4장 오차 수정하기

-경사 하강법

목차

오차 수정하기: 경사 하강법

- a를 무한대로 키우면 오차도 무한대로 커지고 a를 무한대로 작게 해도 역시 오차도 무한대로 커지는 이러한 관계는 이차 함수 그래프로 표현할 수 있음

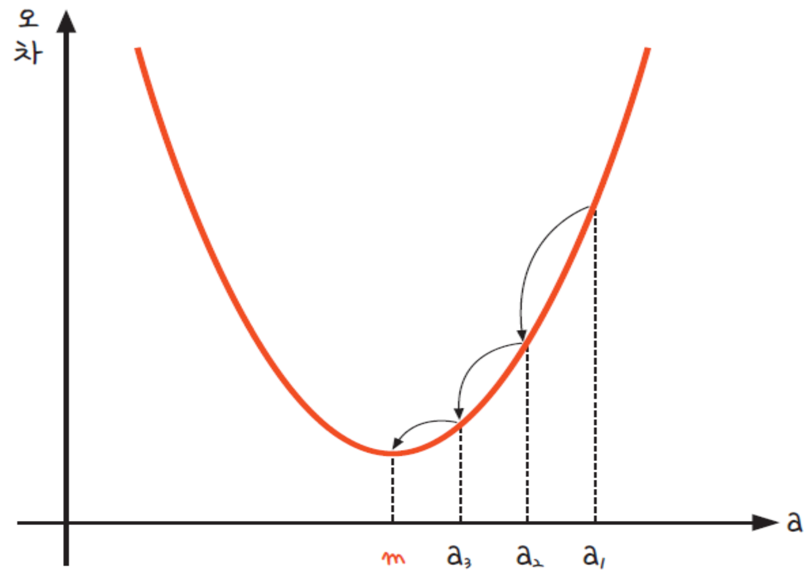

그림 4-1 기울기 a와 오차와의 관계: 적절한 기울기를 찾았을 때 오차가 최소화된다.

오차 수정하기: 경사 하강법

- 컴퓨터를 이용해 m의 값을 구하려면 임의의 한 점(a_1)을 찍고 이 점을 m에 가까운 쪽으로 점점 이동($a_1 \rightarrow a_2 \rightarrow a_3$)시키는 과정이 필요함
- 경사 하강법(gradient descent) :
  그래프에서 오차를 비교하여 가장 작은 방향으로 이동시키는 방법이 있는데 바로 미분 기울기를 이용

1 경사 하강법의 개요

- $y = x^2$ 그래프에서 x에 다음과 같이 a_1, a_2 그리고 m을 대입하여 그 자리에서 미분하면 그림 4-2처럼 각 점에서의 순간 기울기가 그려짐

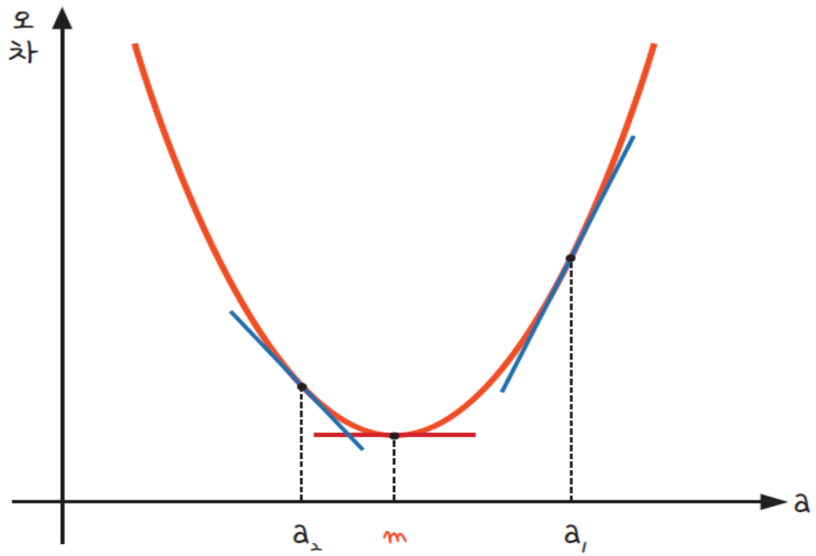

그림 4-2 순간 기울기가 0인 점이 곧 우리가 찾는 최솟값 m이다.

1 경사 하강법의 개요

- 여기서 눈여겨 봐야 할 것은 우리가 찾는 최솟값 m에서의 순간 기울기임
- 그래프가 이차 함수 포물선이므로 꼭짓점의 기울기는 x축과 평행한 선이 됨
- 즉, 기울기가 0임
- 우리가 할 일은 '미분 값이 0인 지점'을 찾는 것이 됨

1 . a_1 에서 미분을 구함

2 . 구해진 기울기의 반대 방향(기울기가 +면 음의 방향, -면 양의 방향)으로 얼마간 이동시킨 a_2에서 미분을 구함(그림 4-3 참조).

3 . 위에서 구한 미분 값이 0이 아니면 위 과정을 반복함

1 경사 하강법의 개요

- 그림 4-3처럼 기울기가 0인 한 점(m)으로 수렴함

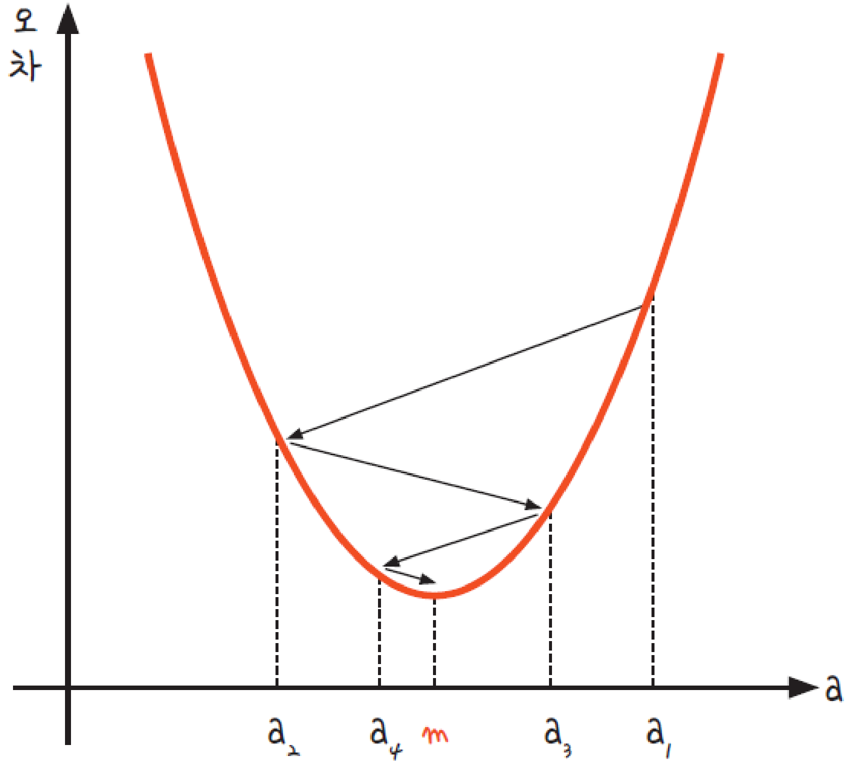

그림 4-3 최솟점 m을 찾아가는 과정

1 경사 하강법의 개요

- 경사 하강법 :

  이렇게 반복적으로 기울기 a를 변화시켜서 m의 값을 찾아내는 방법을 말함

2 학습률

- 기울기의 부호를 바꿔 이동시킬 때 적절한 거리를 찾지 못해 너무 멀리 이동시키면 a값이 한 점으로 모이지 않고 그림 4-4처럼 위로 치솟아 버림

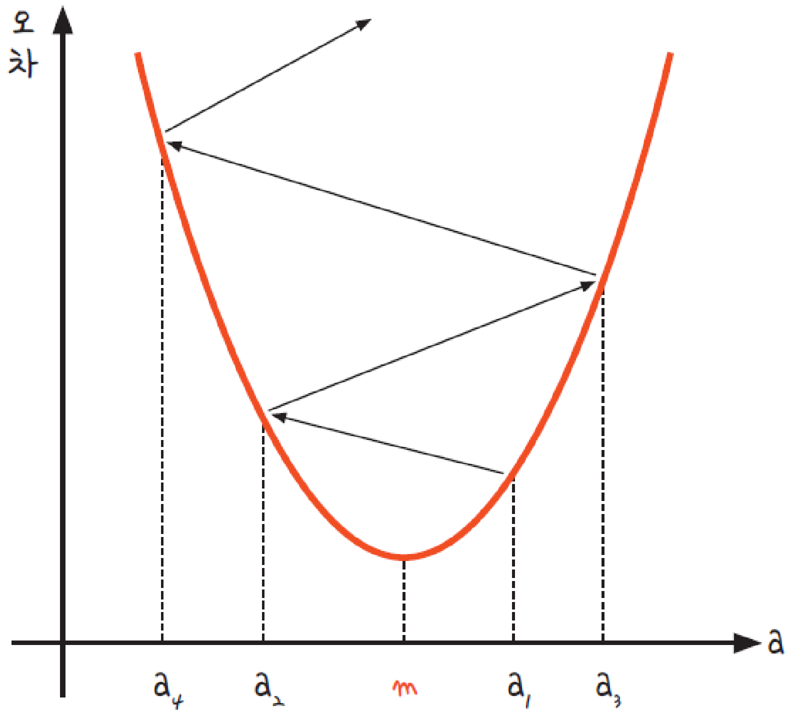

그림 4-4 학습률을 너무 크게 잡으면 한 점으로 수렴하지 않고 발산한다.

2 학습률

- 학습률 :

  어느 만큼 이동시킬지를 신중히 결정해야 하는데, 이때 이동 거리를 정해 주는 것

- 딥러닝에서 학습률의 값을 적절히 바꾸면서 최적의 학습률을 찾는 것은 중요한 최적화 과정 중 하나임

2 학습률

- 경사 하강법
  - 오차의 변화에 따라 이차 함수 그래프를 만들고 적절한 학습률을 설정해 미분 값이 0인 지점을 구하는 것
  - y 절편 b의 값도 이와 같은 성질을 가지고 있음
  - b 값이 너무 크면 오차도 함께 커지고, 너무 작아도 오차가 커짐
  - 최적의 b값을 구할 때 역시 경사 하강법을 사용함

3 코딩으로 확인하는 경사 하강법

- 최솟값을 구하기 위해서는 이차 함수에서 미분을 해야 함
- 그 이차 함수는 평균 제곱 오차를 통해 나온다는 것임
- 평균 제곱 오차의 식을 다시 옮겨 보면 다음과 같음

$$\frac{1}{n}\sum(\hat{y}_i - y_i)^2$$

- 여기서 $\hat{y}_i$은 x_i 를 집어 넣었을 때의 값이므로 $y_i = ax_i + b$ 를 대입하면 다음과 같이 바뀜

$$\frac{1}{n}\sum((ax_i + b) - y_i)^2$$

3 코딩으로 확인하는 경사 하강법

- 이 값을 미분할 때 우리가 궁금한 것은 a 와 b라는 것에 주의해야 함
- 식 전체를 미분하는 것이 아니라 필요한 값을 중심으로 미분해야 하기 때문임

$$a\text{로 편미분 한 결과} = \frac{2}{n}\sum(ax_i + b - y_i)x_i$$

$$b\text{로 편미분 한 결과} = \frac{2}{n}\sum(ax_i + b - y_i)$$

3 코딩으로 확인하는 경사 하강법

TIP

a로 편미분한 결과 유도 과정

$$\begin{aligned}\frac{a}{\partial a}MSE(a,b) &= \frac{1}{n}\sum[(ax_i+b-y_i)^2]' \\ &= \frac{2}{n}(ax_i+b-y_i)[(ax_i+b-y_i)]' \\ &= \frac{2}{n}\sum(ax_i+b-y_i)x_i\end{aligned}$$

b로 편미분한 결과 유도 과정

$$\begin{aligned}\frac{a}{\partial a}MSE(a,b) &= \frac{1}{n}\sum[(ax_i+b-y_i)^2]' \\ &= \frac{2}{n}(ax_i+b-y_i)[(ax_i+b-y_i)]' \\ &= \frac{2}{n}\sum(ax_i+b-y_i)\end{aligned}$$

3 코딩으로 확인하는 경사 하강법

- 이를 각각 파이썬 코드로 바꾸면 다음과 같음

```
y_pred = a * x_data + b   # 오차 함수인 y = ax + b를 정의한 부분
error = y_data - y_pred   # 실제값 - 예측값, 즉 오차를 구하는 식

# 평균 제곱 오차를 a로 미분한 결과
a_diff = -(2 / len(x_data)) * sum(x_data * (error))
# 평균 제곱 오차를 b로 미분한 결과
b_diff = -(2 / len(x_data)) * sum(y_data - y_pred)
```

- 여기에 학습률을 곱해 기존의 a값과 b값을 업데이트해 줌

```
a = a - lr * a_diff   # 미분 결과에 학습률을 곱한 후 기존의 a값을 업데이트
b = b - lr * b_diff   # 미분 결과에 학습률을 곱한 후 기존의 b값을 업데이트
```

- 중간 과정을 그래프로 표현하는 코드를 넣어 모두 정리하면 다음과 같이 코드가 완성됨

코드 4-1 경사 하강법 실습

- 예제 소스: deeplearning_class/03_Linear_Regression.ipynb

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 공부 시간 X와 성적 Y의 리스트를 만들기
data = [[2, 81], [4, 93], [6, 91], [8, 97]]
x = [i[0] for i in data]
y = [i[1] for i in data]
```

3 코딩으로 확인하는 경사 하강법

```
# 그래프로 나타내기
plt.figure(figsize=(8,5))
plt.scatter(x, y)
plt.show( )

#리스트로 되어 있는 x와 y 값을 넘파이 배열로 바꾸기(인덱스를 주어 하나씩 불러와 계산이 가능하게 하기 위함)
x_data = np.array(x)
y_data = np.array(y)

# 기울기 a와 절편 b의 값 초기화
a = 0
b = 0
```

3 코딩으로 확인하는 경사 하강법

```
# 학습률 정하기
lr = 0.05

# 몇 번 반복될지 설정(0부터 세므로 원하는 반복 횟수에 +1)
epochs = 2001


# 경사 하강법 시작
for i in range(epochs):          # 에포크 수만큼 반복
    y_pred = a * x_data + b    # y를 구하는 식 세우기
    error = y_data - y_pred    # 오차를 구하는 식
    # 오차 함수를 a로 미분한 값
    a_diff = -(1/len(x_data)) * sum(x_data * (error))
    # 오차 함수를 b로 미분한 값
    b_diff = -(1/len(x_data)) * sum(y_data - y_pred)
```

3 코딩으로 확인하는 경사 하강법

```
    a = a - lr * a_diff   # 학습률을 곱해 기존의 a값 업데이트
    b = b - lr * b_diff   # 학습률을 곱해 기존의 b값 업데이트

    if i % 100 == 0:      # 100번 반복될 때마다 현재의 a값, b값 출력
        print("epoch=%.f, 기울기=%.04f, 절편=%.04f" % (i, a, b))

# 앞서 구한 기울기와 절편을 이용해 그래프를 다시 그리기
y_pred = a * x_data + b
plt.scatter(x, y)
plt.plot([min(x_data), max(x_data)], [min(y_pred), max(y_pred)])
plt.show()
```

3 코딩으로 확인하는 경사 하강법

실행 결과

```
epoch=0, 기울기=23.2000, 절편=4.5250
epoch=100, 기울기=7.9316, 절편=45.3932
epoch=200, 기울기=4.7953, 절편=64.109
epoch=300, 기울기=3.4056, 절편=72.4022
(중략)
epoch=1800, 기울기=2.3000, 절편=79.0000
epoch=1900, 기울기=2.3000, 절편=79.0000
epoch=2000, 기울기=2.3000, 절편=79.0000
```

3 코딩으로 확인하는 경사 하강법

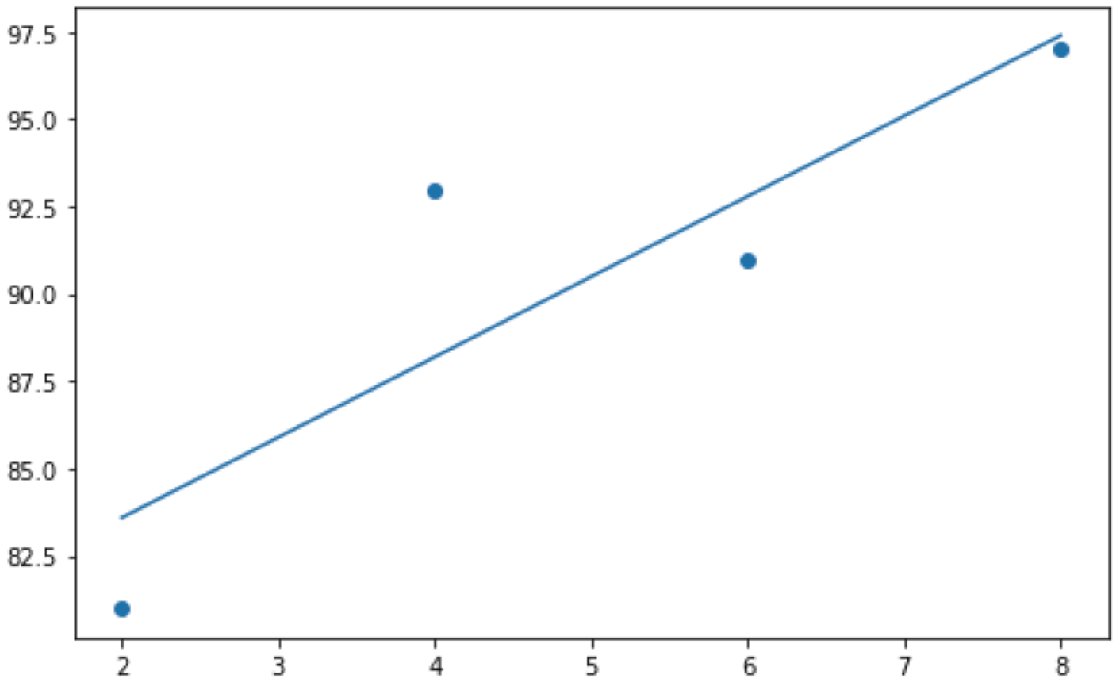

그림 4-5 그래프로 표현한 모습

TIP 여기서 에포크(epoch)는 입력 값에 대해 몇 번이나 반복하여 실험했는지를 나타냅니다. 우리가 설정한 실험을 반복하고 100번마다 결과를 내놓습니다.

4 다중 선형 회귀란

- 더 정확한 예측을 하려면 추가 정보를 입력해야 하며, 정보를 추가해 새로운 예측값을 구하려면 변수의 개수를 늘려 다중 선형 회귀를 만들어 주어야 함

공부한 시간(x_1)	2	4	6	8
과외 수업 횟수(x_2)	**0**	**4**	**2**	**3**
성적(y)	81	93	91	97

표 4-1 공부한 시간, 과외 수업 횟수에 따른 성적 데이터

4 다중 선형 회귀란

- 그럼 지금부터 두 개의 독립 변수 x_1과 x_2 가 생긴 것임
- 이를 사용해 종속 변수 y 를 만들 경우 기울기를 두 개 구해야 하므로 다음과 같은 식이 나옴

$$y = a_1x_1 + a_2x_2 + b$$

5 코딩으로 확인하는 다중 선형 회귀

- 이번에는 X의 값이 두 개이므로 다음과 같이 data 리스트를 만들고 X_1 과 X_2 라는 두 개의 독립 변수 리스트를 만들어 줌

```
data = [[2, 0, 81], [4, 4, 93], [6, 2, 91], [8, 3, 97]]
x1 = [i[0] for i in data]
x2 = [i[1] for i in data]
y = [i[2] for i in data]
```

5 코딩으로 확인하는 다중 선형 회귀

- data가 그래프로 어떻게 보이는지를 확인해 보자
- 먼저 x, y 두 개의 축이던 이전과 달리 x_1, x_2, y 이렇게 세 개의 축이 필요함
- 3D 그래프를 그려주는 라이브러리를 아래와 같이 불러옴

```
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d  # 3D 그래프 그리는 라이브러리 가져오기

ax = plt.axes(projection='3d')     # 그래프 유형 정하기
ax.set_xlabel('study_hours')
ax.set_ylabel('private_class')
ax.set_zlabel('Score')
ax.scatter(x1, x2, y)
plt.show()
```

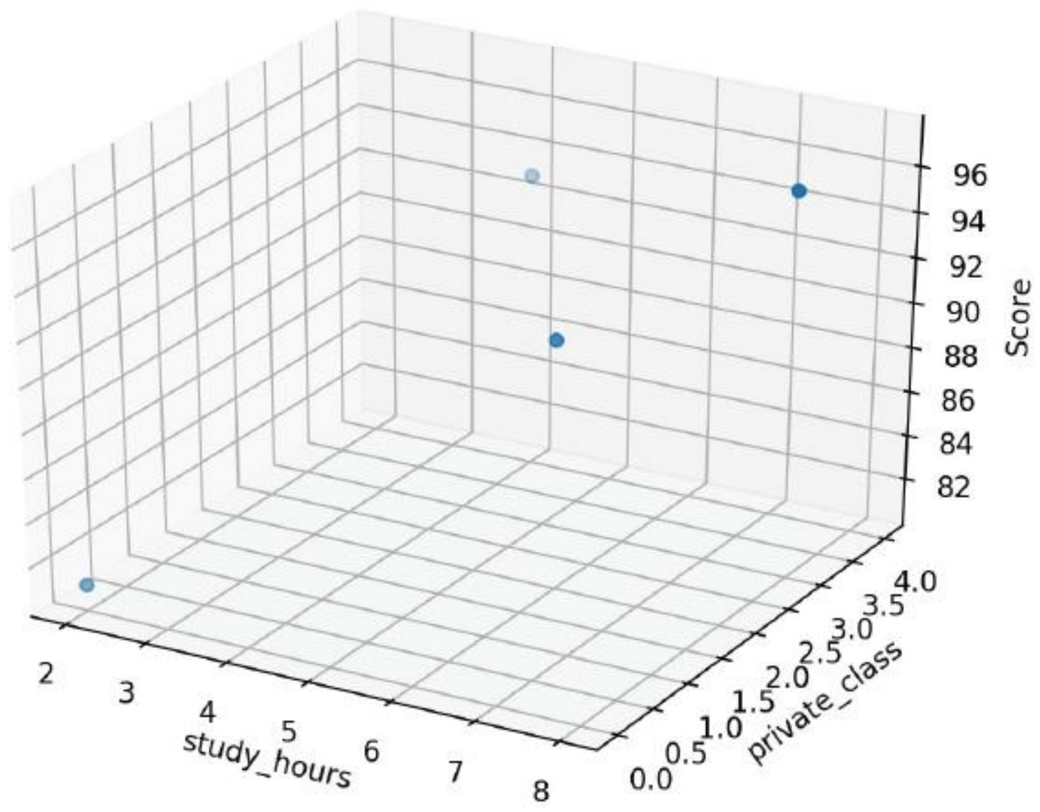

그림 4-6 축이 하나 더 늘어 3D로 배치된 모습

5 코딩으로 확인하는 다중 선형 회귀

- 이제 x가 두 개가 되었으므로 x_1 과 x_2두 가지의 변수를 지정함
- 각각의 값에 기울기 a 값이 다르므로 기울기도a_1 과 a_2 이렇게 두 가지를 만듦
- 각각 앞서 했던 방법과 같은 방법으로 경사 하강법을 적용하고 학습률을 곱해 기존의 값을 업데이트 함

```
y_pred = a1 * x1_data + a2 * x2_data + b   # y를 구하는 식을 세우기
error = y_data - y_pred   # 오차를 구하는 식
a1_diff = -(1/len(x1_data)) * sum(x1_data * (error)) # 오차 함수를 a1로 미분한 값
a2_diff = -(1/len(x2_data)) * sum(x2_data * (error))  # 오차 함수를 a2로 미분한 값
```

5 코딩으로 확인하는 다중 선형 회귀

```
b_new = -(1/len(x1_data)) * sum(y_data - y_pred)   # 오차 함수를 b로 미분한 값
a1 = a1 - lr * a1_diff   # 학습률을 곱해 기존의 a1 값 업데이트
a2 = a2 - lr * a2_diff   # 학습률을 곱해 기존의 a2 값 업데이트
b = b - lr * b_diff      # 학습률을 곱해 기존의 b 값 업데이트
```

코드 4-2 다중 선형 회귀 실습

- 예제 소스: deeplearning_class/04_Multi-Linear-Regression.ipynb

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d

# 공부 시간 X와 성적 Y의 리스트 만들기
data = [[2, 0, 81], [4, 4, 93], [6, 2, 91], [8, 3, 97]]
x1 = [i[0] for i in data]
x2 = [i[1] for i in data]
y = [i[2] for i in data]
```

5 코딩으로 확인하는 다중 선형 회귀

```
# 그래프로 확인
ax = plt.axes(projection='3d')
ax.set_xlabel('study_hours')
ax.set_ylabel('private_class')
ax.set_zlabel('Score')
ax.dist = 11
ax.scatter(x1, x2, y)
plt.show()

# 리스트로 되어 있는 x와 y 값을 넘파이 배열로 바꾸기(인덱스로 하나씩 불러와 계산할 수 있도록 하
기 위함)
x1_data = np.array(x1)
x2_data = np.array(x2)
y_data = np.array(y)
```

5 코딩으로 확인하는 다중 선형 회귀

```
# 기울기 a와 절편 b의 값 초기화
a1 = 0
a2 = 0
b = 0

# 학습률
lr = 0.05

# 몇 번 반복할지 설정(0부터 세므로 원하는 반복 횟수에 +1)
epochs = 2001
```

5 코딩으로 확인하는 다중 선형 회귀

```
# 경사 하강법 시작
for i in range(epochs): # epoch 수 만큼 반복
    y_pred = a1 * x1_data + a2 * x2_data + b  # y를 구하는 식 세우기
    error = y_data - y_pred  # 오차를 구하는 식
    # 오차 함수를 a1로 미분한 값
    a1_diff = -(1/len(x1_data)) * sum(x1_data * (error))
    # 오차 함수를 a2로 미분한 값
    a2_diff = -(1/len(x2_data)) * sum(x2_data * (error))
    # 오차 함수를 b로 미분한 값
    b_new = -(1/len(x1_data)) * sum(y_data - y_pred)
    a1 = a1 - lr * a1_diff  # 학습률을 곱해 기존의 a1 값 업데이트
```

5 코딩으로 확인하는 다중 선형 회귀

```
    a2 = a2 - lr * a2_diff   # 학습률을 곱해 기존의 a2 값 업데이트
    b = b - lr * b_diff      # 학습률을 곱해 기존의 b값 업데이트

    if i % 100 == 0:         # 100번 반복될 때마다 현재의 a1, a2, b 값 출력
        print("epoch=%.f, 기울기1=%.04f, 기울기2=%.04f, 절편=%.04f"
% (i, a1, a2, b))
```

5 코딩으로 확인하는 다중 선형 회귀

실행 결과

```
epoch=0, 기울기 1=23.2000, 기울기 2=10.5625, 절편=4.5250
epoch=100, 기울기 1=6.4348, 기울기 2=3.9893, 절편=43.9757
epoch=200, 기울기 1=3.7255, 기울기 2=3.0541, 절편=62.5766
epoch=300, 기울기 1=2.5037, 기울기 2=2.6323, 절편=70.9656
epoch=400, 기울기 1=1.9527, 기울기 2=2.4420, 절편=74.7491
epoch=500, 기울기 1=1.7042, 기울기 2=2.3562, 절편=76.4554
(중략)
epoch=1500, 기울기 1=1.5001, 기울기 2=2.2857, 절편=77.8567
epoch=1600, 기울기 1=1.5000, 기울기 2=2.2857, 절편=77.8569
epoch=1700, 기울기 1=1.5000, 기울기 2=2.2857, 절편=77.8570
epoch=1800, 기울기 1=1.5000, 기울기 2=2.2857, 절편=77.8571
epoch=1900, 기울기 1=1.5000, 기울기 2=2.2857, 절편=77.8571
epoch=2000, 기울기 1=1.5000, 기울기 2=2.2857, 절편=77.8571
```

5 코딩으로 확인하는 다중 선형 회귀

- 다중 선형 회귀 문제에서의 기울기 a_2, a_1 와 절편 b의 값을 찾아 확인할 수 있음

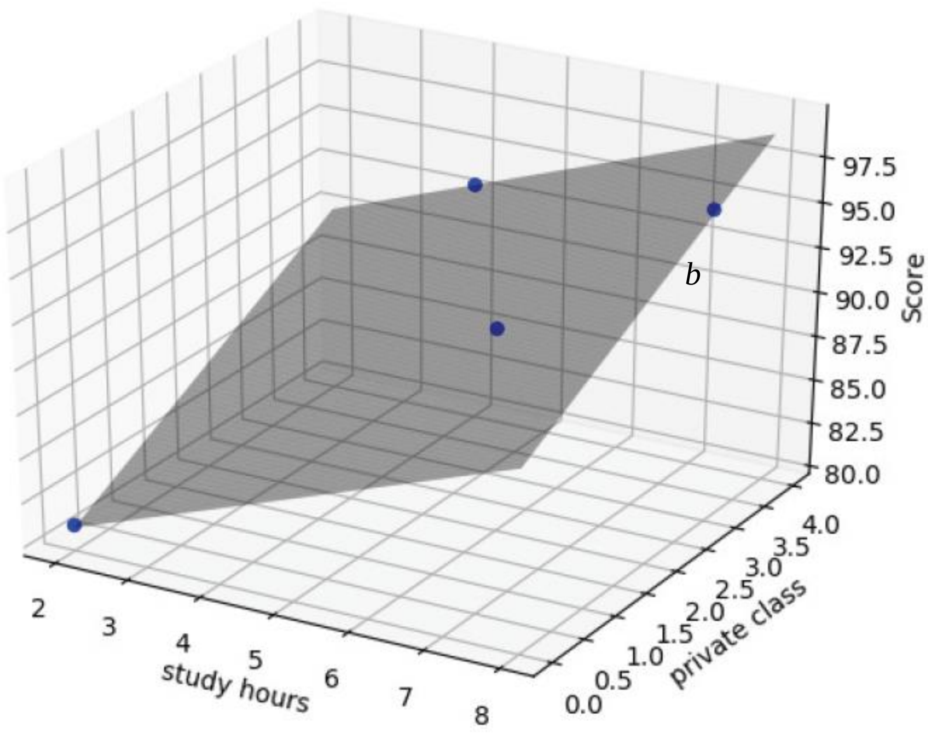

그림 4-7 다중 선형 회귀의 그래프: 차원이 하나 더 늘어난 모습

5 코딩으로 확인하는 다중 선형 회귀

- 1차원 예측 직선이 3차원 '예측 평면'으로 바뀜
- 과외 수업 횟수(privateclass)라는 새로운 변수가 추가됨
- 1차원 직선에서만 움직이던 예측 결과가 더 넓은 평면 범위 안에서 움직이게 됨
- 이로 인해 좀 더 정밀한 예측을 할 수 있게 된 것임