一个用pybrain构建网络识别的单个数字图像的代码：

#-\*- coding:utf-8 -\*-

from pybrain.tools.shortcuts import buildNetwork

from pybrain.datasets import SupervisedDataSet

from pybrain.supervised.trainers import BackpropTrainer

from pybrain.structure import TanhLayer

import numpy as np

import os

from PIL import Image

import pickle as pk

import datetime

def get\_train\_samples(input\_num,output\_num):

'''

从new\_samples文件夹中读图，根据输入数和输出数制作样本，每一原始样本加入随机噪音生成100个样本

'''

print ('getsample start.')

sam\_path='./new\_samples'

samples = SupervisedDataSet(input\_num,output\_num)

nlist = os.listdir(sam\_path)

t=int(np.sqrt(input\_num))

for n in nlist:

file = os.path.join(sam\_path,n)

im = Image.open(file)

im = im.convert('L')

im = im.resize((t,t),Image.BILINEAR)

buf = np.array(im).reshape(input\_num,1)

buf = buf<200

buf = tuple(buf)

buf1=int(n.split('.')[0])

buf2=range(output\_num)

for i in range(len(buf2)):

buf2[i] = 0

buf2[buf1]=1

buf2 = tuple(buf2)

samples.addSample(buf,buf2)

for i in range(100):

buf3 = list(buf)

for j in range(len(buf)/20):

buf3[np.random.randint(len(buf))] = bool(np.random.randint(2))

samples.addSample(tuple(buf3),buf2)

return samples

def get\_test\_samples(input\_num):

'''

从new\_test文件夹读取测试数据

'''

print ('Get test samples start.')

test\_path='./new\_test'

samples = SupervisedDataSet(input\_num,1)

nlist = os.listdir(test\_path)

t=int(np.sqrt(input\_num))

for n in nlist:

file = os.path.join(test\_path,n)

im = Image.open(file)

im = im.convert('L')

im = im.resize((t,t),Image.BILINEAR)

buf = np.array(im).reshape(input\_num,1)

buf = buf<200

buf = tuple(buf)

samples.addSample(buf,1)

return samples

class net:

'''

网络的定义

'''

def \_\_init\_\_(self,input\_num,hide\_node\_num,output\_num):

'''

根据参数初始化网络

'''

self.input\_num = input\_num

self.hide\_node\_num = hide\_node\_num

self.output\_num = output\_num

self.network = buildNetwork(input\_num,hide\_node\_num,output\_num,bias=True) #响应函数和层数可在此调

def train(self,samples,epsilon):

'''

训练函数

'''

print ('Train start.')

trainer = BackpropTrainer(self.network,samples) #学习率可在此调

e = 100

n=0

while e>epsilon:

e=trainer.train()

n+=1

print( n,' done,e=',e)

if not n%10:

self.save()

if n>=100:break

self.save()

print( 'Train end.')

return e

def run(self,samples):

'''

测试

'''

print ('Test start.')

result = []

for sample in samples['input']:

buf = self.network.activate(sample)

buf= list(buf)

result.append(buf.index(max(buf)))

print ('Result ',result)

result\_path = './results/'

filename = str(self.input\_num)+'-'+str(self.hide\_node\_num)+'-'+str(self.output\_num)+'new.txt'

with open(result\_path+filename,'w') as f:

result = str(result)

f.write(result)

def save(self):

'''

保存训练好的网络

'''

print ('saving')

save\_path = './save/'

filename = str(self.input\_num)+'-'+str(self.hide\_node\_num)+'-'+str(self.output\_num)+'new.cPickle'

with open(save\_path+filename,'wb') as f:

pk.dump(self.network,f)

print ('done')

def load(self):

'''

从存档中加载训练好的网络

'''

print ('loading')

save\_path ='./save/'

filename = str(self.input\_num)+'-'+str(self.hide\_node\_num)+'-'+str(self.output\_num)+'new.cPickle'

if filename in os.listdir('./save/'):

with open(save\_path+filename,'rb') as f:

self.network = pk.load(f)

print( 'done')

def main():

'''

主函数，定义程序运行过程

'''

start=datetime.datetime.now()

output\_num = 10

epsilon = 0.01

input\_num=20\*20

hide\_node\_num = 200

filename = str(input\_num)+'-'+str(hide\_node\_num)+'-'+str(output\_num)+'new.cPickle'

net1 = net(input\_num,hide\_node\_num,output\_num)

if filename in os.listdir('./save/'):

net1.load()

else:

samples = get\_train\_samples(input\_num,output\_num)

net1.train(samples,epsilon)

net1.save()

net1.run(get\_test\_samples(input\_num))

end =datetime.datetime.now()

print ('Time ',end-start)

main()