```
# coding: utf-8
import sys, os
sys.path.append(os.pardir)
import numpy as np
from common.optimizer import *

class Trainer:

    def __init__(self, network, x_train, t_train, x_test, t_test,
                 epochs=20, mini_batch_size=100,
                 optimizer='SGD', optimizer_param={'lr':0.01},
                 evaluate_sample_num_per_epoch=None, verbose=True):
        self.network = network
        self.verbose = verbose
        self.x_train = x_train
        self.t_train = t_train
        self.x_test = x_test
        self.t_test = t_test
        self.epochs = epochs
        self.batch_size = mini_batch_size
        self.evaluate_sample_num_per_epoch = evaluate_sample_num_per_epoch#会传入 1000

        # optimizer
        optimizer_class_dict = {'sgd':SGD, 'momentum':Momentum, 'nesterov':Nesterov,
                                'adagrad':AdaGrad, 'rmsprop':RMSprop, 'adam':Adam}
        self.optimizer = optimizer_class_dict[optimizer.lower()](**optimizer_param)

        self.train_size = x_train.shape[0]
        self.iter_per_epoch = max(self.train_size / mini_batch_size, 1)
        self.max_iter = int(epochs * self.iter_per_epoch)
        self.current_iter = 0
        self.current_epoch = 0

        self.train_loss_list = []
        self.train_acc_list = []
        self.test_acc_list = []

    def train_step(self):
        batch_mask = np.random.choice(self.train_size, self.batch_size)
        x_batch = self.x_train[batch_mask]
        t_batch = self.t_train[batch_mask]

        grads = self.network.gradient(x_batch, t_batch)
        self.optimizer.update(self.network.params, grads)#params 中保存着权重
```

```
            loss = self.network.loss(x_batch, t_batch)#CNN 的输出值
            self.train_loss_list.append(loss)
            if self.verbose: print("train loss:" + str(loss))

            # 计算每个 epoch 的识别精度
            if self.current_iter % self.iter_per_epoch == 0:
                self.current_epoch += 1

                x_train_sample, t_train_sample = self.x_train, self.t_train
                x_test_sample, t_test_sample = self.x_test, self.t_test
                if not self.evaluate_sample_num_per_epoch is None:
                    t = self.evaluate_sample_num_per_epoch#
                    x_train_sample, t_train_sample = self.x_train[:t], self.t_train[:t]
                    #取出 1000 个数据，该数据后又会在 accuracy 函数中被切出 100 个数据
                    x_test_sample, t_test_sample = self.x_test[:t], self.t_test[:t]

                #评估训练样例识别精度
                train_acc = self.network.accuracy(x_train_sample, t_train_sample)
                #评估测试样例识别精度
                test_acc = self.network.accuracy(x_test_sample, t_test_sample)
                self.train_acc_list.append(train_acc)#将精确值加入到列表中
                self.test_acc_list.append(test_acc)

                if self.verbose: print("=== epoch:" + str(self.current_epoch) + ", train acc:" + str(train_acc)
+ ", test acc:" + str(test_acc) + " ===")
            self.current_iter += 1

        def train(self):
            for i in range(self.max_iter):
                self.train_step()

            test_acc = self.network.accuracy(self.x_test, self.t_test)

            if self.verbose:
                print("=============== Final Test Accuracy ===============")
                print("test acc:" + str(test_acc))
```