数値解析 2
第5回

追手門学院大学
理工学部　数理・データサイエンス学科
幸谷智紀

https://na-inet.jp/deeplearning/

common/trainer.pyの追加→学習部分をクラス化

```python
# first_trainer_class.py: Trainer Class利用例
import sys
sys.path.append('..') # 親ディレクトリのファイルをインポート
from common.optimizer import SGD
from common.trainer import Trainer
from dataset import spiral
from two_layer_net import TwoLayerNet

# パラメータ設定
max_epoch = 300
batch_size = 30
hidden_size = 10
learning_rate = 1.0

# 学習用データ読み込み
x, t = spiral.load_data()
model = TwoLayerNet(input_size = 2, hidden_size =
hidden_size, output_size = 3)
optimizer = SGD(lr = learning_rate)

# 学習実行
trainer = Trainer(model, optimizer)
trainer.fit(x, t, max_epoch, batch_size, eval_interval = 10)
trainer.plot(png_filename = 'first_trainer_class.png')
```

- first_trainer_class.py・・・メインルーチン
- dataset/spiral.py・・・学習データ生成
- commom/optimizer.py・・・SGD最適化
- commom/trainer.py・・・学習ループ
  - commom/np.py・・・NumPyとCuPyの読み込み
  - common/util.py・・・各種関数
- two_layer_net.py・・・2層NN
  - common/layers.py・・・各種変換クラス
  - common/functions.py・・・各種関数定義

Trainerクラス: commom/trainer.py

- ヘッダ部分
- Remove_duplicate関数
- Trainerクラス：学習部分をすべてクラス化

commom/trainer.py: ヘッダ部分

```python
# trainer.py: 学習用 Trainer class
import sys
sys.path.append('..')
import numpy
import time
import matplotlib.pyplot as plt

# common/np.py
from common.np import *
from common.util import clip_grads
```

- 前回までに構築したメインスクリプトの機能をここで取り込む

commom/trainer.py: remove_dupulicate関数(1/2)

```python
# 重みの重複を一つに集約し、重複分を勾配に加算する(p.117に解説あり)
def remove_duplicate(params, grads):
    params, grads = params[:], grads[:] # リストのコピー

    while True:
        find_flag = False
        L = len(params)

        for i in range(0, L - 1):
            for j in range(i + 1, L):
                # 重みを共有する場合
                if params[i] is params[j]:
                    grads[i] += grads[j] # 勾配の加算
                    find_flag = True
                    params.pop(j)
                    grads.pop(j)
```

commom/trainer.py: remove_dupulicate関数(2/2)

```
28                  # 転置行列として重みを共有する場合(weight tying)
29                  elif params[i].ndim == 2 and params[j].ndim == 2 and
                    params[i].T.shape == params[j].shape and np.all(params
                    [i].T == params[j]):
30                      grads[i] == grads[j].T
31                      find_flag = True
32                      params.pop(j)
33                      grads.pop(j)
34
35                  if find_flag: break
36              if find_flag: break
37
38          if not find_flag: break
39
40      return params, grads
```

commom/trainer.py: Trainerクラス(1/5)

```python
# Trainer class
class Trainer:
    def __init__(self, model, optimizer):
        self.model = model
        self.optimizer = optimizer
        self.loss_list = []
        self.eval_interval = None
        self.current_epoch = 0
```

- 学習用クラスの初期化

commom/trainer.py: Trainerクラス(2/5)

```
52      def fit(self, x, t, max_epoch = 10, batch_size = 32, max_grad =
        None, eval_interval = 20):
53          data_size = len(x)
54          max_iters = data_size // batch_size
55          self.eval_interval = eval_interval
56          model, optimizer = self.model, self.optimizer
57          total_loss = 0
58          loss_count = 0
59
60          start_time = time.time()
61          for epoch in range(max_epoch):
62              # シャッフル
63              idx = numpy.random.permutation(numpy.arange(data_size))
64              x = x[idx]
65              t = t[idx]
```

- 順方向の学習

commom/trainer.py: Trainerクラス(3/5)

```
67              for iters in range(max_iters):
68                  batch_x = x[iters * batch_size: (iters + 1) *
                    batch_size]
69                  batch_t = t[iters * batch_size: (iters + 1) *
                    batch_size]
70
71                  # 勾配を求めてパラメータを更新
72                  loss = model.forward(batch_x, batch_t)
73                  model.backward()
74                  params, grads = remove_duplicate(model.params, model.
                    grads) # 共有重みを一つに集約
75                  if max_grad is not None:
76                      clip_grads(grads, max_grad)
77                  optimizer.update(params, grads)
78                  total_loss += loss
79                  loss_count += 1
```

commom/trainer.py: Trainerクラス(4/5)

```
81                # 評価
82                if (eval_interval is not None) and (iters %
                  eval_interval) == 0:
83                    avg_loss = float(total_loss) / float(loss_count)
84                    elapsed_time = time.time() - start_time
85                    print('| epoch %d | iter %d / %d | time %d[s] |
                      loss %.2f' % (self.current_epoch + 1, iters + 1,
                      max_iters, elapsed_time, avg_loss))
86                    self.loss_list.append(float(avg_loss))
87                    total_loss, loss_count = 0, 0
88
89            self.current_epoch += 1
```

commom/trainer.py: Trainerクラス(5/5)

```
 91     def plot(self, ylim = None, png_filename = None):
 92         #print('self.loss_list: ', self.loss_list)
 93         x = numpy.arange(len(self.loss_list))
 94         if ylim is not None:
 95             plt.ylim(*ylim) # y軸範囲を制限
 96         plt.plot(x, self.loss_list, label = 'Train')
 97         plt.xlabel('Iterations (x' + str(self.eval_interval) + ')')
 98         plt.ylabel('Loss')
 99         #plt.show()
100         if png_filename is None:
101             png_filename = 'test.png'
102         plt.savefig(png_filename)
```

- 学習曲線の描画

commom/np.py

```python
# common/np.py: CuPyとNumPyの切り替え

# GPUの設定
GPU = False # 不使用
#GPU = True # GPU使用

# GPU使用時
if GPU:
    import cupy as np # NumPyの代わりにCuPyを使用

    np.cuda.set_allocator(np.cuda.MemoryPool().malloc) # メモリ割り当て関数の設定

    print('------------------')
    print('---- GPU Mode ----')
    print('------------------')
else: # CPU使用時
    import numpy as np # NumPyを使用
```

- GPU(CuPy)を使用して主として線形計算を高速化
- NVIDIAのGPUで動作するCUDAを利用

common/util.py (1/2)

```python
# util.py: 学習用途に使用する様々な関数

import sys
sys.path.append('..')
import os
from common.np import * # NumPy or CuPy


# CPU配列に変換
def to_cpu(x):
    import numpy
    if type(x) == numpy.ndarray:
        return x
    return np.asnumpy(x)

# GPU配列に変換
def to_gpu(x):
    import cupy
    if type(x) == cupy.ndarray:
        return x
    return cupy.asarray(x)
```

- NumPy(CPU)とCuPy(GPU)の切り替えを自動化

common/util.py (2/2)

```python
# 勾配の計算
def clip_grads(grads, max_norm):
    total_norm = 0
    for grad in grads:
        total_norm += np.sum(grad ** 2)
    total_norm = np.sqrt(total_norm)

    rate = max_norm  / (total_norm + 1.0e-6)
    if rate < 1:
        for grad in grads:
            grad *= rate
```

本日の課題

- trainer_custom_loop.pyをfirst_trainer_class.pyに書き換え，同じ学習効果が得られることを，学習曲線を求めて確認せよ。