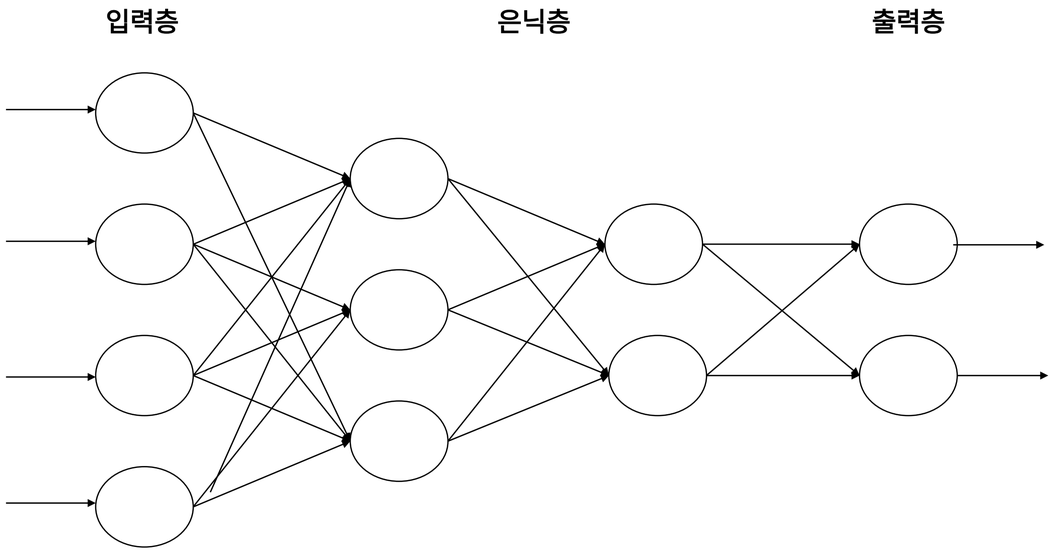
4. 신경망

1 모델 구조와 작동 과정

모델 구조

신경망의 구조는 인간의 뇌 작동 과정을 모방합니다.

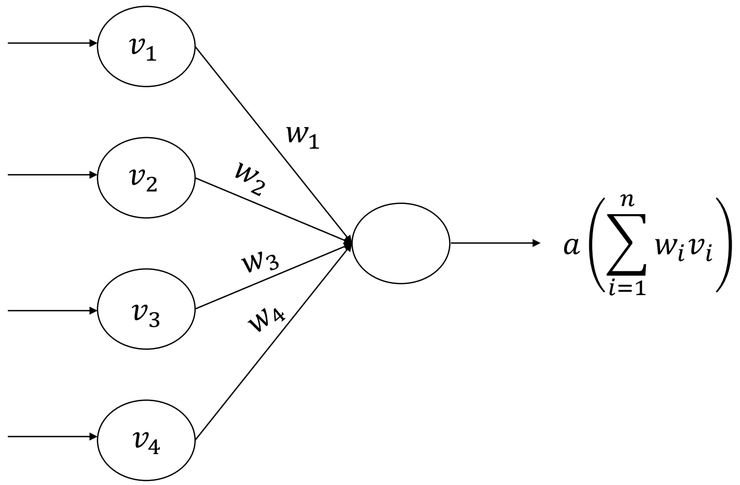


- 신경망은 입력층, 은닉층, 출력층으로 구성됨
- 각 층은 하나 이상의 노드로 구성됨
- 신경망의 각 노드는 다음 층에 있는 각 노드와 연결됨
- 입력층에 있는 각 노드는 특징을 입력받는 역할을 하므로, 입력 노드 수는 특징 개수와 같음
- 출력층에 있는 각 노드는 최종 결과를 출력하는 역할을 하므로 출력 노드 수는 라벨 혹은 클래스의 개수가 됨
- 입력층과 출력층은 사용자가 구조를 지정하지 않지만, 은닉층은 사용자가 직접 지정해야 함

신경망의 은닉층은 복잡도와 직결되는 중요한 하이퍼 파라미터임

작동 과정

신경망은 가중합과 활성화 함수라는 두 가지 연산을 반복하여 라벨을 예측합니다.



입력 노드를 제외한 모든 노드의 연산 과정

(1) 입력값 가중합

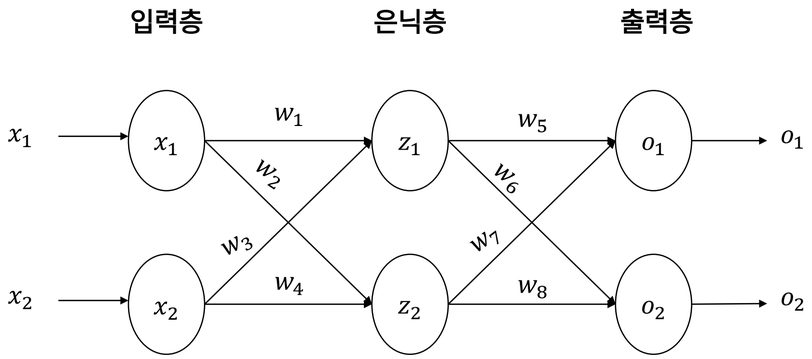
(2) 활성화 함수에 입력

(3) 다음 노드에 전달

신경망은 가중합 연산이 여러 번 반복되는 모델이므로 특징의 스케일에 크게 영향을 받음

작동 과정 (예시)

신경망은 가중합과 활성화 함수라는 두 가지 연산을 반복하여 라벨을 예측합니다.



(1) 두 개의 입력 노드에 특징값 x_1과 x_2가 각각 저장됨

(2) x_1과 x_2가 각 은닉 노드로 전달되며, 이 과정에서 가중합과 활성화 함수 연산이 수행됨

- $z_1 = a_1(w_1x_1 + w_3x_2)$
- $z_2 = a_1(w_2x_1 + w_4x_2)$

(3) z_1과 z_2가 각 출력 노드로 전달되며, 이 과정에서 가중합과 활성화 함수 연산이 수행됨

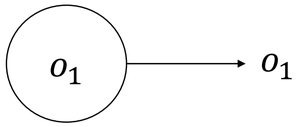
- $o_1 = a_2(w_5z_1 + w_7z_2)$
- $o_2 = a_2(w_6z_1 + w_8z_2)$

(4) 출력 노드의 값 o_1과 o_2를 출력함

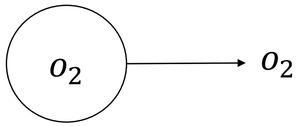
분류: 소프트맥스

소프트맥스는 분류 결과를 종합하는 함수입니다.

출력층



$$\Pr(y = o_1) = \frac{\exp(o_1)}{\exp(o_1) + \exp(o_2)}$$



$$\Pr(y = o_2) = \frac{\exp(o_2)}{\exp(o_1) + \exp(o_2)}$$

활성화 함수

활성화 함수는 신경망에 비선형성을 부여하는 함수로 렐루 함수가 주로 사용됩니다.

활성화 함수가 없다면?

- $z_1 = (w_1x_1 + w_3x_2)$
- $z_2 = (w_2x_1 + w_4x_2)$

⟶

- $o_1 = (w_5z_1 + w_7z_2) = (w_1w_5 + w_2w_7)x_1 + (w_3w_5 + w_4w_7)x_2$
- $o_2 = (w_6z_1 + w_8z_2) = (w_1w_6 + w_2w_8)x_1 + (w_2w_4 + w_3w_6)x_2$

불필요하게 복잡한 선형식

주요 활성화 함수

시그모이드 함수

$$a(z) = \frac{1}{1 + \exp(-z)}$$

기울기 소실 문제 ⟶

렐루 함수

$$a(z) = \begin{cases} z, & if\ z > 0 \\ 0, & otherwise \end{cases}$$

이전 층의 출력이 음수면 학습되지 않음 ⟶

리키 렐루 함수

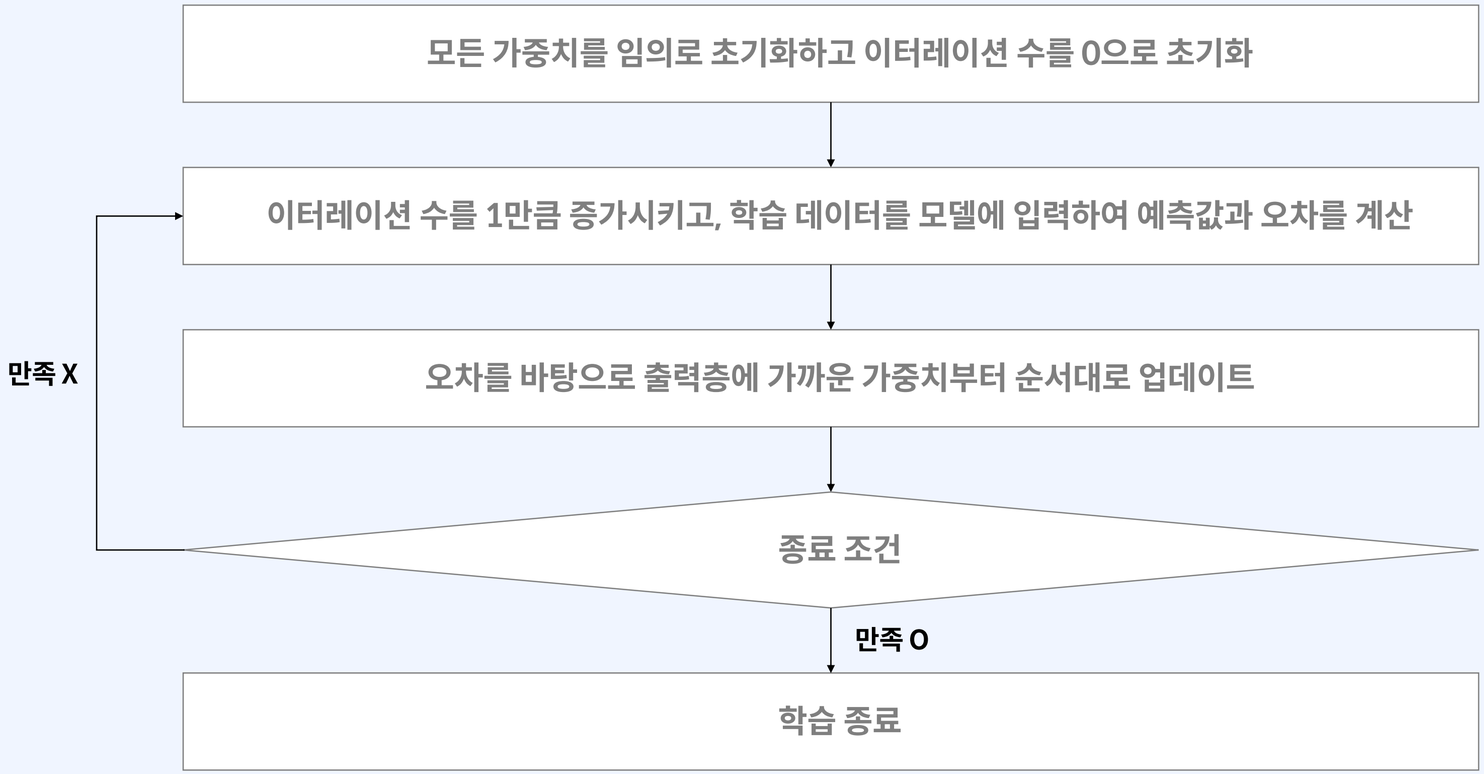
$$a(z) = \begin{cases} z, & if\ z > 0 \\ 0.01z, & otherwise \end{cases}$$

4. 신경망

2 모델 학습과 주요 하이퍼 파라미터

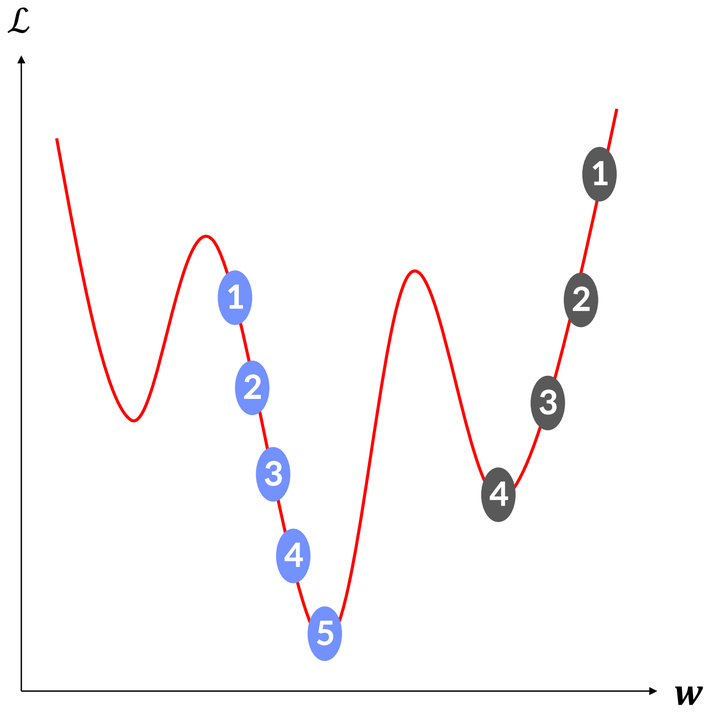
오류 역전파 알고리즘

신경망은 오류 역전파 알고리즘(back propagation algorithm)을 통해 학습됩니다.



내리막 경사법

오류 역전파 알고리즘은 내리막 경사법의 일종으로, 내리막 경사법을 알아야 학습률과 초기 가중치의 중요성을 제대로 이해할 수 있습니다



해 탐색 과정 (1 에서 출발)

(1) 임의의 해 선정: 1

(2) 접선 기울기의 반대 방향(+)으로 이동: 2 에 도달

...

(5) 접선 기울기의 반대 방향(+)으로 이동: 5 에 도달

(6) 접선 기울기가 0이므로 더 이상 이동하지 않음

해 탐색 과정 (1 에서 출발)

(1) 임의의 해 선정: 1

(2) 접선 기울기의 반대 방향(-)으로 이동: 2 에 도달

...

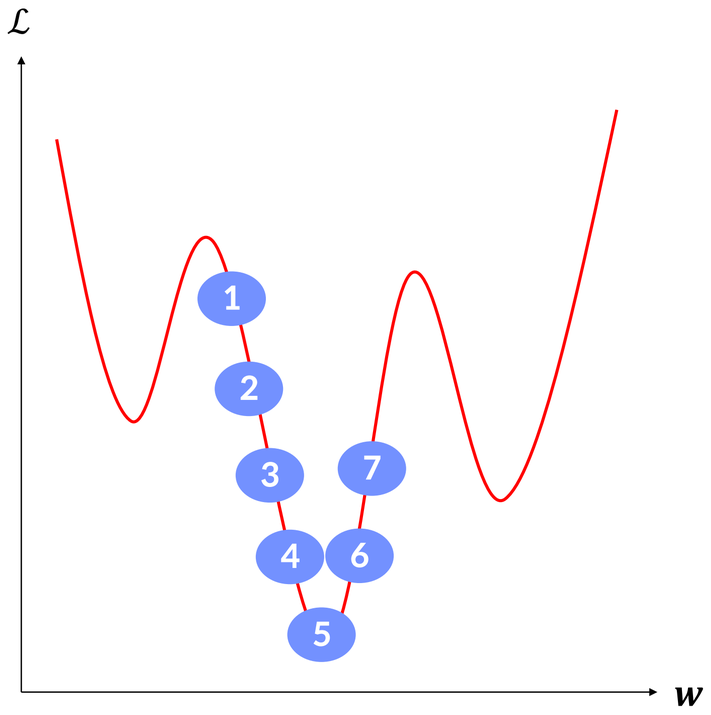
(4) 접선 기울기의 반대 방향(-)으로 이동: 4 에 도달

(5) 접선 기울기가 0이므로 더 이상 이동하지 않음

임의로 설정한 초기 해에 따라 신경망의 성능이 크게 좌우되므로 시드를 잘 설정해야 함

내리막 경사법 (계속)

오류 역전파 알고리즘은 내리막 경사법의 일종으로, 내리막 경사법을 알아야 학습률과 초기 가중치의 중요성을 제대로 이해할 수 있습니다



학습률 = 보폭

보폭이 1일 때: 1 → 2 → 3 → 4 → 5 (느린 수렴)

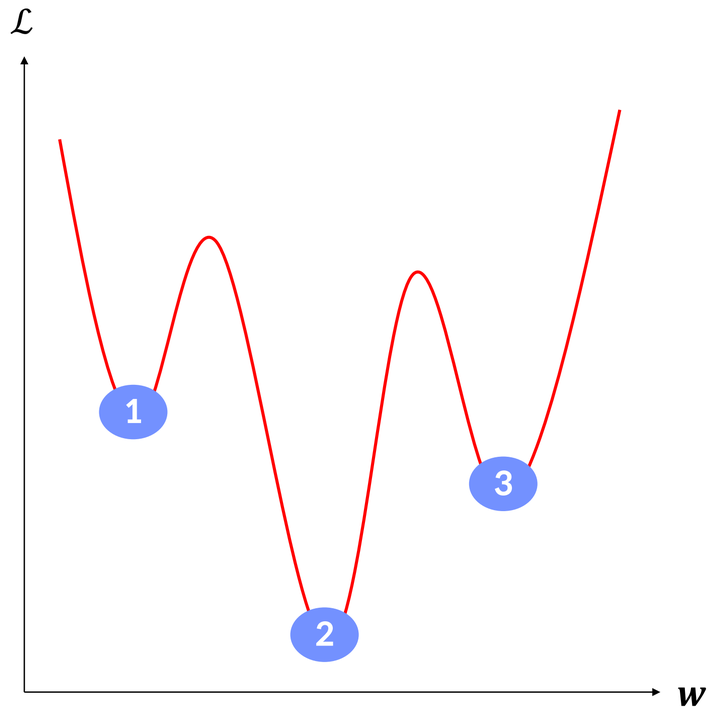
보폭이 2일 때: 1 → 3 → 5 (빠른 수렴)

보폭이 3일 때: 1 → 4 → 7 → 4 → 7 → (발산)

학습률이 클수록 더 빠르게 수렴하지만, 발산할 위험이 있음
또한, 지역 최적을 피할 수도 전역 최적을 못 찾을 수도 있음

내리막 경사법 (계속)

오류 역전파 알고리즘은 내리막 경사법의 일종으로, 내리막 경사법을 알아야 학습률과 초기 가중치의 중요성을 제대로 이해할 수 있습니다



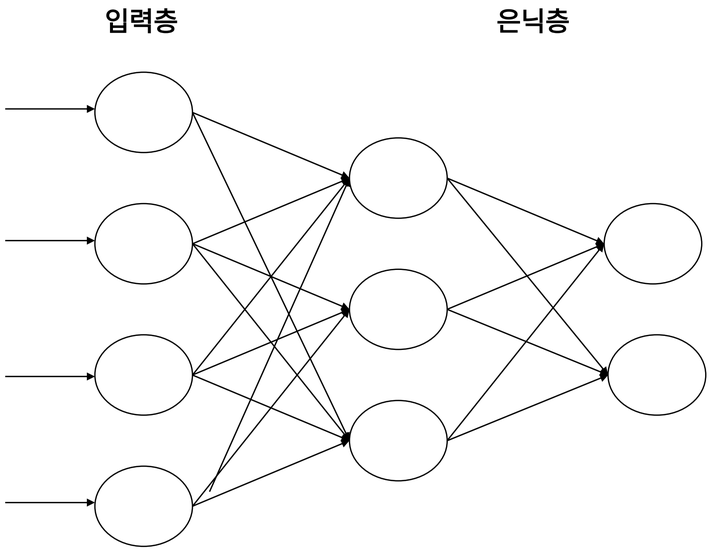
학습 오차 1 > 3 > 2

평가 오차 1 ? 3 ? 2

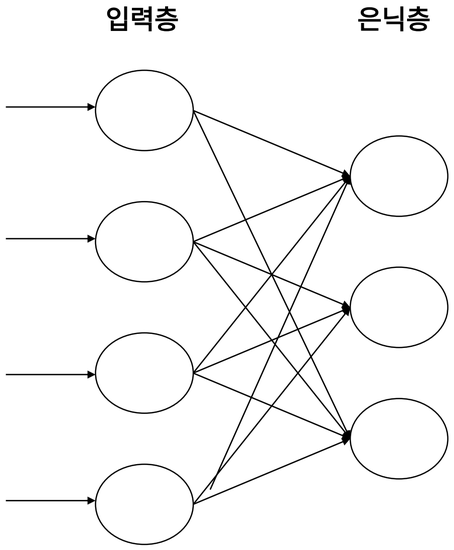
최대 이터레이션 횟수에 도달해서 알고리즘이 종료되는 경우가 많고,
최대 이터레이션 횟수가 너무 크면 과적합 가능성이, 너무 작으면 과소 적합 가능성이 있음

모델 구조

신경망에는 여러 하이퍼파라미터가 있으나 대부분 널리 쓰이는 값을 사용하는 것이 좋습니다. 그런데 신경망의 모델 구조는 그렇지 않으며, 신경망의 성능은 사실상 모델 구조가 결정한다고 해도 과언이 아닙니다.



가중치 수 = 4 * 3 + 3 * 2 = 18



가중치 수 = 4 * 3 = 12

은닉층의 구조가 복잡할수록 모델이 복잡해짐

4. 신경망

3 사이킷런 실습

예제 데이터 불러오기

신경망을 학습할 예제 데이터를 불러옵니다.

예제 데이터 불러오기

```
1  import pandas as pd
2  from sklearn.model_selection import train_test_split
3  df = pd.read_csv("../../data/regression/puma32h.csv")
4  X = df.drop('y', axis = 1)
5  y = df['y']
6  X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 2022)
```

신경망 학습

신경망은 사이킷런의 neural_network 모듈의 MLPClassifier와 MLPRegressor를 이용해 구현할 수 있습니다.

주요 인자

인자	설명	기본 값
hidden_layer_sizes	튜플 형태의 은닉층 구조	(100,)
activation	활성화 함수	“relu”
max_iter	최대 이터레이션 횟수	200

신경망 학습 및 평가 예제

```
1  from sklearn.neural_network import MLPRegressor as NN
2  from sklearn.metrics import mean_absolute_error as MAE
3  model = NN(random_state = 2022).fit(X_train, y_train)
4  y_pred = model.predict(X_test)
5  mae = MAE(y_test, y_pred)
6  print(mae)
```

```
0.06037836487409957
```

모델 구조 비교

hidden_layer_sizes 인자를 바꿔가며 다양한 모델 구조를 정의하고 그 성능을 확인해보겠습니다.

모델 구조 비교 예제

```
1  model1 = NN(hidden_layer_sizes = (5, ), random_state = 2022).fit(X_train, y_train)
2  model2 = NN(hidden_layer_sizes = (10, 10), random_state = 2022).fit(X_train, y_train)
3  model3 = NN(hidden_layer_sizes=(100, 100, 100),random_state = 2022).fit(X_train, y_train)
4
5  y1_pred = model1.predict(X_test)
6  y2_pred = model2.predict(X_test)
7  y3_pred = model3.predict(X_test)
8
9  mae1 = MAE(y_test, y1_pred)
10 mae2 = MAE(y_test, y2_pred)
11 mae3 = MAE(y_test, y3_pred)
12 print(mae1, mae2, mae3)
```

- **라인 1 - 3:** 은닉층의 구조가 (5,), (10, 10), (100, 100, 100)인 신경망 model1, model2, model3을 학습했습니다. 모델 복잡도는 model1 < model2 < model3입니다.

```
0.0769324113849796  0.05711125441447522  0.08956654762313893
```

- 은닉층의 구조가 (10, 10)인 모델의 성능이 가장 좋음
- 은닉층의 구조는 모델의 복잡도와 직결되므로 더 좋은 은닉층의 구조를 찾으려면 (5,)보다는 복잡하고 (100, 100, 100)보다는 단순한 구조를 탐색해야 함

모델 구조 비교 (계속)

hidden_layer_sizes 인자를 바꿔가며 다양한 모델 구조를 정의하고 그 성능을 확인해보겠습니다.

모델 구조 비교 예제

```
model4 = NN(hidden_layer_sizes = (15, 15), random_state = 2022).fit(X_train, y_train)
model5 = NN(hidden_layer_sizes = (10, 5), random_state = 2022).fit(X_train, y_train)
model6 = NN(hidden_layer_sizes = (20, ), random_state = 2022).fit(X_train, y_train)

y4_pred = model4.predict(X_test)
y5_pred = model5.predict(X_test)
y6_pred = model6.predict(X_test)

mae4 = MAE(y_test, y4_pred)
mae5 = MAE(y_test, y5_pred)
mae6 = MAE(y_test, y6_pred)
print(mae4, mae5, mae6)
```

```
0.05044311790884906 0.04886901801873948 0.05072093415834021
```

- (10, 5)에서 가장 좋은 성능을 보였으며, model4, model5, model6 모두 model2보다 나은 성능을 보임

최대 이터레이션 횟수 설정

샘플 수가 적고 상대적으로 특징이 많은 데이터에 대해 최대 이터레이션 횟수를 설정해보겠습니다.

최대 이터레이션 횟수 설정 예제

```
1  df = pd.read_csv("../../data/regression/baseball.csv")
2  X = df.drop('y', axis = 1)
3  y = df['y']
4  X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 2022)
5
6  model = NN(random_state = 2022).fit(X_train, y_train)
7  y_pred = model.predict(X_test)
8  mae = MAE(y_test, y_pred)
9  print(mae)
```

```
748.3964922052431

ConvergenceWarning: Stochastic Optimizer:
Maximum iterations (200) reached and the optimization hasn't converged yet.
```

- 가중치가 수렴하지 못하고 최대 이터레이션 횟수인 200에 도달해서 ConvergenceWarning이 발생함

최대 이터레이션 횟수 설정 (계속)

최대 이터레이션 횟수를 크게 설정하여 수렴시켜 보겠습니다.

최대 이터레이션 횟수 설정 예제

```
1  model = NN(random_state = 2022, max_iter = 10000).fit(X_train, y_train)
2  y_pred = model.predict(X_test)
3  mae = MAE(y_test, y_pred)
4  print(mae)
```

687.3320969527582

- ConvergenceWarning이 발생하지 않음
- 오차가 크게 줄었음
- 그러나 상황에 따라서는 수렴함으로써 과적합될 수 있음