Softmax Classifier와 Logistic Regression

수업 목표

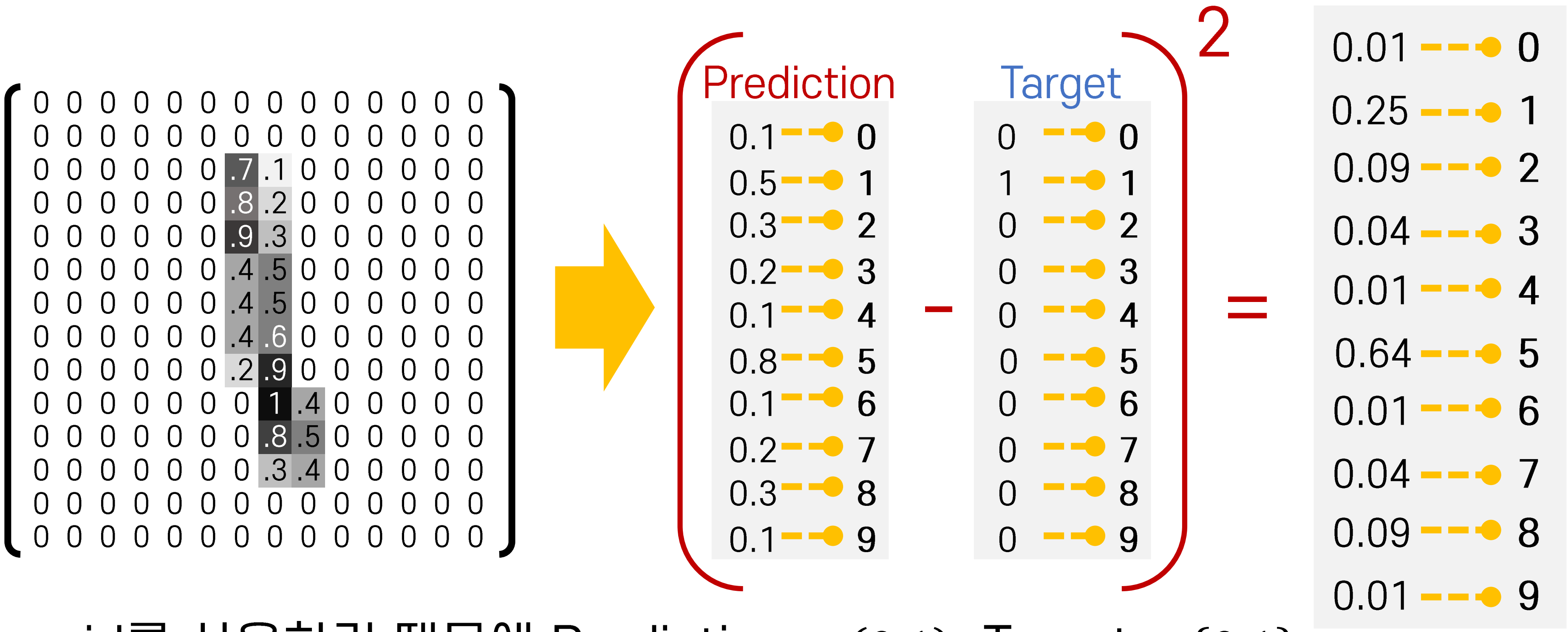
이번 수업의 핵심:

- Mean squared error (MSE)와 Sigmoid를 통한 classifier의 문제점
- Softmax layer의 개념과 Softmax loss 계산
- Softmax loss와 Cross entropy, 그리고 KL divergence의 관계
- Logistic regression과 Softmax classifier의 관계

핵심 개념

- Softmax layer 및 Softmax classifier
- Negative log-likelihood, Softmax loss
- Cross entropy, KL divergence
- Logistic regression

Mean Squared Error (MSE)의 문제점



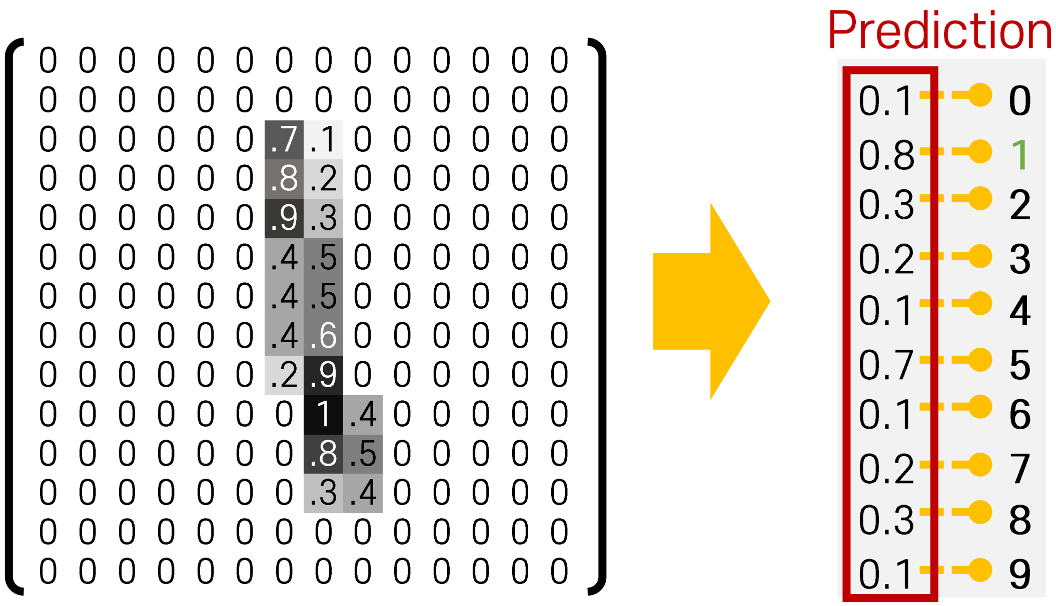
- Sigmoid를 사용하기 때문에 Prediction $\in (0,1)$, Target $\in \{0,1\}$

→ MSE Loss 사용시 Loss과 Gradient 크기에 상한이 존재

$$\max \mathcal{L} = \max_{y\in\{0,1\},\hat{y}\in(0,1)} (\hat{y}-y)^2 < 1, \qquad \max\left|\frac{\partial \mathcal{L}}{\partial \hat{y}}\right| < 2$$

- 따라서, 학습이 느리고, 분류 문제를 위한 더 좋은 Loss function이 존재

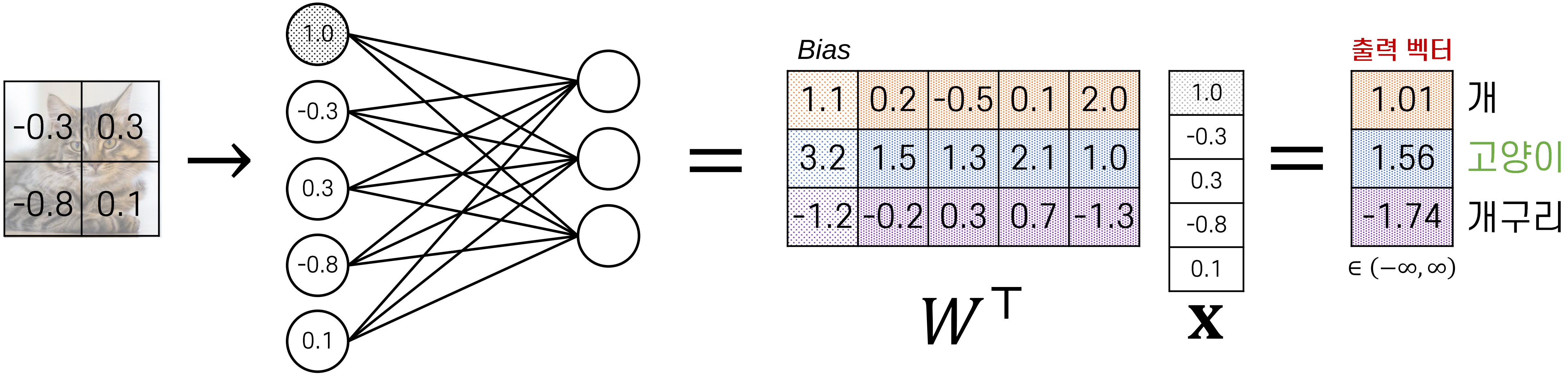
Sigmoid를 통한 Classifier의 문제점



Sigmoid를 통한 Prediction의 결과는 각 숫자가 될 독립적인 확률이지,
어느 숫자가 될지에 대한 Sum-to-one 형태의 상대적인 확률이 아님

- 전자: Multi-label classification
- 후자: Multi-class classification
- MNIST classification은 Multi-class classification 이 적합함

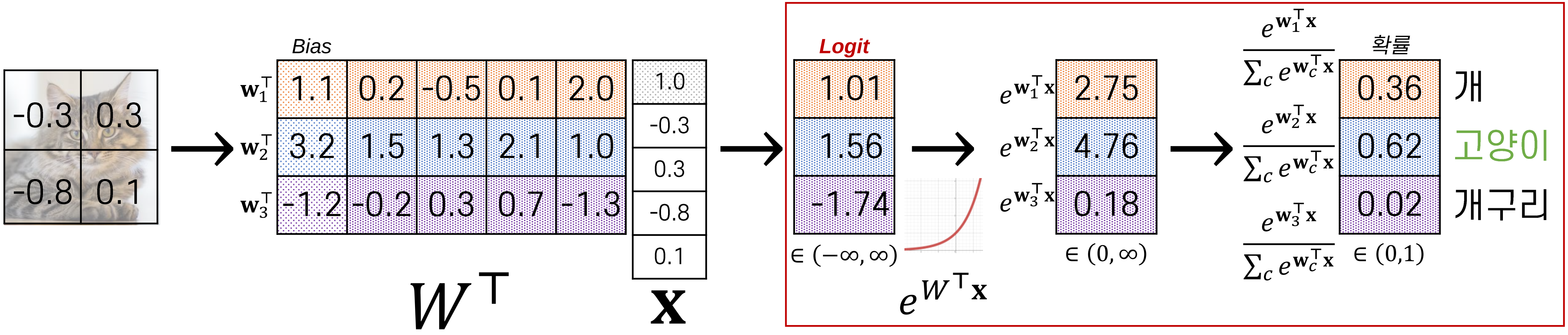
Softmax Layer for Multi-Class Classification



- 먼저 Linear layer를 두고, 출력 벡터의 Dimension을 Class 개수와 동일하게 설정
- 특정 Dimension의 값이 클 수록, 해당 Class에 부여되는 확률 값이 커지도록 함
- 출력 벡터를 Normalize하여 합이 1인 형태의 상대적인 확률 분포로 변환

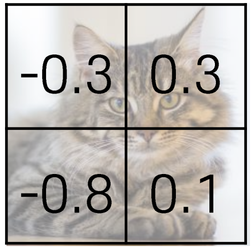
→ 이러한 layer를 Softmax layer라고 부름

Softmax Layer



- **Logit vector:** Softmax의 입력 벡터이자, 직전 Linear layer의 출력 벡터
- Logit vector의 각각의 값에 단조 증가함수인 지수 함수를 적용:
  - $(-\infty, \infty)$ 사이의 Logit을 $(0, \infty)$ 사이의 양수 값으로 변환
  - 변환 후에도 크기 순서가 유지
- 해당 양수 값들의 합에 대한 각 값의 상대적인 크기를 계산
  - 각 Class에 대한 결과 값의 합이 1 → 확률 분포로 해석할 수 있음

Softmax Classifier의 Loss Function



i번째 사진 →

$$\begin{array}{c} \text{Bias} \\ W^\top = \begin{bmatrix} 1.1 & 0.2 & -0.5 & 0.1 & 2.0 \\ 3.2 & 1.5 & 1.3 & 2.1 & 1.0 \\ -1.2 & -0.2 & 0.3 & 0.7 & -1.3 \end{bmatrix} \end{array} \quad \mathbf{x}_i = \begin{bmatrix} 1.0 \\ -0.3 \\ 0.3 \\ -0.8 \\ 0.1 \end{bmatrix} \rightarrow \begin{array}{c} \text{Logit} \\ \begin{bmatrix} 1.01 \\ 1.56 \\ -1.74 \end{bmatrix} \\ \in (-\infty, \infty) \end{array} \rightarrow \begin{array}{c} \text{확률} \\ \begin{bmatrix} 0.36 \\ 0.62 \\ 0.02 \end{bmatrix} \\ \in (0,1) \end{array} \begin{array}{l} \text{개} \\ \text{고양이} \\ \text{개구리} \end{array}$$

$\hat{p}_c$

$c \in \textbf{classes} = \{1, 2, 3\}$

Loss

정답 class
고양이 $:= \mathbf{2}$

$\boxed{2}$

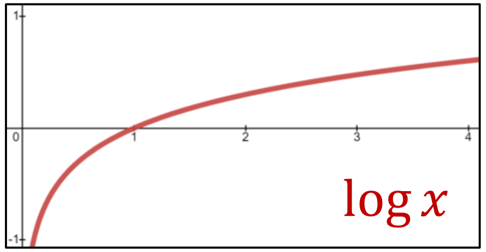
y_i

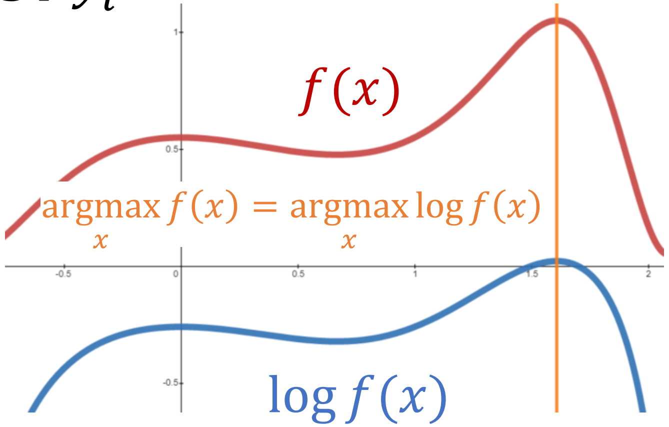
$y_i \in \textbf{classes} = \{1, 2, 3\}$

- 정답 Class에 대한 Likelihood $\hat{p}_{y_i}$를 최대화

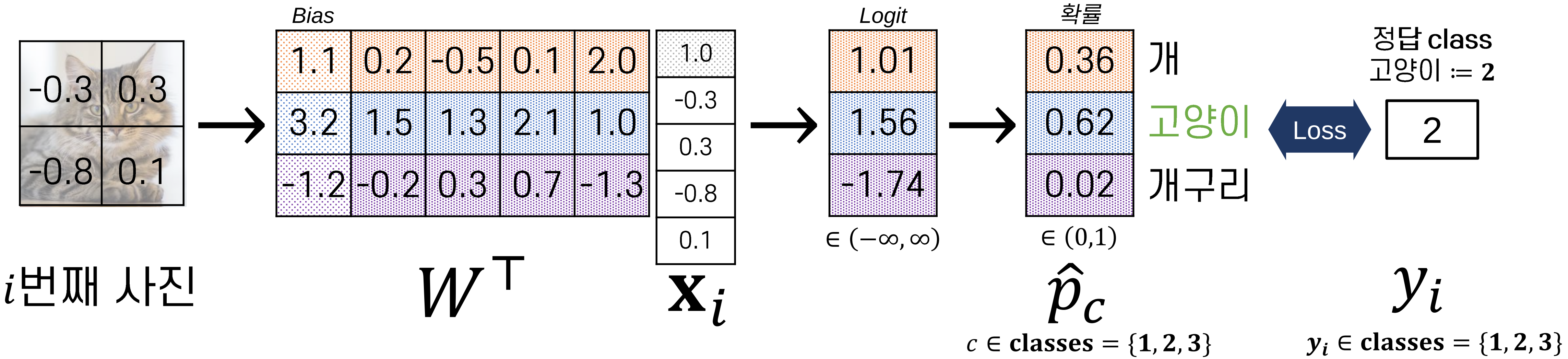
→ 이는 Log-likelihood 최대화와 동등: $\log \hat{p}_{y_i}$

- $(0, 1)$ 사이의 값을 $(-\infty, 0)$ 사이로 변환
- 로그 함수는 단조 증가 → 크기 순서 유지



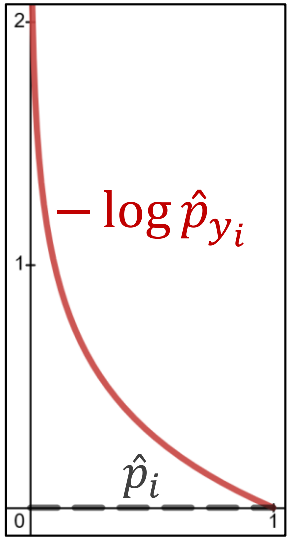


Softmax Loss = Negative Log-Likelihood Loss



- 정답 Class에 대한 Likelihood $\hat{p}_{y_i}$를 최대화하고자 하나,

  우리는 일반적으로 최소화하고자 하는 "Loss"의 형태로 정의함



→ Negative Log-Likelihood (NLL) loss, 즉 $-\log \hat{\boldsymbol{p}}_{y_i}$를 최소화

- $(0, 1)$ 사이의 값을 $(0, \infty)$ 사이로 변환
- 마이너스 로그 함수는 단조 감소 함수

→ 그리고, 이 Loss를 Softmax loss라고도 부름

Softmax Loss = Cross Entropy Loss

이산 확률 분포 P와 Q에 대해서, Cross entropy는 다음과 같음:

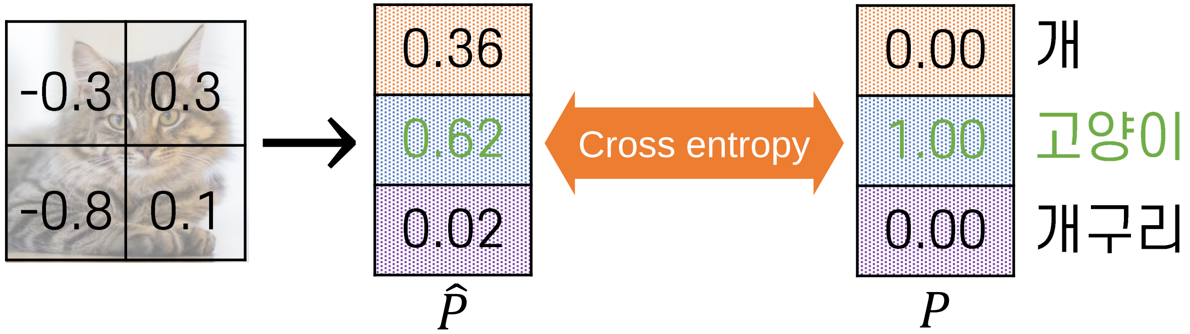
$$H(P,Q) = -\sum_{x \in X} P(x) \log Q(x)$$

- 이 값은 두 분포 P와 Q가 얼마나 다른지를 측정
- Cross entropy가 Loss로 사용될 시, Ground-truth 확률 분포와 예측 확률 분포간 차이를 측정
- 일반적으로 Ground-truth 확률 분포를 P로 두고, 예측 확률 분포 $\hat{P}$를 Q로 둠

Cross entropy에서 Ground-truth 분포를 One-hot vector로 계산 → Softmax loss와 동일

- One-hot vector: $P = [p_1, \ldots, p_C]$의 정답 위치 p_{y_i}에만 1을 넣고 다른 위치에는 0을 할당,

  (예시: [0, 1, 0]) $H(P,Q) = H\left(P,\hat{P}\right) = -\sum_{c=1}^{C} p_c \log \hat{p}_c = -\log\left(\hat{p}_{y_i}\right)$



KL Divergence vs. Cross Entropy

Cross entropy는 KL divergence로도 표현 가능

$$D_{KL}(P \parallel Q) = -\sum_{x \in X} P(x) \log\left(\frac{Q(x)}{P(x)}\right) = -\sum_{x \in X} P(x) \log Q(x) + \sum_{x \in X} P(x) \log P(x) = H(P, Q) - H(P)$$

Cross entropy와 KL divergence의 차이는 추가적인 Scalar 값 $-\boldsymbol{H(P)}$ 존재 유무

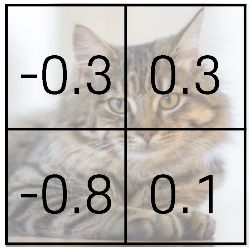
- Ground-truth 확률 분포 P의 Entropy인 $H(P)$는 상수 값
  → 최적화 과정에서는 영향을 주지 않음
- 또한, P가 One-hot vector인 경우, $H(P) = 0$

따라서, Cross entropy 최소화 = KL divergence 최소화

→ Cross entropy 대신 KL divergence도 Loss로 사용 가능

- Ground-truth 확률 분포 P가 Dense vector인 경우도 존재
  - e.g., Knowledge distillation
  - Dense vector 예시: [0.2, 0.4, 0.1, 0.3] (vs. one-hot vector [0, 1, 0, 0])

Logistic Regression은 Softmax Classifier의 Special Case



→

Bias

3.2	1.5	1.3	2.1	1.0

$\mathbf{w}^\top$

1.0
-0.3
0.3
-0.8
0.1

$\mathbf{x}$

→

Logit

1.56
0

$\in (-\infty, \infty)$

→ $e^{\mathbf{w}^\top \mathbf{x}}$

$e^{\mathbf{w}^\top \mathbf{x}}$ 4.76

$e^0 = 1$ 1

$\in (0, \infty)$

→ 확률

$\frac{e^{\mathbf{w}^\top \mathbf{x}}}{e^{\mathbf{w}^\top \mathbf{x}} + 1}$ 0.83 고양이

$\frac{1}{e^{\mathbf{w}^\top \mathbf{x}} + 1}$ 0.17 고양이 아님

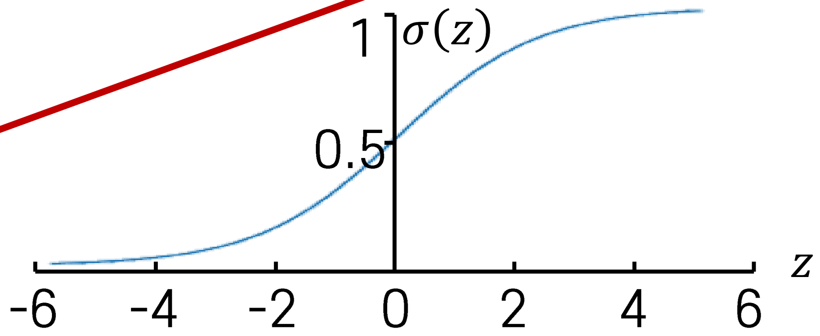
$\in (0,1)$

$$\frac{e^{\mathbf{w}^\top \mathbf{x}}}{e^{\mathbf{w}^\top \mathbf{x}} + 1} = \boxed{\frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}}}}$$

Logistic regression

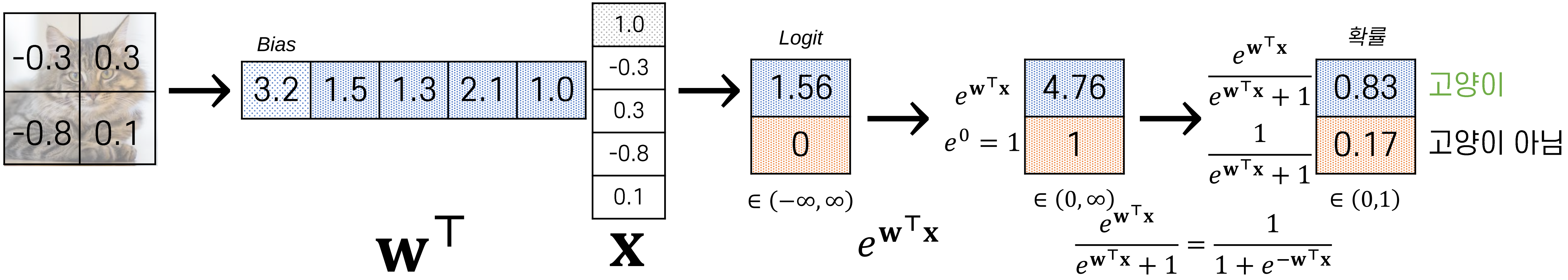
→ Binary classification 문제에서 Positive class가 아닌 나머지 Class의 Logit을 상수 0으로 고정한 형태의 Softmax classifier

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



Logistic function
Sigmoid function

Sigmoid Function



Logistic regression

→ Binary classification 문제에서 Positive class가 아닌 나머지 Class의 Logit을 상수 0으로 고정한 형태의 Softmax classifier

- 2열 크기 행렬 W로 두 Class에 대한 Softmax classification 으로도 가능
  - 이 경우 Logistic regression보다 두 배 많은 수의 Parameter를 가짐

Binary Cross Entropy for Logistic Regression

Logistic Regression의 Binary cross entropy loss

➔ Softmax classifier의 Cross entropy에서 Class가 두 개일 때와 동일함

- Cross Entropy

$$\mathcal{L} = -\sum_{c=1}^{C} y_c \log(p_c)$$

- Binary Cross Entropy (BCE)

$$\mathcal{L} = -\sum_{c=1}^{2} y_c \log(p_c) = -y_1 \log(p_1) - y_2 \log(p_2) = -y_1 \log(p_1) - (1 - y_1)\log(1 - p_1)$$

$$\because y_2 = 1 - y_1, p_2 = 1 - p_1$$

요약

- MSE, Sigmoid classifier의 문제점
- Softmax layer의 개념 및 수학적 배경
- Softmax loss와 Cross entropy loss의 관계
- Cross entropy 최소화와 KL divergence 최소화의 동등성
- Logistic regression이 Softmax classifier의 특수한 예시임을 확인