

생성형AI

Day 13

딥러닝 I



목차

1. 딥러닝이란?
2. 딥러닝 기초
3. 딥러닝 원리
4. 딥러닝 과적합과 해결방법
5. 실습과제



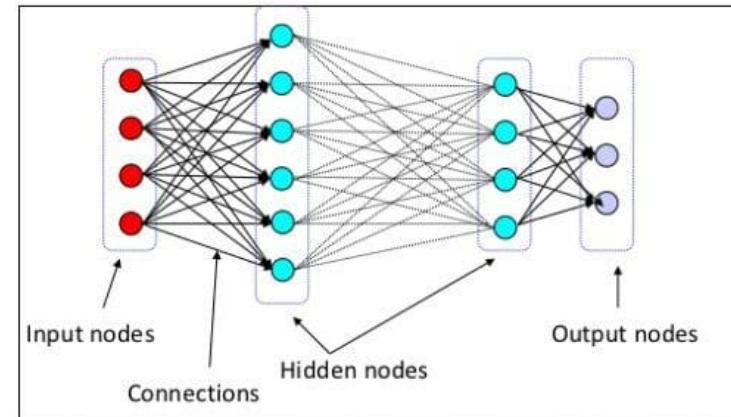
딥러닝(Deep Learning)

딥러닝?

- 딥러닝(Deep Learning)은 인공 신경망(ANN)을 기반으로 한 기계 학습(Machine Learning)의 한 분야
- 인간의 두뇌를 모방한 구조로, 다층 신경망을 사용하여 데이터로부터 특징을 학습하고 패턴을 인식하는 기술

딥러닝의 중요성?

- 딥러닝은 복잡한 데이터 분석과 패턴 인식에서 뛰어난 성능
- 자율주행차, 음성 인식, 이미지 분류, 자연어 처리 등 다양한 분야에서 활용
- 안정성 향상: 모델의 변동성을 줄이고, 예측의 일관성 향상



딥러닝(Deep Learning)

머신러닝 VS 딥러닝?

- 딥러닝은 머신러닝의 한 분야로, 다중 신경망을 사용하여 데이터를 자동으로 학습
- 딥러닝은 대규모 데이터와 복잡한 문제에 적합하며, 피처 엔지니어링을 최소화하고, 엔드 투 엔드(end-to-end) 학습을 수행

뭐가 다를까?

모델의 복잡성

- 머신러닝: 비교적 단순한 알고리즘(예: 선형 회귀, 의사 결정 트리)을 사용
- 딥러닝: 복잡한 모델을 사용하여 데이터의 깊은 패턴을 학습

데이터 요구 사항

- 머신러닝: 적은 양의 데이터로도 학습이 가능하지만, 데이터의 품질이 중요
- 딥러닝: 대량의 데이터가 필요하며, 데이터의 양이 많을수록 성능이 향상

특징 추출

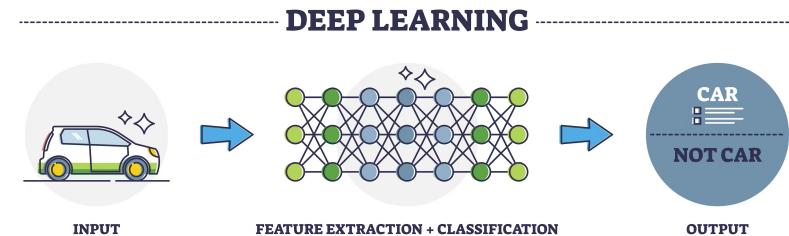
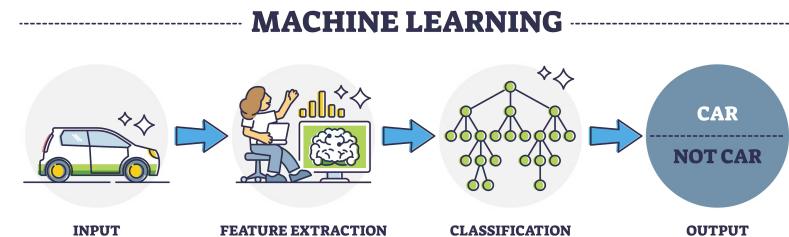
- 머신러닝: 수동으로 특징을 추출해야 하는 경우가 많음
- 딥러닝: 자동으로 특징을 추출하여 데이터의 중요한 부분을 학습

학습 속도

- 머신러닝: 상대적으로 빠르게 학습이 가능
- 딥러닝: 학습 시간이 오래 걸리며, GPU와 같은 고성능 하드웨어가 필요

성능

- 머신러닝: 적은 양의 데이터와 간단한 문제에 대해 좋은 성능
- 딥러닝: 대량의 데이터와 복잡한 문제에서 뛰어난 성능



딥러닝(Deep Learning)

초기 인공지능과 신경망 연구

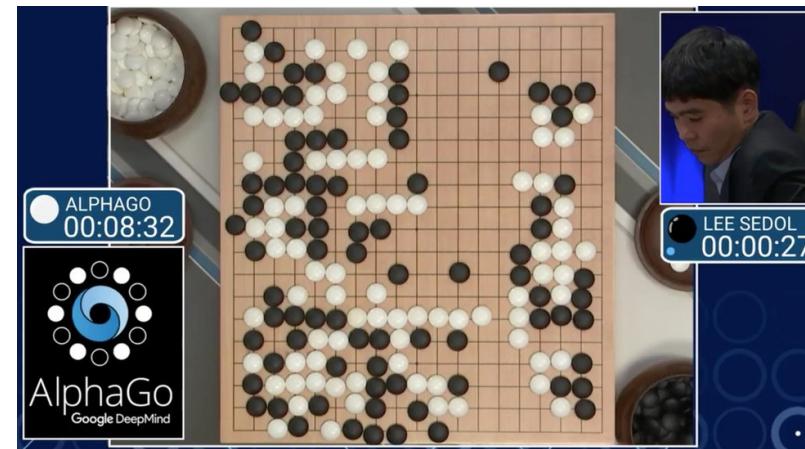
- 1940년대와 1950년대 초기 인공지능 연구에서 신경망의 개념이 처음 등장
- 퍼셉트론(Perceptron) 모델이 개발되었으나, XOR 문제 해결 불가능성 등으로 인해 관심이 줄어

신경망의 재부상과 딥러닝의 발전

- 1980년대와 1990년대 역전파 알고리즘(Backpropagation)의 개발로 신경망 연구가 재부상
- 2000년대 이후, 컴퓨팅 파워와 데이터 양의 증가로 딥러닝이 지속적으로 발전

이후 딥러닝 주요 사건

- AlexNet의 ImageNet 대회 우승
- AlphaGo의 바둑 대결



딥러닝 기초

인공 신경망 구조

뉴런

- 인공 신경망의 기본 구성 요소인 뉴런은 생물학적 뉴런과 유사
- 입력 신호를 받아 가중치와 함께 처리한 후, 활성화 함수를 통해 출력 신호를 생성
- 뉴런의 역할은 주어진 입력 데이터에서 특정 패턴이나 특징을 학습

구성요소

입력 (Input):

- 다른 뉴런이나 입력 데이터로부터 전달받는 값

가중치 (Weight):

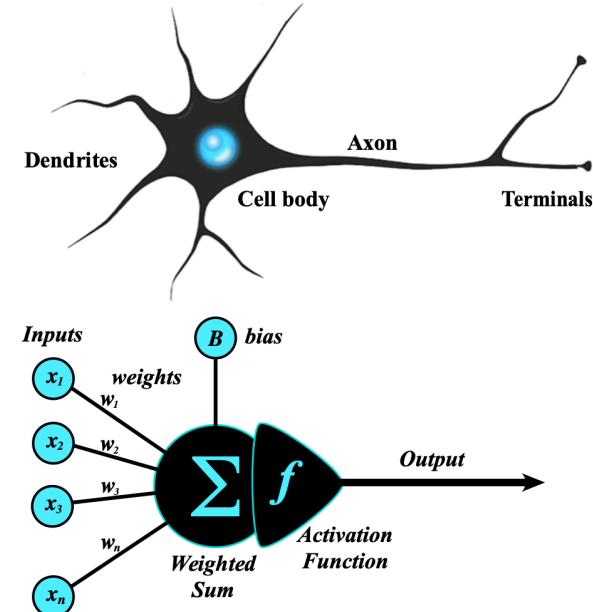
- 각 입력 신호에 곱해지는 값으로 입력 값의 중요도를 조절하는 역할

바이어스 (Bias):

- 출력 값에 더해지는 상수, 모델의 높은 유연성과, 다양한 출력을 생성하도록 조절하는 역할

활성화 함수 (Activation Function):

- 뉴런의 가중 합을 비선형 변환하여 최종 출력 값을 생성



딥러닝 기초

뉴런 작동 원리

1. 입력 신호 수집

여러 입력값 $x_1, x_2, x_3, \dots, x_n$ 받기

2. 가중치와 입력 값 곱셈

- 각 입력 값에 가중치를 곱함, $w_1, w_2, w_3, \dots, w_n$

3. 가중 합 계산

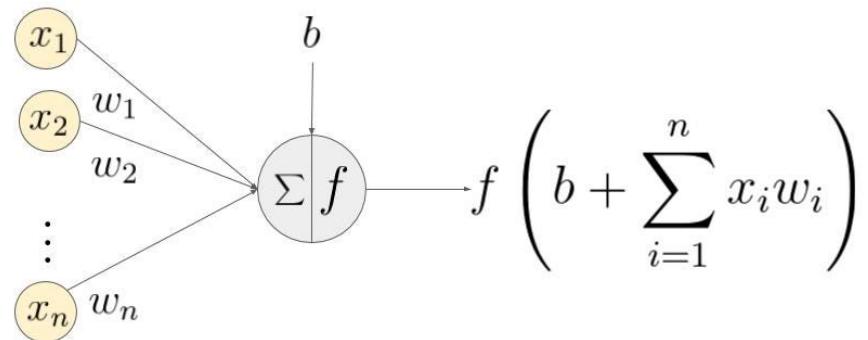
가중치와 입력 값을 곱한 후, 이 값을 모두 더함, $\sum_{i=1}^n w_i x_i$

4. 바이어스 추가

가중합에 바이어스 추가, $\sum_{i=1}^n w_i x_i + b$

5. 활성화 함수 적용

최종합에 활성화 함수 σ 를 적용하여 출력값을 계산, $y = \sigma(\sum_{i=1}^n w_i x_i + b)$



An example of a neuron showing the input ($x_1 - x_n$), their corresponding weights ($w_1 - w_n$), a bias (b) and the activation function f applied to the weighted sum of the inputs.

딥러닝 기초

활성화 함수 (Activation Function)

- 활성화 함수는 입력 신호의 선형 결합에 비선형성을 도입
- 신경망이 복잡한 패턴을 학습하고, 비선형 문제를 해결할 수 있게 함

활성화 함수 종류

- Sigmoid 함수: 출력 값을 (0, 1) 사이로 변환하여 이진 분류에 유용

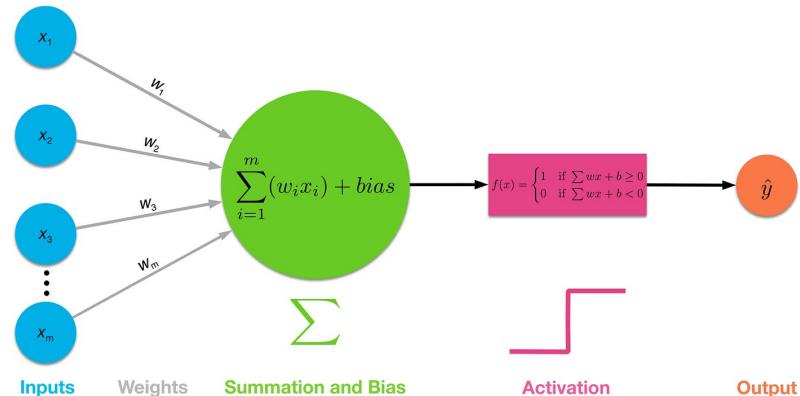
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- ReLU 함수: 기울기 소실 문제 완화 음수 값 0으로 변환, 양수 값은 그대로 반환

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

- Tanh 함수: 출력 값을 (-1, 1) 사이로 변환하여 신경망 학습에 유리

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



딥러닝 기초

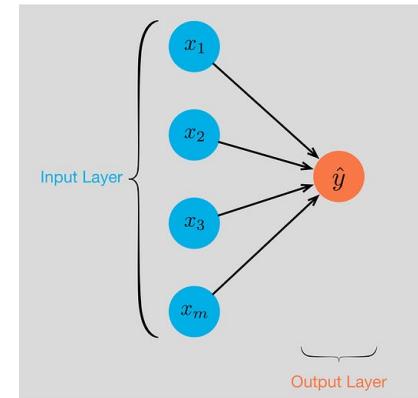
신경망의 구조

- 입력층(Input Layer): 원본 데이터를 받아들이는 층
- 은닉층(Hidden Layer): 입력 데이터를 처리하고 특징을 추출하는 층
- 출력층(Output Layer): 최종 결과를 출력하는 층

단층 신경망

- 단층 신경망은 입력층과 출력층으로만 구성된 신경망
- 단순한 문제 해결에는 유용하지만, XOR 문제와 같은 비선형 문제를 해결하지 못함

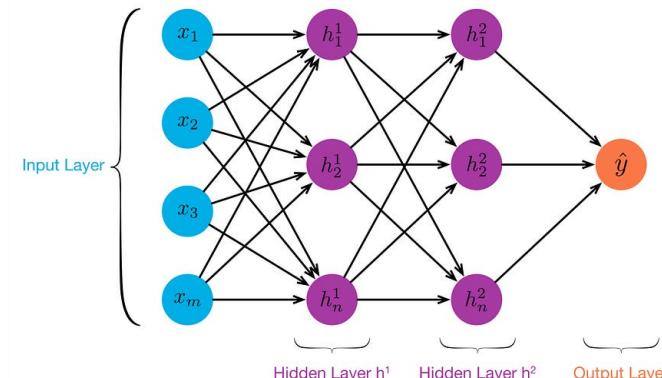
입력층 → 출력층



다층 신경망

- 다층 신경망은 입력층, 은닉층, 출력층으로 구성된 신경망
- 은닉층을 통해 비선형 문제를 해결할 수 있으며, 더 복잡한 패턴을 학습

입력층 → 은닉층(들) → 출력층



딥러닝 원리

딥러닝 학습 원리

학습과정

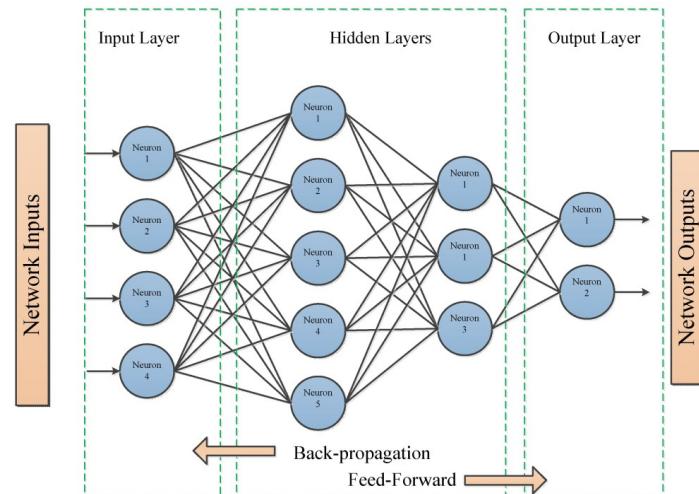
1. 순방향 계산(Feedforward process): 입력 데이터를 네트워크를 통해 전달하여 출력 값을 계산
2. 손실 함수(Loss Function): 예측값과 실제값의 차이를 계산
3. 역전파(Backpropagation): 손실 함수를 기준으로 가중치를 업데이트

학습방법

- 배치 학습: 전체 데이터를 한 번에 학습
- 미니 배치 학습: 데이터를 여러 작은 배치로 나누어 학습

최적화 알고리즘

- 경사 하강법(Gradient Descent)
- 확률적 경사 하강법(SGD)
- Adam



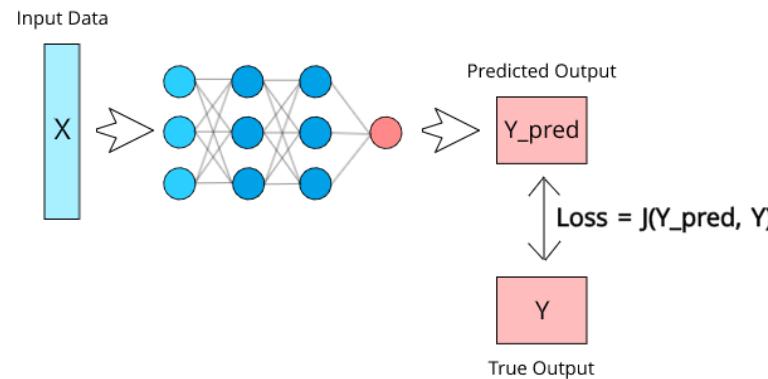
딥러닝 원리

손실함수(Loss Function)

- 손실 함수는 모델의 예측 값과 실제 값 간의 차이를 측정하는 함수
- 모델이 얼마나 잘 학습하고 있는지를 평가하는 기준

손실함수 역할

- 모델이 예측한 값과 실제 값의 차이를 줄이기 위해 사용
- 손실함수의 값이 작을수록 모델의 성능이 좋다는 것을 의미
- 손실함수를 최소화하는 방향으로, 학습 과정에서 가중치를 업데이트하는 데 중요한 역할



딥러닝 원리

교차 엔트로피 (Cross-Entropy)

- 교차 엔트로피는 예측 확률 분포와 실제 분포 간의 차이를 측정하는 손실 함수로 주로 분류 문제에 사용

$$\begin{aligned} \text{Cross - Entropy} &= -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \\ \text{Cross - Entropy} &= - \sum_{i=1}^n \sum_{j=1}^m y_{ij} \log(\hat{y}_{ij}) \end{aligned}$$

- y_i : 실제 값, \hat{y}_i : 예측 확률 값, n : 데이터 포인트의 수

교차 엔트로피 장점

- 예측된 확률과 실제 라벨 간의 차이를 직접적으로 측정하여 모델의 예측 성능을 정밀하게 평가 가능
- 확률 분포를 다루기 때문에 모델의 출력을 확률로 해석 가능

교차 엔트로피 단점

- 모델이 잘못된 예측을 할 때 큰 손실 값을 가지기 때문에, 초기 학습 단계에서 손실 값이 크게 나타날 수 있음

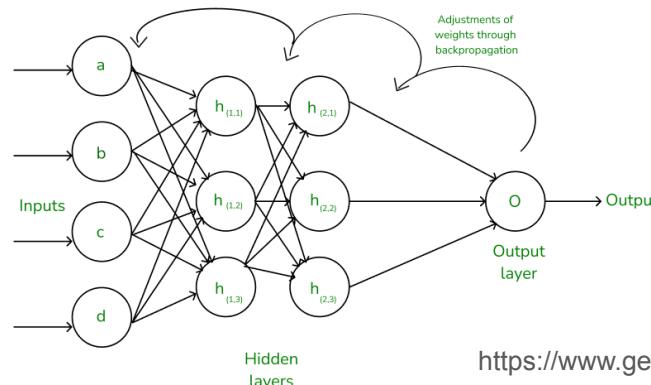
딥러닝 원리

역전파(Backpropagation)

- 역전파는 모델의 예측 오류를 줄이기 위해 가중치를 조정하는 방법
- 딥러닝 모델에서 가중치를 학습하고 최적화하는 데 핵심적인 역할
- 역전파를 통해 모델이 점점 더 정확한 예측을 할 수 있도록 가중치를 조정

최적화 알고리즘

- 최적화 알고리즘은 딥러닝 모델이 손실함수를 최소화하기 위해 가중치(weight)를 조정하는 방법
- 모델이 학습하면서 손실값을 줄이고, 예측 성능을 향상



딥러닝 원리

경사 하강법 (Gradient Descent)

- 경사 하강법은 손실 함수를 최소화하기 위해 반복적으로 가중치를 업데이트하는 최적화 알고리즘
- 경사 하강법은 가중치의 기울기를 계산하여 손실 함수가 가장 빠르게 감소하는 방향으로 이동

$$w = w - \alpha \nabla L(w)$$

- w 가중치, α 학습률, $\nabla L(w)$ 손실함수의 기울기
- 손실함수 L 의 기울기 $\nabla L(w)$ 를 계산하여 가중치 업데이트
- 적절한 학습률을 선택하는 것이 중요

경사 하강법의 종류

- 배치 경사 하강법: 전체 데이터셋을 사용하여 기울기를 계산하고 가중치를 업데이트
- 확률적 경사 하강법: 하나의 데이터 포인트를 사용하여 기울기를 계산하고 가중치를 업데이트
- 미니배치 경사 하강법: 데이터셋을 작은 배치로 나누어 기울기를 계산하고 가중치를 업데이트

딥러닝 원리

배치 경사 하강법(Batch Gradient Descent)

- 배치 경사하강법은 전체 데이터셋을 사용하여 손실함수의 기울기를 계산하고 가중치를 업데이트

배치 경사 하강법의 작동 원리

1. 전체 데이터셋에 대해 손실함수의 기울기를 계산
2. 계산된 기울기를 사용하여 가중치를 업데이트

$$w := w - \alpha \nabla L(w)$$

- w 가중치, α 학습률, $\nabla L(w)$ 손실함수의 기울기

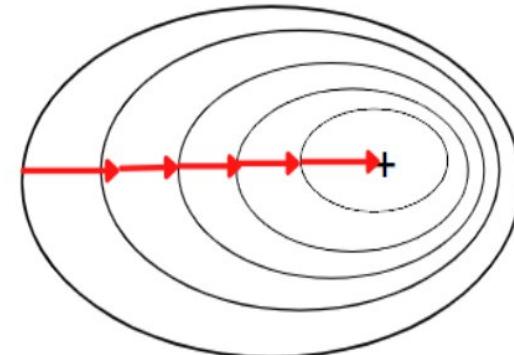
배치 경사 하강법의 장점

- 기울기 계산이 안정적이며, 수렴 과정이 안정적

배치 경사 하강법의 단점

- 계산 비용이 높고, 메모리 사용량도 높음

Batch Gradient Descent



딥러닝 원리

확률적 경사 하강법(Stochastic Gradient Descent, SGD)

- 확률적 경사하강법(SGD)은 하나의 데이터 포인트를 사용하여 손실함수의 기울기를 계산하고 가중치를 업데이트

확률적 경사 하강법의 작동 원리

1. 무작위로 선택된 하나의 데이터 포인트에 대해 손실함수의 기울기를 계산
2. 계산된 기울기를 사용하여 가중치를 업데이트

$$w := w - \alpha \nabla L(w; x_i, y_i)$$

- w 가중치, α 학습률, $\nabla L(w)$ 손실함수의 기울기, x_i 와 y_i 는 i 번째 데이터

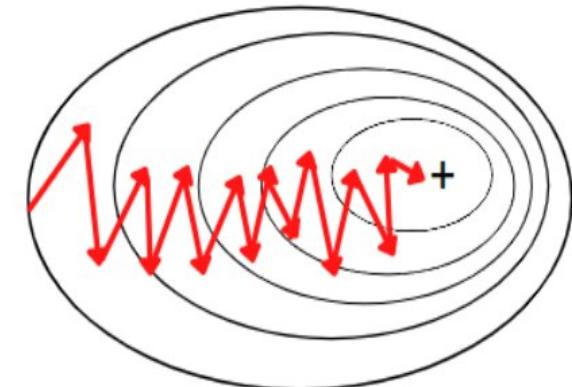
확률적 경사 하강법의 장점

- 계산이 빠르고, 메모리 사용량이 적음

확률적 경사 하강법의 단점

- 기울기가 불안정하고, 수렴 과정에서 진동 가능성 있음

Stochastic Gradient Descent



딥러닝 원리

미니배치 경사 하강법(Mini-Batch Gradient Descent)

- 미니배치 경사하강법은 전체 데이터셋을 작은 배치로 나누어, 각 배치에 대해 손실함수의 기울기를 계산하고 가중치를 업데이트

미니배치 경사 하강법의 작동 원리

1. 데이터셋을 크기 m 의 미니배치로 분할
2. 각 미니배치에 대해 손실함수의 기울기를 계산
3. 계산된 기울기를 사용하여 가중치를 업데이트

$$w := w - \alpha \nabla L(w; B)$$

- w 가중치, α 학습률, $\nabla L(w)$ 손실함수의 기울기, B 는 미니배치 데이터

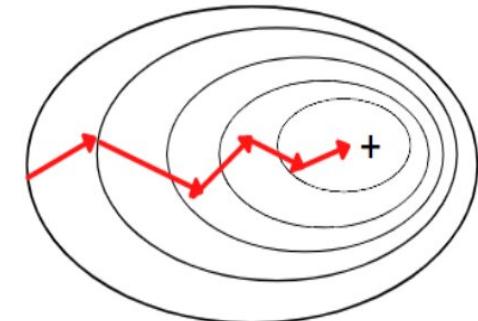
미니배치 경사 하강법의 장점

- 계산 효율성과 메모리 사용량의 균형을 잡을 수 있고, 수렴 과정이 비교적 안정적

미니배치 경사 하강법의 단점

- 배치 크기 선택이 어려울 수 있으며, 하이퍼파라미터 튜닝이 필요

Mini-Batch Gradient Descent



딥러닝 원리

미니배치 경사 하강법의 배치

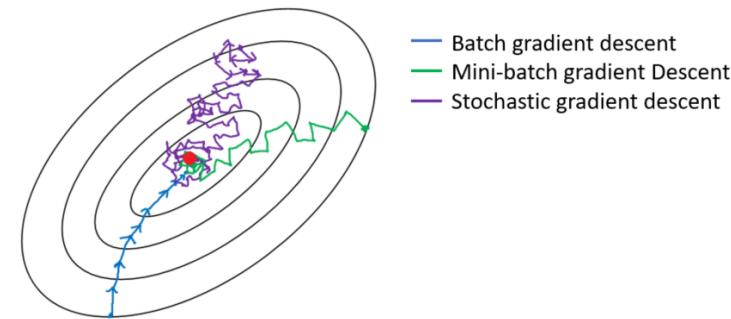
- 작은 배치 크기: 더 자주 업데이트가 이루어지며, SGD와 유사해지고 계산이 빠르지만 불안정해짐
- 큰 배치 크기: 더 안정적이고 배치 경사하강법과 유사해지고, 계산이 느리지만 안정적임

학습률

- 학습률이 너무 크면: 최적점 주변에서 진동하거나 발산할 가능성이 높음
- 학습률이 너무 작으면: 수렴 속도가 느리고, 최적점에 도달하는 시간이 길어짐

최적화 알고리즘의 비교

- 배치 경사하강법: 데이터셋이 작고, 높은 안정성이 필요한 경우
- 확률적 경사하강법: 데이터셋이 뭉쳐 크고, 빠른 계산이 필요한 경우
- 미니배치 경사하강법: 배치와 확률적 경사하강법의 중간 특성



딥러닝 원리

역전파의 한계

- 계산 비용

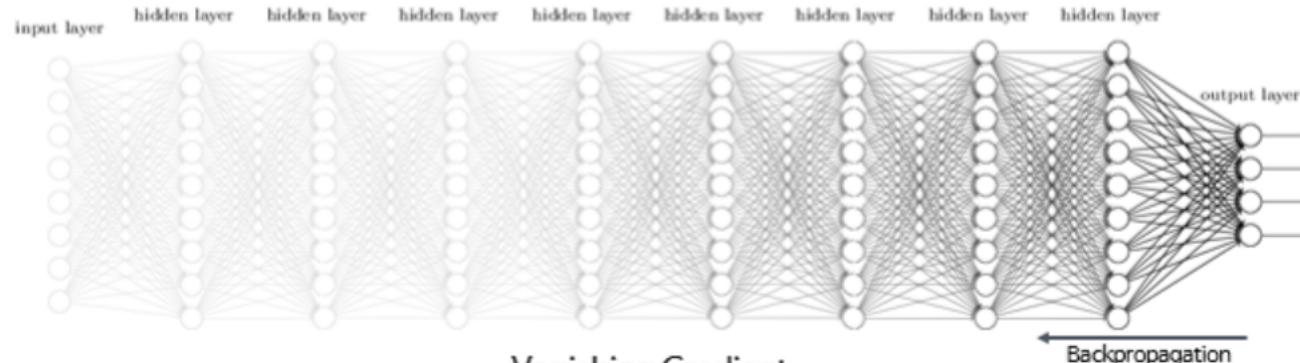
: 역전파 알고리즘은 계산 비용이 높아 큰 신경망에서 학습 시간이 길어질 수 있음

- 기울기 소실(Gradient Vanishing)

: 역전파 과정에서 기울기가 점점 작아져서 학습이 잘 되지 않는 현상

- 기울기 폭발(Gradient Exploding)

: 역전파 과정에서 기울기가 너무 커져서 학습이 불안정해지는 현상



Vanishing Gradient

<https://iambeginnerdeveloper.tistory.com/188>

딥러닝의 과적합과 해결 방법

과적합(Overfitting)

- 딥러닝은 모델이 복잡하고 사용하는 데이터셋이 큼
- 모델이 학습 데이터에 지나치게 적응하여 과적합될 가능성이 높음

과적합 방지 방법

- 데이터 증강(Data Augmentation): 학습 데이터를 다양하게 변형하여 데이터 양을 늘림
- 정규화(Regularization): 모델의 복잡성을 제한하여 과적합을 방지 (예: L1, L2 정규화)
- 드롭아웃(Dropout): 학습 시 무작위로 뉴런을 비활성화하여 모델의 일반화 능력을 향상

이론 실습

colab: https://colab.research.google.com/drive/1Dbcjfxkykc2uIReh_Bu2GV1To6cI0bdIN?usp=sharing

실습 과제

1. 기본 딥러닝 모델 확장

- 신경망(SimpleNN)을 확장하여 더 깊고 복잡한 모델을 만들어보기

2. 다른 데이터셋 사용

- MNIST 외의 다른 이미지 데이터셋으로 신경망 학습해보기

3. 학습률 최적화

- 학습률을 여러 값으로 설정하여 모델을 학습시켜보고, 각 학습률에서의 모델 성능을 비교해보기

4. 다양한 최적화 함수 사용해보기

- SGD, Adam, ...

실습 진행