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

from sklearn import datasets

from sklearn.neural\_network import MLPClassifier

import matplotlib.pyplot as plt

# 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

# 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)

# データを分類器に与え、予測を得る

result = clf.predict(X)

print('ターゲット')

print(y)

print('機械学習による予測')

print(result)

# データ数をtotalに格納

total = len(X)

# ターゲット（正解）と予測が一致した数をsuccessに格納

success = sum(result==y)

# 正解率をパーセント表示

print('正解率')

print(100.0\*success/total)

# 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()