1 기초 수학 복습

1.1 행렬 연산

해의 존재: $\text{rank}(A) = \text{rank}([A|b])$
정사영: $\text{proj}_b(a) = \frac{a \cdot b}{|b|^2} b$
Rank: 선형독립 벡터 개수

1.2 확률/정보이론

$H(X) = -\sum p(x) \log_2 p(x)$
BCE: $L = -[y \log(\hat{y}) + (1-y) \log(1-\hat{y})]$
균등분포일 때 엔트로피 최대 (이진: $p = 0.5 \rightarrow H = 1$)

2 신경망 핵심

2.1 체인룰 계산 예제

조건: $x = 2, W_1 = 3, b_1 = -1, W_2 = 0.5, b_2 = 0.2, y = 1$
순전파: $z_1 = W_1 x + b_1 = 3 \times 2 + (-1) = 5$
$a_1 = \text{ReLU}(z_1) = \text{ReLU}(5) = 5$
$z_2 = W_2 a_1 + b_2 = 0.5 \times 5 + 0.2 = 2.7$
$\hat{y} = \sigma(z_2) = \frac{1}{1+e^{-2.7}} \approx 0.937$
$L = (y - \hat{y})^2 = (1 - 0.937)^2 = 0.004$
역전파 ($\frac{\partial L}{\partial W_1}$): $\frac{\partial L}{\partial \hat{y}} = 2(\hat{y} - y) = 2(0.937 - 1) = -0.126$
$\frac{\partial \hat{y}}{\partial z_2} = \sigma'(z_2) = 0.937 \times 0.063 = 0.059$
$\frac{\partial z_2}{\partial a_1} = W_2 = 0.5$
$\frac{\partial a_1}{\partial z_1} = \text{ReLU}'(z_1) = 1 \ (\because z_1 > 0)$
$\frac{\partial z_1}{\partial W_1} = x = 2$
$\frac{\partial L}{\partial W_1} = (-0.126) \times 0.059 \times 0.5 \times 1 \times 2 = -0.0074$
2차 역전파 ($\frac{\partial z_2}{\partial W_1}$): $\frac{\partial z_2}{\partial W_1} = \frac{\partial z_2}{\partial a_1} \times \frac{\partial a_1}{\partial z_1} \times \frac{\partial z_1}{\partial W_1} = 0.5 \times 1 \times 2 = 1.0$

2.2 손실함수

MSE: $L = \frac{(y-\hat{y})^2}{2}$, $\frac{\partial L}{\partial \hat{y}} = \hat{y} - y$
BCE + Sigmoid: $\frac{\partial L}{\partial z} = \hat{y} - y$ (핵심!)

3 CNN 필수 공식

3.1 CNN 출력 크기

$$\text{Output} = \left\lfloor \frac{\text{Input} + 2 \times \text{Padding} - \text{Filter}}{\text{Stride}} + 1 \right\rfloor$$

예시: 28×28, Conv$(5 \times 5, s = 2, p = 3)$
$\rightarrow \lfloor \frac{28+2\times3-5}{2} + 1 \rfloor = \lfloor \frac{29}{2} \rfloor + 1 = 15$

3.2 주요 아키텍처

LeNet: Conv → Pool → Conv → Pool → FC
ResNet: $y = F(x) + x$ (Skip Connection)
1×1 Conv: 채널 수 조절, 비선형성 추가

3.3 Batch Normalization

위치: Conv/Linear → BN → Activation → Dropout
효과: 내부 공변량 이동 방지, 학습 안정화

4 Transformer 완전정복

4.1 Self-Attention 수식

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V$$
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

스케일링 이유: 큰 d_k에서 softmax 극값 방지

4.2 Multi-Head Attention

$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h) W^O$
$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$
8개 헤드 이유: 다양한 representation 부공간 학습

4.3 Positional Encoding

필요 이유: Self-attention은 순서 무관한 집합 연산

4.4 Encoder vs Decoder

구성요소	Encoder	Decoder
Self-Attention	✓	Masked ✓
Cross-Attention	×	✓
Feed-Forward	✓	✓

Masked Attention: 미래 토큰 정보 차단

5 Language Models 비교

5.1 BERT vs GPT vs T5 핵심 차이

특성	BERT	GPT	T5
구조	Encoder-only	Decoder-only	Encoder-Decoder
방향성	양방향(Enc)	단방향(Dec)	양방향 + 단방향
학습	MLM + NSP	Auto-regressive	Span Masking
용도	이해 중심	생성 중심	텍스트→텍스트 통합

5.2 BERT MLM 과정

1. 15% 토큰 선택 → 80%[MASK], 10%무작위, 10%유지
2. 양방향 인코딩으로 전체 문맥 학습
3. 마스킹된 위치만 예측하여 손실 계산

5.3 GPT Auto-regressive

$P(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | x_1, \ldots, x_{i-1})$
이전 토큰들만 참조하여 다음 토큰 예측

GPT 발전 과정:
GPT-1: Transformer 디코더, Next Token Prediction
GPT-2: 1.5B 파라미터, Zero-shot 학습 가능
GPT-3: 175B 파라미터, Few-shot Learning
ChatGPT: RLHF(인간피드백 강화학습) 4단계 학습
1. **Pretraining**: GPT-3 수준 대규모 비지도 학습
2. **Supervised Fine-tuning**: 인간 작성 질문-답변 쌍 지도학습
3. **Reward Model Training**: 인간 순위 매김 기반 보상 모델
4. **Reinforcement Learning (PPO)**: 보상 모델 기반 PPO 최적화

6 Vision Transformer & Style Transfer

6.1 ViT vs CNN

특성	ViT	CNN
귀납적 편향	없음	지역성, 평행이동 불변성
데이터 요구량	많음	적음
패치 처리	16×16 패치	픽셀 단위

6.2 Swin Transformer (핵심 개선)

ViT 문제점: 모든 패치간 attention → $O(n^2)$ 복잡도
해결방법: Window-based + Shifted Windows
3단계 프로세스:
1. Patch Partitioning: 4×4 패치 분할
2. Window-based Attention: 윈도우 내부만 attention
3. Shifted Windows: 층마다 윈도우 위치 이동
Cyclic Shift: Feature map을 작은 거리만큼 순환 이동
복잡도 개선: $O(n^2) \rightarrow O(n)$ (선형으로 감소)
계층적 구조: $4 \times 4 \rightarrow 8 \times 8 \rightarrow 16 \times 16$
Patch Merging: Resolution 점진적 감소 → 계층적 표현 학습
장점: CNN의 지역성 + Transformer의 장거리 의존성

6.3 Neural Style Transfer

Content Loss: $J_{\text{content}}(C, G) = \frac{1}{2} \sum_l (A^{[l]}(C) - A^{[l]}(G))^2$
Style Loss: $J_{\text{style}}(S, G) = \sum_l \frac{1}{(2n_H^{[l]} n_W^{[l]} n_C^{[l]})^2} \sum_{i,j} (G_{i,j}^{[l]}(S) - G_{i,j}^{[l]}(G))^2$
Gram Matrix: $G_{ij} = \sum_k F_{ik} \times F_{jk}$ (채널 간 상관관계)
Total Loss: $J(G) = \alpha J_{\text{content}}(C, G) + \beta J_{\text{style}}(S, G)$

7 최적화 & 정규화

7.1 Transfer Learning

Feature Extraction: 하위층 고정, 상위층만 학습
Fine-tuning: 하위층 작은 LR, 상위층 큰 LR

8 Adversarial Machine Learning

8.1 Adversarial Attack 기본 개념

정의: $x' = x + \delta$, where $\|\delta\|_p \leq \epsilon$
목표: $f(x) \neq f(x')$ but $\|x - x'\|$ is small

8.2 공격 방법론

FGSM: $x' = x + \epsilon \cdot \text{sign}(\nabla_x \ell(f(x), y))$
PGD: $x^{t+1} = \Pi_S(x^t + \alpha \cdot \text{sign}(\nabla_x \ell(f(x^t), y)))$

8.3 손실함수 설계

Untargeted: $\max_\delta \ell(f(x + \delta), y)$
Targeted: $\min_\delta \ell(f(x + \delta), y_{\text{target}})$
CW Loss: $g(x) = \max(\max_{i \neq t} Z_i(x) - Z_t(x), -\kappa)$
Logit 차이: $\ell_{\text{diff}} = Z_y - \max_{i \neq y} Z_i$

8.4 Adversarial Training

Min-Max: $\min_\theta \mathbb{E}_{(x,y)} \left[\max_{\|\delta\|_\infty \leq \epsilon} \ell(f_\theta(x + \delta), y)\right]$
TRADES: $\min_\theta \mathbb{E}\left[\ell(f_\theta(x), y) + \beta \cdot D_{KL}(f_\theta(x) \| f_\theta(x + \delta))\right]$
MART: Misclassified Adversarial Training

8.5 방어 기법

Input Transform: JPEG compression, bit reduction
Gradient Masking: 그래디언트 정보 은닉
Certified Defense: Randomized smoothing
Detection: Statistical tests, reconstruction error

9 고급 주제

9.1 ViT Patch Embedding

1. 이미지 → 16×16 패치 분할
2. 선형 변환 → d_{model} 차원
3. Position Embedding 추가

4. [CLS] 토큰 concat

10 실습 코드 핵심

10.1 PyTorch 텐서 계산

Conv2d 출력 크기:

```
out_h = (in_h + 2*pad - kernel) // stride + 1
```

Conv2d(a,b,c,d,e) 매개변수 분석:
a: input_channels, b: output_channels, c: kernel_size
d: stride, e: padding
예시: $(32, 512, 14, 14) \rightarrow$ Conv2d$(512, 256, 3, 2, 1)$
$\rightarrow (32, 256, 7, 7)$

10.2 Adversarial Attack 코드

FGSM 구현:

```
delta = eps * torch.sign(x.grad.data)
x_adv = x + delta
x_adv = torch.clamp(x_adv, 0, 1)
```

PGD 구현:

```
for i in range(num_steps):
x_adv.requires_grad_()
loss = criterion(model(x_adv), y)
loss.backward()
x_adv = x_adv + alpha * torch.sign(x_adv.grad)
x_adv = torch.clamp(x_adv, x-eps, x+eps)
```

Loss 함수 설정:

```
target_loss = -nn.CrossEntropyLoss()(pred, target)
untarget_loss = nn.CrossEntropyLoss()(pred,
true_label)
```

10.3 논리게이트 구현 (중간고사 실제 문제)

10.3.1 퍼셉트론 모델

$$y = f(w_1x_1 + w_2x_2 + b)$$

$$f(z) = \begin{cases} 1 & \text{if } z \geq 0.5 \\ 0 & \text{if } z < 0.5 \end{cases}$$

10.3.2 AND 게이트 (교수님 제시 파라미터)

파라미터: $w_1 = 20, w_2 = 20, b = -30$

x_1	x_2	$z = w_1x_1 + w_2x_2 + b$	$z \geq 0.5$?	출력	AND
0	0	-30	X	0	0
0	1	-10	X	0	0
1	0	-10	X	0	0
1	1	10	O	1	1

10.3.3 XOR/XNOR - 선형분리 불가능

단층 퍼셉트론: 불가능 (직선으로 분리 안됨)
시험 정답: "만들 수 없습니다" (4명 정답처리)
해결: 2층 신경망 필요 (AND + NOR → OR)

10.3.4 XNOR 2층 신경망 구현

논리: XNOR = AND(x_1,x_2) OR NOR(x_1,x_2)
"둘 다 같으면 1" = "둘 다 1" OR "둘 다 0"
네트워크 구조:

```
x₁,x₂ → [AND, NOR] → OR → XNOR
```

파라미터: 1층 AND: $w_{11} = 20, w_{12} = 20, b_1 = -30$
1층 NOR: $w_{21} = -20, w_{22} = -20, b_2 = 10$
2층 OR: $v_1 = 20, v_2 = 20, b_3 = -10$
계산 검증:

x_1	x_2	h_1(AND)	h_2(NOR)	output	XNOR
0	0	f(-30)=0	f(10)=1	f(10)=1	1
0	1	f(-10)=0	f(-10)=0	f(-10)=0	0
1	0	f(-10)=0	f(-10)=0	f(-10)=0	0
1	1	f(10)=1	f(-30)=0	f(10)=1	1

10.3.5 시험 답안 작성 템플릿

Step 1: 파라미터 제시 (3점)
"제시한 파라미터: $w_1 = [\,`\,], w_2 = [\,`\,], b = [\,`\,]$"
Step 2: 계산 과정 (12점)
각 입력조합 (x_1, x_2)에 대해: $z = w_1x_1 + w_2x_2 + b$ 계산 → 임계값 비교 → 출력 결정
Step 3: 결론 (3점)
"모든 경우에서 신경망 출력이 [게이트명] 진리표와 일치함"
XNOR 함정 주의: "만들 수 없습니다" = 정답!

10.3.6 XNOR 선형분리 불가능 증명

단층 퍼셉트론의 결정경계: $w_1x_1 + w_2x_2 + b = 0.5$
XNOR 조건들: $(0,0)$: $b \geq 0.5$ (출력=1)
$(0,1)$: $w_2 + b < 0.5$ (출력=0)
$(1,0)$: $w_1 + b < 0.5$ (출력=0)
$(1,1)$: $w_1 + w_2 + b \geq 0.5$ (출력=1)
모순 도출: 조건 2,3: $w_1 + w_2 < 1 - 2b$
조건 4: $w_1 + w_2 \geq 0.5 - b$
조건 1: $b \geq 0.5$이면 $1 - 2b \leq 0$
$\therefore$ $w_1 + w_2 < 0$ 그리고 $w_1 + w_2 \geq 0$ (모순!)
따라서 단층으로는 구현 불가능!

11 암기 필수 공식

ResNet: $y = F(x) + x$
Swin 복잡도: ViT $O(n^2) \rightarrow$ Swin $O(n)$
LoRA 분해: $\Delta W = A \times B$, $r \ll \min(d, k)$
Adversarial Risk: $\mathbb{E}[\max_{\|\delta\| \leq \epsilon} \ell(f(x + \delta), y)]$
Neural Style Transfer: $J(G) = \alpha J_{\text{content}}(C, G) + \beta J_{\text{style}}(S, G)$
GAN: $\min_G \max_D [\mathbb{E}[\log D(x)] + \mathbb{E}[\log(1 - D(G(z)))]]$
Language Model: $P(w_1, \ldots, w_n) = \prod_{i=1}^{n} P(w_i | w_1, \ldots, w_{i-1})$
BERT: $L = L_{MLM} + L_{NSP}$ (MLM: Masked Language Model, NSP: Next Sentence Prediction)

12 최종 체크리스트

12.1 LeNet 실습 코드 (중간고사 기출)

입력: $28 \times 28 \times 1$ (MNIST)
Conv1: kernel=5, out=6 $\rightarrow 24 \times 24 \times 6$
Pool1: kernel=2 $\rightarrow 12 \times 12 \times 6$
Conv2: kernel=5, out=16 $\rightarrow 8 \times 8 \times 16$
Pool2: kernel=2 $\rightarrow 4 \times 4 \times 16$
Flatten: $4 \times 4 \times 16 = 256$
FC: $256 \rightarrow 120 \rightarrow 84 \rightarrow$ output_dim
특성벡터 h: 256차원 (Flatten 후)

13 고급 주제 (15주차)

13.1 LoRA (Low-Rank Adaptation)

문제: GPT-3 (174B), PaLM (540B) 파라미터 fine-tuning 비용
해결: 저차원 분해로 효율적 적응 **수식**: $\Delta W = A \times B$
$A \in \mathbb{R}^{d \times r}, B \in \mathbb{R}^{r \times k}$ where $r \ll \min(d, k)$
파라미터 수: $(d \times r) + (r \times k) \ll d \times k$
예시: $d = k = 4096, r = 16 \rightarrow$ 128배 파라미터 감소

13.2 GAN (Generative Adversarial Networks)

구조: Generator vs Discriminator 경쟁
목적함수: $\min_G \max_D [\mathbb{E}[\log D(x)] + \mathbb{E}[\log(1 - D(G(z)))]]$
학습: Generator는 가짜 생성, Discriminator는 진위 판별

13.3 멀티모달: LLaVA 완전정복

구조: LLM + ViT + Projection Layer
구성: CLIP ViT-L/14 + Vicuna LLM
2단계 학습 과정:
1단계 - Feature Alignment:
- Projection W만 학습 (LLM, Vision 고정)
- 단일 문장 이미지 캡션 기반

2단계 - End-to-End Fine-tuning:
- LLM + W 동시 업데이트
- 158K 시각 지시 데이터: 대화(58K) + 설명(23K) + 추론(77K)

Visual Instruction Tuning: 시각정보 + 언어지시 결합 학습
한계: 개체 분리 상태 판단 어려움 (딸기+요거트≠딸기요거트)

13.4 고급 학습 기법

In-Context Learning: 프롬프트 예시로 규칙 유추
Chain-of-Thought: "Let's think step by step" - 중간 추론 과정 표시
Prefix Tuning: 입력 앞에 학습가능 가상 토큰 추가

13.5 언어모델 추가

Word2Vec: 주변단어→중심단어 예측, 문맥고려 불가
T5: 모든 NLP → 텍스트입력→텍스트출력 통합
Linear Attention: $O(N^2) \rightarrow O(N)$ 복잡도 개선
핵심: 기존 Attention 식의 행렬 곱 순서 변경
$A(x) = \sum_{j=1}^{N} P_j V_j, \quad K(q,k) = \phi(q)^T \phi(k)$