Numpy

numpy: basic array manipulation

Description

Welcome! This is Deep Learning, Machine Learning, and Data Science Prerequisites: The Numpy Stack in Python.

One question or concern I get a lot is that people want to learn deep learning and data science, so they take these courses, but they get left behind because they don’t know enough about the Numpy stack in order to turn those concepts into code.

Even if I write the code in full, if you don’t know Numpy, then it’s still very hard to read.

This course is designed to remove that obstacle - to show you how to do things in the Numpy stack that are frequently needed in deep learning and data science.

So what are those things?

Numpy. This forms the basis for everything else. The central object in Numpy is the Numpy array, on which you can do various operations.

The key is that a Numpy array isn’t just a regular array you’d see in a language like Java or C++, but instead is like a mathematical object like a vector or a matrix.

That means you can do vector and matrix operations like addition, subtraction, and multiplication.

The most important aspect of Numpy arrays is that they are optimized for speed. So we’re going to do a demo where I prove to you that using a Numpy vectorized operation is faster than using a Python list.

Then we’ll look at some more complicated matrix operations, like products, inverses, determinants, and solving linear systems.

Pandas. Pandas is great because it does a lot of things under the hood, which makes your life easier because you then don’t need to code those things manually.

Pandas makes working with datasets a lot like R, if you’re familiar with R.

The central object in R and Pandas is the DataFrame.

We’ll look at how much easier it is to load a dataset using Pandas vs. trying to do it manually.

Then we’ll look at some dataframe operations, like filtering by column, filtering by row, the apply function, and joins, which look a lot like SQL joins.

So if you have an SQL background and you like working with tables then Pandas will be a great next thing to learn about.

Since Pandas teaches us how to load data, the next step will be looking at the data. For that we will use Matplotlib.

In this section we’ll go over some common plots, namely the line chart, scatter plot, and histogram.

We’ll also look at how to show images using Matplotlib.

99% of the time, you’ll be using some form of the above plots.

Scipy.

I like to think of Scipy as an addon library to Numpy.

Whereas Numpy provides basic building blocks, like vectors, matrices, and operations on them, Scipy uses those general building blocks to do specific things.

For example, Scipy can do many common statistics calculations, including getting the PDF value, the CDF value, sampling from a distribution, and statistical testing.

It has signal processing tools so it can do things like convolution and the Fourier transform.

In sum:

If you’ve taken a deep learning or machine learning course, and you understand the theory, and you can see the code, but you can’t make the connection between how to turn those algorithms into actual running code, this course is for you.

All the code for this course can be downloaded from my github: /lazyprogrammer/machine\_learning\_examples

In the directory: numpy\_class

Make sure you always "git pull" so you have the latest version!

HARD PREREQUISITES / KNOWLEDGE YOU ARE ASSUMED TO HAVE:

linear algebra

probability

Python coding: if/else, loops, lists, dicts, sets

you should already know "why" things like a dot product, matrix inversion, and Gaussian probability distributions are useful and what they can be used for

scipy: scipy.ndimage submodule dedicated to image processing (n-dimensional images). See the documentation:

the submodule scipy.ndimage provides functions operating on n-dimensional NumPy arrays.

>>> from scipy import ndimage

Common tasks in image processing:

Input/Output, displaying images

Basic manipulations: cropping, flipping, rotating, ...

Image filtering: denoising, sharpening

Image segmentation: labeling pixels corresponding to different objects

Classification

Feature extraction

Registration

...

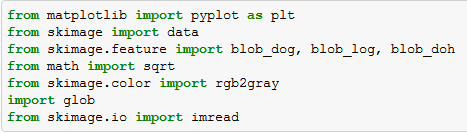
I've never done any image processing with Python (a shame, I know), but I've done a bit with Processing (an odd little Java-y framework) and Javascript. Processing is a bit clunky, but it comes with tons of example code that should give you ideas on how to implement something in Python. Image processing in the browser was surprisingly easy to do, now that we have the canvas element in HTML 5.

Your first step should be to learn how to calculate a 2D histogram of an image, which contains information on the distribution of colour intensities. Armed this information, you can start to do things like thresholding, which tends to make "interesting" areas of an image more prominent.

From there, you'll want to play around with applying different kernels to an image so you get an understanding of how the various image effects (e.g. brighten, sharpen, blur) work. One of the more important ones for image analysis is edge detection, which is one way you might find features. For your example task, you'll probably want to look into blob detection. <https://en.wikipedia.org/wiki/Kernel_(image_processing)>

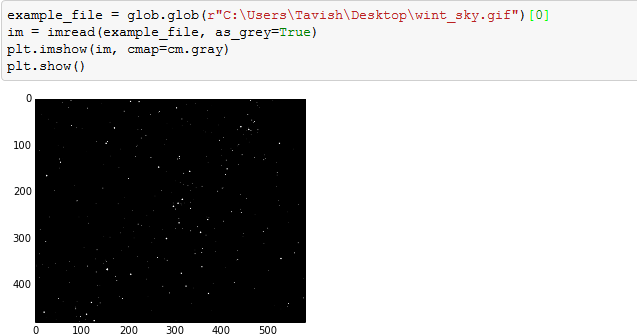
Step 1: Import the required library

Skimage package enables us to do image processing using Python. The language is extremely simple to understand but does some of the most complicated tasks. Here are a few library you need to import to get started,



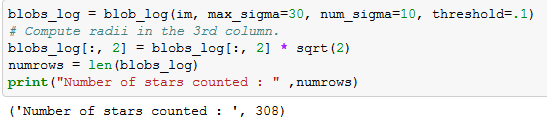
Step 2 : Import the image

Once we have all the libraries in place, we need to import our image file to python. Following is the code you can use to import the image file. Note that the image is imported in grey scale, which basically means that each pixel is a shade of grey. And each pixel essentially becomes one cell in a matrix. In this case the image is matrix of 480\*581 cells (or image of 480\*581 pixels).



Step 3 : Find the number of Stars

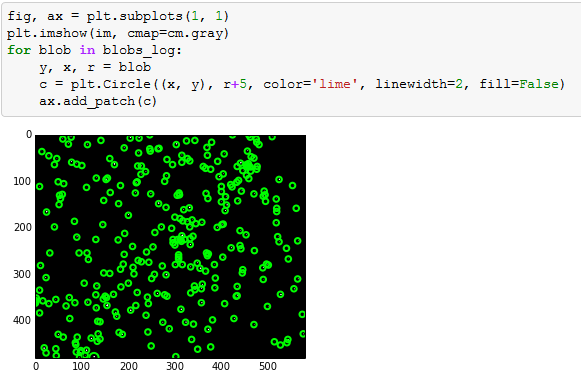
Now comes the critical part where our major labor is done by a few commands. These few command go out for searching continuous objects in the picture. Blobs\_log gives three outputs for each object found. First two are the coordinates and the third one is the area of the object. The radius of each blob/object can be estimated using this column (area of the object).



As we can see that the algorithm has estimated 308 visible stars. Let’s now look at how accurate are these readings.

Step 4 : Validated whether we captured all the stars

The number 308 is still coming out of a black box. Let’s look at whether we have spotted all the stars correctly. For this I am circling each estimated star position. And the look at the image if we are missing any star.



Here is the complete code :

from matplotlib import pyplot as plt

from skimage import data

from skimage.feature import blob\_dog, blob\_log, blob\_doh

from math import sqrt

from skimage.color import rgb2gray

import glob

from skimage.io import imread

example\_file = glob.glob(r"C:UsersTavishDesktopwint\_sky.gif")[0]

im = imread(example\_file, as\_grey=True)

plt.imshow(im, cmap=cm.gray)

plt.show()

blobs\_log = blob\_log(im, max\_sigma=30, num\_sigma=10, threshold=.1)

# Compute radii in the 3rd column.

blobs\_log[:, 2] = blobs\_log[:, 2] \* sqrt(2)

numrows = len(blobs\_log)

print("Number of stars counted : " ,numrows)

fig, ax = plt.subplots(1, 1)

plt.imshow(im, cmap=cm.gray)

for blob in blobs\_log:

y, x, r = blob

c = plt.Circle((x, y), r+5, color='lime', linewidth=2, fill=False)

ax.add\_patch(c)

End Notes

Image Processing is fascinating! I started my journey with Python Image Processing not more than 5 days. For the benefit of the community, I will encourage any suggestions or best practices to be shared on this forum. This exercise might not have any practical application but similar analysis can be done for purity estimations. For instance, in glass industry we need the number of silica particles in the glass at a microscopic level. By capturing the frames in a video, you can use this simple code to do a lot of things. For example, estimation of traffic through CCTV footage. This code can easily be tailored to achieve the same.