SWHarden.com

Scott W Harden: neuroscientist, dentist, molecular biologist, code monkey

Home About Scott Publications Projects Contact

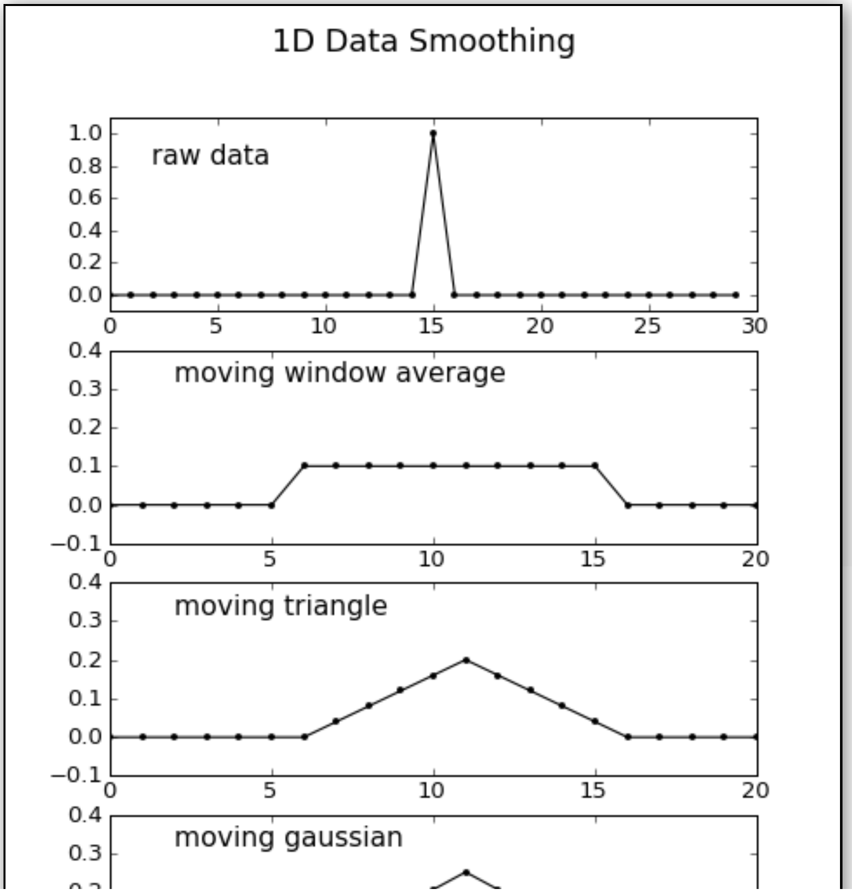
Linear Data Smoothing in Python

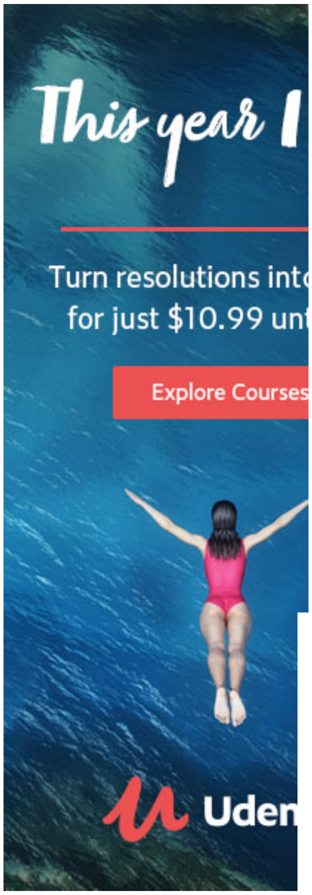
November 17, 2008 by Scott | General, Python | 1 Comment

Here's a scrumptious morsel of juicy python code for even the most stoic of scientists to get excited about. Granted, it's a very simple concept and has surely been done countless times before, but there aren't any good resources for this code on the internet. Since I had to write my own code to perform a variety of different linear 1-dimensional array data smoothing in python, I decided it would be nice to share it. At the bottom of this post you can see a PNG image which is the file output by the code listen even further below. If you copy/paste the code into an empty text file and run it in Python, it will generate the exact same PNG file (assuming you have pylab and numpy libraries configured).

```
### This is the Gaussian data smoothing function I wrote ###
def smoothListGaussian(list,degree=5):
    window=degree*2-1
    weight=numpy.array([1.0]*window)
    weightGauss=[]
    for i in range(window):
        i=i-degree+1
        frac=i/float(window)
        gauss=1/(numpy.exp((4*(frac))**2))
        weightGauss.append(gauss)
    weight=numpy.array(weightGauss)*weight
    smoothed=[0.0]*(len(list)-window)
    for i in range(len(smoothed)):
        smoothed[i]=sum(numpy.array(list[i:i+window])*weight)/sum(weight)
    return smoothed
```

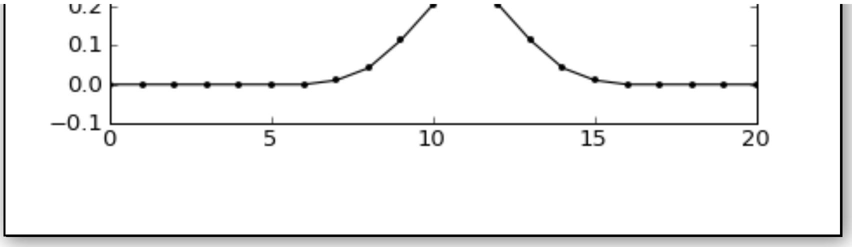
Basically, you feed it a list (it doesn't matter how long it is) and it will return a smoother version of the data. The Gaussian smoothing function I wrote is leagues better than a moving window average method, for reasons that are obvious when viewing the chart below. Surprisingly, the moving triangle method appears to be very similar to the Gaussian function at low degrees of spread. However, for huge numbers of data points, the Gaussian function should perform better.





Categories

C/C++ Circuitry DIY ECG Electronics General GitHub HAB (high altitude balloon) Linux Microcontroller Molecular Biology My Website PHP Prime Numbers Programming Python QRSS / MEPT



```
### This is the code to produce the image displayed above ###

import pylab,numpy


def smoothList(list,strippedXs=False,degree=10):

    if strippedXs==True: return Xs[0:-(len(list)-(len(list)-degree+1))]

    smoothed=[0]*(len(list)-degree+1)

    for i in range(len(smoothed)):

        smoothed[i]=sum(list[i:i+degree])/float(degree)

    return smoothed


def smoothListTriangle(list,strippedXs=False,degree=5):

    weight=[]

    window=degree*2-1

    smoothed=[0.0]*(len(list)-window)

    for x in range(1,2*degree):weight.append(degree-abs(degree-x))

    w=numpy.array(weight)

    for i in range(len(smoothed)):

        smoothed[i]=sum(numpy.array(list[i:i+window])*w)/float(sum(w))

    return smoothed


def smoothListGaussian(list,strippedXs=False,degree=5):

    window=degree*2-1

    weight=numpy.array([1.0]*window)
```

(manned experimental propagation transmistter)
RF (Radio Frequency)
Thermoregulation UCF Lab
Uncategorized

Categories

- C/C++ (46)
- Circuitry (103)
- DIY ECG (16)
- Electronics (15)
- General (196)
- GitHub (16)
- HAB (high altitude balloon) (4)
- Linux (15)
- Microcontrollers (77)
- Molecular Biology (3)
- My Website (10)
- PHP (4)
- Prime Numbers (16)
- Programming (4)
- Python (69)
- QRSS / MEPT (manned experimental propagation transmistter) (4)
- RF (Radio Frequency) (84)
- Thermoregulation (3)
- UCF Lab (4)
- Uncategorized (2)

Archives

```
    weightGauss=[]
    for i in range(window):
        i=i-degree+1
        frac=i/float(window)
        gauss=1/(numpy.exp((4*(frac))**2))
        weightGauss.append(gauss)
    weight=numpy.array(weightGauss)*weight
    smoothed=[0.0]*(len(list)-window)
    for i in range(len(smoothed)):
        smoothed[i]=sum(numpy.array(list[i:i+window])*weight)/sum(weight)
    return smoothed

### DUMMY DATA ###
data = [0]*30 #30 "0"s in a row
data[15]=1    #the middle one is "1"

### PLOT DIFFERENT SMOOTHING FUNCTIONS ###

pylab.figure(figsize=(550/80,700/80))
pylab.suptitle('1D Data Smoothing', fontsize=16)

pylab.subplot(4,1,1)
p1=pylab.plot(data,".k")
p1=pylab.plot(data,"-k")
a=pylab.axis()
pylab.axis([a[0],a[1],-.1,1.1])
pylab.text(2,.8,"raw data",fontsize=14)

pylab.subplot(4,1,2)
```

December 2017 (1)

September 2017 (2)

August 2017 (3)

June 2017 (1)

April 2017 (1)

February 2017 (2)

October 2016 (1)

September 2016 (4)

August 2016 (4)

July 2016 (6)

December 2015 (1)

April 2014 (1)

February 2014 (1)

June 2013 (5)

May 2013 (4)

April 2013 (3)

January 2013 (1)

December 2012 (2)

August 2012 (1)

June 2012 (3)

August 2011 (5)

July 2011 (7)

June 2011 (3)

March 2011 (4)

February 2011 (6)

January 2011 (4)

December 2010 (1)

November 2010 (5)

September 2010 (5)

August 2010 (6)

July 2010 (3)

June 2010 (19)

May 2010 (14)

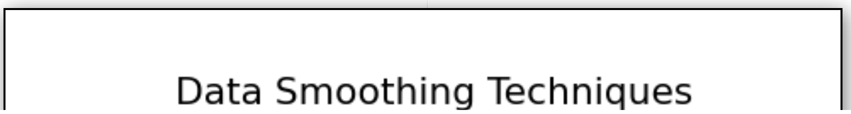
April 2010 (2)

March 2010 (13)

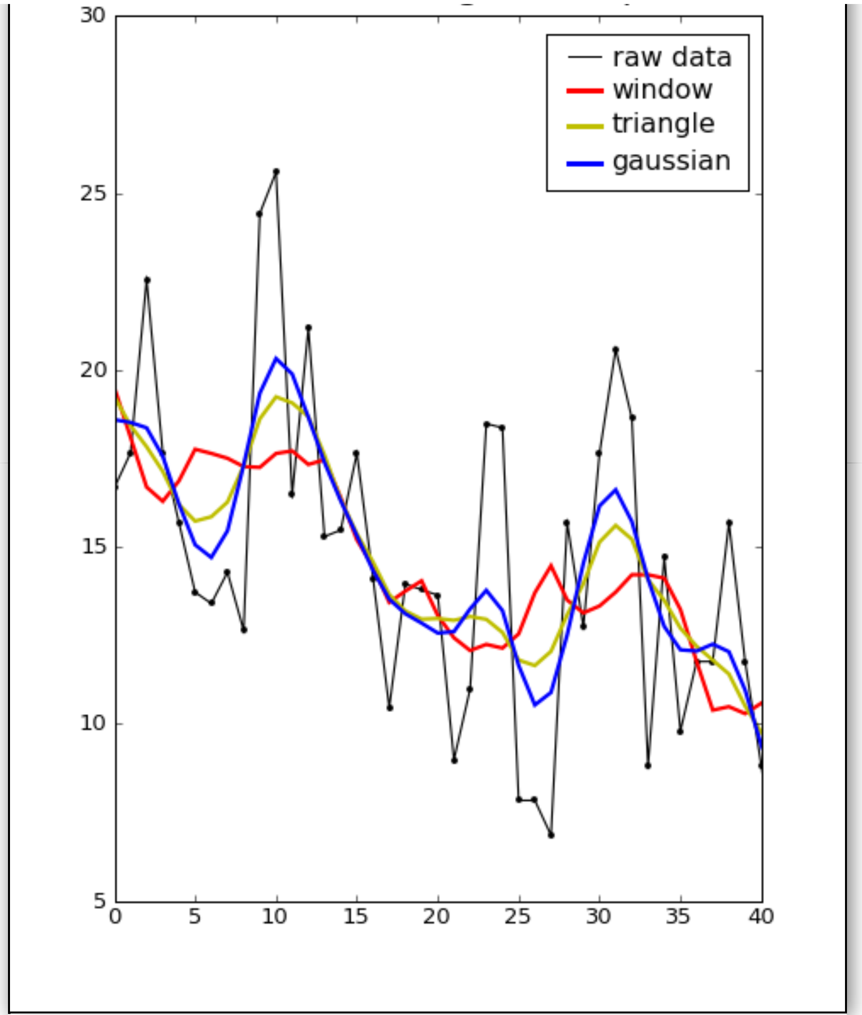
February 2010 (2)

```
p1=pylab.plot(smoothList(data),".k")
p1=pylab.plot(smoothList(data),"-k")
a=pylab.axis()
pylab.axis([a[0],a[1],-.1,.4])
pylab.text(2,.3,"moving window average",fontsize=14)

pylab.subplot(4,1,3)
p1=pylab.plot(smoothListTriangle(data),".k")
p1=pylab.plot(smoothListTriangle(data),"-k")
pylab.axis([a[0],a[1],-.1,.4])
pylab.text(2,.3,"moving triangle",fontsize=14)

pylab.subplot(4,1,4)
p1=pylab.plot(smoothListGaussian(data),".k")
p1=pylab.plot(smoothListGaussian(data),"-k")
pylab.axis([a[0],a[1],-.1,.4])
pylab.text(2,.3,"moving gaussian",fontsize=14)

#pylab.show()
pylab.savefig("smooth.png",dpi=80)
```

Hey, I had a great idea, why don't I test it on some of my own data? Due to the fact that I don't want the details of my thesis work getting out onto the internet too early, I can't reveal exactly what this data is from. It will suffice to say that it's fractional density of neurite coverage in thick muscle tissue. Anyhow, this data is wild and in desperate need of some smoothing. Below is a visual representation of the differences in the methods of smoothing. Yayness! I like the gaussian function the best.

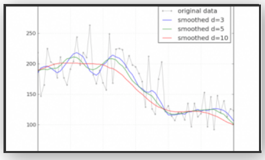


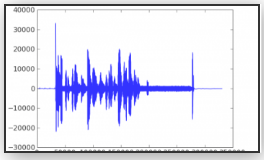
January 2010 (2)
December 2009 (2)
September 2009 (1)
August 2009 (7)
July 2009 (4)
June 2009 (18)
May 2009 (11)
April 2009 (10)
March 2009 (2)
February 2009 (4)
January 2009 (18)
November 2008 (5)

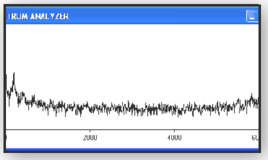


I should note that the degree of window coverage for the moving window average, moving triangle, and gaussian functions are 10, 5, and 5 respectively. Also note that (due to the handling of the "degree" variable between the different functions) the actual number of data points assessed in these three functions are 10, 9, and 9 respectively. The degree for the last two functions represents "spread" from each point, whereas the first one represents the total number of points to be averaged for the moving average. Enjoy.

Share this:







Smoothing Window Data Averaging in Python - Moving Triangle Tecnique
June 20, 2010
In "General"

Reading PCM Audio with Python
June 19, 2009
In "Python"

Realtime FFT Graph of Audio WAV File or Microphone Input with Python, Scipy, and WCKgraph
March 5, 2010
In "General"

This year I will
Turn resolutions into action
with courses up to 90% off until
Explore Courses
Udemy

General, Python | 1 Comment

‹ Free Damask Seamless Tiling Backgrounds

› Run Ubuntu Live CD From a USB Drive

One thought on “Linear Data Smoothing in Python”

Ramon Crehuet
February 6, 2015 at 3:58 am | Log in to Reply

This is an interesting post, but there are some practices I wouldn’t recommend:
1) Using ‘list’ as a name of a variable.
2) Working with lists and transforming them to arrays at the end. I would work with arrays all the time. It’s more elegant and more efficient.
3) Using weight and weightGauss to create essentially the same thing. Once weight is created, you can fill it with:
weight[i]=gauss

Numpy pad, ones and zeros can help you create the arrays you need. For example:
np.array([1.0]*window) is the same as np.ones(window)
Finally, you can turn your code more efficient removing for loops with numpy constructs. My final version of your code is the following:

```
def smoothListGaussian2(myarray, degree=5):
myarray = np.pad(myarray, (degree-1,degree-1), mode=’edge’)
window=degree*2-1
weight=np.arange(-degree+1, degree)/window
weight = np.exp(-(16*weight**2))
weight /= sum(weight)
smoothed = np.convolve(myarray, weight, mode=’valid’)
return smoothed
```

You must log in to post a comment.

About Scott

Meta

Log in

Scott Harden lives in Gainesville, Florida and works at the University of Florida as a biological research scientist studying cellular neurophysiology. Scott has lifelong passion for computer programming and electrical engineering, and in his spare time enjoys building small electrical devices and writing cross-platform open-source software. more →

Entries RSS

Comments RSS

WordPress.org

Contact

SWHarden@gmail.com

Subscribe via Email

Enter your email address to subscribe to this website and receive notifications of new posts by email.

Email Address

Subscribe