[12주차] 딥러닝의 첫걸음: MLP로 고전 모델 뛰어넘기 🚀

안녕하세요! 12주차 수업에 오신 것을 환영합니다.
지난 시간까지 우리는 경청 트리, 랜덤 포레스트와 같은
강력한 고전 머신러닝 모델들을 배웠습니다.
이 모델들은 **'어떻게 질문할지(규칙)'** 를 학습했습니다.

오늘은 드디어 **딥러닝(Deep Learning)** 의 가장 기본적인 모델인
MLP (Multi-Layer Perceptron, 다층 퍼셉트론) 를 만나봅니다.

MLP는 2주차 역사 시간에 잠깐 등장했던, 'AI 겨울'을 초래했던
단순한 퍼셉트론을 '여러 겨 쌓아' 비선형 문제를 해결한 모델입니다.
`scikit-learn`의 `MLPClassifier`를 이용해 이 모델을 구현하고,
10주차에 풀었던 붓꽃 문제를 다시 풀어보며 성능을 확인해 보겠습니다.

🧠 모듈 1: MLP (다층 퍼셉트론)란 무엇인가?

MLP의 구조를 **'복잡한 의사결정을 내리는 회사'** 에 비유해 봅시다.

1. 입력층 (Input Layer) - '신입 사원'

- **역할:** 외부로부터 원본 데이터(e.g., 붓꽃의 특징 4개)를 받아들입니다.
- 고객의 원본 요청서(데이터)를 받아서 다음 부서로 전달하는 신입 사원들입니다.
- **구분:** 특징 개수만큼 노드가 존재합니다. (e.g., 붓꽃 문제에서는 4개)

2. 은닉층 (Hidden Layer(s)) - '중간 관리자'

- **역할:** MLP의 핵심. 입력 데이터를 받아 **복잡한 특징을** 추출하고 조합합니다.
- **첫 번째 은닉층 (주임/대리):** 신입들이 넘긴 원본 데이터를 보고,
  "꽃잎이 1cm 이하인가?", "너비가 0.5cm 이상인가?" 같은 **단순한 판단을** 합니다.
- **두 번째 은닉층 (과장/팀장):** 주임/대리들의 단순 판단들을 **조합**해,
  "꽃잎이 매우 작으면서 너비도 좁고" 같은 더 **복잡하고 추상적인 판단을** 합니다.
- **구분:** 이 층의 개수와 각 층의 노드(뉴런) 개수는 우리가 정하는 **하이퍼파라미터**입니다.
  (e.g., `hidden_layer_sizes=(10, 5)` -> 10명의 주임, 5명의 과장)

3. 출력층 (Output Layer) - '최고 경영자(CEO)'

- **역할:** 은닉층에서 가공된 모든 정보를 최종적으로 취합하여 **결론(예측)** 을 내립니다.
- **비유:** 과장/팀장들의 복잡한 판단 보고서를 모두 검토한 후,
  "이것은 Setosa다", "이것은 Versicolor다" 라고 최종 결재를 내립니다.
- **구분:** 분류 문제의 클래스 개수만큼 노드가 존재합니다. (e.g., 붓꽃 문제에서는 3개)

뉴런과 비선형 활성화 함수 (ReLU)

MLP의 각 노드(뉴런)는 기본적으로 두 가지 일을 합니다.

1. 선형 계산: 로지스틱 회귀처럼 입력값(x)과 가중치(W)를 곱하고 편향(b)을 더합니다.

- $z = W_1x_1 + W_2x_2 + ... + b$

2. 비선형 활성화 (Activation): 계산된 z값을 **비선형 함수**에 통과시킵니다.

- **이것이 딥러닝의 핵심입니다!**
- 만약 활성화 함수가 없다면 (즉, 선형이라면), 선형 계산을 아무리 여러 겹 쌓아도
  그 결과는 결국 하나의 거대한 선형 계산과 같습니다.
  (e.g., 1차 함수 * 1차 함수 = 1차 함수)
- 비선형 함수가 층간에 끼어듦으로써,
  층을 쌓을수록 더 복잡하고 굽은 경계선을 만들 수 있습니다.

ReLU (Rectified Linear Unit):

- 가장 널리 사용되는 비선형 활성화 함수입니다.
- $f(z) = max(0, z)$
- **의미:** 계산된 값 z가 0보다 작으면(e.g., 관련 없는 정보) 0으로 무시하고,
  0보다 크면(e.g., 의미 있는 정보) 그 값을 그대로 통과시킵니다.
- **비유:** "이 정보가 판단에 도움이 되면(양수) 보고하고, 아니면(음수) 무시해!"

학습 과정: 역전파 (Backpropagation)

MLP는 어떻게 학습할까요? (2주차 역사 시간 복습)

1. 데이터를 입력층에 넣고 출력층까지 계산을 수행합니다. (순전파, Forward Propagation)
2. 출력층의 **예측**과 실제 **정답**을 비교하여 **오차(Error)** 를 계산합니다.
3. 이 오차를 'C다'가 신입 사원까지 **역방향으로(Back)** 전파(Propagation)시킵니다.
4. 각 노드(뉴런)는 자신이 오차에 얼마나 기여했는지 책임을 지게 되고,
   자신의 **편향 가중(가중치 W, 편향 b)** 을 조금씩 수정합니다.
5. 이 과정을 모든 데이터에 대해 수백, 수천 번 반복(Epoch)하여 오차를 최소화합니다.

🌸 모듈 2: MLP로 붓꽃 문제 다시 풀기

10주차에 경청 트리와 랜덤 포레스트로 풀었던 붓꽃 분류 문제를
이번에는 MLP로 해결해보겠습니다.

[중요] 딥러닝 모델을 위한 데이터 전처리: 스케일링

결정 트리는 각 특징의 range(e.g., 5cm)를 기준으로 분할하므로 스케일에 영향을 덜 받지만,
MLP와 같은 신경망 모델은 **입력 데이터의 스케일에 매우 민감합니다.**

- **이유:** $z = W_1x_1 + W_2x_2 + b$ 계산을 생각해 보세요.
  - x_1은 (0 ~ 1) 범위이고 x_2는 (0 ~ 1000) 범위이면,
    모델은 x_2의 값에만 거의 전적으로 의존하게 되고 x_1의 영향력은 무시됩니다.
    이는 가중치(W) 학습을 매우 불안정하고 비효율적으로 만듭니다.
- **해결:** `StandardScaler`를 사용하여 모든 특징의 **평균을 0, 분산을 1로**
  표준화하여, 모든 특징이 동등한 스케일에서 학습에 기여하도록 만듭니다.

```
In [2]:
```

```
# 필요한 라이브러리 임포트
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import (classification_report,
                             accuracy_score)

# 1. 데이터 로드 및 준비
iris = load_iris()
X = iris.data  # 4개의 특징 모두 사용
y = iris.target
feature_names = iris.feature_names
target_names = iris.target_names

# 2. 데이터 분리
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

# 3. 데이터 스케일링 (중요!)
# StandardScaler 객체 생성
scaler = StandardScaler()

# [중요] scaler는 *학습(Train) 데이터로만 fit 해야 합니다!
# (실전 시험 데이터의 정보가 학습 과정에 유출되면 안 됨)
scaler.fit(X_train)

# fit된 scaler를 이용해 train/test 데이터 모두를 transform
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

print("--- 스케일링 전 Train 데이터 (일부) ---")
print(np.round(X_train[:5], 2))
print("\n--- 스케일링 후 Train 데이터 (일부) ---")
print(np.round(X_train_scaled[:5], 2))
```

```
--- 스케일링 전 Train 데이터 (일부) ---
[[5.1 2.5 3.  1.1]
 [6.2 2.2 4.5 1.5]
 [5.1 3.8 1.5 0.3]
 [6.8 3.2 5.9 2.3]
 [5.7 2.8 4.1 1.3]]

--- 스케일링 후 Train 데이터 (일부) ---
[[-0.9  -1.22 -0.44 -0.14]
 [ 0.38 -1.88  0.4   0.38]
 [-0.9   1.64 -1.29 -1.17]
 [ 1.88  0.32  1.19  1.41]
 [-0.2  -0.56  0.18  0.12]]
```

MLP 모델 생성 및 학습

이제 `MLPClassifier`를 생성하고 **스케일링된 데이터로** 학습시킵니다.
핵심은 **`hidden_layer_sizes`** 파라미터로 모델의 구조를 설계하는 것입니다.

```
In [49]:
```

```
# 1. MLP 모델 생성
# hidden_layer_sizes=(10, 5)
# : 10개 뉴런의 첫 번째 은닉층,
#   5개 뉴런의 두 번째 은닉층을 의미
#   (회사 비유: 주임 10명, 과장 5명)
# activation='relu' : 비선형 함수로 ReLU를 사용
# max_iter=1000 : 학습 에포크(총 전체 1000번 반복
# random_state=42 : 결과 재현을 위한 시드 고정
# solver='adam' : 최적화 알고리즘 (기본값, 효율적)
model_mlp = MLPClassifier(
    hidden_layer_sizes=(10, 5),
    activation='relu',
    max_iter=1000,
    random_state=42,
    solver='adam'
)

# 2. '스케일링된 연습 문제'로 모델 학습 (fit)
model_mlp.fit(X_train_scaled, y_train)

print("MLP 모델 학습 완료!")
```

```
MLP 모델 학습 완료!
```

성능 평가 및 비교

학습된 MLP 모델의 성능을 **스케일링된 실전 시험 데이터로** 평가합니다.

```
In [50]:
```

```
# '스케일링된 실전 시험' 데이터로 예측 수행
y_pred_mlp = model_mlp.predict(X_test_scaled)

# 성능 평가
accuracy_mlp = accuracy_score(y_test, y_pred_mlp)
report_mlp = classification_report(y_test, y_pred_mlp,
                                   target_names=target_names)

print("--- MLPClassifier 성능 평가 (Test Data) ---")
print(f"정확도: {accuracy_mlp:.4f}")
print("\nClassification Report:")
print(report_mlp)
```

```
--- MLPClassifier 성능 평가 (Test Data) ---
정확도: 0.9556

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        15
  versicolor       0.93      0.93      0.93        15
   virginica       0.93      0.93      0.93        15

    accuracy                           0.96        45
   macro avg       0.96      0.96      0.96        45
weighted avg       0.96      0.96      0.96        45
```

✨ 마무리

오늘은 딥러닝의 기본 모델인 **MLP (다층 퍼셉트론)** 를 배웠습니다.

- **핵심 원리:** 여러 층(Layer)의 뉴런(Neuron)과 **비선형 활성화 함수(ReLU)** 를 조합하여,
  복잡하고 부드러운 **비선형 결정 경계를** 학습합니다.
- **필수 전처리:** MLP는 입력 특징의 스케일에 민감하므로, `StandardScaler` 를 이용한
  데이터 **스케일링**이 매우 중요합니다.
- **성능:** 고전적인 모델(e.g., 랜덤 포레스트)과 동등하거나 그 이상의 성능을 보이며,
  더 복잡한 문제(이미지, 언어)에서 진가를 발휘합니다.

다음 시간에는 좀 더 복잡한 문제인 고차원 이미지 데이터 분류를 MLP를 이용해 해결해봅시다.