# -\*- coding: utf-8 -\*-

from sklearn import datasets

from sklearn.neural\_network import MLPClassifier

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.colors import ListedColormap

# アヤメのデータをロードし、変数irisに格納

iris = datasets.load\_iris()

# 特徴量のセットを変数Xに、ターゲットを変数yに格納

X = iris.data

y = iris.target

# 特徴量を外花被片の長さ(sepal length)と幅(sepal width)の

# 2つのみに制限(2次元で考えるため)

X = X[:,:2]

# 分類用に多層ニューラルネットワークを用意

# ランダムな要素を固定した場合（毎回同じ結果）

clf = MLPClassifier(hidden\_layer\_sizes=(100, ), max\_iter=10000, tol=0.00001, random\_state=1)

# 毎回ランダムな場合

#clf = MLPClassifier(hidden\_layer\_sizes=(100, ), max\_iter=10000, tol=0.00001, random\_state=None)

# ニューラルネットワークの学習

print('学習中…')

clf.fit(X, y)

##### 分類結果を背景の色分けにより表示

# 外花被片の長さ(sepal length)と幅(sepal width)の

# 最小値と最大値からそれぞれ1ずつ広げた領域を

# グラフ表示エリアとする

x\_min = min(X[:,0]) - 1

x\_max = max(X[:,0]) + 1

y\_min = min(X[:,1]) - 1

y\_max = max(X[:,1]) + 1

# グラフ表示エリアを縦横400ずつのグリッドに区切る

# (分類クラスに応じて背景に色を塗るため)

XX, YY = np.mgrid[x\_min:x\_max:400j, y\_min:y\_max:400j]

# グリッドの点をscikit-learn用の入力に並べなおす

Xg = np.c\_[XX.ravel(), YY.ravel()]

# 各グリッドの点が属するクラス(0～2)の予測をZに格納

Z = clf.predict(Xg)

# グリッド上に並べなおす

Z = Z.reshape(XX.shape)

# クラス0 (iris setosa) が薄オレンジ (1, 0.93, 0.5, 1)

# クラス1 (iris versicolor) が薄青 (0.5, 1, 1, 1)

# クラス2 (iris virginica) が薄緑 (0.5, 0.75, 0.5, 1)

cmap0 = ListedColormap([(0, 0, 0, 0), (1, 0.93, 0.5, 1)])

cmap1 = ListedColormap([(0, 0, 0, 0), (0.5, 1, 1, 1)])

cmap2 = ListedColormap([(0, 0, 0, 0), (0.5, 0.75, 0.5, 1)])

# グラフウインドウの大きさを横長に定める。

plt.subplots(figsize=(11.2,4.2))

# 縦に1枚、横に2枚のグラフの1枚目を設定

plt.subplot(121)

# 背景の色を表示

plt.pcolormesh(XX, YY, Z==0, cmap=cmap0)

plt.pcolormesh(XX, YY, Z==1, cmap=cmap1)

plt.pcolormesh(XX, YY, Z==2, cmap=cmap2)

# 軸ラベルを設定

plt.xlabel('sepal length')

plt.ylabel('sepal width')

##### ターゲットに応じた色付きでデータ点を表示

# iris setosa (y=0) のデータのみを取り出す

Xc0 = X[y==0]

# iris versicolor (y=1) のデータのみを取り出す

Xc1 = X[y==1]

# iris virginica (y=2) のデータのみを取り出す

Xc2 = X[y==2]

# iris setosa のデータXc0をプロット

plt.scatter(Xc0[:,0], Xc0[:,1], c='#E69F00', linewidths=0.5, edgecolors='black')

# iris versicolor のデータXc1をプロット

plt.scatter(Xc1[:,0], Xc1[:,1], c='#56B4E9', linewidths=0.5, edgecolors='black')

# iris virginica のデータXc2をプロット

plt.scatter(Xc2[:,0], Xc2[:,1], c='#008000', linewidths=0.5, edgecolors='black')

# 縦に1枚、横に2枚のグラフの2枚目を設定

plt.subplot(122)

# 損失関数のグラフの軸ラベルを設定

plt.xlabel('time step')

plt.ylabel('loss')

# グラフ縦軸の範囲を0以上と定める

plt.ylim(0, max(clf.loss\_curve\_))

# 損失関数の時間変化を描画

plt.plot(clf.loss\_curve\_)

# 描画したグラフを表示

plt.show()