```
# -*- coding: utf-8 -*-
"""
Cap 12
Created on Mon Apr  6 13:31:08 2020

@author: Sebastian Raschka

adattamento: Massimo Campioni
"""
'======='
'1 FASE'
'======='
import numpy as np


class Perceptron(object):
    """ Classificatore del Perceptron.

    Parametri
    ------------
    eta : float
        Tasso d'apprendimento (tra 0.0 and 1.0)
    n_iter : int
        Passi attraverso la serie di dati d'apprendimento.

    Attributi
    -----------
    w_ : 1d-array
        Pesi dopo l'adattamento.
    errors_ : list
        Numero di errete classificazioni (aggiornate) in ogni epoch.

    """
    def __init__(self, eta=0.01, n_iter=10):
        self.eta = eta
        self.n_iter = n_iter

    def fit(self, X, y):

        """Adatta i dati.

        Parametri
        ----------
        X : {tipo-array}, shape = [n_campioni, n_caratteristiche]
            Vettori di addestramento, dove n_campioni è il numerodi campioni e
            n_caratteristiche è ilnumero delle caratteristiche.
        y : tipo-array, shape = [n_campioni]
            Valori obiettivo.

        Returns
        -------
        self : object

        """
        self.w_ = np.zeros(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):
        """Calcola l'ingresso di rete"""
        return np.dot(X, self.w_[1:]) + self.w_[0]

    def predict(self, X):
        """Return: etichetta di classe dopo un gradino unitario"""
        return np.where(self.net_input(X) >= 0.0, 1, -1)
import pandas as pd  # modulo di gestione dei dati

'======='
'2 FASE'
'======='

df = pd.read_csv('D:/RETI NEURALI/iris.data', header=None) #importa il file
iris.data
print(df.tail())  #mostra le ultime n righe (predefinito=5)

import matplotlib.pyplot as plt

# sceglie setosa e  versicolor
y = df.iloc[0:100, 4].values # prende le prime 100 righe
y = np.where(y == 'Iris-setosa', -1, 1) # assegna -1 a iris-setosa, 1 a
versicolor

# estrae la lunghezza di sepal e di petal
X = df.iloc[0:100, [0, 2]].values

# disegna i dati in modo dispersione
fig, ax = plt.subplots(facecolor='teal', alpha=0.1)
plt.style.use('ggplot')
plt.tick_params(labelcolor='y')

plt.scatter(X[:50, 0], X[:50, 1],
            color='red', marker='o', label='setosa')
plt.scatter(X[50:100, 0], X[50:100, 1],
            color='blue', marker='x', label='versicolor')

plt.xlabel('Lunghezza del sepalo (cm)')
plt.ylabel('Lunghezza del petalo (cm)')
plt.legend(loc='upper left')

plt.tight_layout()

plt.show()
'======='
'3 FASE'
'======='
ppn = Perceptron(eta=0.1, n_iter=10)

ppn.fit(X, y)

fig, ax = plt.subplots(facecolor='teal', alpha=0.1)
plt.style.use('ggplot')
plt.tick_params(labelcolor='y')



plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
plt.xlabel('Epoch',color='y')
plt.ylabel('N. errori errata classificazione',color='y')
```

```
plt.tight_layout()
# plt.savefig('./perceptron_1.png', dpi=300)
plt.show()
'======='
'4 FASE'
'======='
from matplotlib.colors import ListedColormap

def plot_decision_regions(X, y, classifier, resolution=0.02):

    # setup marker generator and color map
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])

    # plot the decision surface
    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=0.4, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())

    # plot class samples

    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],alpha=0.8,
c=cmap(idx),edgecolor='black',marker=markers[idx],label=cl)

fig, ax = plt.subplots(facecolor='teal', alpha=0.1)
plt.style.use('ggplot')
plt.tick_params(labelcolor='y')
plot_decision_regions(X, y, classifier=ppn)
plt.xlabel('Lunghezza del sepalo (cm)',color='y')
plt.ylabel('Lunghezza del petalo (cm)',color='y')
plt.legend(loc='lower right')

plt.tight_layout()
# plt.savefig('./perceptron_2.png', dpi=300)
plt.show()
```