확률적 경사 하강법

우도경

2025-07-16

INDEX

- 점진적인 학습
- SGDClassifier
- 에포크와 과대/과소적합
- 마무리

1. 점진적인 학습

- 훈련 데이터가 한 번에 준비되는 것이 아니라 조금씩 전달되는 상황

  - 기존의 훈련 데이터에 새로운 데이터를 추가하여 모델을 매일매일 다시 훈련하면?
    - 시간이 지날수록 데이터가 늘어나므로 서버를 늘려야 함. 지속 가능한 방법은 아님
  - 새로운 데이터를 추가할 때, 이전 데이터를 버림으로써 훈련 데이터 크기를 일정하게 유지
    - 데이터를 버릴 때, 다른 데이터에 없는 중요한 정보가 유실될 가능성 존재

  - 훈련한 모델을 버리지 않고 새로운 데이터에 대해서만 조금씩 더 훈련 가능하다면?

이러한 훈련 방식을 점진적 학습이라고 부름. 대표적으로 확률적 경사하강법이 있음

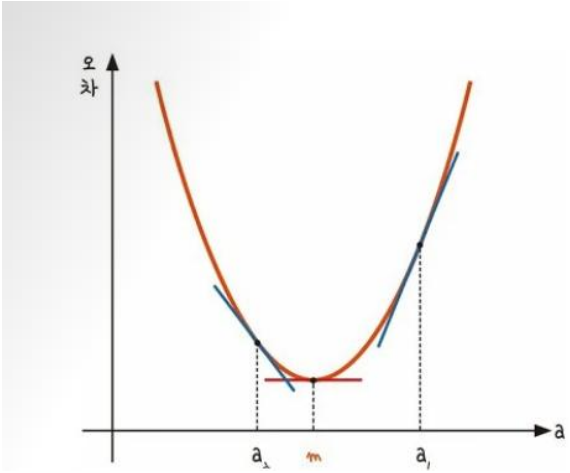
1. 점진적인 학습

- 손실함수
  - “모델이 틀린 정도"를 수치로 나타낸 함수
  - 모델이 예측한 값과 실제 정답 사이의 오차를 수치화하여, 이 값을 최소화하는 방향으로 학습이 진행됨

- 선형회귀에서의 손실함수

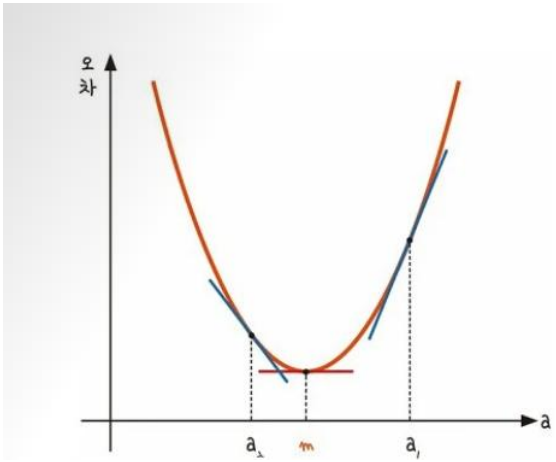
$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - y_i)^2}$$

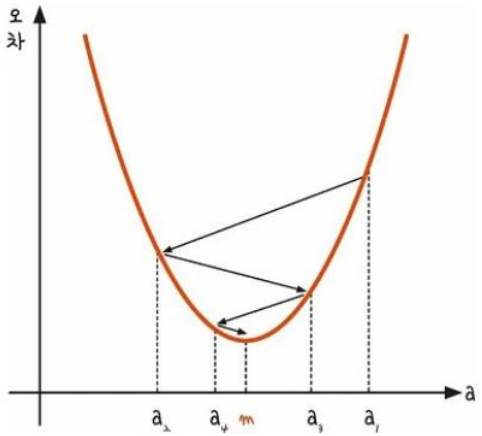
- p_i: 예측값
- y_i: 실제값
- n: 데이터 개수

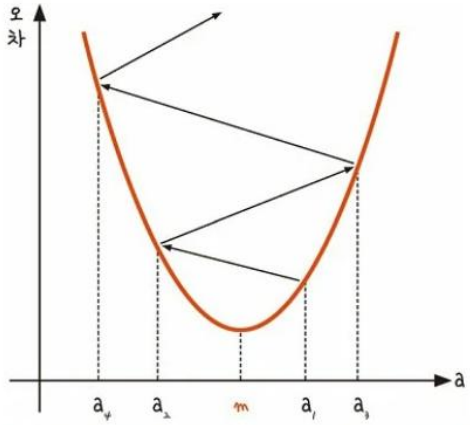


1. 점진적인 학습

- 경사하강법





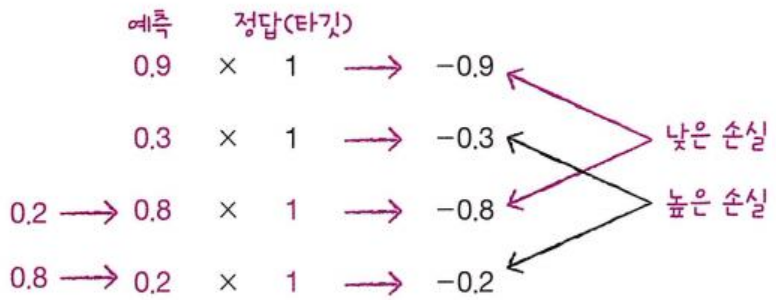


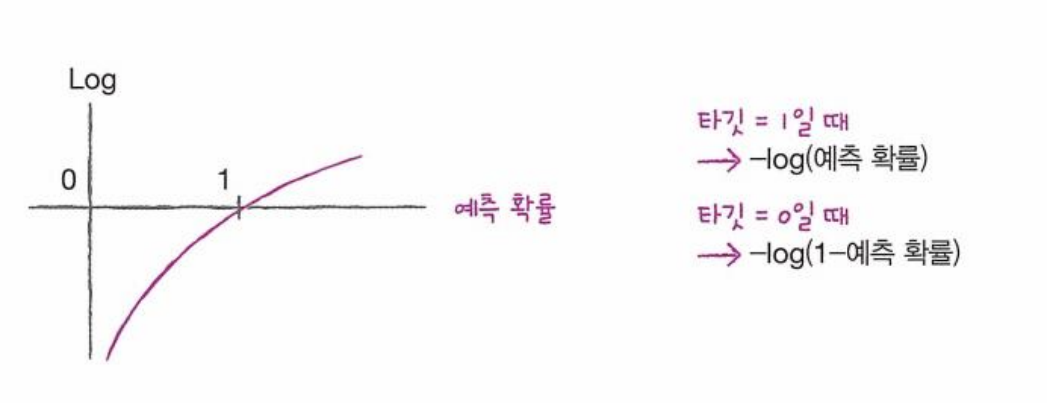
1. a_1에서 미분을 구한다.
2. 구한 기울기의 반대방향(기울기가 +면 음의 방향, -면 양의 방향)으로 특정 값만큼 이동시킨 a_2 에서 미분을 구한다.
3. 앞에서 구한 미분 값이 0이 아니면 1과 2 과정을 반복한다.

* 단, 학습률을 너무 크게 잡으면 한 점으로 수렴하지 않고 발산한다.

1. 점진적인 학습

- 로지스틱 손실함수





$$\mathcal{L} = -\left[y \cdot \log(\hat{y}) + (1-y) \cdot \log(1-\hat{y})\right]$$

- y: 실제 정답 (0 또는 1)
- $\hat{y}$: 예측값 (0 ~ 1 사이 확률)

1. 점진적인 학습

- MSE vs 로지스틱 손실함수

✅ 1. "곱을 로그로 바꾸면 계산과 최적화가 쉬워진다"는 뜻

이 말은 **수학적으로도, 컴퓨터 계산상으로도** 매우 중요한 의미가 있어.

📌 상황 설정: 우도(likelihood)

로지스틱 회귀에서 우리는 다음을 최대화하려 해:

$$\mathcal{L}(\theta) = \prod_{i=1}^{n} P(y^{(i)} \mid x^{(i)}; \theta)$$

- 즉, 모델이 각 샘플을 맞출 확률을 **전부 곱해서** 하나의 성능으로 봐
- 하지만 이 **곱**은 계산하기 매우 어렵고, 특히 확률이 작으면 **수치적으로 0에 수렴**해버려

🪛 해결책: 로그(log)를 취한다

$$\log \mathcal{L}(\theta) = \log \prod_{i=1}^{n} P(y^{(i)} \mid x^{(i)}; \theta) = \sum_{i=1}^{n} \log P(y^{(i)} \mid x^{(i)}; \theta)$$

- **곱 → 합으로 변환**되면서 계산이 쉬워지고
- **최적화 알고리즘(GD 등)**이 훨씬 효율적으로 작동
- **수치 안정성(numerical stability)**도 보장됨

✅ 2. MSE vs 로지스틱 손실 (Cross Entropy): 극단적으로 틀린 경우 어떻게 다를까?

이건 진짜로 Cross Entropy가 얼마나 **날카롭게 오차를 인식하는지** 보여주는 부분이야.

🔁 다시 비교해보자

✅ **MSE 손실:**

예측값 $\hat{y}$	실제값 $y = 1$	MSE 손실
0.49	1	$(1 - 0.49)^2 = 0.2601$
0.01	1	$(1 - 0.01)^2 = 0.9801$

→ 틀렸긴 한데, **확신을 완전히 잘못했음에도 벌점 차이가 3.7배밖에 안 됨**

✅ **로지스틱 손실 (Cross Entropy):**

$$\mathcal{L} = -\left[y \log \hat{y} + (1 - y) \log(1 - \hat{y})\right]$$

예측값 $\hat{y}$, 실제값 $y = 1$일 때:

예측값 $\hat{y}$	Cross Entropy 손실
0.49	$-\log(0.49) \approx 0.71$
0.01	$-\log(0.01) \approx 4.61$

→ 벌점 차이가 **6.5배 이상**,
→ 특히 **확신을 가지고 틀릴수록 손실이 급격히 커짐**

2. SGDClassifier

- 확률적 경사 하강법

```
[ ]  import pandas as pd

     fish = pd.read_csv('https://bit.ly/fish_csv_data')
```

```
[ ]  fish_input = fish[['Weight','Length','Diagonal','Height','Width']].to_numpy()
     fish_target = fish['Species'].to_numpy()
```

```
[ ]  from sklearn.model_selection import train_test_split

     train_input, test_input, train_target, test_target = train_test_split(
         fish_input, fish_target, random_state=42)
```

```
[ ]  from sklearn.preprocessing import StandardScaler

     ss = StandardScaler()
     ss.fit(train_input)
     train_scaled = ss.transform(train_input)
     test_scaled = ss.transform(test_input)
```

```
[ ]  from sklearn.linear_model import SGDClassifier
```

```
[ ]  sc = SGDClassifier(loss='log_loss', max_iter=10, random_state=42)
     sc.fit(train_scaled, train_target)

     print(sc.score(train_scaled, train_target))
     print(sc.score(test_scaled, test_target))
```

```
0.773109243697479
0.775
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_stochastic_gradient.py:744:
  warnings.warn(
```

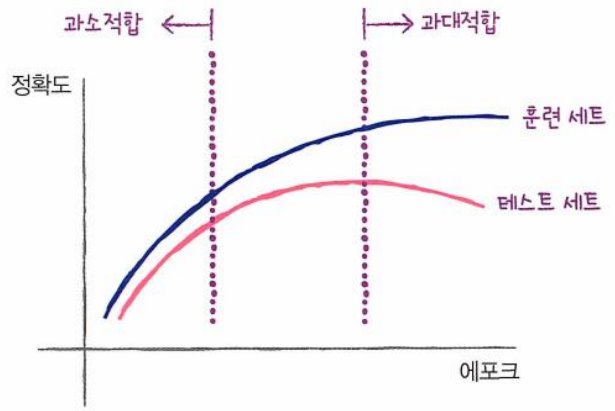
```
[ ]  sc.partial_fit(train_scaled, train_target)

     print(sc.score(train_scaled, train_target))
     print(sc.score(test_scaled, test_target))
```

```
0.8151260504201681
0.85
```

- Partial_fit() 사용 시, 1 에포크씩 이어서 훈련
- 에포크를 한 번 더 실행하니 정확도 향상된 결과
- 그렇다면 얼마나 더 훈련해야 할까?

2. 에포크와 과대/과소적합

- 에포크와 모델의 정확도 간의 관계



- 에포크가 진행될수록 훈련 세트 점수는 꾸준히 증가하지만, 테스트 세트 점수는 어느 순간 감소하기 시작
- 앞서 말한 지점이 모델이 과대적합되기 시작하는 지점
- 과대적합이 시작하기 전에 훈련을 멈추는 것을 조기 종료라고 함

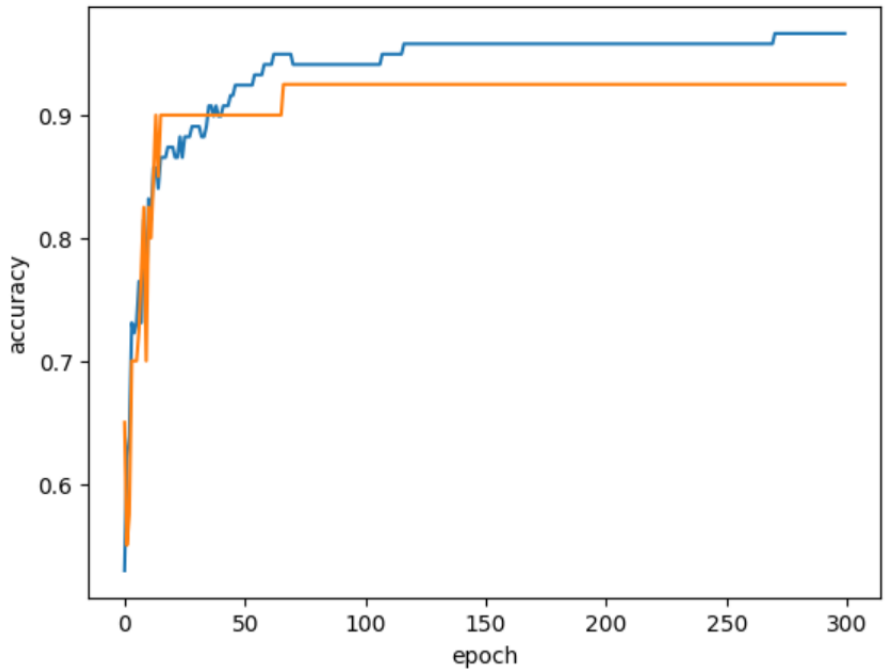
2. 에포크와 과대/과소적합

- 에포크와 모델의 정확도 간의 관계

```
[ ]  import numpy as np

     sc = SGDClassifier(loss='log_loss', random_state=42)

     train_score = []
     test_score = []

     classes = np.unique(train_target)
```

```
[ ]  for _ in range(0, 300):
         sc.partial_fit(train_scaled, train_target, classes=classes)

         train_score.append(sc.score(train_scaled, train_target))
         test_score.append(sc.score(test_scaled, test_target))
```

```
[ ]  import matplotlib.pyplot as plt

     plt.plot(train_score)
     plt.plot(test_score)
     plt.xlabel('epoch')
     plt.ylabel('accuracy')
     plt.show()
```



- 100번째 에포크 이후 훈련 세트와 테스트 세트의 점수 차이가 벌어지기 시작함

2. 에포크와 과대/과소적합

- 모델 훈련

```
[ ]  sc = SGDClassifier(loss='log_loss', max_iter=100, tol=None, random_state=42)
     sc.fit(train_scaled, train_target)

     print(sc.score(train_scaled, train_target))
     print(sc.score(test_scaled, test_target))
```

```
0.957983193277311
0.925
```

- SGDClassfier는 일정 에포크 동안 성능이 향상되지 않으면 더 훈련하지 않고 자동으로 멈춤
- tol 매개변수에서 향상될 최솟값 지정
- tol 매개변수를 None으로 지정하면, max_iter = 100 만큼 무조건 반복

3. 마무리

- 점진적 학습을 위한 확률적 경사 하강법
  - 대량의 데이터를 이용할 때, 데이터를 한 번에 모두 컴퓨터 메모리에 읽을 수 없음
  - 따라서, 데이터를 조금씩 사용해 점진적으로 학습하는 방법이 필요
  - 확률적 경사하강법은 손실 함수라는 산을 정의하고 가장 가파른 경사를 따라 조금씩 내려오는 알고리즘
  - 이를 충분히 반복하여 훈련하면 훈련 세트에서 높은 점수를 얻는 모델을 만들 수 있음

Thank You

참고문헌

- 박해선. 혼자 공부하는 머신러닝 + 딥러닝. 한빛미디어, 2025.
- 조태호. 모두의 딥러닝 - 누구나 쉽게 이해하는 딥러닝 개정3판. 길벗, 2022.