サンプルプログラムの解説

この文書では、このフォルダに存在するサンプルプログラムが何をしているのかを説明していきます。

load-image.py

猫画像を読み込むサンプルプログラムです。画像といってもただの配列変数であることを実感していいただければと思います。

get-sdo-image.py

上記の応用で、京大の宇宙天気予報チームが保有しているサーバーから任意の時刻の太陽画像を取得してくるサンプルプログラムです。

convolution.py

Convolution という、配列変数の隣接する要素を演算する操作のサンプルプログラムです。

convolution-2.py

Convolution を応用すれば、図形の辺や頂点が検出できることを示すサンプルプログラムです。

cat-or-dog.py

これまでにもご紹介した、犬と猫の画像を識別するプログラムです。

目 次

1 load-image.py 2

2 get-sdo-image.py 3

3 convolution.py 4

4 convolution-2.py 5

5 cat-or-dog.py 6

1 load-image.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# 配列変数を扱うのに必要なライブラリ
import numpy as np
# 画像の入出力につかうライブラリたち
from PIL import Image
import matplotlib
matplotlib.use('Agg')
import pylab

# "PetImages/Cat/0.jpg" というファイルを開いて画像を RGB 形式で読み込みます
img=Image.open("PetImages/Cat/0.jpg").convert('RGB')

# 読み込まれた画像を numpy の配列変数形式に変換します
# 3次元配列の3つの添字は、順に色 (channel) 、y座標、x座標を表します
img =np.asarray(img).astype(np.float32).transpose(2, 0, 1)

# 画像データの中身や、配列の寸法を表示します
print img
print img.shape
print img[0].shape

# ファイル名とnumpy形式の画像を受け取って、画像ファイルを書き出す関数を定義します
def write_image(fn, img):
    pylab.rcParams['figure.figsize'] = (6.4,6.4)
    pylab.clf()
    pylab.imshow((img.transpose(1,2,0))/255.0)
    pylab.savefig(fn)


# まずは読み込んだとおりの猫画像を'test-normal-cat.jpg'書き出してみます
write_image('test-normal-cat.jpg', img)

# 元々の画像の赤色を2倍にした画像を作ります
red_img = 1.0 * img
red_img[0, :, :] =  np.minimum(255, 2.0 * img[0] )
write_image('test-red-cat.jpg', red_img)

# 元々の画像の緑色を2倍にした画像を作ります
green_img = 1.0 * img
green_img[1, :, :] =  np.minimum(255, 2.0 * img[1] )
write_image('test-green-cat.jpg', green_img)

# 左上が暗い画像をつくります
halfdark_img = 1.0 * img
n_color, n_y, n_x = img.shape
for c in range(n_color):
    for y in range(n_y):
        for x in range(n_x):
            if x < 300 and y < 200:
                halfdark_img[c, y, x] = halfdark_img[c, y, x] /2
write_image('test-halfdark-cat.jpg', halfdark_img)

# 猫のプライバシーに配慮した画像を作ります
privacy_img = 1.0 * img
privacy_img[:, 150:170, 150:220] = 0
write_image('test-privacy-cat.jpg', privacy_img)
```

2 get-sdo-image.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# 必要なライブラリのimport文です
import datetime, StringIO, urllib
import numpy as np
import matplotlib
matplotlib.use('Agg')
import pylab

# 時刻tにおける、波長wavelengthの太陽画像を取得します
# SDO衛星が撮影した元データは http://sdo.gsfc.nasa.gov/data/ にあります。
#
# 村主へのメモ
# http://jsoc2.stanford.edu/data/aia/synoptic/を使った方がいいかも？
def get_image(wavelength,t):
    url = 'http://sdo.s3-website-us-west-2.amazonaws.com/aia{}/720s-x1024
        /{:04}/{:02}/{:02}/{:02}{:02}.npz'.format(wavelength, t.year, t.month
        , t.day, t.hour, t.minute)
    resp = urllib.urlopen(url)
    strio = StringIO.StringIO(resp.read())
    return np.load(strio)

# 2011年1月1日、00:00に相当するdatetime型の値を作ります。
t = datetime.datetime(2011,1,1,0,0)
print t

# 画像データを取得します
img = get_image(171, t)['img']

# とれた画像の中身、型、寸法を表示します
print img
print type(img)
print img.shape

# 画像データを'test-sun.png'というファイル名で出力してみます
pylab.rcParams['figure.figsize'] = (6.4,6.4)
pylab.clf()
pylab.imshow(img)
pylab.savefig('test-sun.png')
```

3 convolution.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# chainerのconvolution関数を使ってみます
import numpy as np
import chainer
from chainer import computational_graph
from chainer import cuda
import chainer.functions as F
import chainer.links as L
from chainer import optimizers
from chainer import serializers

# (x,y)の位置にx+10*yが格納された配列imgを作ります
img = np.zeros((1,1,10,10), dtype=np.float32)
for y in range(10):
    for x in range(10):
        img[0,0,y,x] = x+10*y

# convolutionの係数は4次元配列で定義されます
# 4つの配列添字は、それぞれ
# 「出力画像の色(channel)」「入力画像の色(channel)」「y座標」「x座標」に対応
    しています
w = np.zeros((1,1,3,3), dtype=np.float32)
w[0,0,0,1] = 1
w[0,0,1,0] = 100

# 上記のwを使ってconvolutionを行います
v = chainer.Variable(img)
f = L.Convolution2D(1,1,3,stride=1, initialW = w)

# convolutionを行う前と、後の配列の内容を出力してみます
print "Original data"
print v.data
print "The data after convolution"
print f(v).data
```

4 convolution-2.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# convolutionを使って、図形の辺や頂点などの特徴を検出できることを示します
import numpy as np
import chainer
from chainer import computational_graph
from chainer import cuda
import chainer.functions as F
import chainer.links as L
from chainer import optimizers
from chainer import serializers

img = np.zeros((1,1,10,10), dtype=np.float32)
for y in range(2,8):
    for x in range(2,8):
        img[0,0,y,x] = 99


# 1色(1 channel)の画像から3色(3 channel)の画像を出力するような
# あるconvolutionを作ってみます
w = np.zeros((3,1,2,2), dtype=np.float32)

# 0番目のチャンネルではy方向の差をとります
w[0,0,0,0] =  1
w[0,0,1,0] = -1

# 1番目のチャンネルではx方向の差をとります
w[1,0,0,0] =  1
w[1,0,0,1] = -1

# 2番目のチャンネルでは格子状の4点の差をとります
w[2,0,0,0] =  1
w[2,0,1,0] = -1
w[2,0,0,1] = -1
w[2,0,1,1] =  1


v = chainer.Variable(img)
f = L.Convolution2D(1,3,2,stride=1, initialW = w)

# convolution前と後の画像を出力してみます
print "Input:"
print v.data

print "Output:"
print f(v).data
```

5 cat-or-dog.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import glob,random, sys, subprocess, os.path

import matplotlib
matplotlib.use('Agg')
import pylab

import numpy as np
from PIL import Image

import chainer
from chainer import computational_graph
from chainer import cuda
import chainer.functions as F
import chainer.links as L
from chainer import optimizers
from chainer import serializers

# batchsizeは、一度に学習する画像の枚数です
batchsize = 10
# 元画像から、このサイズの領域を取り出して学習します
imgsize = 64

model_filename = 'model.save'
state_filename = 'state.save'


def read_command(cmd):
    stdout, stderr = subprocess.Popen(cmd, shell=True, stdout=subprocess.PIPE
        , stderr=subprocess.PIPE).communicate()
    return stdout

catfns = glob.glob('PetImages/Cat/*')
dogfns = glob.glob('PetImages/Dog/*')


# 猫と犬の画像を分類するニューラルネットワークです。
class CatDog(chainer.Chain):
    def __init__(self):
        super(CatDog, self).__init__(
            l1=L.Convolution2D(3,64,3,stride=2),
            l2=L.Convolution2D(64,128,3,stride=2),
            l3=L.Convolution2D(128,256,3,stride=2),
            l4=L.Linear(256*7*7,2)
        )

    def __call__(self, x):
        h1 = F.relu(self.l1(x))
        h2 = F.relu(self.l2(h1))
        h3 = F.relu(self.l3(h2))
        return F.reshape(self.l4(h3), (x.data.shape[0],2))


# fns が None なら、教師データのなかからランダムに画像を読み込みます。
# fns が None でなければ、指定されたファイル名の画像を読み込みます。
def load_image(fns = None):
    if fns is not None:
        my_batchsize = len(fns)
```

```
    else:
        my_batchsize = batchsize

    ret = np.zeros((my_batchsize, 3, imgsize, imgsize), dtype=np.float32)
    ret_ans = np.zeros((my_batchsize), dtype=np.int32)


    for j in range(my_batchsize):
        # 正解ラベルをランダムに選ぶ
        ans = np.random.randint(2)
        img = None
        if fns is not None:
            # fnsで指定されたファイル名の画像を読み込む
            img=Image.open(fns[j]).convert('RGB')
            ans=-1
        else:
            while True:
                try:
                    # 正解ラベルに従って猫、犬のいずれかの画像を読み込む
                    if ans==0:
                        fn = random.choice(catfns)
                    else:
                        fn = random.choice(dogfns)

                    with open(fn,'r') as fp:
                        img=Image.open(fp).convert('RGB')
                    break
                except:
                    continue

        # ランダムな回転と拡大縮小を加える
        img=img.rotate(np.random.random()*20.0-10.0, Image.BICUBIC)
        w,h=img.size
        scale = 80.0/min(w,h)*(1.0+0.2*np.random.random())
        img=img.resize((int(w*scale),int(h*scale)),Image.BICUBIC)
        img = np.asarray(img).astype(np.float32).transpose(2, 0, 1)

        # 画像の中央付近をランダムに切り出す
        oy = (img.shape[1]-imgsize)/2
        ox = (img.shape[2]-imgsize)/2
        oy=oy/2+np.random.randint(oy)
        ox=ox/2+np.random.randint(ox)

        # 1/2の確率で、左右を反転させる
        if np.random.randint(2)==0:
            img[:,:,:] = img[:,:,::-1]

        ret[j,:,:,:] = (img[:,oy:oy+imgsize,ox:ox+imgsize]-128.0)/128.0
        ret_ans[j] = ans
    return (chainer.Variable(ret), chainer.Variable(ret_ans))

# 与えられた画像をファイルに書き出す関数です
def write_image(fn, img, ans):
    pylab.rcParams['figure.figsize'] = (6.4,6.4)
    pylab.clf()
    pylab.imshow((img.transpose(1,2,0) + 1)/2)
    if ans == 0:
        pylab.title("cat")
    else:
        pylab.title("dog")
    pylab.savefig(fn)
```

```
################################################################
# メインプログラム開始
################################################################


# ニューラルネットワークによるモデルと、モデルの最適化機を作ります
model = L.Classifier(CatDog())
optimizer = optimizers.Adam()
optimizer.setup(model)

# セーブファイルが存在する場合は、セーブファイルから状態を読み込みます
if os.path.exists(model_filename) and os.path.exists(state_filename):
    print 'Load␣model␣from', model_filename, state_filename
    serializers.load_npz(model_filename, model)
    serializers.load_npz(state_filename, optimizer)

# 現在までの学習状態をファイルに書き出す関数です。
def save():
    print('save␣the␣model')
    serializers.save_npz(model_filename, model)
    print('save␣the␣optimizer')
    serializers.save_npz(state_filename, optimizer)

# 機械学習を行う関数です
def learn():
    e = 0
    while True:
        e+=1
        print e,
        imgs, anss = load_image()
        optimizer.update(model, imgs, anss)
        print model.loss.data

        write_image('test.png', imgs.data[0], anss.data[0])

        if e & (e-1) == 0 or e%1000 == 0:
            save()
    save()

# コマンドライン入力から与えられたファイル名の画像を、猫犬判定する関数です。
def test():
    fns = sys.argv[1:]
    for fn in fns:
        imgs, anss = load_image(11*[fn])
        model(imgs,anss)
        count_cat = 0
        count_dog = 0
        for c,d in model.y.data:
            if c>d:
                count_cat += 1
            else:
                count_dog += 1
        if count_cat > count_dog:
            result = "cat"
        else:
            result = "dog"

        print fn, "␣is␣a␣", result

# コマンドライン引数からファイル名が与えられていれば、
# test()を呼びます。
```

```
# ファイル名が与えられていなければ学習を行います
if len(sys.argv) > 1:
    test()
else:
    learn()
```