소프트맥스와 크로스엔트로피

최적화 관점에서 이해하기

대전대학교 컴퓨터공학과 | 조교수 박상돈

복습

회귀는 원래 이런 식이었다

$$y = a_1x_1 + a_2x_2 + a_3x_3 + b$$

입력값을 잘 섞어서 정답에 가깝게 만드는 계수들을 찾는 것

벡터로 쓰면 $\mathbf{x} = (x_1, x_2, x_3)^\top$, $\mathbf{w} = (a_1, a_2, a_3)^\top$ 라서 $\mathbf{w}^\top\mathbf{x} = a_1x_1 + a_2x_2 + a_3x_3$ 가 된다.

여기서 $^\top$ 는 transpose로, 세로 벡터를 가로로 돌렸다는 뜻이다.

쉽게 말해, 이런 오차를 줄이는 문제다:

$$\min \frac{1}{N}\sum_{i=1}^{N}(y_i - (a_1x_{i1} + a_2x_{i2} + a_3x_{i3} + b))^2$$

확장

분류는 클래스마다 식이 하나씩 필요하다

클래스가 5개면 점수 식도 5개 만든다

$$z_1 = a_1x_1 + b_1x_2 + c_1x_3 + d_1$$ 클래스 1

$$z_2 = a_2x_1 + b_2x_2 + c_2x_3 + d_2$$ 클래스 2

$$z_3 = a_3x_1 + b_3x_2 + c_3x_3 + d_3$$ 클래스 3

$$z_4 = a_4x_1 + b_4x_2 + c_4x_3 + d_4$$ 클래스 4

$$z_5 = a_5x_1 + b_5x_2 + c_5x_3 + d_5$$ 클래스 5

클래스마다 식이 하나씩 있으니, 계수도 그만큼 많이 필요하다

점수 z를 확률처럼 바꾸기

문제

- z는 그냥 계산 점수라서 아무 값이나 나올 수 있다.
- 예: $z_1 = 2.1,\ z_2 = -0.5,\ z_3 = 5.3, \ldots$
- 그래서 바로 "확률"이라고 부르기 어렵다.

→

해결: 소프트맥스

- 값을 0과 1 사이로 눌러 준다.
- 클래스별 값을 다 더하면 정확히 1이 된다.
- 그래서 "각 클래스일 가능성"처럼 읽을 수 있다.

$$\text{softmax}(z_k) = \frac{e^{z_k}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4} + e^{z_5}}$$

쉽게 말해, 큰 z는 더 큰 확률을 갖게 하고 전체 합은 1로 맞춘다.

손실은 결국 정답 확률만 보면 된다

$$\mathcal{L} = -\log(p_{\text{correct}})$$

p_{correct}는 모델이 정답 클래스에 준 확률이다.
예: 정답이 클래스 4이고 softmax 결과 벡터 $\mathbf{p} = (0.05, 0.10, 0.15, 0.60, 0.10)$이면 $p_{\text{correct}} = 0.60$

$-\log$를 붙이면 뭐가 좋아질까?

정답 클래스에 준 확률	$-\log$(확률)	해석
0.95	0.05	거의 맞음 → 손실이 아주 작다
0.50	0.69	애매함 → 손실이 중간 정도다
0.01	4.60	거의 틀림 → 손실이 매우 크다

구체적 예시

샘플 하나를 넣고 따라가 보자

입력 벡터 $\mathbf{x} = (5, 3, 7)^\top$ | 정답: 클래스 4

$$
\begin{aligned}
z_1 &= a_{11}(5) + a_{12}(3) + a_{13}(7) + b_1 \\
z_2 &= a_{21}(5) + a_{22}(3) + a_{23}(7) + b_2 \\
z_3 &= a_{31}(5) + a_{32}(3) + a_{33}(7) + b_3 \\
z_4 &= a_{41}(5) + a_{42}(3) + a_{43}(7) + b_4 \\
z_5 &= a_{51}(5) + a_{52}(3) + a_{53}(7) + b_5
\end{aligned}
$$

정답이 클래스 4라면, 일단 z_4를 가장 중요하게 보면 된다.

구체적 예시

이 샘플에서 손실이 어떻게 나오나

① 소프트맥스 통과

$$\frac{e^{z_4}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4} + e^{z_5}}$$

정답이 클래스 4이면 이 값이 바로 $p_{\text{correct}} = p_4$다.
예를 들어 이 값이 0.60이라면 정답 클래스 확률은 60%라는 뜻이다.

② $-\log$ 씌우기

$$\mathcal{L}_{\text{sample}} = -\log\left(p_{\text{correct}}\right) = -\log(0.60) \approx 0.51$$

이것이 이 샘플의 손실이다.
반대로 $p_{\text{correct}} = 0.05$면 손실은 약 3.00이 되어 훨씬 크게 혼난다.

이걸 모든 샘플에 대해 반복하고 평균 내면 전체 손실이 된다.

일반 수식

이 흐름을 수식으로 쓰면 이런 모양이다

$$z_k(\mathbf{x}_i) = a_{k1}x_{i1} + a_{k2}x_{i2} + a_{k3}x_{i3} + b_k, \qquad k = 1, \ldots, 5$$

$$\min_{\{a_{k1}, a_{k2}, a_{k3}, b_k\}} -\frac{1}{N}\sum_{i=1}^{N} \log\left(\frac{e^{z_{y_i}(\mathbf{x}_i)}}{\sum_{j=1}^{5} e^{z_j(\mathbf{x}_i)}}\right)$$

$\mathbf{x}_i = (x_{i1}, x_{i2}, x_{i3})^\top$: i번째 샘플의 입력 벡터
$z_{y_i}(\mathbf{x}_i)$: 그 샘플에서 정답 클래스가 받은 점수
분모: 모든 클래스 점수를 e^z로 바꾼 뒤 더한 값

왜 정확도 말고 손실을 보나

정확도는 바로 미분하기 어렵다

- 맞고 틀림만 보니 값이 뚝뚝 바뀐다.
- 계수를 조금 고쳐도 정확도가 안 바뀔 수 있다.
- 그래서 어느 쪽으로 움직일지 잡기 어렵다.

손실은 부드럽게 변해서 다루기 쉽다

- 틀린 정도를 숫자로 보여 준다.
- 확률이 조금 바뀌면 손실도 같이 조금 바뀐다.
- 그래서 경사 하강법을 쓰기 좋다.

$$J(W,b) = \frac{1}{N}\sum_{i=1}^{N} \ell_i + \alpha R(W)$$

ℓ_i: i번째 샘플이 얼마나 틀렸는지, $R(W)$: 큰 가중치에 주는 벌점, α: 그 벌점의 세기

결국 모델은 "평균 손실은 줄이고, 가중치는 너무 커지지 않게" 학습한다.

규제는 큰 가중치에 벌점을 주는 장치다

규제가 없으면

- 훈련 데이터를 억지로 맞추려고 가중치가 불필요하게 커질 수 있다.
- 데이터가 조금만 바뀌어도 예측이 크게 흔들릴 수 있다.
- 훈련 데이터엔 잘 맞아도 새 데이터엔 약해질 수 있다.

규제를 넣으면

- 큰 가중치에 벌점을 줘서 더 단순한 해를 선호한다.
- 그래서 과하게 휘는 모델을 막는 데 도움이 된다.
- 하지만 α가 너무 크면 반대로 너무 단순해져 과소적합된다.

$$R(W) = \frac{1}{2}\|W\|_2^2, \qquad R(W) = \|W\|_1$$

왼쪽은 L2 규제(기본), 오른쪽은 L1 규제다.
$\|W\|_2^2$는 모든 가중치의 제곱합, $\|W\|_1$은 절댓값 합이다.
`SGDClassifier`의 기본 규제는 보통 L2이고, 평균 손실에 $\frac{\alpha}{2}\|W\|_2^2$를 더한 목적함수를 줄인다고 생각하면 된다.

이진 분류의 log_loss를 쉬운 말로 보면

$$z = \mathbf{w}^\top \mathbf{x} + b, \qquad p = \sigma(z) = \frac{1}{1 + e^{-z}}$$

$\mathbf{x} = (x_1, x_2, x_3)^\top$는 입력 벡터, $\mathbf{w} = (w_1, w_2, w_3)^\top$는 가중치 벡터다.
그래서 $\mathbf{w}^\top \mathbf{x} = w_1 x_1 + w_2 x_2 + w_3 x_3$는 "각 입력에 가중치를 곱해 더한 값"이라고 보면 된다.

$$\ell(y, p) = -\Big[y \log p + (1 - y) \log(1 - p)\Big], \qquad y \in \{0, 1\}$$

y: 실제 정답 라벨, p: 모델이 "1일 확률"이라고 본 값

정답이 1일 때

$\ell = -\log p$
정답인데 p가 작으면 크게 혼난다.
반대로 p가 크면 손실은 거의 없다.

정답이 0일 때

$\ell = -\log(1 - p)$
정답이 0인데 p를 크게 주면,
손실이 크게 올라간다.

이 식은 사실 두 경우를 한 줄로 합친 것이다

$$\ell(y, p) = -\Big[y \log p + (1-y) \log(1-p)\Big]$$

여기서 y와 $(1-y)$는 스위치처럼 작동한다.
정답이 1이면 첫 항만 남고, 정답이 0이면 둘째 항만 남는다.

$y = 1$ 일 때

$$\ell = -\big[1 \cdot \log p + 0 \cdot \log(1-p)\big] = -\log p$$

예를 들어 $p = 0.9$면 손실은 약 0.10이고,
$p = 0.1$이면 손실은 약 2.30으로 커진다.

$y = 0$ 일 때

$$\ell = -\big[0 \cdot \log p + 1 \cdot \log(1-p)\big] = -\log(1-p)$$

예를 들어 $p = 0.1$이면 손실은 약 0.10이고,
$p = 0.9$이면 손실은 약 2.30으로 커진다.

즉, 이 식은 "정답 쪽 확률이 낮을수록 크게 벌준다"를 한 줄로 쓴 것이다.

SGDClassifier도 결국 같은 손실을 쓴다

$$\ell(y, z) = \log\big(1 + \exp(-yz)\big), \qquad y \in \{-1, +1\}$$

표기만 다를 뿐 뜻은 같다. 구현에서는 y를 $-1, +1$로 두는 경우가 많다.

중간 계산은 이 한 줄을 풀어쓴 것이다

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

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

$$y = +1 \quad \Rightarrow \quad \ell = \log(1 + e^{-z}) = -\log \sigma(z)$$

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

$$y = -1 \quad \Rightarrow \quad \ell = \log(1 + e^{z}) = -\log(1 - \sigma(z))$$

여기서 점프가 있는 것이 아니라 $\sigma(z) = \frac{1}{1+e^{-z}}$를 대입한 뒤 중간 줄을 생략했던 것이다.

특히 $\frac{e^{-z}}{1+e^{-z}} = \frac{1}{1+e^{z}}$는 양변을 나눈 것이 아니라 분자와 분모에 같은 수 e^{z}를 곱해서 정리한 것이다.

크로스엔트로피

다중 분류는 softmax와 one-hot으로 본다

$$p_k = \text{softmax}(\mathbf{z})_k = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

여기서 $\mathbf{z} = (z_1, \ldots, z_K)$는 점수 벡터, $\mathbf{p} = (p_1, \ldots, p_K)$는 확률 벡터다.

$$\ell(\mathbf{y}, \mathbf{p}) = -\sum_{k=1}^{K} y_k \log p_k$$

y는 정답 클래스만 1로 표시한 one-hot 벡터라고 생각하면 된다.

크로스엔트로피

그래서 결국 정답 클래스 확률 하나만 남는다

정답 클래스가 c라면 $y_c = 1$이고 나머지는 0이다.

$$\ell(\mathbf{y}, \mathbf{p}) = -\sum_{k=1}^{K} y_k \log p_k \quad \Longrightarrow \quad \ell = -\log p_c = -\log p_{\text{correct}}$$

여기서 p_c와 p_{correct}는 둘 다 "정답 클래스에 준 확률"이라는 같은 뜻이다.

정답 클래스 확률이 높아질수록 손실은 작아진다. 그래서 모델은 정답 클래스 확률을 키우는 쪽으로 학습된다.

왜 기울기의 반대 방향으로 갈까

여기서 우리가 줄이고 싶은 전체 목적함수는

$$J(\mathbf{w}, b) = \frac{1}{N}\sum_{i=1}^{N} \ell_i + \frac{\alpha}{2}\|\mathbf{w}\|_2^2$$

로 둘 수 있다. 즉, "전체 데이터의 평균 손실 + 규제"다.
∇J는 현재 위치에서 J가 가장 빨리 커지는 방향이므로, 손실을 줄이려면 그 반대 방향인 $-\nabla J$로 조금 움직여야 한다.

1차원으로 생각하면

$\frac{dJ}{dw} > 0$이면 오른쪽으로 갈수록 J가 커지므로 왼쪽으로 간다.
$\frac{dJ}{dw} < 0$이면 오른쪽으로 갈수록 J가 작아지므로 오른쪽으로 간다.

식으로 보면

여기서 $\theta = (\mathbf{w}, b)$는 파라미터 전체를 묶어 쓴 것이고, Δ는 아주 작은 이동량이다.

$$J(\theta + \Delta) \approx J(\theta) + \nabla J(\theta)^\top \Delta, \qquad \theta = (\mathbf{w}, b)$$

$$\Delta = -\eta \nabla J \Rightarrow J(\theta+\Delta) - J(\theta) \approx -\eta \|\nabla J(\theta)\|^2 < 0$$

그래서 작은 걸음에서는 J가 줄어드는 쪽으로 간다.

수식으로 유도하면 $-\nabla J$가 나온다

현재 점에서 아주 조금 움직인다고 하면, 손실 변화는 1차항으로

$$J(\theta + \Delta) - J(\theta) \approx \nabla J(\theta)^\top \Delta, \qquad \theta = (\mathbf{w}, b)$$

로 쓸 수 있다. 즉, 지금은 "아주 작은 걸음 Δ를 어느 방향으로 놓을까"를 보는 단계다.

이제 크기가 같은 작은 걸음 $\|\Delta\| = \varepsilon$들 중에서 J를 가장 많이 줄이는 방향을 고르자.
그러면 $\nabla J(\theta)^\top \Delta$를 가장 작게 만들면 되고, 코시-슈바르츠 부등식으로

$$\nabla J(\theta)^\top \Delta \geq -\|\nabla J(\theta)\| \, \|\Delta\| = -\varepsilon \|\nabla J(\theta)\|$$

이다. 등호는

$$\Delta = -\varepsilon \frac{\nabla J(\theta)}{\|\nabla J(\theta)\|}$$

일 때 성립하므로, 가장 빨리 내려가는 방향은 $-\nabla J$다.

걸음 크기 ε를 학습률 η에 흡수하면 실제 업데이트를 $\Delta = -\eta \nabla J$로 쓸 수 있다.

그래서 식으로 쓰면 이렇게 조금씩 고친다

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} J, \qquad b \leftarrow b - \eta \nabla_b J$$

여기서 $J(\mathbf{w}, b)$는 줄이고 싶은 전체 목적함수다. 이 발표에서는 보통 "평균 손실 + 규제"를 뜻한다.

$$J(\mathbf{w}, b) \approx \frac{1}{N} \sum_{i=1}^{N} \ell_i + \frac{\alpha}{2} \|\mathbf{w}\|_2^2$$ 처럼 생각하면 된다.

η는 한 번에 얼마나 움직일지 정하는 값이다. 너무 크면 튀고, 너무 작으면 느리다.

배치 GD
한 번에 전체 데이터를 다 보고 업데이트

미니배치 GD
몇 개씩 묶어서 한 번에 업데이트

확률적 GD
샘플 하나씩 보고 바로 업데이트

에포크(epoch)는 훈련 데이터를 한 바퀴 도는 것이고, `max_iter`는 그 반복 횟수다.

$p - y$는 얼마나, 어느 쪽으로 고칠지 알려준다

$\frac{\partial \ell}{\partial z} = p - y$는 기울기(gradient)다.
하지만 실제 업데이트는 기울기를 더하는 것이 아니라 $z \leftarrow z - \eta \frac{\partial \ell}{\partial z}$처럼 뺀다.
따라서 $p - y < 0$이면 z는 커지고, $p - y > 0$이면 z는 작아진다.

이진 분류

$$\frac{\partial \ell}{\partial z} = p - y$$

$$\nabla_{\mathbf{w}} \ell = (p - y)\mathbf{x}, \qquad \frac{\partial \ell}{\partial b} = p - y$$

예: $y = 1,\ p = 0.2$이면 $p - y = -0.8$이므로 $z \leftarrow z - \eta(-0.8) = z + 0.8\eta$라서 올라간다.
예: $y = 0,\ p = 0.9$이면 $z \leftarrow z - \eta(0.9)$라서 내려간다.

다중 분류

$$\frac{\partial \ell}{\partial z_k} = p_k - y_k$$

$$\nabla_{\mathbf{w}_k} \ell = (p_k - y_k)\mathbf{x}$$

예: $\mathbf{p} = (0.10, 0.20, 0.70),\ \mathbf{y} = (1, 0, 0).$
정답 클래스는 $0.10 - 1 = -0.90$이므로 $z_1 \leftarrow z_1 - \eta(-0.90) = z_1 + 0.90\eta$라서 올라간다.
오답 클래스는 $0.20,\ 0.70$이므로 내려간다.

SGDClassifier가 한 번 학습할 때 하는 일

1. 샘플 벡터 $\mathbf{x}_i$ 하나(또는 샘플 묶음 하나)를 고른다.
2. 현재 식으로 점수 $z = \mathbf{w}^\top \mathbf{x}_i + b$를 계산한다.
3. `log_loss`라면 이 점수를 확률로 바꾼다.
4. 손실과 기울기를 계산한다.
5. 가중치를 아주 조금 고친다.

기본 설정이 L2 규제라면

$$R(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2 \quad \Rightarrow \quad \frac{\partial R}{\partial \mathbf{w}} = \mathbf{w}$$

$$\mathbf{w} \leftarrow \mathbf{w} - \eta\big((p-y)\mathbf{x} + \alpha\mathbf{w}\big), \qquad b \leftarrow b - \eta(p-y)$$

데이터가 주는 수정량은 $(p-y)\mathbf{x}$이고, 규제가 주는 수정량은 $\alpha\mathbf{w}$다.
즉, 정답에 맞게 고치되 가중치 크기는 너무 커지지 않게 같이 눌러 준다.

SGDClassifier

실습할 때 기억할 점

요즘 표기

SGDClassifier(loss="log_loss", penalty=..., max_iter=100, tol=None)

보통 규제는 L2를 많이 쓰고, 코드 옵션 이름은 소문자 l + 숫자 2인 "l2"다.

옛 코드의 `loss='log'`는 지금 `loss='log_loss'`로 쓰면 된다.

`fit` vs `partial_fit`

- `fit`: 정해 둔 에포크 수만큼 한 번에 학습
- `partial_fit`: 부를 때마다 이어서 조금 더 학습
- 첫 호출에는 보통 전체 클래스 목록이 필요

`penalty`와 규제

- L2: 큰 가중치를 제곱으로 벌점. 기본값이고 가장 무난하다.
- L1: 절댓값으로 벌점. 일부 가중치를 0으로 만들기 쉽다.
- `alpha`가 클수록 규제가 강해지고, 특성 표준화가 중요하다.

비교

회귀와 분류, 뼈대는 같다

다중선형회귀 (MSE)

$$\min \frac{1}{N}\sum_{i=1}^{N}\left(y_i - f(\mathbf{x}_i)\right)^2$$

$$f(\mathbf{x}) = a_1x_1 + a_2x_2 + a_3x_3 + b$$

선형식 1개
예측값과 정답의 차이를 제곱해서 본다

분류 (크로스엔트로피)

$$\min -\frac{1}{N}\sum_{i=1}^{N}\log p_{\text{correct}}(\mathbf{x}_i)$$

$$\mathbf{p}(\mathbf{x}) = \text{softmax}(W\mathbf{x} + \mathbf{b})$$

선형식 여러 개 + 소프트맥스
여기서 $p_{\text{correct}}(\mathbf{x}_i)$는 샘플 $\mathbf{x}_i$의 정답 클래스 확률이다

공통 구조: 샘플마다 손실을 만들고 평균낸 뒤, 그 값을 가장 작게 만드는 파라미터를 찾는다

정리

1. 클래스가 5개면 점수 식도 5개 만든다
2. 각 식의 출력 z를 softmax로 확률처럼 바꾼다
3. 정답 확률이 낮으면 손실이 커진다
4. 평균 손실과 규제를 함께 줄이도록 파라미터를 조금씩 고친다

"결국 뼈대는 회귀와 비슷하고, 마지막 해석만 확률로 바뀐다."