2021 Spring

Artificial Intelligence & Deep Learning

Prof. Minsuk Koo

Department of Computer Science & Engineering
Incheon National University

5.1.2 교차 엔트로피 목적함수

교차 엔트로피

- 레이블에 해당하는 y가 확률변수 (부류가 2개라고 가정하면 $y \in \{0,1\}$)
- 확률 분포: P는 정답 레이블, Q는 신경망 출력

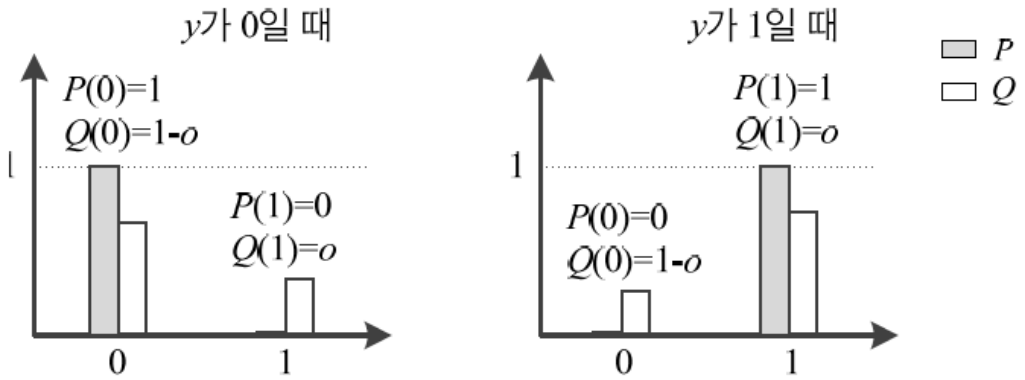

그림 5-3 레이블 y가 0일 때와 1일 때의 P와 Q의 확률분포

- 확률분포를 통일된 수식으로 쓰면,

$$P(0) = 1 - y \quad Q(0) = 1 - o$$
$$P(1) = y \quad Q(1) = o$$

- 교차 엔트로피 식은 $H(P,Q) = -\sum_{y\in\{0,1\}} P(y)\log_2 Q(y)$

5.1.2 교차 엔트로피 목적함수

교차 엔트로피 목적함수

$$e = -(y \log_2 o + (1-y)\log_2(1-o)), \quad \text{이때, } o = \sigma(z) \text{이고} z = wx + b \qquad (5.4)$$

- 제구실 하는지 확인
  - y가 1, o가 0.98일 때 (예측이 잘된 경우)
    - 오류 $e = -(1 \log_2 0.98 + (1-1)\log_2(1-0.98)) = 0.0291$로서 낮은 값
  - y가 1, o가 0.0001일 때 (예측이 엉터리인 경우)
    - 오류 $e = -(1 \log_2 0.0001 + (1-1)\log_2(1-0.0001)) = 13.2877$로서 높은 값

5.1.2 교차 엔트로피 목적함수

- 공정한 벌점을 부여하는지 확인 (MSE의 느린 학습 문제를 해결하나?)
  - 도함수를 구하면,

$$\begin{aligned}\frac{\partial e}{\partial w} &= -\left(\frac{y}{o}-\frac{1-y}{1-o}\right)\frac{\partial o}{\partial w}\\ &= -\left(\frac{y}{o}-\frac{1-y}{1-o}\right)x\sigma'(z)\\ &= -x\left(\frac{y}{o}-\frac{1-y}{1-o}\right)o(1-o)\\ &= x(o-y)\end{aligned}$$

$$\left.\begin{aligned}\frac{\partial e}{\partial w} &= x(o-y)\\ \frac{\partial e}{\partial b} &= (o-y)\end{aligned}\right\} \quad (5.5)$$

  - 그레이디언트를 계산해 보면, 오류가 더 큰 오른쪽에 더 큰 벌점(그레이디언트) 부과

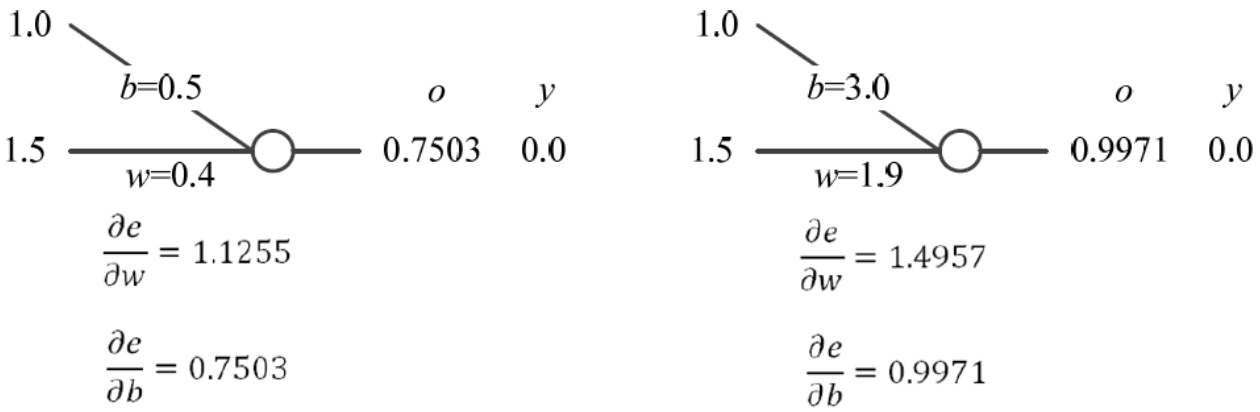

그림 5-4 교차 엔트로피를 목적함수로 사용하여 느린 학습 문제를 해결

5.1.2 교차 엔트로피 목적함수

- 식 (5.4)를 c개의 출력 노드를 가진 경우로 확장
  - 출력 벡터 $\mathbf{o} = (o_1, o_2, \cdots, o_c)^{\mathrm{T}}$인 상황으로 확장 ([그림 4-3]의 DMLP)

$$e = -\sum_{i=1,c} (y_i \log_2 o_i + (1 - y_i) \log_2 (1 - o_i)) \qquad (5.6)$$

5.1.3 softmax 활성함수와 로그우도 목적함수

- softmax 활성함수

$$o_j = \frac{e^{s_j}}{\sum_{i=1,c} e^{s_i}} \qquad (5.7)$$

  - 동작 예시
    - softmax는 max를 모방(출력 노드의 중간 계산 결과 s_i^L에서 최댓값은 더욱 활성화하고 작은 값은 억제
    - 모두 더하면 1이 되어 확률 모방

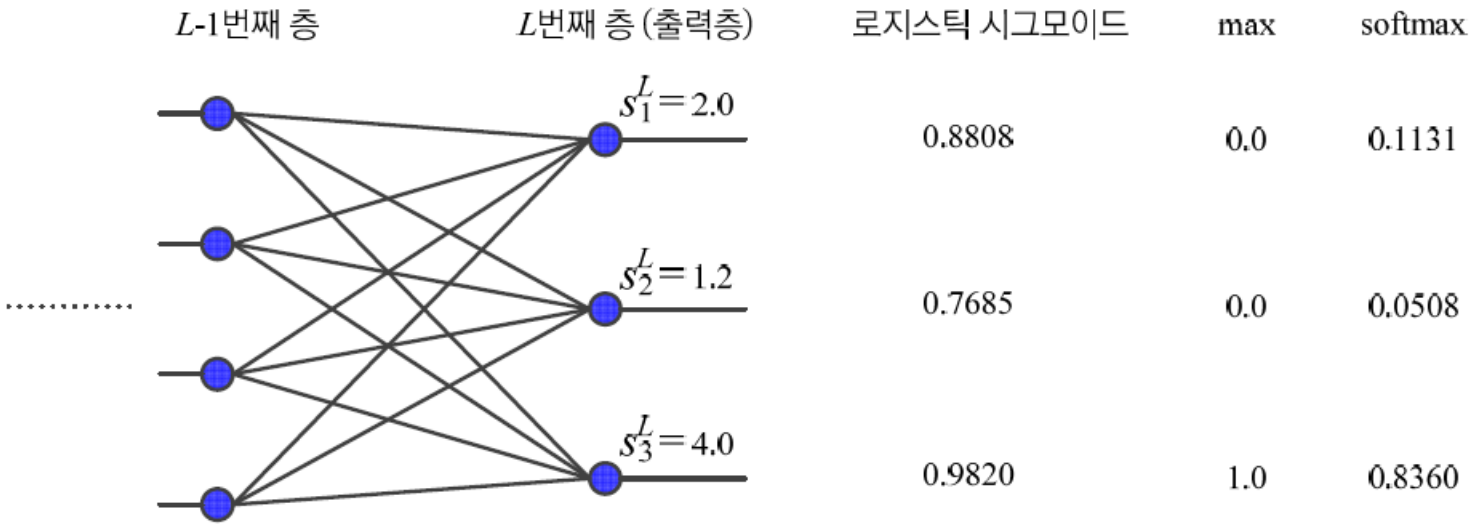

그림 5-5 출력층의 활성함수로 로지스틱 시그모이드와 softmax 비교

5.1.3 softmax 활성함수와 로그우도 목적함수

로그우도 목적함수

$$e = -\log_2 o_y \qquad (5.8)$$

- 모든 출력 노드값을 사용하는 MSE나 교차 엔트로피와 달리 o_y라는 하나의 노드만 사용
- o_y는 샘플의 레이블에 해당하는 노드의 출력값
- 동작 예시1) [그림 5-5]에서 현재 샘플이 두 번째 부류라면 o_y는 o_2 $e = -\log_2 0.0508 = 4.2990$. 잘못 분류한 셈이므로 목적함수값이 큼
- 동작 예시2) [그림 5-5]에서 현재 샘플이 세 번째 부류라면 o_y는 o_3 $e = -\log_2 0.8360 = 0.2584$. 제대로 분류한 셈이므로 목적함수값이 작음

Softmax와 로그우도

- Softmax는 최댓값이 아닌 값을 억제하여 0에 가깝게 만든다는 의도 내포
- 학습 샘플이 알려주는 부류에 해당하는 노드만 보겠다는 로그우도와 잘 어울림
- 따라서 둘을 결합하여 사용하는 경우가 많음

Cross Entropy Loss

For multi-class classification, we use the cross entropy loss.

$$H(P,Q) = -\sum_{y \in Y} P(y) \log Q(y) \qquad L = \frac{1}{N} \sum -y \log(\hat{y})$$

where $\hat{y}$ is the predicted probability and y is the correct probability (0 or 1).

```
z = torch.rand(3, 5, requires_grad=True)
hypothesis = F.softmax(z, dim=1)
print(hypothesis)
```

```
tensor([[0.2645, 0.1639, 0.1855, 0.2585, 0.1277],
        [0.2430, 0.1624, 0.2322, 0.1930, 0.1694],
        [0.2226, 0.1986, 0.2326, 0.1594, 0.1868]], grad_fn=<SoftmaxBackwa
rd>)
```

```
y = torch.randint(5, (3,)).long()
print(y)
```

```
tensor([0, 2, 1])
```

Cross Entropy Loss

```
y_one_hot = torch.zeros_like(hypothesis)
y_one_hot.scatter_(1, y.unsqueeze(1), 1)
```

```
tensor([[1., 0., 0., 0., 0.],
        [0., 0., 1., 0., 0.],
        [0., 1., 0., 0., 0.]])
```

```
cost = (y_one_hot * -torch.log(hypothesis)).sum(dim=1).mean()
print(cost)
```

```
tensor(1.4689, grad_fn=<MeanBackward1>)
```

```
# Low level
torch.log(F.softmax(z, dim=1))
```

```
tensor([[-1.3301, -1.8084, -1.6846, -1.3530, -2.0584],
        [-1.4147, -1.8174, -1.4602, -1.6450, -1.7758],
        [-1.5025, -1.6165, -1.4586, -1.8360, -1.6776]], grad_fn=<LogBackw
ard>)
```

```
# High level
F.log_softmax(z, dim=1)
```

```
tensor([[-1.3301, -1.8084, -1.6846, -1.3530, -2.0584],
        [-1.4147, -1.8174, -1.4602, -1.6450, -1.7758],
        [-1.5025, -1.6165, -1.4586, -1.8360, -1.6776]],
       grad_fn=<LogSoftmaxBackward>)
```