|  |  |
| --- | --- |
| 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

グラフ, 折れ線グラフ, ヒストグラム

自動的に生成された説明