追手門学院大学　理工学部　幸谷担当講義

レポート用紙

講義名　：数値解析２	年月日：　2025 年　10 月　10 日（ 金 ）
学籍番号：2318082	氏名：鈴木　祐亮

<train_custom_loop.py>

```
# train_custom_loop.py: 学習用コード
import sys
sys.path.append('..')
import numpy as np
import matplotlib.pyplot as plt

# dataset/spiral.py
from dataset import spiral
# common/optimizer.py
from common.optimizer import SGD
# two_layer_net.py
from two_layer_net import TwoLayerNet

# (1) Hyper parameters の設定
max_epoch = 300 # epoch は学習の単位: 「1 epoch = 全てのデータを 1 回学習」
batch_size = 30
hidden_size = 10
learning_rate = 1.0

# (2) データの読み込み、モデルとオプティマイザの設定
x, t = spiral.load_data()
model = TwoLayerNet(input_size = 2, hidden_size = hidden_size, output_size = 3)
optimizer = SGD(lr = learning_rate)

# 学習で使用する変数
data_size = len(x)
max_iters = data_size // batch_size # floor
total_loss = 0
loss_count = 0
loss_list = []

# 学習のためのメインループ
for epoch in range(max_epoch):
    # (3) データのシャッフル
    idx = np.random.permutation(data_size)
    x = x[idx]
    t = t[idx]

    for iters in range(max_iters):
        batch_x = x[iters * batch_size:(iters + 1) * batch_size]
        batch_t = t[iters * batch_size:(iters + 1) * batch_size]

        # (4) 勾配を求め、パラメータを更新
        loss = model.forward(batch_x, batch_t)
        model.backward()
        optimizer.update(model.params, model.grads)

        total_loss += loss
        loss_count += 1

        # 定期的に学習効果を出力（この場合は 10 回ごと）
        if (iters + 1) % 10 == 0:
            avg_loss = total_loss / loss_count
            print('epoch %d | iter %d / %d | loss %.2f' % (epoch + 1, iters + 1, max_iters,
avg_loss))
            loss_list.append(avg_loss)
            total_loss, loss_count = 0, 0
```

```
# グラフ描画：学習効果(エラー)のプロット
fig, ax = plt.subplots()
plt.plot(loss_list)
plt.plot(np.arange(len(loss_list)), loss_list, label = 'train')
plt.xlabel('iterations(x10)')
plt.ylabel('loss')
fig.savefig('two_layer_nn.png')
plt.show()
```

<two_layer_net.py>

```
# two_layer_net.py: 2 層 NN
# P.43 -
import sys
sys.path.append('..')
import numpy as np
# common/layers.py
from common.layers import Affine, Sigmoid, SoftmaxWithLoss
import matplotlib.pyplot as plt   # グラフ描画用

# TwoLayerNet class
class TwoLayerNet:
    def __init__(self, input_size, hidden_size, output_size):
        I, H, O = input_size, hidden_size, output_size

        # 重みとバイアスの初期化
        # 重みを小さいランダム値にすると学習が進みやすい
        W1 = 0.01 * np.random.randn(I, H)
        b1 = np.zeros(H)
        W2 = 0.01 * np.random.randn(H, O)
        b2 = np.zeros(O)

        # Layer の生成
        self.layers = [
            Affine(W1, b1),
            Sigmoid(),
            Affine(W2, b2)
        ]
        self.loss_layer = SoftmaxWithLoss()

        # 全ての重みと勾配をリストにまとめる
        self.params, self.grads = [], []
        for layer in self.layers:
            self.params += layer.params
            self.grads += layer.grads

    # predict 関数
    def predict(self, x):
        for layer in self.layers:
            x = layer.forward(x) # 予測を進める
        return x

    # forward 関数
    def forward(self, x, t):
        score = self.predict(x)
        loss = self.loss_layer.forward(score, t)
        return loss

    # backward 関数
    def backward(self, dout = 1):
        dout = self.loss_layer.backward(dout)
        for layer in reversed(self.layers):
            dout = layer.backward(dout)
        return dout
```

レポート用紙