딥러닝 관련 학습 기법들

Deep Session 3차시

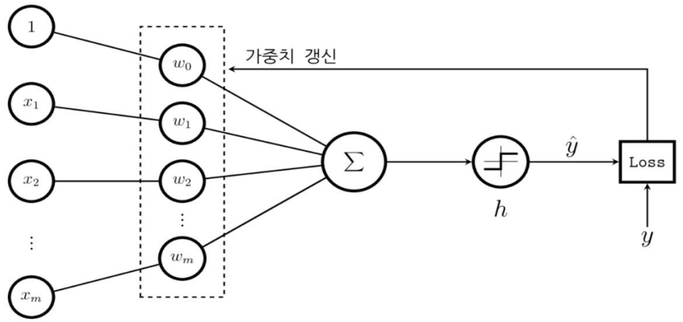
빅데이터 분석 학회 D&A

CONTENTS.

활성화 함수의 정의

활성화 함수란?

입력 신호의 총합을 출력 신호로 변환하는 함수



선형함수 $f(x) = ax$

| 순전파에서의 활성화 함수 역할

- 여러 층을 쌓는 이점을 살릴 수 없음

$$f(f(...f(x)...)) = a^n x$$

| 역전파에서의 활성화 함수 역할

- 미분값이 상수여서 기울기 입력에 관계없이 일정

$$f'(x) = a$$

$$\text{weight의 업데이트} = \underbrace{-\gamma\nabla F(\mathbf{a}^n)}_{\text{에러 낮추는 방향 (decent)}} \times \underbrace{\gamma}_{\text{한발자국 크기 (learning rate)}} \times \underbrace{\nabla F(\mathbf{a}^n)}_{\text{현 지점의 기울기 (gradient)}}$$

보폭 방향

비선형함수 $\sigma(x) = \frac{1}{1+e^{-x}}$

| 순전파에서의 활성화 함수 역할 $y = \sigma(wx + b)$

- 각 뉴런의 출력값 결정
- 신경망에 비선형성을 추가하여 비선형 문제(XOR게이트) 해결

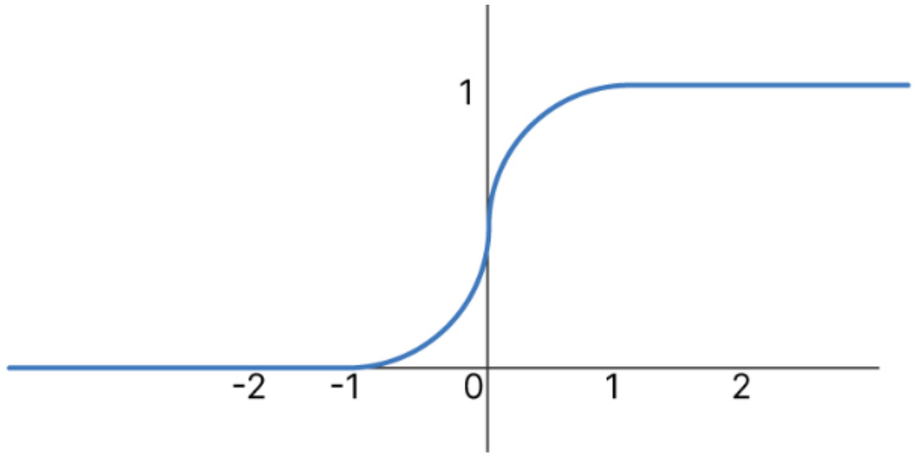
| 역전파에서의 활성화 함수 역할 $\frac{d}{dx}\sigma(x) = \sigma(x)(1 - \sigma(x))$

- 입력값에 따라 변하는 미분값으로 각 층에 다양한 기울기 전파
- 미분값은 가중치를 얼마나 조정할지 결정 $\frac{\partial L}{\partial w} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial w}$

활성화 함수의 종류

Sigmoid 함수

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



특징

- 이진분류의 출력층에 주로 사용되는 함수
- 계단 함수와 달리 출력 값이 0부터 1사이의 값이 출력

단점

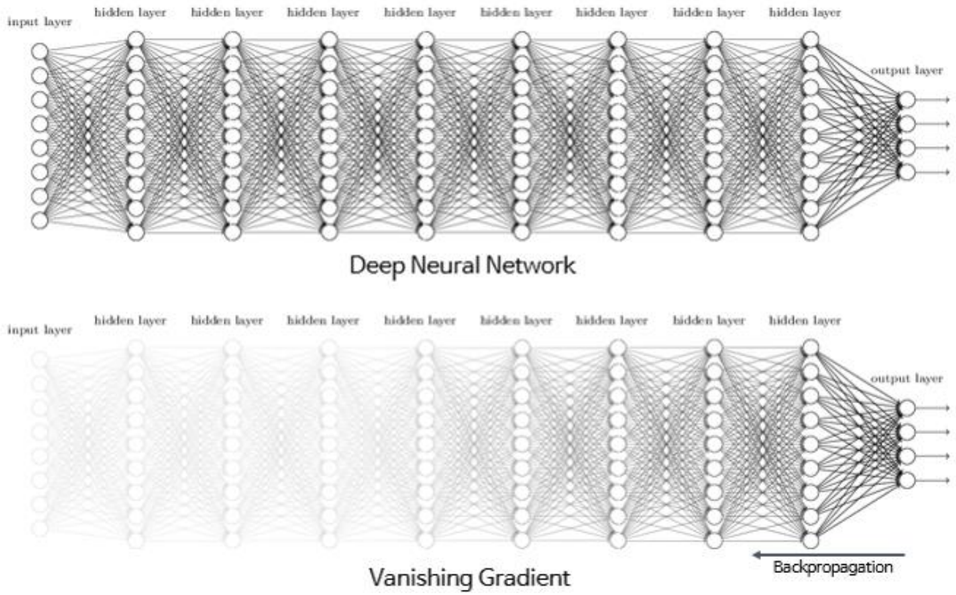
1. Gradient vanishing
2. Non-zero centered

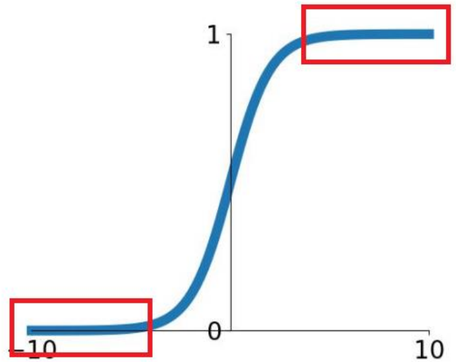
활성화 함수의 종류

Gradient Vanishing

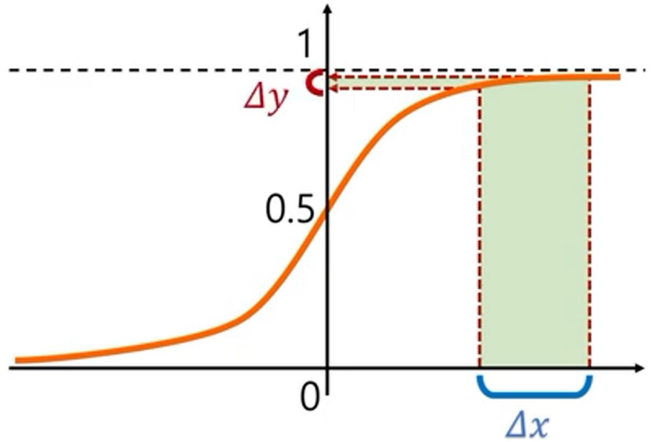
기울기가 계층을 이동하면서 점차 기울기가 사라지는 문제

= 오차 역전파의 학습효과가 0에 가까워진다





Sigmoid



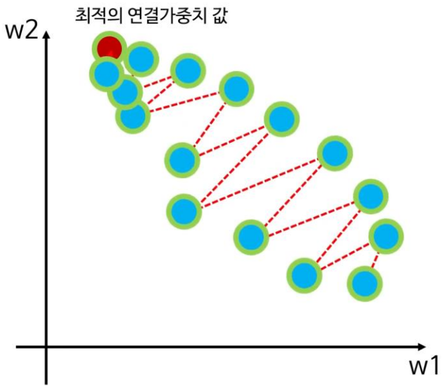
활성화 함수의 종류

Non – zero Centered

가중치의 학습과정이 지그재그 패턴을 띄는 문제 = 학습과정의 능률 감소

Sigmoid의 출력값이 항상 양수여서 발생

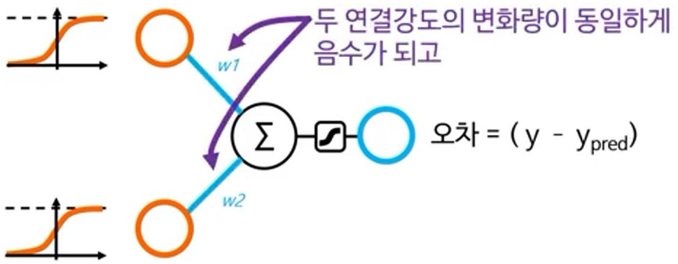
-> Tanh함수로 해결



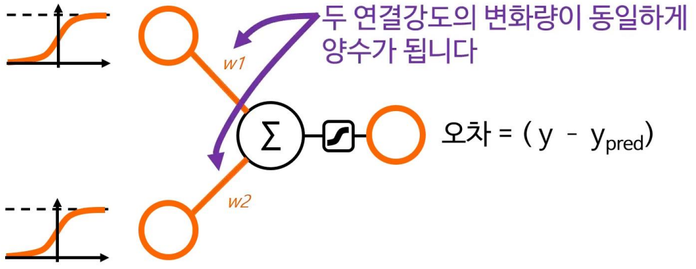
다층신경망 학습 방법 : 새로운 가중치 변화량 ∝ [현 입력값 x 오차]

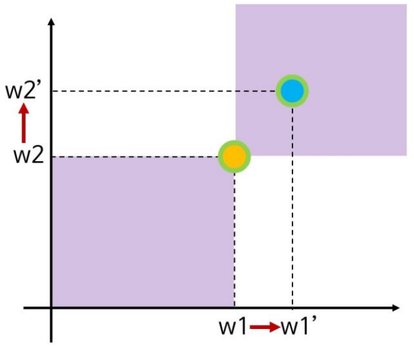
- 시그모이드 함수에 의해 출력된 값은 항상 (+)

- 오차가 (-)



- 오차가 (+)

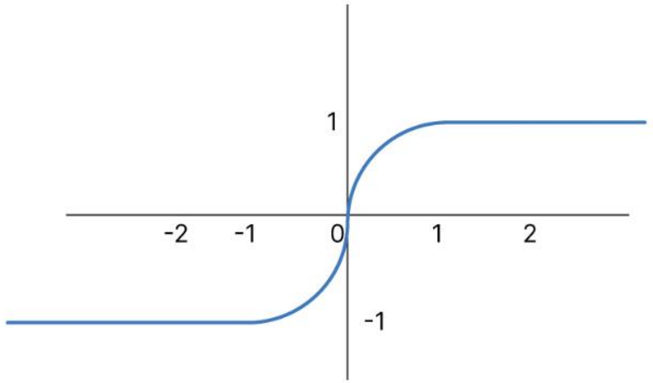




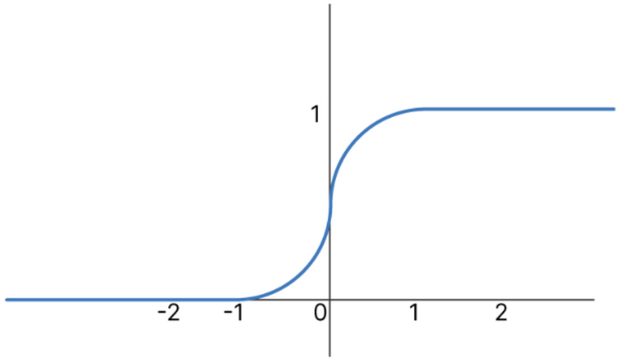
활성화 함수의 종류

Tanh 함수

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



tanh



Sigmoid

특징

- Sigmoid 함수에서 non-zero centered 단점 극복
- 출력값이 -1 ~ 1

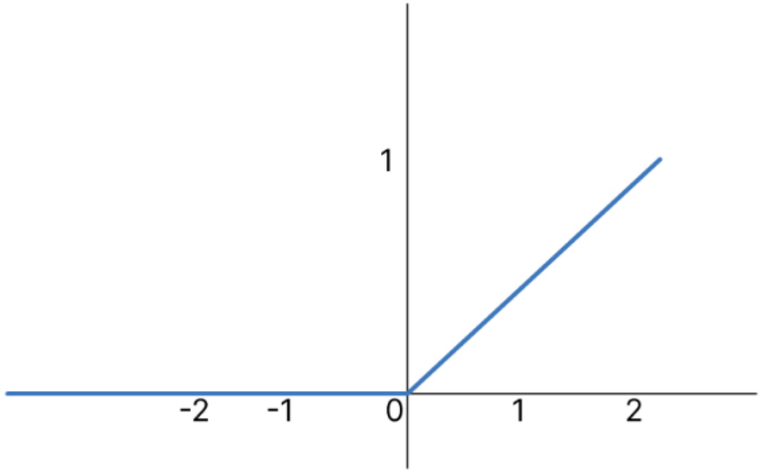
단점

- 여전히 Gradient vanishing 문제가 존재
-> 깊은 네트워크에서는 덜 선호

활성화 함수의 종류

ReLU 함수

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



특징

- Gradient vanishing 문제를 해결 (양수에서)
- 현재 가장 많은 논문에서 사용되고 있는 함수

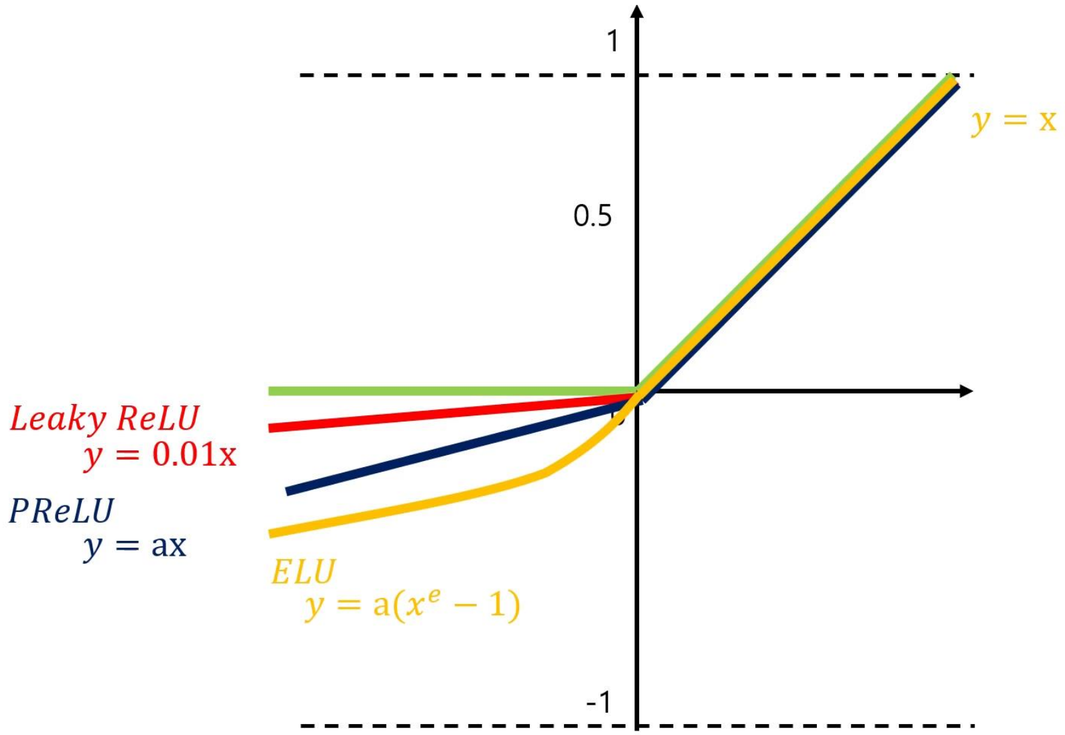
단점

- 입력 값이 음수일 때는 모든 값이 0으로 수렴
  = Dying ReLU 현상

활성화 함수의 종류

변형 ReLU 함수

=> 음수 입력값에 변화 줌



Leaky ReLU

- 음수 입력값에 작은 기울기 (0.1) 부여

PReLU

- 음수에 a라는 변수 부여
- 학습 모델에 따라 자유롭게 a값 설정 -> 학습 효율 높임

ELU

- 음수에 곡선이 들어간 지수 함수 사용

-> 보다 빠르고 정확한 결과

활성화 함수의 종류

지금까지는 은닉층, 이진분류(sigmoid)에 주로 쓰이는 활성화 함수들

Softmax 함수

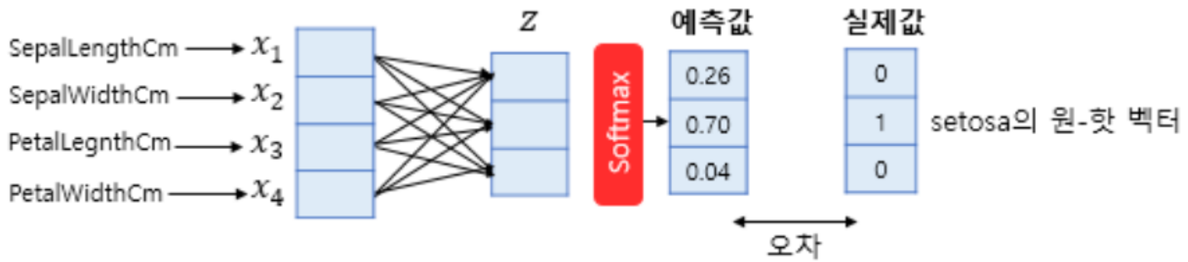
시그모이드와 비슷하게, 0~1사이로 변환하여 출력하지만,
출력값들의 합이 1이 되도록 하는 함수

$$p_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_j}} \qquad j = 1, 2, \ldots, K$$

특징

K: 클래스 수, z_j는 소프트맥스 함수의 입력값
p_j의 직관적으로 해석 = (j번째 입력값) / (입력값의 합)

예시



Optimizer의 정의

Optimizer란?

손실 함수를 최소화하기 위해 파라미터를 조정(갱신)하기 위한 방법

다양한 방법으로 발전되었으며, 모델 학습에 중요한 요소 중 하나

손실함수

신경망의 예측값과 실제 타겟 값 사이의 차이

신경망의 성능을 측정하는 지표

경사하강법

weight의 업데이트 = 에러 낮추는 방향 (decent) X 한발자국 크기 (learning rate) X 현 지점의 기울기 (gradient)

$$-\gamma \nabla F(\mathbf{a}^{\mathbf{n}})$$

$-$: 에러 낮추는 방향 (decent)

γ : 한발자국 크기 (learning rate) — 보폭

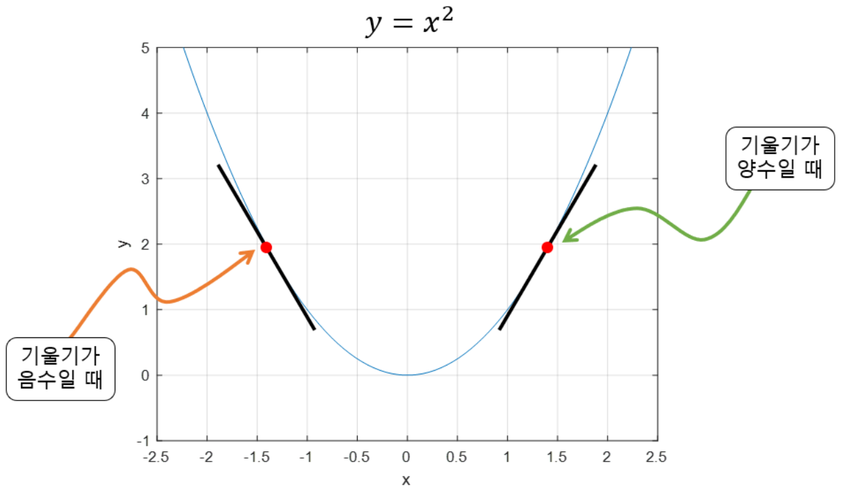
$\nabla F(\mathbf{a}^{\mathbf{n}})$: 현 지점의 기울기 (gradient) — 방향

손실 함수(오차)를 최소화하는 과정에서

핵심적인 역할을 하는 알고리즘

손실 함수의 현재 위치에서 기울기(미분값)를 계산,

기울기가 가장 급격하게 감소하는 방향(기울기 반대 방향)으로 파라미터 조정



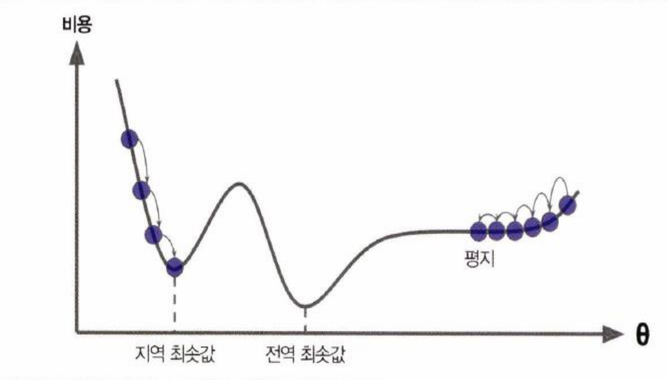
02. 최적화

경사하강법(Gradient Descent)

경사하강법 배경

| 저차원

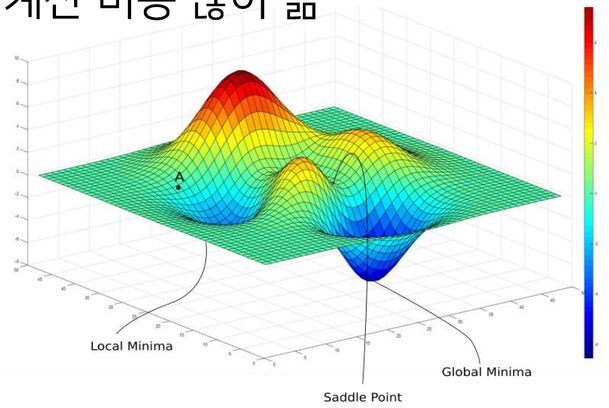
파라미터의 수가 적어,
손실 함수를 직관적으로 이해하고 계산하기가 수월



손실함수 그래프 1

| 고차원

솔루션을 알기 어려움
계산 비용 많이 듦



손실함수 그래프 2

경사하강법 단점

- 학습속도가 매우매우 느리다.
- 극솟값이나 안장점에서 학습이 중단될 수 있다.
- learning rate 결정 까다로움

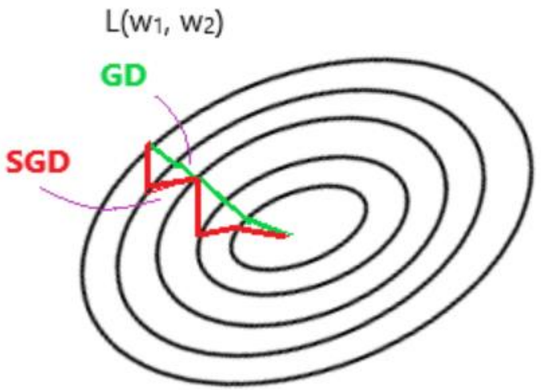
SGD (Stochastic Gradient Descent)

정의

Gradient Descent를 전체 데이터(Batch)가 아닌
일부 데이터의 모음(Mini-Batch)를 사용하는 방법

$$W(t+1) = W(t) - \alpha \frac{\partial}{\partial w} Cost(w)$$

α : 학습률, 하이퍼파라미터로써 기울기 갱신 속도 조절 가능



특징

- GD의 속도 문제 보완
  일부 데이터만을 사용하여 학습 속도 증가
- 오차가 굉장히 작으면서 계산 비용은 많이 줄여 준다

단점

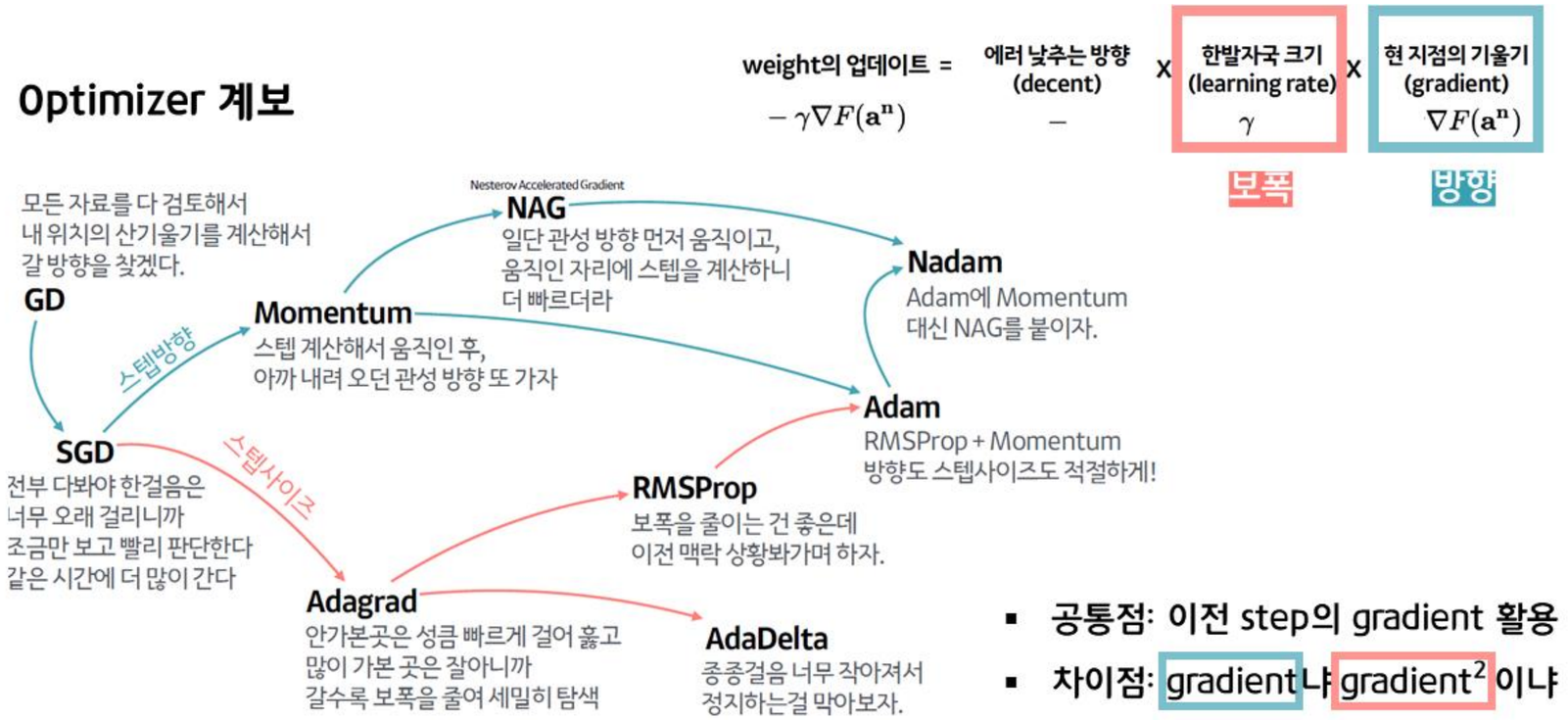
- 미니 배치를 기준으로 학습하기 때문에 비효율
  -> 샘플링으로 인한 지그재그 패턴
- 여전히 극소점에 수렴할 수 있다는 문제점 존재
- learning rate 결정 까다로움

GD, SGD의 개선

① 과거의 진행**방향**을 참고하는 관성(Momentum)을 사용함 ~ (ex) Momentum, NAG

② 이제까지 계산된 Gradient에 따라 다른 **학습률**(Adaptive Learing Rate)을 가지도록 함 ~ (ex) Adagrad, RMSProp

③ **관성**(Momentum)과 **스텝사이즈** 변화(Adaptive Learning Rate) 두가지를 다 고려함 ~ (ex) Adam



Momentum

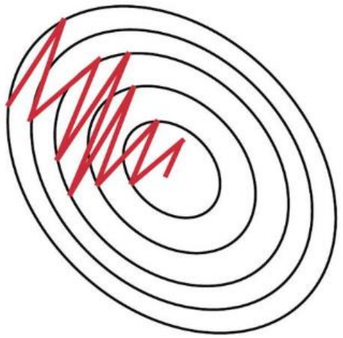
정의

$$W(t+1) = W(t) - \alpha \frac{\partial}{\partial w} Cost(w)$$

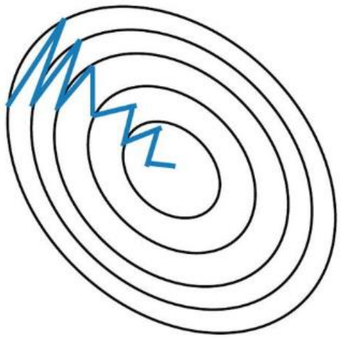
SGD에 관성(과거의 속도)을 더해 주는 방법
이전의 가중치의 수정 방향을 참고하여 갱신 방향을 결정

$$V(t) = m * V(t-1) - \alpha \frac{\partial}{\partial w} Cost(w)$$
$$W(t+1) = W(t) + V(t)$$

※ V(t): 속도, V(t-1): 관성, m: 관성계수



Stochastic Gradient Descent **without** Momentum



Stochastic Gradient Descent **with** Momentum

특징

- 관성이 붙어 극솟값이나 안장점에서 벗어날 수도 있음
- 이전의 갱신 값 고려하여 더욱 빠르게 갱신 가능
- 궤적이 크게 변동하지 않아, SGD보다 안정적으로 Gradient가 하강

단점

- 여전히 진동하기 때문에 최적해에 지그재그로
- 하이퍼파라미터(관성 계수) 추가 및 까다로움
  -> 최솟값 지나칠 수 있음
- learning rate 결정 까다로움

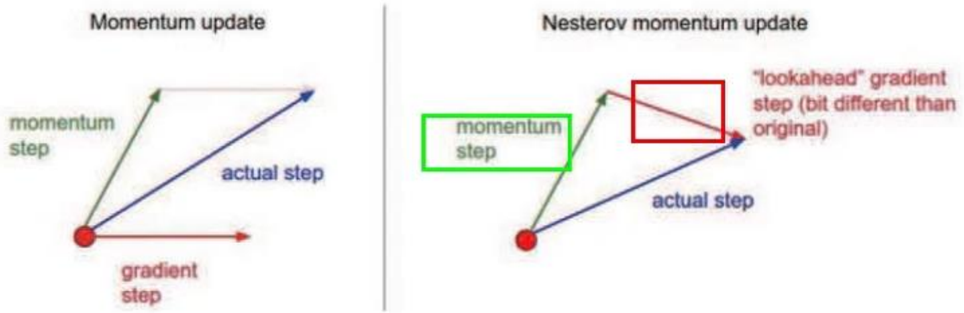
NAG (Nesterov Accelated Gradient)

정의

$$W(t+1) = W(t) - \alpha \frac{\partial}{\partial w} Cost(w)$$

Momentum 방식을 베이스로 하며
미리 경사를 확인할 수 있게 하고 경사에 맞춰 속도를 조절하는 방식

$$V(t) = m * V(t-1) - \alpha \frac{\partial}{\partial (w + m * V(t-1))} Cost(w)$$
$$W(t+1) = W(t) + V(t)$$



$$\mathbf{v}_n = \alpha \mathbf{v}_{n-1} + \eta \nabla f(\mathbf{x}_n + \alpha \mathbf{v}_{n-1}), \qquad \mathbf{v}_{-1} = \mathbf{0}$$
$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{v}_n$$

특징

- 모멘텀 방법에 비해 진동 폭 감소
- 모멘텀 값을 기준으로 기울기를 계산하기 때문에
  더 빠르게 최솟값에 접근

단점

- 하이퍼파라미터(관성 계수) 추가 및 까다로움
  -> 최솟값 지나칠 수 있음
- learning rate 결정 까다로움

AdaGrad (Adaptive Gradient)

정의

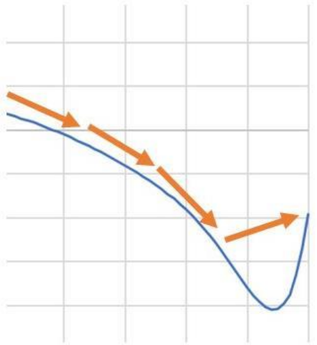
$$W(t+1) = W(t) - \alpha \frac{\partial}{\partial w} Cost(w)$$

SGD에서 학습률을 조절하여 기울기를 갱신하는 방법

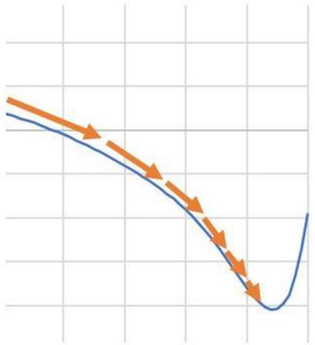
이제까지 계산된 Gradient에 따라 다른 학습률 조절

$$G(t) = G(t-1) + \left(\frac{\partial}{\partial w(t)} Cost(w(t))\right)^2 = \sum_{i=0}^{t} \left(\frac{\partial}{\partial w(i)} Cost(w(i))\right)^2$$

$$W(t+1) = W(t) - \alpha * \frac{1}{\sqrt{G(t)+\epsilon}} * \frac{\partial}{\partial w(i)} Cost(w(i))$$



SGD



AdaGrad

특징

- 과거의 업데이트된 총량을 기준으로 학습률 조정
- 학습률 조정을 통해 학습 속도 증가
- 극솟값에 수렴하지 못하고 발산하는 경우 방지

단점

- 과거의 모든 업데이트 총량을 고려하기 때문에
  시간이 지날수록 학습률이 점점 감소하여 0이 됨

RMSProp

정의

Adagrad에서 과거의 정보와 현재 정보의 비중을 조율하는 방법

감쇠를 계속 이어나갈지 VS 현재 것만 중요시 할지

$$G(t) = G(t-1) + \left(\frac{\partial}{\partial w(t)} Cost(w(t))\right)^2$$

$$G(t) = \gamma G(t-1) + (1-\gamma)\left(\frac{\partial}{\partial w(i)} Cost(w(i))\right)^2$$

$$W(t+1) = W(t) - \alpha * \frac{1}{\sqrt{G(t)+\epsilon}} * \frac{\partial}{\partial w(i)} Cost(w(i))$$

※γ: decay_rate (감쇠율)

특징

- 지수 이동평균을 활용하여 최근 값에 더욱 집중 가능
- 먼 과거의 정보는 잊기 때문에 지속하여 학습 가능

단점

- 초반에는 과거의 정보가 존재하지 않아 0으로 편향된 값 추정 발생

Adam (Adaptive Moment Estimation)

정의

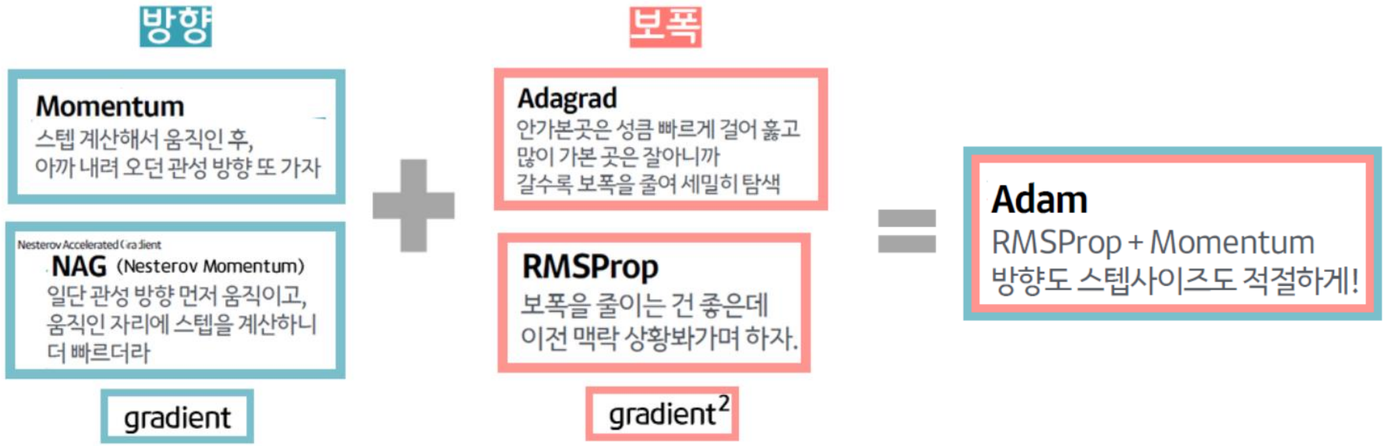
RMSProp + Momentum

$$M(t) = \beta_1 M(t-1) + (1-\beta_1)\frac{\partial}{\partial w(t)} Cost(w(t))$$

$$V(t) = \beta_2 V(t-1) + (1-\beta_2)\left(\frac{\partial}{\partial w(i)} Cost(w(i))\right)^2$$

$$\hat{M}(t) = \frac{M(t)}{1-\beta_1{}^t} \qquad \hat{V}(t) = \frac{V(t)}{1-\beta_2{}^t}$$

$$W(t+1) = W(t) - \alpha * \frac{\hat{M}(t)}{\sqrt{\hat{V}(t)} + \epsilon}$$



특징

- 초반 과거 정보 부족 문제를 해결, 최근 가장 많이 사용되는 방법
- 가장 최적화에 근사한 값에 도달

단점

- 계산 비용이 다른 방법보다 큼

03. 가중치 초기화

초기값 세팅의 중요성

가중치 초기화란?

모델을 본격적으로 학습하기 전, 모델에 존재하는 가중치를 초기화하는 것

가중치 최적화와 함께 모델 학습에 있어서 중요한 요소 중 하나

Random Initialization

정규분포를 활용하여 가중치를 초기화하는 방법

평균과 표준편차를 설정하여 가중치를 초기화

초기값 세팅의 중요성

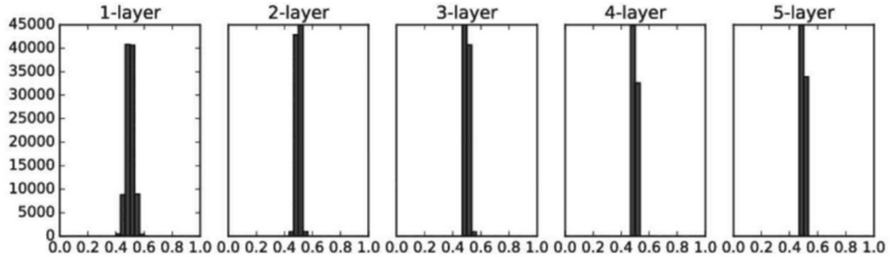
Random Initialization 시 가능한 문제점

| 표준 편차를 작게 잡으면

e.g) 표준편차 0.01 → 가중치 간의 차이가 거의 없음

→ 표현력 제한 문제

데이터의 정보를 담을 수 있는 파라미터의 숫자가 너무 적어서
가중치가 모든 뉴런에 대해 동일하거나 유사한 방식으로 갱신됨

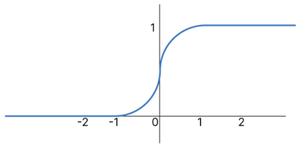


| 표준 편차를 크게 잡으면

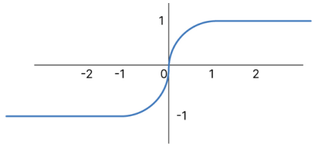
e.g) 표준편차가 1 → 가중치가 평균으로부터 멀리 떨어진 값일 경향이 큼

→ 활성화 함수별 문제가 다름

Gradient Vanishing

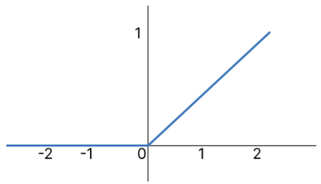


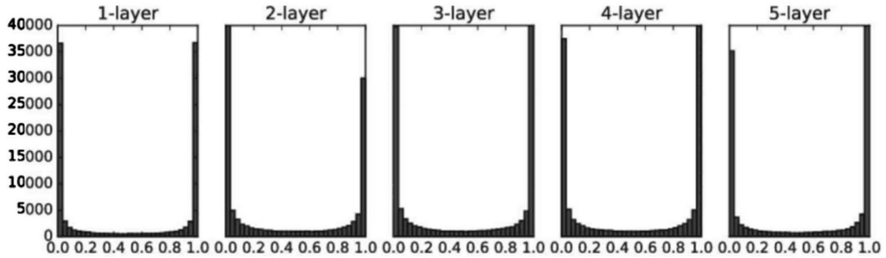
Sigmoid



tanh

Dying ReLU





가중치 초기화 방법

Xavier Initialization

Layer들의 입력/출력 노드 개수에 따라 파라미터값을 초기화

Sigmoid함수, tanh함수와 함께 주로 사용되는 초기화 방법

$$W \sim N(0, Var(W))$$

$$Var(W) = \sqrt{\frac{2}{n_{in} + n_{out}}}$$

He Initialization

ReLU 함수에 최적화된 파라미터 초기화 방법

$$W \sim N(0, Var(W))$$

$$Var(W) = \sqrt{\frac{2}{n_{in}}}$$

배치 정규화 (Batch Normalization)

Batch Normalization

Affine 층, 활성화 층 사이에 데이터의 분포를 컨트롤 해주는 Batch Norm(배치 정규화)층 을 끼워 넣는 것

이름과 같이 미니배치를 단위로 정규화 한다

$$BN(X) = \gamma\left(\frac{X - \mu_{BN}}{\sigma_{BN}}\right) + \beta$$

Batch Norm 층 역할

= 데이터의 평균과 표준 편차를 재설정

평균과 편차를 사람이 정해주는 게 아니라 ,

머신이 데이터로부터 가장 효율적 인 평균과 표준 편차를 스스로 학습

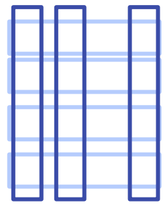
장점

- 학습 속도 개선
- 초기값 의존도 감소
- 오버피팅 억제 (드롭아웃 필요성 감소)

배치 정규화 (Batch Normalization)

Batch Norm 층 연산

$$BN(X) = \gamma\left(\frac{X - \mu_{BN}}{\sigma_{BN}}\right) + \beta$$



$x_1, x_2, \cdots, x_N$: 피쳐 개수 D인 데이터 N개

$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix}$: 배치 묶음, N×D 행렬

$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$: x의 열에 대한 평균, 길이 D인 벡터

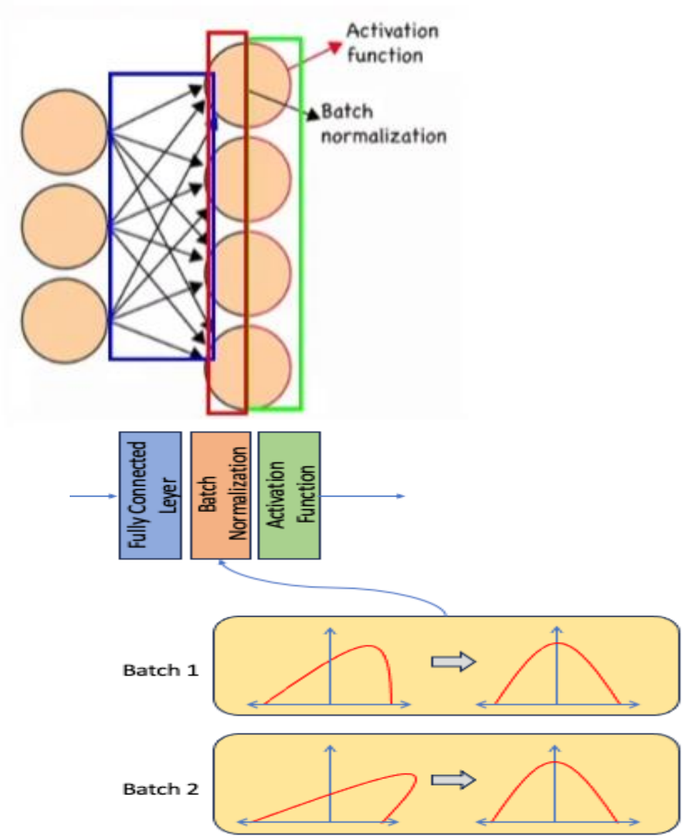
$\mathbf{x}_c = \mathbf{x} - \mu$: x의 각 열의 평균을 0으로 만듦, N×D 행렬

$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$: x의 각 열의 분산, 길이 D인 벡터

$\mathbf{x}_n = \frac{\mathbf{x}_c}{\sigma}$: x의 각 열을 normalize함, N×D 행렬

γ : 길이 D인 벡터, $\mathbf{x}_n$의 각 행에 곱해준다. (확장)

β : 길이 D인 벡터, $\mathbf{x}_n$의 각 행에 더해준다. (이동)

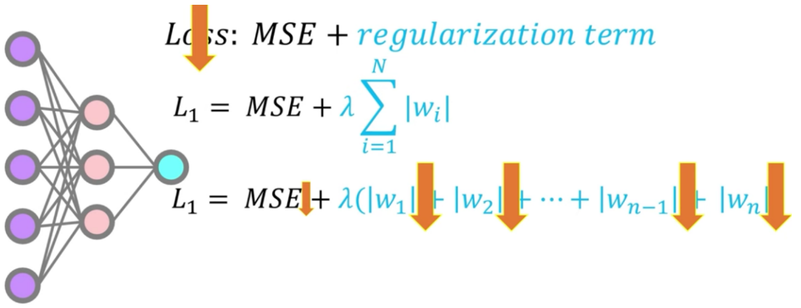


오버피팅 억제

가중치 감소 (Weight Decay)

큰 가중치에 비례하여 큰 패널티를 부여
모델이 특정 가중치에 지나치게 의존하는 것을 방지하는 원리
'가중치가 클수록 오차에 많은 영향을 미친다'

ex) L2 norm (제곱), L1 norm (절댓값)
가중치의 합을 줄여나가되, 손실함수 값도 최소가 되는 밸런스를 찾는 과정



드롭아웃

- 신경망 학습 시, 은닉층에 존재하는 뉴런의 일정 비율(하이퍼파라미터)을 사용하지 않고 학습하는 방법
- 오차역전파를 수행할 때에도 순전파 과정에 참여한 뉴런만 수행
- 특정 뉴런에 의존하지 않고 모든 뉴런을 공평하게 사용하도록 함

드롭아웃의 장점

- 서로 다른 뉴런들이 채택
  → 매번 다른 모델을 학습시키는 것으로 해석

: 앙상블 효과

Deep Session 3차시

코드 실습

2024_Deep_3주차_실습.ipynb를 열어주세요!

Deep Session 3차시

과제

과제 1

dropout, 활성화 함수, Initialization, Batch Normalization

부분 변경해보며 모델 성능 비교해보기

과제 2

활성화 함수와 경사하강법과 손실함수 개념을 이해하고

학습부터 역전파를 활용한 매개변수 갱신 과정을 이해한다.

REFERENCE

밑바닥부터 시작하는 딥러닝1 – 사이토 고키

https://sites.google.com/site/kyunghoonhan/deep-learning-i

https://youtu.be/iICZlmAhnf0?si=Oy0u8znDXZhQ4YAH

https://www.shiksha.com/online-courses/articles/activation-functions-with-real-life-analogy-and-python-code/

https://www.youtube.com/@phdshinAI

THANK YOU

Deep Session 3차시