실습 과제

> **data 폴더의 wine.csv를 read 하여 데이터 분석 및 Perceptron을 이용한 결정 경계 그래프 그리기**

- wine.csv dataset의 alcohol(1번째 열)과 proline(13번째 열)을 특징으로 사용하여 wine의 class 예측
- 5주차 실습에서 사용한 MinMaxScaler 혹은 StandardScaler를 사용하여 데이터셋 정규화 수행
- 목표: **wine.csv의 산점도**, **epoch당 number of updates 그래프** 및 **결정 경계 그래프** 등 시각화
- 과제는 해당 셀 아래에 작성
- 권장사항: 다른 특징을 사용한 결정 경계 그래프 확인
- [wine dataset 정보]

```
# google drive 연결
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
colab_path = "/content/drive/MyDrive/MachineLearning/6wk/과제"
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import numpy as np

class Perceptron(object):
    def __init__(self, eta=0.01, n_iter=50, random_state=1):
        ''' 초기화 함수 구현 '''
        self.eta = eta
        self.n_iter = n_iter
        self.random_state = random_state


    def fit(self, X, y):
        rgen = np.random.RandomState(self.random_state)
        self.w_ = rgen.normal(loc=0.0, scale = 0.01, size = 1 + X.shape[1])
        self.errors_ = []

        for _ in range(self.n_iter):
          errors = 0
          for xi, target in zip(X,y):
            update = self.eta * (target - self.predict(xi))
            self.w_[1:] += update * xi
            self.w_[0] += update
            errors += int(update != 0.0)
          self.errors_.append(errors)
        return self

    def net_input(self, X):
        return np.dot(X, self.w_[1:]) +self.w_[0]

    def predict(self, X):
        return np.where(self.net_input(X) >= 0.0, 1, -1)
```

더블클릭 또는 Enter 키를 눌러 수정

```
# data 폴더의 wine.csv read
import pandas as pd
import os

# 실습자료 data 폴더의 wine.csv 파일 read
df = pd.read_csv(os.path.join(colab_path,'data/wine.csv'))
df.shape
```

```
(121, 14)
```

```
# 붓꽃 예제의 np.where 함수 용법을 참고하여, wine.csv의 정답 클래스 전처리
# Hint: wine의 정답 class는 0번 열에 위치

import numpy as np

# 정답 클래스 전처리:
y = df.iloc[0:121, 0].values
y = np.where(y == 1, 1, -1)
y
```

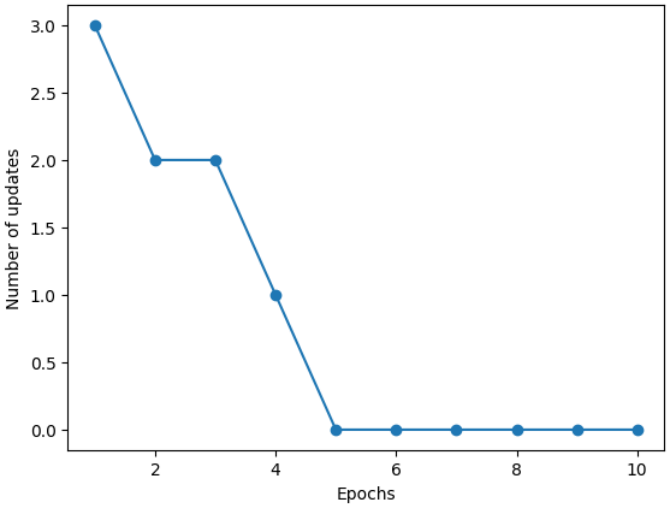
```
array([ 1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
```

```
        -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
        -1, -1])
```

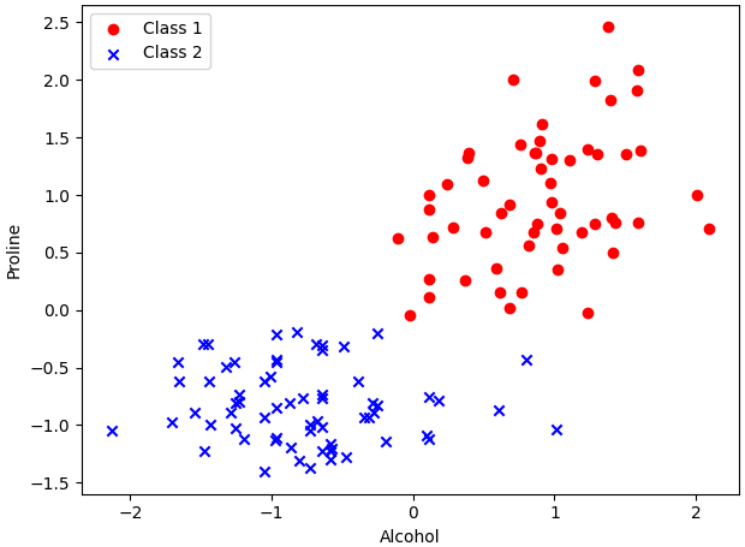
```
# X 값으로 사용할 변수에 dataframe의 alchol(1번 열) proline(13번 열)의 값을
# 담아 특징으로 사용
X = df.iloc[0:121, [1, 13]].values
```

```
# X값에 대해 MinMaxScaler or StandardScaler 함수로 정규화 수행
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X = scaler.fit_transform(X)
```

```
# Perceptron 설정 및 학습 (eta, n_iter 조정 추천)
ppn = Perceptron(eta=0.01, n_iter=10)

ppn.fit(X, y)
# Epochs당 Number of updates 그래프 출력
plt.figure()
plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.xlabel('Epochs')
plt.ylabel('Number of updates')
plt.show()
```




```
# 산점도 그리기 - class 1 -> (x_wine 0 ~ 55), class2 -> (x_wine 56 ~ 121)
plt.scatter(X[:56, 0], X[:56, 1], color='red', marker='o' ,label='Class 1')
plt.scatter(X[56:121, 0], X[56:121, 1], color='blue',marker='x', label='Class 2')
plt.xlabel('Alcohol')
plt.ylabel('Proline')
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```



```
# 결정 경계 그래프
from matplotlib.colors import ListedColormap

# 결정 경계 그래프 함수 정의
def plot_decision_regions(X, y, classifier, resolution=0.01):
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])

    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution), np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)

    plt.contourf(xx1, xx2, Z, alpha=0.3, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())
    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1], alpha=0.8, c=colors[idx], marker=markers[idx], label=cl, edgecolor=None if idx ==1 els
# 결정 경계 그래프 출력
plot_decision_regions(X, y, classifier=ppn)
plt.xlabel('Alcohol (Standardized)')
plt.ylabel('Proline (Standardized)')
plt.legend(loc='upper left')
plt.show()
```