딥러닝 (0203~0208)

📌 이번 주 학습 흐름 정리

1️⃣ 신경망(Neural Network)의 기본 개념 이해

- 신경망은 인간의 뇌를 모방한 모델로, 여러 개의 **퍼셉트론(Perceptron)**을 쌓아 만든 구조.
- 퍼셉트론은 기본적으로 입력을 받아 가중치를 곱하고 활성화 함수를 통해 출력을 내는 단위(뉴런).

2️⃣ 신경망 학습 방법 이해

- 신경망이 학습하는 과정에서 예측값과 실제값의 차이를 측정하기 위해 **손실 함수(Loss Function)**를 사용.
- 손실을 줄이기 위해 오차역전파법(Backpropagation)을 활용하여 가중치를 조정하는 방법을 배움.

▼ 오차역전파법

✅ 오차역전파법(Backpropagation) = 신경망이 잘못된 예측을 수정하는 학습 과정

✅ 순전파(Forward Propagation) = 예측값 계산

✅ 역전파(Backward Propagation) = 오차를 줄이기 위해 가중치 업데이트

- 학습을 더 효과적으로 하기 위해 **기울기 소실 문제**를 해결하는 **다양한 활성화 함수(예: ReLU, Sigmoid, Softmax)**를 적용.

3️⃣ 신경망의 학습 최적화 및 초기화 방법 이해

- 신경망이 잘 학습하도록 하기 위해 **가중치 초기화 기법(He 초기화, Xavier 초기화)**을 사용.
- 학습 속도를 높이고, 안정적으로 학습하기 위해 Adam, SGD 등 다양한 Optimizer(최적화 알고리즘)도 배움.

- 배치 정규화(Batch Normalization)를 활용하여 학습을 더 효율적으로 만드는 방법도 학습.

4️⃣ 합성곱 신경망(CNN)으로 확장

- 기존 신경망(DNN)은 숫자 데이터를 처리하는 데는 좋지만, **이미지 처리에는 한계**가 있었음.
- 그래서 CNN(Convolutional Neural Network, 합성곱 신경망)을 배워서 이미지의 **특징을 자동으로 추출**하는 방법을 익힘.
- CNN에서는 합성곱 계층(Convolution Layer) + 풀링 계층(Pooling Layer) + 완전연결층(Fully Connected Layer)로 구성된 **구조**임.
- 기존 신경망은 모든 뉴런을 연결(완전연결층)하는 방식이지만, CNN은 **필터(커널)를 이용해 이미지의 중요한 부분을 자동으로 찾는 방식임**.

5️⃣ 전이학습(Transfer Learning)과 ResNet 학습

- CNN을 직접 학습시키는 것도 좋지만, **기존에 학습된 모델(ResNet, VGG 등)을 활용하면 학습 시간을 단축**할 수 있음.
- 전이학습을 사용하면 **기존 모델의 특징을 가져와서 새로운 이미지 분류 모델을 빠르게 만들 수 있다**는 걸 배움.

▼ 전이학습

기본적인 패턴을 재사용해, 모델의 일부를 새롭게 학습하는 것임.

- Resnet : 딥러닝에서 사용되는 신경망 모델.
  기존 cnn? 기울기 소실 문제 발생했는데 Resnet은
  Residual Block을 도입하면서 이를 완화함

▼ Residual Block이 뭔데?

- 일반 CNN은 **입력 → 합성곱 → 활성화 함수(ReLU) → 다음 층**의 구조
- ResNet은 여기에 **잔차 연결(Residual Connection)**을 추가함

✔ **잔차 연결(Residual Connection)이란?**

- **입력값을 그대로 다음 층으로 더해주는 구조**
- 즉, F(x)를 계산한 결과에 원래 입력값 x를 더하는 방식

- 수식으로 표현하면:
  $y=F(x)+x$
- 여기서 F(x)는 CNN을 거쳐 변화한 값, x는 원래 입력값

✔️ 이 방식의 장점?

- 기울기가 소실되지 않고 **초기층까지 더 잘 전달됨**
- 역전파 과정에서 x의 기울기는 항상 **1로 유지되므로 학습이 잘 진행됨**
- 깊은 신경망에서도 **기울기 값이 0으로 사라지지 않음**

6️⃣ CNN을 활용한 실제 이미지 분류 실습 (암석)

- NASA의 달 암석 데이터를 활용해서 현무암 vs 고지대 암석을 분류하는 모델을 만들어 봄.
- 기존 신경망보다 CNN이 더 빠르고 정확하게 이미지 분류를 수행하는 걸 **확인**함.

📌 딥러닝 핵심 개념 정리

(👉 "이게 뭔데?" + "왜 하는데?"까지 포함!)

---

1. 퍼셉트론 (Perceptron)

📌 이게 뭔데?

- 인간 뉴런을 모방한 가장 간단한 인공 신경망 모델
- **입력 값(x) × 가중치(w) + 편향(b) → 활성화 함수 → 출력(y)**
- 특정 기준(임계값 θ) 이상이면 1을 출력, 아니면 0을 출력하는 방식

📌 왜 하는데?

- 논리 연산(AND, OR, NAND 등)을 수행할 수 있음
- 선형 분리가 가능한 문제를 해결하는 데 사용됨
- 하지만 XOR 문제는 해결 못함 → 다층 퍼셉트론(MLP) 필요!

---

2. 논리 게이트 (AND, OR, NAND, XOR)

📌 **이게 뭔데?**

- **AND 게이트**: 두 입력이 모두 1일 때만 1 출력
- **OR 게이트**: 입력 중 하나라도 1이면 1 출력
- **NAND 게이트**: AND의 반대, 둘 다 1이면 0 출력
- **XOR 게이트**: 입력이 서로 다를 때만 1 출력

📌 **왜 하는데?**

- 기본적인 논리 연산을 신경망으로 표현할 수 있음
- XOR은 단일 퍼셉트론으로 해결 불가능 → MLP가 필요!

---

3. 활성화 함수 (Activation Function)

📌 **이게 뭔데?**

- 뉴런의 출력 값을 결정하는 함수
- 입력 값이 특정 조건을 만족하면 신호를 전달

📌 **왜 하는데?**

- 신경망에비선형성(Non-linearity)을 추가해 복잡한 패턴 학습 가능
- 대표적인 활성화 함수:
  - **계단 함수 (Step Function)**: 0과 1만 출력 (비추천)
  - **시그모이드 (Sigmoid)**: 부드러운 곡선, 0~1 범위 (기울기 소실 문제)
  - **ReLU (Rectified Linear Unit)**: 0 이상만 전달, 계산 효율적 (가장 많이 사용됨)

---

4. 다층 퍼셉트론 (MLP)

📌 **이게 뭔데?**

- 여러 개의 퍼셉트론을 연결한 신경망 모델
- **입력층(Input) → 은닉층(Hidden) → 출력층(Output)** 구조

📌 **왜 하는데?**

- XOR 문제처럼 **비선형 문제** 해결 가능
- 은닉층이 많을수록 복잡한 패턴을 학습할 수 있음

---

5. Softmax 함수

📌 **이게 뭔데?**

- 출력 값을 확률(0~1)로 변환하는 함수
- 모든 출력 값의 합이 1이 되도록 조정

📌 **왜 하는데?**

- *분류 문제(Classification)**에서 사용됨
- 예를 들어, 손글씨 숫자 인식에서 "이 숫자가 7일 확률은 80%"처럼 확률을 출력

---

6. 수치 미분 (Numerical Differentiation)

📌 **이게 뭔데?**

- 미분을 수식이 아니라 **근사값**으로 계산하는 방법

📌 **왜 하는데?**

- 신경망의 가중치를 조정하려면 **기울기(Gradient)**를 계산해야 함
- 해석적으로 미분하기 어려운 경우 수치 미분을 사용

---

7. Affine 변환

📌 **이게 뭔데?**

- **행렬 곱 + 편향 추가**로 입력 값을 변형하는 과정
- 수식:

  $y=Wx+by = Wx + b$

📌 **왜 하는데?**

- 신경망에서 데이터를 변형해 학습할 수 있도록 함

- 은닉층에서 입력을 새로운 공간으로 변환하는 역할

---

8. 오차역전파 (Backpropagation)

📌 **이게 뭔데?**

- 신경망이 학습하는 핵심 알고리즘
- 출력층에서 발생한 **오차를 역방향(뒤에서 앞으로)으로 전파**해 가중치를 업데이트

📌 **왜 하는데?**

- 신경망이 스스로 최적의 가중치를 찾을 수 있도록 함
- 경사하강법(Gradient Descent)과 함께 사용됨

---

9. 순전파 (Forward Propagation)

📌 **이게 뭔데?**

- **입력 → 가중치 곱 → 활성화 함수 → 출력** 과정

📌 **왜 하는데?**

- 데이터를 모델에 넣어서 예측을 하기 위해
- 순전파 결과를 보고 오차를 계산한 후, 역전파를 수행

---

10. 역전파 (Backward Propagation)

📌 **이게 뭔데?**

- 오차(손실)를 줄이기 위해 가중치를 수정하는 과정
- **출력층 → 은닉층 → 입력층 방향**으로 진행

📌 **왜 하는데?**

- 신경망이 학습하려면 가중치를 조정해야 함
- 역전파를 통해 가중치를 업데이트하면 모델이 점점 더 정확해짐

▼ 1~10 최종 정리

개념	설명	왜 하는데?

퍼셉트론	가장 기본적인 신경망 모델	간단한 분류 문제 해결 (XOR은 해결 못함)
AND/OR/NAND/XOR 게이트	논리 연산을 수행하는 퍼셉트론 모델	컴퓨터의 기본 논리를 신경망으로 표현 가능
활성화 함수	입력을 변형해 출력하는 함수	신경망에 비선형성을 추가해 더 복잡한 패턴 학습 가능
다층 퍼셉트론 (MLP)	여러 퍼셉트론을 연결한 신경망	XOR 문제 해결 가능, 더 깊은 학습 가능
Softmax 함수	출력 값을 확률(0~1)로 변환	분류 문제에서 확률 기반 예측 수행
수치 미분	미분을 근사적으로 계산	신경망 학습에서 기울기 계산에 사용
Affine 변환	행렬 곱 + 편향 추가	데이터를 변형해 신경망이 학습할 수 있도록 함
오차역전파	신경망이 학습하는 알고리즘	가중치를 수정해 정확도를 높임
순전파	입력 → 가중치 → 출력 과정	예측을 수행하는 과정
역전파	오차를 역방향으로 전파해 가중치 수정	모델을 학습시키기 위해 필수

📌 (2탄) 가중치 초기화부터 합성곱 신경망까지

1️⃣ 활성화 함수 (Activation Functions)

🔷 ReLU 함수 (Rectified Linear Unit)

- **설명**: 0보다 크면 그대로 출력, 0 이하는 0으로 출력.
- **이유**: 시그모이드보다 학습이 잘 되고, 깊은 신경망에서도 효과적.
- **비유**: 출발 신호(양수)가 있어야 자동차(뉴런)가 움직이고, 없으면 멈춘다.

🔷 시그모이드 함수 (Sigmoid)

- **설명**: 입력을 0~1 사이 값으로 변환.
- **이유**: 확률적인 문제(이진 분류)에 유용하지만, 값이 0 또는 1에 가까우면 학습이 어려움.
- **비유**: 빛의 밝기를 0~100% 사이에서 조절하는 것.

🔷 탄젠트 함수 (Tanh)

- **설명**: 입력을 -1~1 사이 값으로 변환.
- **이유**: 시그모이드보다 중앙값이 0이라 학습이 좀 더 잘됨.
- **비유**: 온도 조절기 (-100도~100도)처럼 양방향 조절 가능.

2️⃣ 가중치 초기화 (Weight Initialization)

🔷 std=0.01

- **설명**: 가중치를 평균 0, 표준편차 0.01인 작은 값으로 설정.
- **문제점**: 값이 너무 작아 학습이 거의 진행되지 않음.

🔷 Xavier 초기화

- **설명**: **시그모이드, Tanh** 같은 활성화 함수에 적합한 초기화 방법.
- **이유**: 층이 깊어질수록 데이터가 적당히 분포하도록 가중치를 조절.
- **비유**: 각 층이 적절한 양의 데이터를 받을 수 있도록 퍼뜨리는 역할.

🔷 He 초기화

- **설명**: **ReLU 함수**에 적합한 초기화 방법.
- **이유**: 값이 더 넓게 분포되도록 조정해 학습을 원활하게 함.
- **비유**: ReLU는 0 이하에서 죽으니까, 처음부터 더 큰 값으로 설정해 학습을 잘 되게 함.

번외) **뉴런이 죽는다?**

Relu의 경우, 음수를 0으로 만든다.

⇒출력이 0이면, 역전파 과정에서 기울기가 0이 된다.

⇒ 기울기가 0이면 가중치 업데이트가 되지 않고,

⇒ 해당 뉴런은 죽는다.

3️⃣ 에포크 (Epoch)

- **설명**: **훈련 데이터를 한 바퀴 다 학습하는 횟수**.

- **비유**: 책 한 권을 다 읽고 복습하는 횟수.
- **주의**: 너무 많으면 오버피팅, 너무 적으면 과소적합.

4️⃣ 파라미터 vs 하이퍼파라미터

🔷 파라미터 (Parameter)

- **설명**: 모델이 학습을 통해 자동으로 찾는 값. (예: 가중치, 편향)
- **비유**: 문제를 풀면서 직접 찾아내는 해답.

🔷 하이퍼파라미터 (Hyperparameter)

- **설명**: 사람이 직접 설정해야 하는 값. (예: 학습률, 층의 개수, 뉴런 수, 배치 크기)
- **비유**: 시험 공부 방법(공부 시간, 문제 유형 선택)을 사람이 조정하는 것.

5️⃣ 필터 (Filter)

- **설명**: CNN에서 특정한 특징을 찾아내는 작은 행렬(커널).
- **이유**: 이미지에서 중요한 패턴(모서리, 선, 질감 등)을 감지하는 역할.
- **비유**: 돋보기처럼 이미지에서 특정한 부분을 강조하는 것.

6️⃣ 패널티 (Penalty)

- **설명**: 가중치가 너무 커지지 않도록 **규제를 걸어주는 기법**.
- **이유**: 오버피팅을 막기 위해 사용됨.
- **비유**: 너무 긴 논문을 간결하게 정리하는 것.

7️⃣ Stride (스트라이드)

- **설명**: **필터가 이미지를 이동하는 간격**.
- **비유**: 걸음걸이 크기. (크면 빠르게 지나가지만, 자세한 정보를 놓칠 수 있음)

8️⃣ CNN (합성곱 신경망)

🔷 CNN이란?

CNN(Convolutional Neural Network)

- **구조:** 기존 DNN과 다르게 **합성곱 계층(Convolution Layer) + 풀링 계층(Pooling Layer) + Fully Connected Layer**로 구성됨.
- **특징:**
  - 이미지의 공간적 특징(Spatial Features)을 유지하면서 학습 가능.
  - 합성곱 연산(Convolution)과 풀링(Pooling)을 통해 **특징 추출(feature extraction) → 차원 축소 → 최종 분류** 과정 진행.
  - **데이터 전처리 없이도 패턴을 자동으로 학습**할 수 있어 이미지 처리에 강점.

▼ 기존 DNN?

DNN(Deep Neural Network)

- **구조:** Fully Connected Layer(FCL) 기반의 신경망.

▼ FCL?

직역하면, 완전 연결 계층. 신경망 내 모든 뉴런이 서로 연결된 구조를 갖는 네트워크. 이를 MLP(Multi-Layer Perceptron, 다층 퍼셉트론)이라고도 한다.

▼ MLP

다층 퍼셉트론, 즉 여러 개의 은닉층을 가진 신경망.

완전 연결 형태인 FC (Fully Connected) 구조를 가짐.

입력층, 은닉층, 출력층으로 구성. (각 층이 연결. 왜? FC니까!)

1️⃣ FCL 구조

FCL 신경망은 다음과 같은 계층(Layer)으로 구성됨.

1. **입력층(Input Layer)**
   - 데이터를 받아들이는 층.
   - 예를 들어, 이미지(28×28)를 입력하면 1차원으로 변환하여 784개의 뉴런(28×28 = 784)으로 표현.

2. **은닉층(Hidden Layer, 1개 이상 존재)**
    - 입력층과 출력층 사이에서 연산을 수행하는 층.
    - 뉴런마다 **가중치(Weight)와 바이어스(Bias)**가 존재하며, 활성화 함수(Activation Function)를 적용하여 비선형성을 부여.
    - 일반적으로 **ReLU, Sigmoid, Tanh** 같은 활성화 함수 사용.
3. **출력층(Output Layer)**
    - 최종 예측 값을 출력하는 층.
    - 분류 문제의 경우, **Softmax(다중 분류) 또는 Sigmoid(이진 분류)**를 사용하여 결과 도출.

- **특징:**
    - 데이터를 **1차원 배열로 변환** 후 학습.
    - **이미지 데이터(2D)를 1D로 변환**해야 하므로 **공간적 상관관계(Spatial Relationship)**가 손실됨.
    - 고차원의 데이터를 처리할 수 있지만, **연산량이 많고 학습 속도가 느림**.
    - 이미지에서 중요한 부분을 효과적으로 인식하지 못하는 문제가 있음.

🔷 CNN의 장점

- 이미지의 공간 정보를 유지하면서 학습.
- 필터를 통해 중요한 특징을 자동으로 추출.
- **비유**: 사람의 시각적 인식 과정과 유사함.

▼ 1차원 데이터와 2차원 데이터의 차이

FC-DNN과 CNN의 차이로 봐도 무방.

1차원은 점, 2차원은 평면 (이미지와 같은 데이터)

CNN에서는 이 2D 구조(가로 x 세로 픽셀)를 유지하면서 학습.

단, CNN은 DNN의 한 종류임을 인지할 것.

DNN은 신경망 전체를 의미하는 큰 개념이고, CNN은 그 중 하나.

다시말해, DNN = 은닉층이 2개 이상 있는 '신경망'

CNN은 DNN의 한 종류로, 주로 이미지 처리에 특화됨.

기존 DNN(MLP)와 달리, 모든 뉴런을 FC(fully connected)하지 않음.

▼ 합성곱계층

💡

합성곱 계층(Convolutional Layer)은 이미지에서 중요한 특징(Feature_모서리, 패턴 등)을 자동으로 추출하는 층이다.

필터를 사용해 입력 데이터를 학습 가능한 형태로 변환하는 과정이기도 하다.

1️⃣ 왜 필요한가? (FC 신경망의 한계)

- *기존 DNN(MLP)**은 이미지를 **1차원 벡터로 변환**해야 했기 때문에 **픽셀 간의 공간적 관계(Spatial Relationship)가 사라짐**.
- CNN은 **이미지의 공간 구조를 유지하면서도 학습할 수 있도록 설계**됨.
- 핵심 원리는 **필터(Filter, Kernel)를 이용해 이미지에서 중요한 특징을 찾아내는 것**.

2️⃣ 합성곱 계층의 핵심 개념

✅ ① 필터(Filter, Kernel) 적용

- 작은 **행렬(예: 3×3, 5×5 크기)**을 사용하여 이미지를 훑으면서 중요한 패턴을 찾아냄.
- 필터는 학습을 통해 자동으로 조정됨.

✅ ② 스트라이드(Stride) 조절

- 필터가 한 번에 이동하는 거리.
- **Stride가 크면** 연산량이 줄지만, **자세한 정보가 손실**될 수 있음.

✅ ③ 패딩(Padding) 적용

- 이미지의 가장자리를 보존하기 위해 **0 값을 추가**하여 크기를 유지.

✅ ④ 활성화 함수(ReLU) 적용

- 비선형성을 추가하여 **복잡한 패턴을 학습**할 수 있도록 함.

▼ 풀링계층

💡

풀링 계층은 이미지 크기를 줄이면서 중요한 정보를 유지하는 역할을 한다.

▼ 여기서 중요한 정보?

A. 눈에 띄는 패턴 = 엣지(경계선), 윤곽, 패턴 등 혹은

이미지 내에서 밝기가 급격히 변하는 부분 (=경계, 특징이 있는 부분)

1️⃣ 풀링 계층의 역할

✅ ① 특징을 유지하면서 차원 축소 (Downsampling)

- 풀링을 사용하면 **데이터 크기를 줄여 연산량을 감소**시킴.
- 하지만, 중요한 특징(Feature)은 유지됨.

✅ ② 모델의 일반화 성능 향상

- 특정 픽셀 값에 너무 의존하지 않도록 함 → **오버피팅 방지**.
- 이미지가 조금 회전되거나 이동해도 같은 특징을 유지할 수 있도록 만듦.

✅ ③ 계산 속도 향상

- 뉴런의 개수가 줄어들어 연산량이 감소 → 학습 속도가 빨라짐.

📌 (3탄) PyTorch vs TensorFlow 차이점

특징	**PyTorch**	**TensorFlow**
개발사	Meta (Facebook)	Google
사용 방식	동적 그래프 (Dynamic Graph)	정적 그래프 (Static Graph)
학습 과정	직관적인 Pythonic 코드	그래프를 미리 정의해야 함
디버깅	디버깅이 쉬움 (Python 코드처럼 실행)	실행 속도가 빠름
배포(Serving)	상대적으로 불편	**TensorFlow Serving 지원 (배포에 강함)**
모바일 지원	가능하지만 TensorFlow보다 적음	**TensorFlow Lite로 모바일에서 강점**

TPU 지원	지원하지만 상대적으로 적음	**Google TPU 지원 (대규모 학습에 강함)**

추가) pytorch는 모델 돌리면서 weight 꺼내볼 수 있지만, tensorflow는 그렇지 못함.

📌 그럼, TensorFlow와 PyTorch. 언제 사용하면 좋을까?

✅ PyTorch 추천

- 연구 목적 (Deep Learning 연구, 논문 구현)
- 모델을 빠르게 만들고 실험할 때
- 코드가 직관적이라 배우기 쉬움

✅ TensorFlow 추천

- 실제 제품 배포 (Production)
- 대규모 학습, TPU 사용
- 모바일, 웹에서 모델을 실행할 때

💡 정리하면!

- **연구/개발** → PyTorch
- **배포/대규모 학습** → TensorFlow