five_layer_net.py

```
# coding: utf-8
import numpy as np
from common.layers import *
from common.gradient import numerical_gradient
from collections import OrderedDict
from common.optimizer import Adam

class fiveLayerNet:

    def __init__(self, input_size, hidden_size1, hidden_size2, hidden_size3, hidden_size4, output_size,
weight_init_std=0.1):
        # 가중치 초기화
        self.params = {}
        self.params['W1'] = weight_init_std * np.random.randn(input_size, hidden_size1)
        self.params['b1'] = np.zeros(hidden_size1)
        self.params['W2'] = weight_init_std * np.random.randn(hidden_size1, hidden_size2)
        self.params['b2'] = np.zeros(hidden_size2)
        self.params['W3'] = weight_init_std * np.random.randn(hidden_size2, hidden_size3)
        self.params['b3'] = np.zeros(hidden_size3)
        self.params['W4'] = weight_init_std * np.random.randn(hidden_size3, hidden_size4)
        self.params['b4'] = np.zeros(hidden_size4)
        self.params['W5'] = weight_init_std * np.random.randn(hidden_size4, output_size)
        self.params['b5'] = np.zeros(output_size)

        # 정규화 파라미터
        self.gamma1 = np.ones(hidden_size1)
        self.beta1 = np.zeros(hidden_size1)
        self.gamma2 = np.ones(hidden_size2)
        self.beta2 = np.zeros(hidden_size2)
        self.gamma3 = np.ones(hidden_size3)
        self.beta3 = np.zeros(hidden_size3)
        self.gamma4 = np.ones(hidden_size4)
        self.beta4 = np.zeros(hidden_size4)

        # 계층 생성
        self.layers = OrderedDict()
        self.layers['Affine1'] = Affine(self.params['W1'], self.params['b1'])
        self.layers['BatchNorm1'] = BatchNormalization(self.gamma1, self.beta1)
        self.layers['Relu1'] = Relu()
        self.layers['Affine2'] = Affine(self.params['W2'], self.params['b2'])
        self.layers['BatchNorm2'] = BatchNormalization(self.gamma2, self.beta2)
        self.layers['Relu2'] = Relu()
        self.layers['Affine3'] = Affine(self.params['W3'], self.params['b3'])
        self.layers['BatchNorm3'] = BatchNormalization(self.gamma3, self.beta3)
        self.layers['Relu3'] = Relu()
        self.layers['Affine4'] = Affine(self.params['W4'], self.params['b4'])
        self.layers['BatchNorm4'] = BatchNormalization(self.gamma4, self.beta4)
```

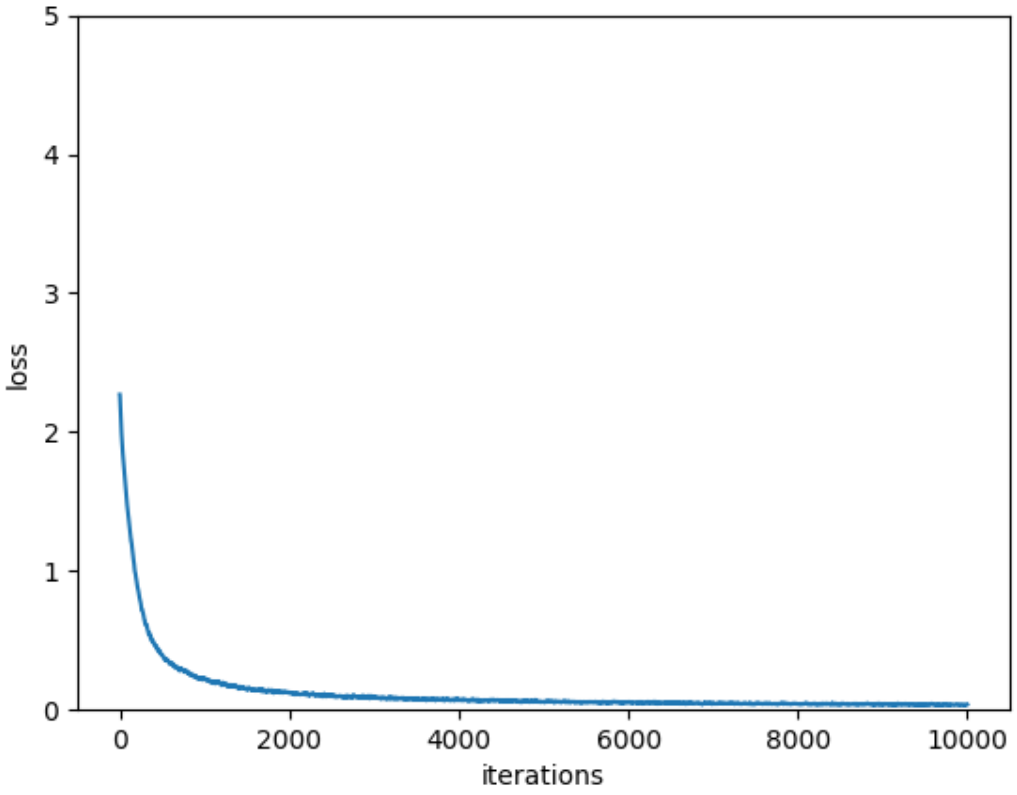
```
        self.layers['Relu4'] = Relu()
        self.layers['Affine5'] = Affine(self.params['W5'], self.params['b5'])
        self.lastLayer = SoftmaxWithLoss()

        self.optimizer = Adam()

    def predict(self, x):
        for layer in self.layers.values():
            x = layer.forward(x)
        return x

    def loss(self, x, t):
        y = self.predict(x)
        return self.lastLayer.forward(y, t)

    def accuracy(self, x, t):
        y = self.predict(x)
        y = np.argmax(y, axis=1)
        if t.ndim != 1: t = np.argmax(t, axis=1)
        accuracy = np.sum(y == t) / float(x.shape[0])
        return accuracy

    def numerical_gradient(self, x, t):
        loss_W = lambda W: self.loss(x, t)
        grads = {}
        grads['W1'] = numerical_gradient(loss_W, self.params['W1'])
        grads['b1'] = numerical_gradient(loss_W, self.params['b1'])
        grads['W2'] = numerical_gradient(loss_W, self.params['W2'])
        grads['b2'] = numerical_gradient(loss_W, self.params['b2'])
        grads['W3'] = numerical_gradient(loss_W, self.params['W3'])
        grads['b3'] = numerical_gradient(loss_W, self.params['b3'])
        grads['W4'] = numerical_gradient(loss_W, self.params['W4'])
        grads['b4'] = numerical_gradient(loss_W, self.params['b4'])
        grads['W5'] = numerical_gradient(loss_W, self.params['W5'])
        grads['b5'] = numerical_gradient(loss_W, self.params['b5'])
        return grads

    def gradient(self, x, t):
        # forward
        self.loss(x, t)
        # backward
        dout = 1
        dout = self.lastLayer.backward(dout)
        layers = list(self.layers.values())
        layers.reverse()
        for layer in layers:
            dout = layer.backward(dout)
```

```
        # 결과 저장
        grads = {}
        grads['W1'], grads['b1'] = self.layers['Affine1'].dW, self.layers['Affine1'].db
        grads['W2'], grads['b2'] = self.layers['Affine2'].dW, self.layers['Affine2'].db
        grads['W3'], grads['b3'] = self.layers['Affine3'].dW, self.layers['Affine3'].db
        grads['W4'], grads['b4'] = self.layers['Affine4'].dW, self.layers['Affine4'].db
        grads['W5'], grads['b5'] = self.layers['Affine5'].dW, self.layers['Affine5'].db
        grads['gamma1'], grads['beta1'] = self.layers['BatchNorm1'].dgamma, self.layers['BatchNorm1'].dbeta
        grads['gamma2'], grads['beta2'] = self.layers['BatchNorm2'].dgamma, self.layers['BatchNorm2'].dbeta
        grads['gamma3'], grads['beta3'] = self.layers['BatchNorm3'].dgamma, self.layers['BatchNorm3'].dbeta
        grads['gamma4'], grads['beta4'] = self.layers['BatchNorm4'].dgamma, self.layers['BatchNorm4'].dbeta
        self.optimizer.update(self.params, grads)
        return grads
```

train_neuralnet2.py

```
# coding: utf-8
import sys, os
sys.path.append(os.pardir)

import numpy as np
from dataset.mnist import load_mnist
#from two_layer_net import TwoLayerNet
from five_layer_net import fiveLayerNet
import matplotlib.pyplot as plt
from common.util import smooth_curve

# 데이터 읽기
(x_train, t_train), (x_test, t_test) = load_mnist(normalize=True,
one_hot_label=True)

network = fiveLayerNet(input_size=784, hidden_size1=15, hidden_size2=13,
hidden_size3=13, hidden_size4 = 11, output_size=10)
iters_num = 10000
train_size = x_train.shape[0]
batch_size = 1200
learning_rate = 0.019

train_loss_list = []
train_acc_list = []
test_acc_list = []

iter_per_epoch = max(train_size / batch_size, 1)

for i in range(iters_num):
    batch_mask = np.random.choice(train_size, batch_size)
    x_batch = x_train[batch_mask]
```

```
    t_batch = t_train[batch_mask]

    # 기울기 계산
    grad = network.gradient(x_batch, t_batch)

    # 갱신
    for key in ('W1', 'b1', 'W2', 'b2', 'W3', 'b3', 'W4', 'b4', 'W5', 'b5'):
        network.params[key] -= learning_rate * grad[key]

    loss = network.loss(x_batch, t_batch)
    train_loss_list.append(loss)

    if i % iter_per_epoch == 0:
        train_acc = network.accuracy(x_train, t_train)
        test_acc = network.accuracy(x_test, t_test)
        train_acc_list.append(train_acc)
        test_acc_list.append(test_acc)
        print("train acc:", train_acc, "test acc:", test_acc)

max_iterations = len(train_loss_list)
x = np.arange(max_iterations)

plt.plot(x, smooth_curve(train_loss_list))
plt.xlabel("iterations")
plt.ylabel("loss")
plt.ylim(0, 5)
plt.show()
```



터미널 로그

```
PS C:₩Users₩장주훈₩Desktop₩대학교₩3학년₩인공지능개론 - 최인엽교수님₩deep-learning-from-scratch-master> & C:/Users/장주훈/AppData/Local/Programs/Python/Python38-32/python.exe "c:/Users/장주훈/Desktop/대학교/3학년/인공지능개론 - 최인엽교수님/deep-learning-from-scratch-master/ch05/train_neuralnet2.py"
train acc: 0.10688333333333333 test acc: 0.1038
train acc: 0.5525666666666667 test acc: 0.5536
train acc: 0.6644333333333333 test acc: 0.672
train acc: 0.73635 test acc: 0.7402
train acc: 0.77715 test acc: 0.7786
train acc: 0.8738 test acc: 0.8705
train acc: 0.89905 test acc: 0.896
train acc: 0.9140666666666667 test acc: 0.9113
train acc: 0.92435 test acc: 0.9173
train acc: 0.9337166666666666 test acc: 0.9265
train acc: 0.9393833333333333 test acc: 0.9303
train acc: 0.9448 test acc: 0.9364
train acc: 0.9469 test acc: 0.9382
train acc: 0.9511 test acc: 0.9423
```

train acc: 0.9533 test acc: 0.9435
train acc: 0.9555333333333333 test acc: 0.9459
train acc: 0.9570833333333333 test acc: 0.9449
train acc: 0.9589833333333333 test acc: 0.9471
train acc: 0.9604166666666667 test acc: 0.9491
train acc: 0.9611166666666666 test acc: 0.9504
train acc: 0.9627833333333333 test acc: 0.9512
train acc: 0.9639 test acc: 0.9525
train acc: 0.9645666666666667 test acc: 0.9513
train acc: 0.9647333333333333 test acc: 0.9525
train acc: 0.9657333333333333 test acc: 0.9518
train acc: 0.96635 test acc: 0.953
train acc: 0.96735 test acc: 0.952
train acc: 0.9688833333333333 test acc: 0.9539
train acc: 0.9690833333333333 test acc: 0.955
train acc: 0.9695333333333334 test acc: 0.9533
train acc: 0.9706 test acc: 0.955
train acc: 0.9705833333333334 test acc: 0.9551
train acc: 0.9708 test acc: 0.955
train acc: 0.9714666666666667 test acc: 0.9554
train acc: 0.9719333333333333 test acc: 0.955
train acc: 0.97255 test acc: 0.9536
train acc: 0.97295 test acc: 0.9557
train acc: 0.9736833333333333 test acc: 0.9563
train acc: 0.9738666666666667 test acc: 0.9559
train acc: 0.9747666666666667 test acc: 0.9558
train acc: 0.9749333333333333 test acc: 0.9558
train acc: 0.9756333333333334 test acc: 0.9572
train acc: 0.9755166666666667 test acc: 0.9561
train acc: 0.9758833333333333 test acc: 0.9566
train acc: 0.9769833333333333 test acc: 0.9563
train acc: 0.9764666666666667 test acc: 0.9565
train acc: 0.97725 test acc: 0.9556
train acc: 0.9771833333333333 test acc: 0.9562
train acc: 0.9775 test acc: 0.9572
train acc: 0.97755 test acc: 0.9576
train acc: 0.97705 test acc: 0.9546
train acc: 0.9786833333333333 test acc: 0.9555
train acc: 0.9784666666666667 test acc: 0.9574
train acc: 0.9789 test acc: 0.9563

train acc: 0.9784333333333334 test acc: 0.9557
train acc: 0.9789 test acc: 0.9571
train acc: 0.9798333333333333 test acc: 0.9562
train acc: 0.9798 test acc: 0.9567
train acc: 0.9798666666666667 test acc: 0.9552
train acc: 0.9797333333333333 test acc: 0.956
train acc: 0.9803333333333333 test acc: 0.957
train acc: 0.98105 test acc: 0.9565
train acc: 0.9815833333333334 test acc: 0.9559
train acc: 0.9815 test acc: 0.9555
train acc: 0.9810833333333333 test acc: 0.9565
train acc: 0.9815833333333334 test acc: 0.9563
train acc: 0.98195 test acc: 0.9554
train acc: 0.9812833333333333 test acc: 0.9563
train acc: 0.9821666666666666 test acc: 0.9557
train acc: 0.9819166666666667 test acc: 0.9548
train acc: 0.9824666666666667 test acc: 0.9549
train acc: 0.9823333333333333 test acc: 0.9547
train acc: 0.98205 test acc: 0.9554
train acc: 0.98255 test acc: 0.9564
train acc: 0.98295 test acc: 0.9558
train acc: 0.9826333333333334 test acc: 0.9559
train acc: 0.9828666666666667 test acc: 0.955
train acc: 0.98345 test acc: 0.9546
train acc: 0.98345 test acc: 0.9547
train acc: 0.9837 test acc: 0.9541
train acc: 0.9837666666666667 test acc: 0.9547
train acc: 0.9839833333333333 test acc: 0.955
train acc: 0.9841666666666666 test acc: 0.9554
train acc: 0.9846666666666667 test acc: 0.9519
train acc: 0.9844 test acc: 0.9537
train acc: 0.9846333333333334 test acc: 0.9549
train acc: 0.9841833333333333 test acc: 0.955
train acc: 0.9846666666666667 test acc: 0.9546
train acc: 0.98445 test acc: 0.9551
train acc: 0.9851 test acc: 0.9536
train acc: 0.9845333333333334 test acc: 0.9546
train acc: 0.9848833333333333 test acc: 0.9539
train acc: 0.9855333333333334 test acc: 0.956
train acc: 0.9854333333333334 test acc: 0.9544

train acc: 0.98565 test acc: 0.9546
train acc: 0.9848333333333333 test acc: 0.9526
train acc: 0.9856166666666667 test acc: 0.9543
train acc: 0.9859333333333333 test acc: 0.9544
train acc: 0.9860666666666666 test acc: 0.9545
train acc: 0.9858666666666667 test acc: 0.956
train acc: 0.98605 test acc: 0.9539
train acc: 0.98605 test acc: 0.9544
train acc: 0.986 test acc: 0.9541
train acc: 0.98635 test acc: 0.9544
train acc: 0.9865 test acc: 0.9535
train acc: 0.9863333333333333 test acc: 0.9537
train acc: 0.9866833333333334 test acc: 0.9556
train acc: 0.9870166666666667 test acc: 0.9553
train acc: 0.9869833333333333 test acc: 0.9547
train acc: 0.9869333333333333 test acc: 0.9527
train acc: 0.9874 test acc: 0.9539
train acc: 0.98725 test acc: 0.9517
train acc: 0.98695 test acc: 0.9545
train acc: 0.98705 test acc: 0.9538
train acc: 0.9867666666666667 test acc: 0.9535
train acc: 0.9870333333333333 test acc: 0.9521
train acc: 0.9878333333333333 test acc: 0.9541
train acc: 0.9874666666666667 test acc: 0.952
train acc: 0.9877833333333333 test acc: 0.9521
train acc: 0.9872333333333333 test acc: 0.9517
train acc: 0.9878166666666667 test acc: 0.9533
train acc: 0.988 test acc: 0.9522
train acc: 0.9873166666666666 test acc: 0.9531
train acc: 0.9880666666666666 test acc: 0.9526
train acc: 0.98855 test acc: 0.9525
train acc: 0.9887666666666667 test acc: 0.9547
train acc: 0.9885666666666667 test acc: 0.9532
train acc: 0.9885666666666667 test acc: 0.952
train acc: 0.9884833333333334 test acc: 0.9522
train acc: 0.9876666666666667 test acc: 0.9516
train acc: 0.9881 test acc: 0.9523
train acc: 0.9889 test acc: 0.9531
train acc: 0.9888833333333333 test acc: 0.9511
train acc: 0.9889833333333333 test acc: 0.9529

train acc: 0.98875 test acc: 0.9523
train acc: 0.9887333333333334 test acc: 0.953
train acc: 0.9884833333333334 test acc: 0.9524
train acc: 0.9888833333333333 test acc: 0.9514
train acc: 0.9887166666666667 test acc: 0.9531
train acc: 0.9891 test acc: 0.9529
train acc: 0.9891833333333333 test acc: 0.9516
train acc: 0.9898666666666667 test acc: 0.9522
train acc: 0.9892833333333333 test acc: 0.9522
train acc: 0.9894833333333334 test acc: 0.9537
train acc: 0.9895333333333334 test acc: 0.9527
train acc: 0.9891833333333333 test acc: 0.952
train acc: 0.99 test acc: 0.9519
train acc: 0.9899 test acc: 0.9505
train acc: 0.9903833333333333 test acc: 0.9505
train acc: 0.9898166666666667 test acc: 0.951
train acc: 0.9899333333333333 test acc: 0.9517
train acc: 0.9895833333333334 test acc: 0.9521
train acc: 0.9896333333333334 test acc: 0.9526
train acc: 0.98955 test acc: 0.9507
train acc: 0.9900333333333333 test acc: 0.9512
train acc: 0.99005 test acc: 0.9513
train acc: 0.99035 test acc: 0.9507
train acc: 0.9902666666666666 test acc: 0.9511
train acc: 0.9903 test acc: 0.9514
train acc: 0.9902833333333333 test acc: 0.9514
train acc: 0.9906833333333334 test acc: 0.9533
train acc: 0.9901666666666666 test acc: 0.9539
train acc: 0.9900833333333333 test acc: 0.9516
train acc: 0.9908166666666667 test acc: 0.9506
train acc: 0.9900166666666667 test acc: 0.9509
train acc: 0.99045 test acc: 0.9518
train acc: 0.9906833333333334 test acc: 0.9503
train acc: 0.9909 test acc: 0.9516
train acc: 0.9908166666666667 test acc: 0.9512
train acc: 0.9905 test acc: 0.9509
train acc: 0.9908833333333333 test acc: 0.9523
train acc: 0.99105 test acc: 0.9491
train acc: 0.9906333333333334 test acc: 0.9513
train acc: 0.9906333333333334 test acc: 0.9511

```
train acc: 0.99065 test acc: 0.95
train acc: 0.9907833333333333  test acc: 0.9499
train acc: 0.99075 test acc: 0.9505
train acc: 0.9909833333333333  test acc: 0.9494
train acc: 0.9914833333333334  test acc: 0.9493
train acc: 0.9914 test acc: 0.9498
train acc: 0.9913666666666666  test acc: 0.9514
train acc: 0.9904666666666667  test acc: 0.9503
train acc: 0.9911666666666666  test acc: 0.9506
train acc: 0.9905166666666667  test acc: 0.9494
train acc: 0.99155 test acc: 0.9517
train acc: 0.9915166666666667  test acc: 0.9509
train acc: 0.9914333333333334  test acc: 0.9503
train acc: 0.99155 test acc: 0.9503
train acc: 0.9916666666666667  test acc: 0.9483
train acc: 0.9916333333333334  test acc: 0.9493
train acc: 0.9917 test acc: 0.9486
train acc: 0.9918666666666667  test acc: 0.9513
train acc: 0.9909 test acc: 0.9508
train acc: 0.99115 test acc: 0.9497
train acc: 0.9918666666666667  test acc: 0.9501
train acc: 0.9917166666666667  test acc: 0.9499
train acc: 0.9916166666666667  test acc: 0.9488
train acc: 0.9915166666666667  test acc: 0.9493
train acc: 0.9917833333333334  test acc: 0.9505
train acc: 0.9920333333333333  test acc: 0.9506
```