Tutorial Three: Loops and Conditionals

Imad Pasha
Chris Agostino

February 18, 2015

1 Introduction

In lecture Monday we learned that combinations of conditionals and loops could make our code much more powerful. In last week's tutorial, you loaded up a spectrum, performed some basic array manipulation on it, and viewed it. Today we are going to continue with spectral analysis, this time performing a step known as "centroiding". The code you will need to write to load up, correct, and centroid a spectrum will require you to use both loops and conditionals.

2 Emission Spectra

In the spectrum you worked with last week, (A neon emission spectrum), there were sharp emission peaks at very distinct wavelengths. This occurs because when atoms are energized and release photons, they can only do so at several quantized energies (wavelengths). Think of the bohr model of the atom. An energized electron can drop from the 4th to the 2nd energy level, or the 3rd to 2nd, or the 2nd to 1st, etc. Because those energy levels are quantized, the energies of the photons emitted when an electron falls n levels is also quantized. Every atom has a specific arrangement of electron orbitals that makes it such that the combination of wavelengths it can output due to electron cascading is unique (so a trained spectroscopist can look at almost any emission spectrum and recognize the element!).

However, when we view spectra like the one you saw last week, we notice that the sharp peaks do in fact have a certain width, despite the description above indicating they should be perfect vertical lines at some defined wavelengths. There are numerous reasons for the widening: spectrometers introduce some widening, fast moving gas has doppler shifts associated with some photons, etc.

In any case, if we are to perform any sort of scientific analysis on a spectrum, we need to know for each of those peaks what our best guess for the single "right" wavelength for that line should be. One typical method of determining this is centroiding.

3 Centroiding

Centroiding amounts to finding the "center of mass" of a given peak. The formula for determining the centroid of a range of values (such as intensity/signal) is

$$x_{cm} = \frac{\sum x_i I_i}{\sum I_i}. \tag{1}$$

It basically represents a "weighted average" of the peak. Note: we can't simply choose the x value where the peak hits its maximum because the limitations of the spectrometer result in the true "peak value" not lining up with the real wavelength we are looking for. This weighted average of the whole peak provides a better guess.

4 Writing the Code

Goal: Write a script that will go through the spectrum array and find centroids for all the peaks.

Strategy: Determine how to find the centroid of a single peak, then use loops and conditionals to automatically find centroids for the whole array.

4.1 Centroid of one peak

It's relatively simple to understand how to centroid a single peak, when done manually.

1. There is a neon spectrum located on the tutorial page (not the same as last weeks!) – load it and extract an array with the pixel numbers and an array with the intensities (see tutorial 2).
2. Plot the spectrum, and pick one of the peaks to centroid. Use the "magnifying glass" icon in the plot window to zoom in on the peak and see at around what pixel number it 'starts' and 'ends'. (note those numbers, you'll need them).
3. Go back to your code and use the center of mass formula (eqn. 1) to determine a center of mass for the peak. *Hint: remember that you can use the sum or np.sum function to sum an array or part of an array.*
4. Use the function plt.axvline(x, '-.') to plot a vertical dotted line at the value of the centroid you just found over the original spectrum. (Make sure the line plt.show() comes after your last plotting command or you won't see the dotted line). Check that it seems to be at the weighted center of the peak.

4.2 Many peaks

The spectrum we have given you only has 20 or so peaks, so it's conceivable to centroid them all manually. If you had a full Neon spectrum it could have hundreds to thousands of lines! We want to automate the centroid finding process. First we have to automate the peak finding process.

1. Find the pixel positions of the peaks in the spectrum. There are multiple ways to do this; feel free to try your own way. One possible method is to find everywhere in the array where a value is higher than both the one before and the one after it, all while being over some threshold. (A conditional statement might be useful here)! You will need to iterate over the signal array to implement it. Also keep in mind that you need to initialize your list/array *outside* the for loop.

    - Note: you will be wanting to append values to the end of the list/arrays you are working with. Lists and arrays have slightly different syntax for this: for lists, use listname.append(new value), while for arrays use arrayname = np.append(arrayname, new value).

2. This spectrum is very sharp, so the window size within which to calculate the centroid is a few pixels on either side of the peak. Experiment with values between 5 and 10 pixels.

3. Write a block of code that produces a list or array of peak positions, and then use a for loop to iterate through each peak position, calculating the centroid in the vicinity of that peak, appending them to a new list/array of of ”centroids”.

4. Define a function that simply uses plt.axvline, taking an x coordinate as input (and just hard code the ’-.’ option.

5. Use a for-loop to quickly plot vertical dashed lines (using the function you just made) for all your centroids over the original spectrum (you don’t need the plot of the single centroid you did earlier anymore).

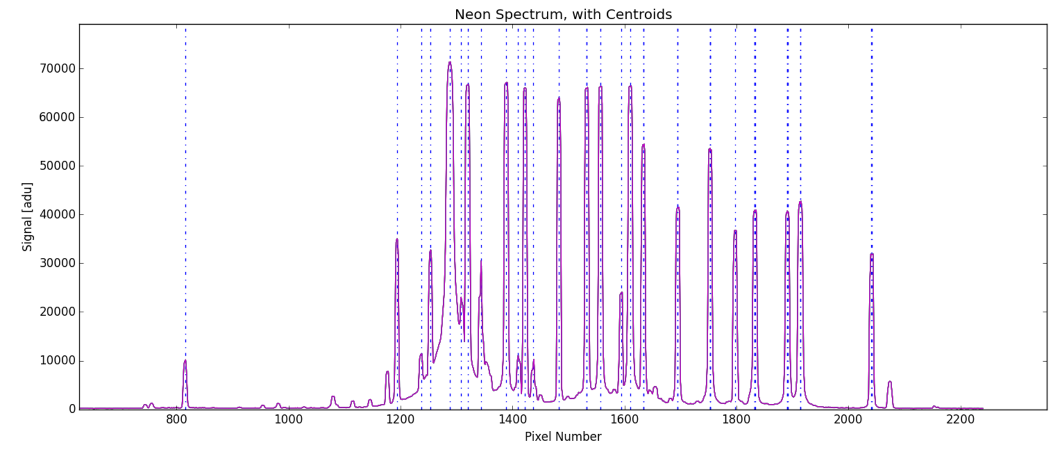

Figure 1: The final product: a spectrum with the centroids of the peaks marked. More importantly, you now have an array with the exact centroid values in it.

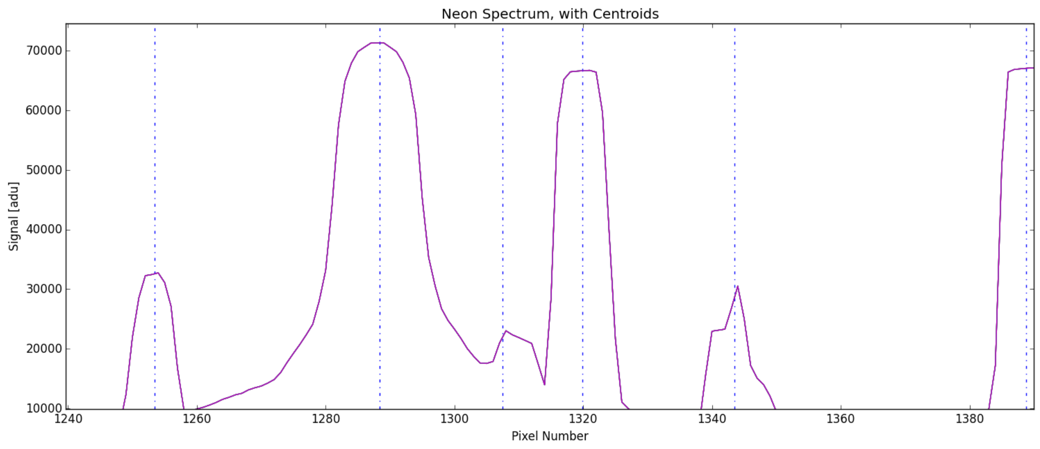

Figure 2: A zoomed in figure demonstrates how the lines lie near the center of the peaks measured.

That was by no means a trivial task. But now we have a piece of code which we can use to automatically find the centroids of the peaks in a spectrum over some threshold. As of now, you could run this code on different spectra, but you'd have to edit your code to load the right file and set an appropriate threshold, save it, then run it in python. You could also just indent everything in the code, stick it all in one function that takes a string filename and a threshold as inputs. Then theoretically, you wouldn't have to edit your code to run it on different spectra.

5 Homework

The homework this week is to finish the tutorial above, as it likely took more than an hour. We would like you to turn in the final python file that performs the centroiding (remember to comment your code). Also upload an image titled "your_username_tut3.png" of the final plot- the spectrum with all centroids marked by dashed lines (see fig. 1). Both files should be in a zip before uploading.

Next week we will be going more in depth into plotting and reviewing everything we've covered so far. Please read up through Chapter 5 of the textbook, which is on plotting.

6 Optional Homework!

If you found the above relatively straightforward and want more to do, (or you're just really bored or want extra practice), here's a little something for you.

6.1 The Centroid Window

You may have noticed that the choice of the range of pixels over which we calculated the centroid for each peak was rather arbitrary. We can actually be a bit more systematic about how we select that window.

6.2 FWHM

FWHM, or "Full-width at Half-Maximum" is, as it states, a measure of how wide a certain peak is at half of its maximum value(fig. 3). Finding the FWHM of a peak isn't too difficult: find the peak, divide the intensity by two, and then find the first place before and after the peak that are that value (look up np.where!). Change your code to calculate the

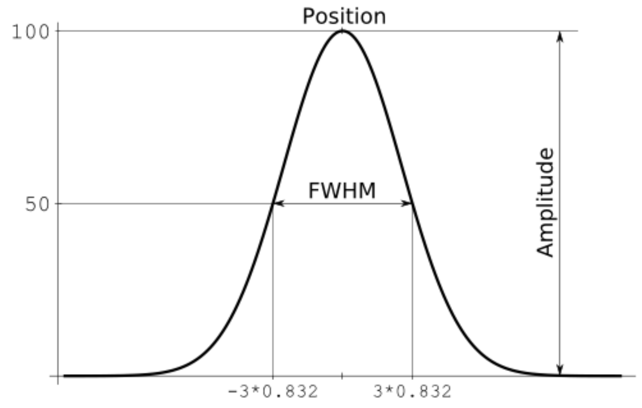

Figure 3: Full Width Half Maximum

centroid of each peak in a window the width of that peak's FWHM. This is where functional programming comes in handy. You can write a function to generate a FWHM for any peak, and then call that function for each peak you are centroiding to determine the range of pixels that are considered in the center of mass determination. This method rightfully uses small windows for narrower peaks and wider windows for wider peaks.

Furthermore, if you wanted to experiment and do a little research, you could try out a second method for turning a wide peak into a single value. The method is to fit a gaussian curve to the peak, and then take the maximum of the gaussian. (This is quite a bit more complicated).