스터디 **8**주차 중간정리본**(1~3**주차 강의**)**

홍송은

1주차는 기본적으로 파이썬 활용, 텐서 이해 등 기본 내용이 대부분이라서
2주차와 3주차에 집중했습니다. (5월 7일에 배운 내용도 연결되는 부분이라 조금 넣었습니다!)
중간 점검인 만큼, 전반적인 흐름을 이해하며 실수하기 좋은 부분을 짚고,
여러 개념을 비교하며 핵심적인 부분을 이해할 수 있게 정리했습니다.

1. **PyTorch** 기본 개념 및 미분

- **PyTorch** 특징
  - Meta 개발, 연구/학계 중심
  - 동적 계산 그래프 지원 → 직관적 디버깅 가능
  - GPU 연산 지원 (torch.cuda.FloatTensor)
- **Tensor vs NumPy**
  - .data.numpy()로 변환 가능 (단, GPU에선 .cpu().data.numpy() 필요함)

⚠️ 실수 주의

- requires_grad=True 설정하지 않으면 자동 미분 불가
- backward() 실행 시 경사 값 누적 → x.grad.zero_()로 초기화 필수

---

2. 자동 미분 **(Autograd)**

- backward() 호출 → .grad 속성에 미분 결과 저장
- 최종 결과는 스칼라 값이어야 미분 가능 (.sum() 사용)

✅ **Torchviz**

- make_dot(z, params={'x': x}) 로 계산 그래프 시각화

---

3. 시그모이드 함수와 미분

- 내장 함수: torch.nn.Sigmoid()
- 직접 정의: 1 / (1 + torch.exp(-x))

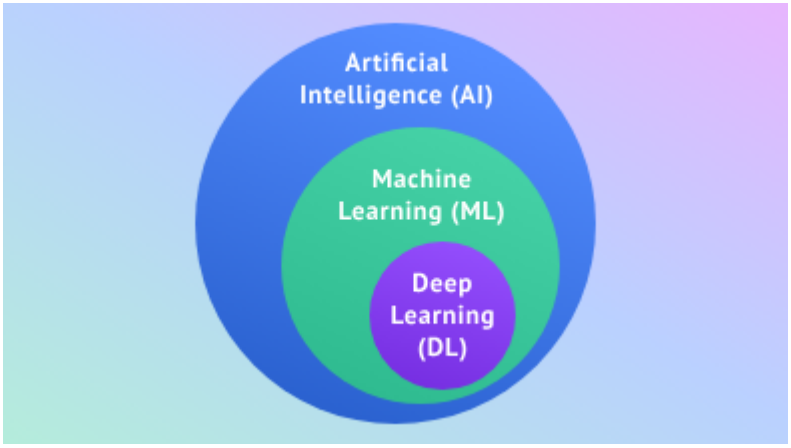
⚠️ 실수 주의

- sum() 없이 backward() 사용하면 에러 발생
- 다시 미분하기 전에는 .zero_()로 초기화 필수

---

4. AI/ML/DL 개요 비교

구분	설명
AI	인간처럼 사고하는 기술 (규칙 기반 포함)
ML	데이터 기반 학습, 예측 (지도/비지도/강화 학습)
DL	다층 신경망 사용, 특징 추출도 자동



✅ 머신러닝 학습 방법

- 지도학습: 정답 있음 → 분류/회귀
- 비지도학습: 정답 없음 → 군집
- 강화학습: 보상 기반 → 행동 학습

---

5. 퍼셉트론과 신경망

- 단층 퍼셉트론: 선형 분리 문제만 가능
- 다층 퍼셉트론 **(MLP)**: XOR 해결 가능 (hidden layer 존재)

---

6. 신경망 학습 및 역전파

- 순전파 → 손실 계산 → 역전파 → 파라미터 업데이트
- 기울기 소실**(Vanishing Gradient)** 문제
    - Sigmoid, Tanh 사용 시 발생
    - 해결책: ReLU, BatchNorm 등

---

7. 활성화 함수 비교

함수	특징	기울기 문제
Sigmoid	출력 (0, 1)	O (기울기 소실)
Tanh	출력 (-1, 1)	O
ReLU	0 이상만 활성화	X (0 이하에서 죽는 문제 있음)
Leaky ReLU	음수에서도 기울기 존재	해결됨

---

8. 경사 하강법 **(Gradient Descent)**

- 단계: 초기화 → 기울기 계산 → 파라미터 업데이트 → 반복
- 종류
    - Batch: 전체 데이터 사용, 안정적이지만 느림
    - SGD: 1개 샘플 사용, 빠르지만 불안정
    - Mini-batch: 균형형

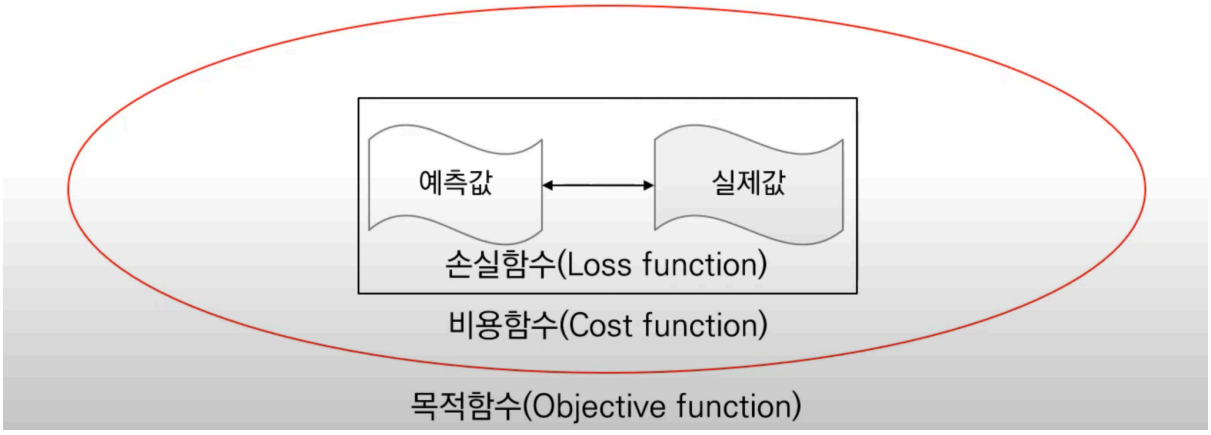
⚠️ 실수 주의

- 학습률**(lr)** 너무 크면 발산, 너무 작으면 느림
- 적절한 learning rate 설정이 중요

---

9. 손실 함수 **vs** 비용 함수

항목	손실 함수 **(Loss)**	비용 함수 **(Cost)**
범위	샘플 단위	전체 데이터셋
목적	개별 오차 측정	전체 성능 평가
예시	MSE, CrossEntropy	평균 MSE, 전체 CrossEntropy



---

10. 회귀와 분류

- 회귀
    - MSE 사용
    - 예: 집값, 혈압 예측
- 이진 분류
    - Sigmoid + Binary Crossentropy
    - 예: P/F, 악성/양성
- 다중 분류
    - Softmax + Categorical Crossentropy
    - 예: 붓꽃, 성적 등급

---

11. 선형 회귀

- $y = \beta_0 + \beta_1 x + \epsilon$
- **OLS (**최소제곱법**)**: 오차 제곱합 최소화

12. 다항 회귀

- 독립변수가 2차 이상 → 비선형 관계 모델링 가능

---

13. 이진 분류 **(Binary Classification)**

- 기본 구조: 입력층 → 활성화 함수 (예: Sigmoid) → 출력층 (확률 기반 이진 분류)
- 결정 경계 **(Hyperplane)**: 2D에서는 선, 3D에서는 평면

⚠️ 실수 주의

- 확률로 예측하되, 임계값 기준 (보통 0.5) 이상이면 양성으로 판단

---

14. 데이터셋 분할 전략

구분	용도
Training Set	학습
Validation Set	하이퍼파라미터 튜닝
Test Set	최종 성능 평가

- 교차 검증: k-fold 사용
- 부트스트랩핑: 데이터 적을 때 사용, 여러 샘플 셋 생성

---

15. 혼동 행렬 **(Confusion Matrix)**

실제/예측	**Positive**	**Negative**
Positive	TP	FN
Negative	FP	TN

- **TP, FP, FN, TN** 개념 정확히 이해 필요**!!**

- Accuracy = TP + TN / (TP + FP + FN + TN)
- Precision = TP / (TP + FP)
- Recall = TP / (TP + FN)
- F1 = 조화 평균

✅ 지표 선택 예시**(**참고만 해도 괜찮을 듯**)**

중점 사항	지표	예시
FP 줄이기	Precision	스팸 필터
FN 줄이기	Recall	암 진단
균형	F1-score	일반적인 분류

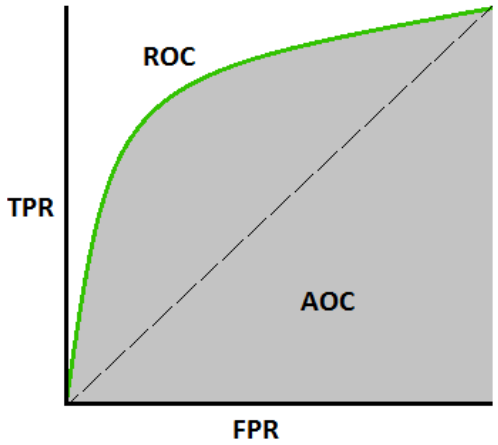
전체 성능	ROC-AUC	분류 전반 성능 평가

---

16. ROC-AUC

- x축: FPR (False Positive Rate) = 잘못 양성 판정한 비율
- y축: TPR (True Positive Rate) = 올바르게 양성 판정한 비율 (= 재현율, Recall)

- AUC (Area Under Curve) 값이 **1**에 가까울수록 좋음
- 곡선의 면적이 크다는 것은 좋은 분류 성능을 의미



---

17. 다중 분류 (Multinomial Classification)

- **Softmax** 함수 사용
    - 각 클래스에 대해 확률 계산, 총합은 1
    - 예측은 가장 확률 높은 클래스로 결정

---

18. Negative Log-Likelihood (NLL)

- 손실 함수
    - 예측이 정확할수록 NLL 값 작음

    - 확률 기반 예측에서 자주 사용됨
- 다중 클래스에서도 NLL 사용 (Multiclass NLL)

---

19. CNN (Convolutional Neural Network)

구성 요소

- **Convolutional Layer**: 필터로 특징 추출
- **Pooling Layer**: 다운샘플링 (Max, Average)
- **FC Layer**: 분류 수행
- **Activation**: ReLU 주로 사용
- **Loss Function**: CrossEntropy
- **Flatten Layer**: FC로 연결되기 위한 1D 변환

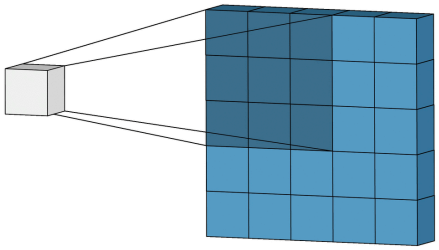
https://aayushmaan1306.medium.com/basics-of-convolutional-neural-networks-using-pytorch-lightning-474033093746

https://dotiromoook.tistory.com/19

---

20. CNN 연산 핵심

요소	설명
Stride	필터 이동 간격. 작을수록 세밀
Padding	경계 보존 목적. Same vs Valid
Pooling	Max, Average
Flatten	FC 연결 위한 1D 변환



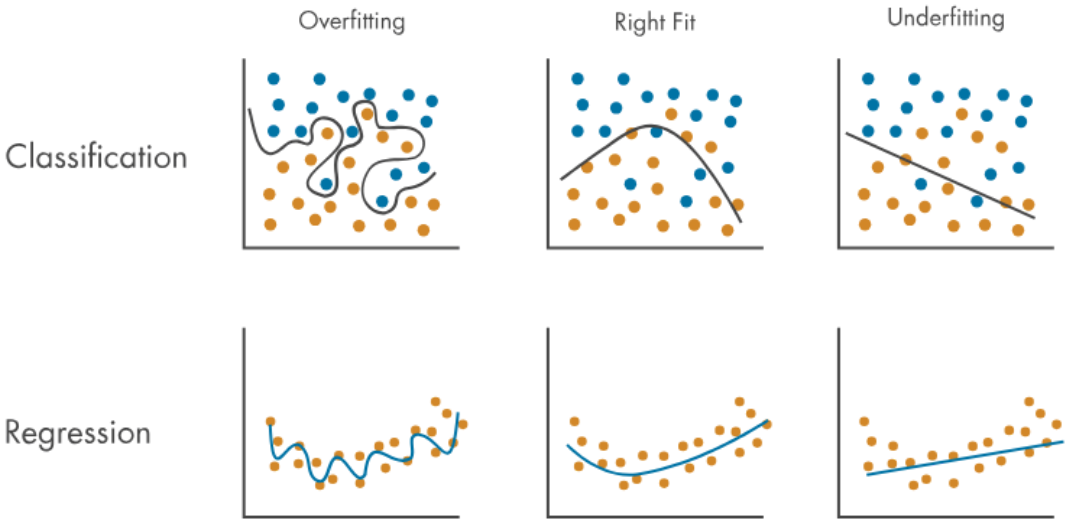
---

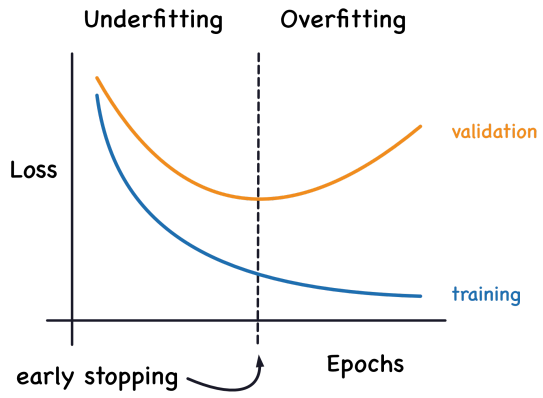
21. 최적화 알고리즘 (Optimizer 종류)

알고리즘	특징
SGD	빠르지만 진동 가능
Momentum	관성으로 수렴 속도 증가
NAG	예측 위치에서 기울기 계산
Adagrad	학습률 자동 조절 (단점: 너무 작아질 수 있음)
RMSProp	Adagrad 개선, 안정성 ↑
Adam	모멘텀 + RMSProp, 가장 널리 사용됨

---

22. 과적합 (Overfitting)





원인

- 데이터 부족
- 모델 크기 과도
- Epoch 수 과다

해결 방법

- 데이터 증강
- 정규화 **(L1, L2)**
- **Dropout**
- **Early Stopping**

- 적절한 모델 구조 선택

---

23. Batch Normalization

- **Internal Covariate Shift** 해결
- 학습 속도 증가 + 안정성 향상 + 과적합 감소
- Mini-batch 단위 정규화 수행

---

24. 데이터 증강 (Data Augmentation)

기법	설명
회전/이동/확대	기하학 변환
밝기/대비	컬러 조절
노이즈 삽입	Gaussian Noise 등
블러링	이미지 흐림

필요 이유

- 데이터 불균형, 과적합 완화, 비용 절감
