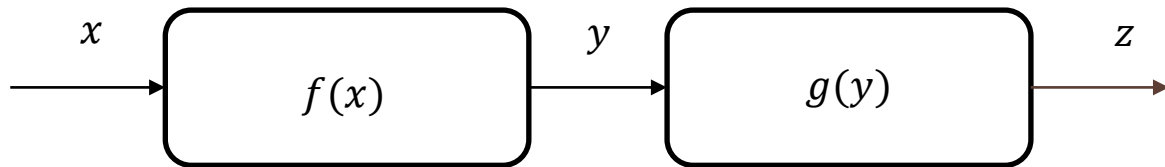
- N : 딥러닝 모델의 가중치 개수

- 미분을 위해 필요한 손실 평가 횟수 : N+1
- 손실 함수 평가에 필요한 곱 연산 : N

>>> gradient descent 한 스텝 계산을 위해 **N(N+1)번의 연산** 필요

역전파 알고리즘의 필요성

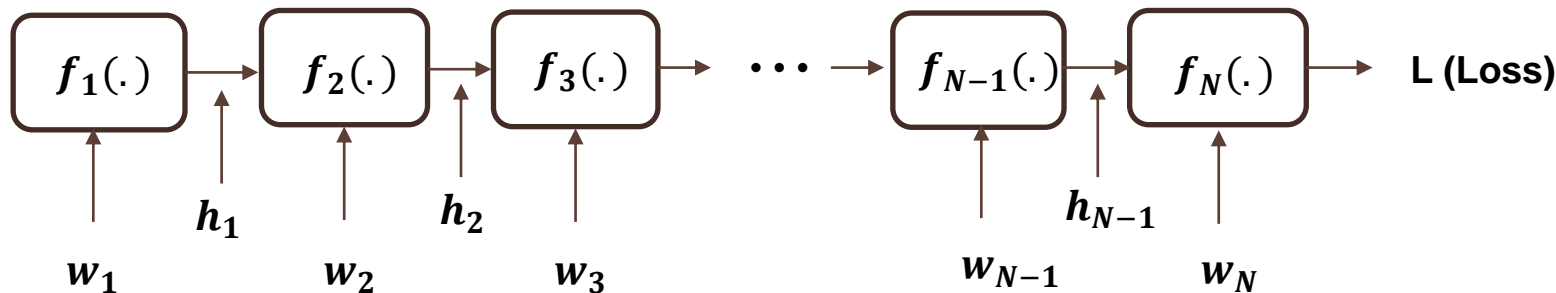
미분과 연쇄법칙



$$\frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx} = \frac{dg(y)}{dy} \frac{df(x)}{dx} = \frac{dg(f(x))}{dx}$$

- 딥러닝의 각 layer를 하나의 함수로 보면
신경망을 합성 함수로 표현 가능
- 연쇄 법칙을 이용하면 연속된 함수의 미분을 각각의 미분의 곱으로 표현 가능

딥러닝의 연쇄 법칙



$$\frac{\partial L}{\partial w_n} = \frac{\partial L}{\partial h_N} \left(\prod_{i=n+1}^N \frac{\partial f_i(h_{i-1})}{\partial h_{i-1}} \right) \frac{\partial f_n(w_n)}{\partial w_n}$$

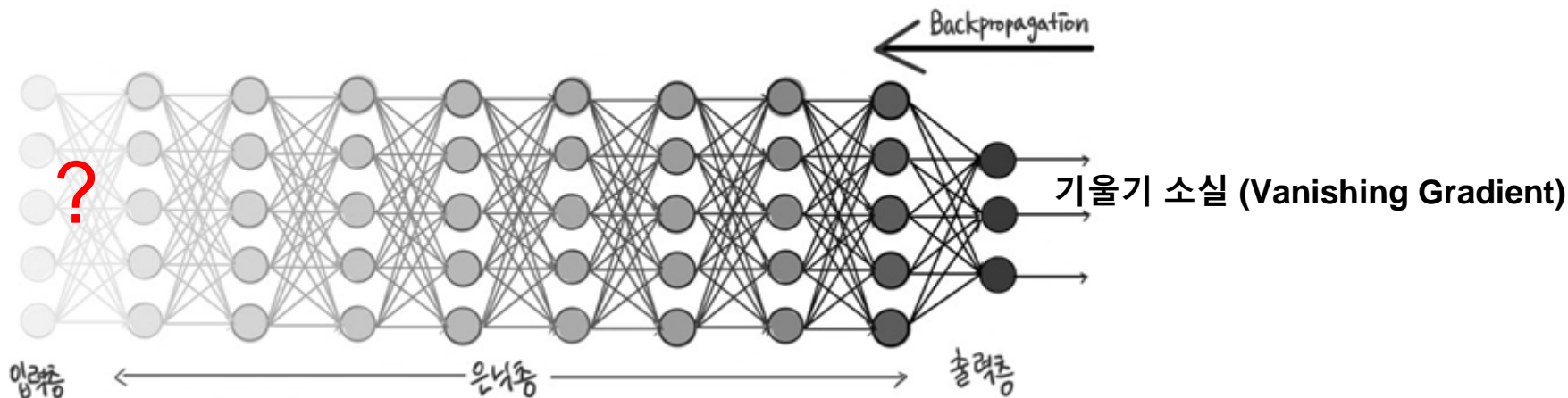
- 미분하고자 하는 경로 사이에 있는 모든 미분 값을 곱하면 원하는 미분을 구할 수 있다
- 앞선 layer들의 가중치의 영향을 받음
- Output layer에서의 손실 점수 변화량이 앞선 layer들로 전파 됨

오류 역전파 알고리즘

1. 학습 데이터로 정방향 연산을 하여 **loss**를 계산
2. 정방향 연산 시, 계층별로 역전파에 필요한 중간 결과를 저장
3. **Loss**를 각 파라미터로 미분 (연쇄 법칙에 의한 역방향 연산 이용)
 - 마지막 계층부터 하나씩 이전 계층으로 연쇄적으로 계산
 - 역방향 연산 시, 정방향 연산에서 저장한 중간 결과를 사용(메모리 추가 사용)

역전파 알고리즘의 문제

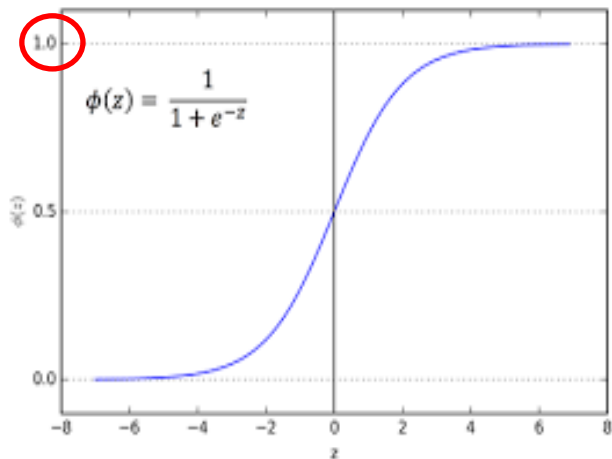
- Layer가 많아질 수록 표현력 증가, 역전파로 많은 수의 가중치 학습
- But, 이론과 달리 역전파 계층이 깊어질 수록 가중치의 학습이 잘 이루어지지 않음



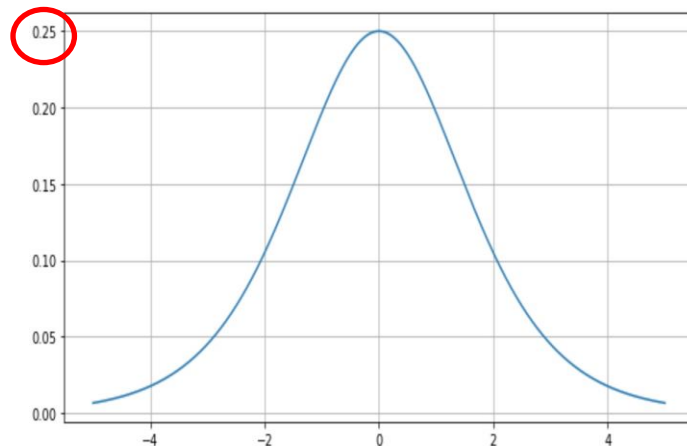
기울기 소실의 원인 - 활성화 함수의 특성

- $\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$

로지스틱 함수 (0~1의 값 출력, 확률 표현)



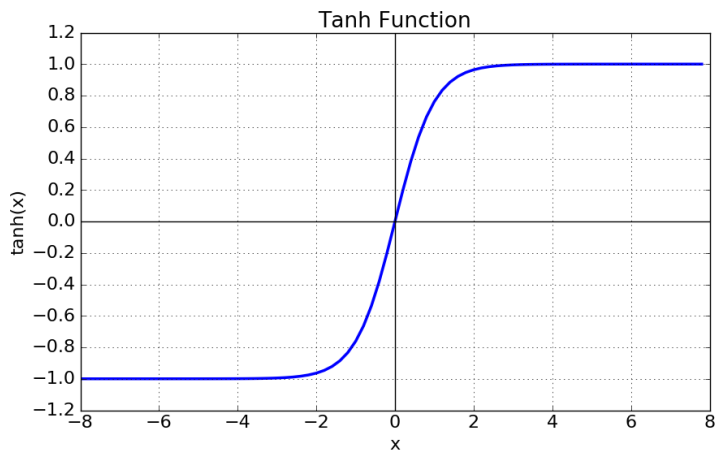
- $\frac{d\text{sigmoid}(x)}{dx} = \text{sigmoid}(x)(1 - \text{sigmoid}(x))$



- Sigmoid 함수의 미분이 거듭 곱해지면서 앞쪽 계층의 gradient 소실

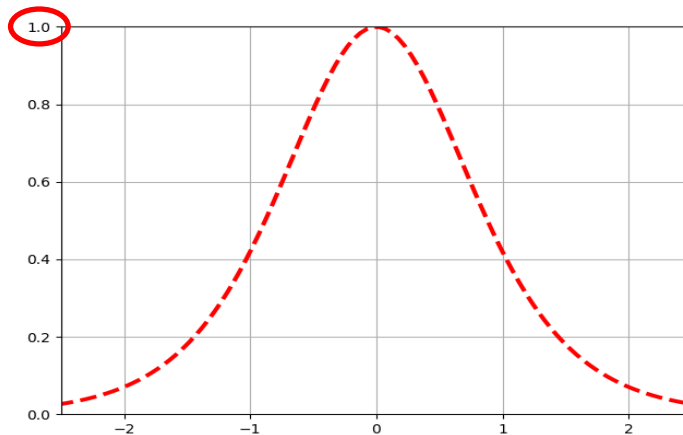
활성함수 개선-Hyperbolic Tangent

- $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$



- Sigmoid를 위 아래로 늘림 (출력범위 2배)

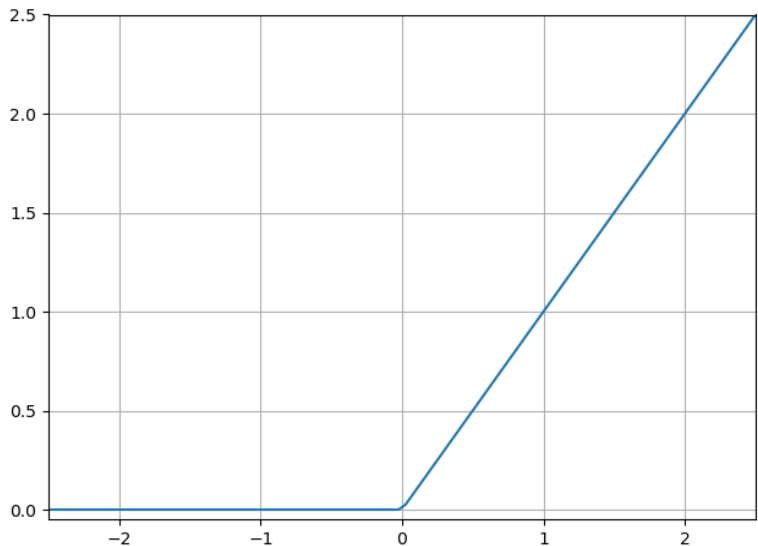
- $\frac{d}{dx} \tanh(x)$



- 여전히 기울기 소실 발생

활성함수 개선-Rectified Linear Unit (ReLU)

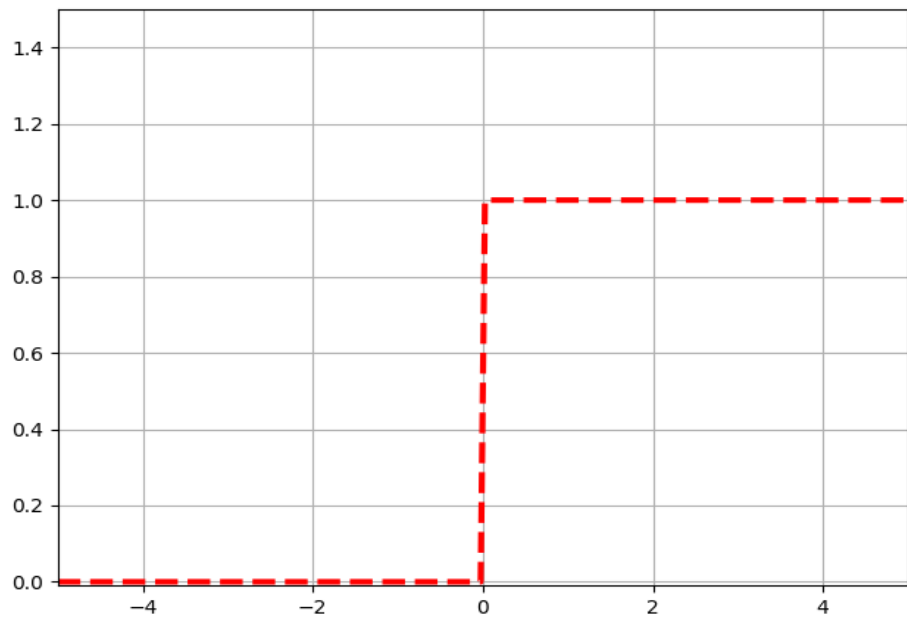
- $ReLU(x) = \max(0, x)$



- 0 보다 작은 값을 0으로 강제
- 가장 사용빈도가 높은 활성 함수
- 미분 값이 일정(0 or 1)해서 학습에 용이
- 단순한 구현, 매우 빠른 연산

ReLU

- $\frac{d}{dx} \text{ReLU}(x)$

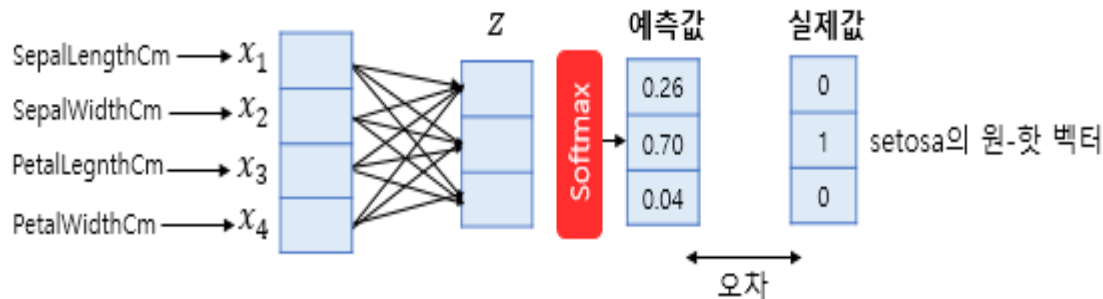


- 양수라면 입력에 관계없이 동일한 미분값 출력
- 기울기 소실 해결

Softmax function

- $$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

- 각 입력의 지수함수 정규화
- 각 출력은 0~1 사이의 값
- 여러 경우의 수 중 한 가지에 속할 확률 표현
- 최종 출력 단계에서 N가지 범주로 분류 가능한 **Multi class classification**에 사용



- Sigmoid는 하나의 입력을 0으로 강제한 2-class softmax