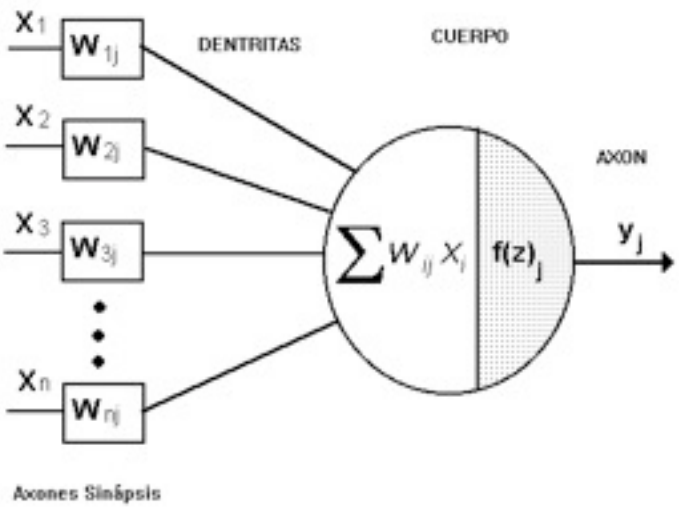
Computation and Mind: ***Lab 1 - An Introduction to Artificial Neural Networks***

It has long been argued that the fundamental unit of computation in the brain is embodied in the neuron. During the middle part of the last century, a considerable amount of research was devoted to understanding the electrical and computational properties of neurons that allow collections of them to give rise to intelligent neural behavior. While the biological operation of the neuron reveals increasingly complexity the closer we are able to observe them, artificial neural networks have long used a simple neuron-like model. ANN's have gone a long way in showing the computational power that maybe created using simple neuron-like elements in various interconnection and learning schemes. Since we are currently studying aspects of perception, in this first Lab, we will explore the behavior of simple sensory neurons and how they can solve useful representation/computational problems in vision.

First, remember the simple *McCulloch-Pitts* model of the neuron that we discussed in class (this time in Spanish for a bit of international flair):



The key elements of this model are that the neuron receives input as a set of *x* values (i.e., x1, x2, x3, etc...). The input values are weighted by the synaptic weights *w1j, w2j, w3j,* etc... and summed up to compute a total internal activation, *z*. This is then stepped through a nonlinear output function *f(z)* to compute a final output. The f(z) takes various forms (such as a sigmoid or step function).

For sensory neurons, we discussed that the form of the "weighting" function (i.e., distribution of the *w1j, w2j, w3j,* values was such that the neurons typically had a "receptive" field for the stimulus space. This means that for input in a particular regions of space the neuron responds vigorously and for other areas it doesn't respond much at all (where "space" here could mean physical areas of the visual field but also the abstract space of color, sound intensity, etc...). Critically, this implies we have to assign a particular kind of meaning to x1, x2, x3 (i.e., that they are neighboring values on some uni-dimensional scale, for examples points in space). There will be a particular xN (where N could be any number) that will make the neuron fire like crazy.

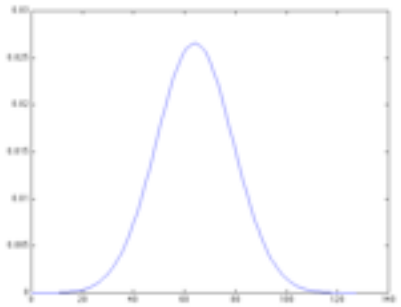
In this lab, we will explore some interesting properties of simple sensory neurons. The lab is divided into two parts, where the third part is at the "advanced level" for the expert programmers in class. Note that

this lab will be divided into a couple sections that slowly build upon each other. Thus, it is critical that you keep up with the class for each part.

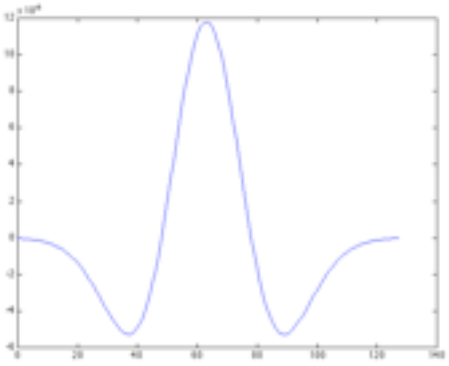
Part I: Building a Simple Artificial Neuron

In the first part of the lab, we are going to design a simple perceptual neuron that has a receptive field with various weighting functions and explore how various input signal and connection weighting schemes influence the behavior of the individual neuron. It's like we are scientists in the lab holding a signal neuron in a micro-pipet and we are going to explore how the neuron responds to various inputs. However, instead of all the messy wet stuff, we do this comfortably from home with our cup of coffee and our laptops.

The two key ideas we are going to play with is different types of receptive fields and their respective properties. The first receptive field we will consider is a "Gaussian" receptive field, which has the following "bell-shaped" curve:



The height of each value is the value of the weight, w, given to a particular input (in this cases there are 128 inputs). We interpret this as the neuron gives the most weight to items that fall in the "middle" of its receptive field and less weight to other regions. The second type of weighting that we will considered is known as the "mexican hat" field, which looks like a mexican hat:



There are two things to notice about the mexican hat RF. First, like the Gaussian, it has a peak in the middle, meaning it really likes stimuli that fall in the middle. However, unlike the Gaussian, it gets negative in the regions on other side of the peak, meaning it really DOESN'T like these other parts. In other words, it has an INHIBITORY region where if it gets input in that area its overall activation is actually LOWERED.

We are going to construct neurons that have these two kinds of weighting functions and explore their response to a variety of inputs.

We will design our neuron using a object or "class" in python. Remember object-oriented programming lets us create "entities" in the computer that we can add behaviors do. Then we can later clone these entities to create more complex problems. Each entity will act according to its pre-specified rules. In this case we want to make a neuron entity.

Here is the scaffold syntax:

```
class RFNeuron():
      def __init__(self, ninputs):
            print "this is what happens when the object is first created"

      def setup_N_wts(self, mean, sd):
            print "init the network to have a gaussian receptive field"

      def setup_MH_wts(self, mean, sdwide, sdnarrow):
            print "init the network to have a mexican hat receptive field"

      def o(self, inputvec):
            print "this is where compute the output for a given input"
```

You goal in the first lab is to implement the missing functions of this neuron and to stimulate it for various inputs.

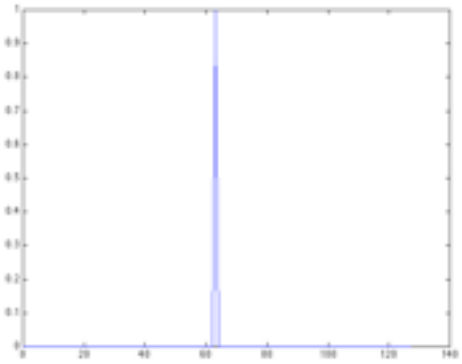
Step 1: Implement your basic neuron
We will do this in class, together.

The key things we will implement are the `__init__()` function, which initializes the neuron to its starting state. The second will be the `o()` function which computes the output of the neuron for a given input. The output will simply be a weighted sum of the inputs (also known as the "dot-product" in linear algebra). We will assume there is no step function or output transformation for our simple sensory neuron.

Next, we will implement two special function (`setup_N_wts()` and `setup_MH_wts()`) which will set up the weights for the two receptive fields described above (the gaussian and the mexican hat).

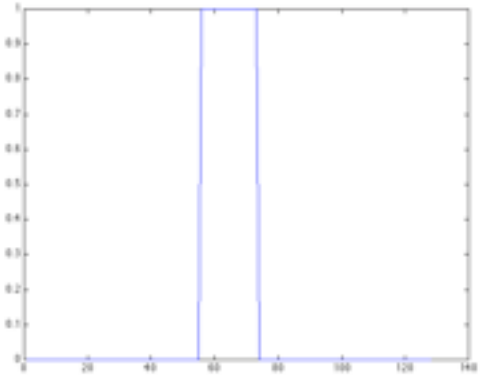
Next, we will explore the response of the neuron to various kinds of inputs: 1.) a sliding Dirac delta function 2.) a centered unit pulse of various widths and 3.) a square pulse with phase shift (a sliding box). Each input type will be analyzed to discover the relationship between the input and the output.

Step 2: Probe your neuron for various "dirac" impulses (only a single input in a single location)
To do this part of the lab, create an empty vector and set only a single value to 1.0. Record the neurons output to this input. Do this repeatedly for different positions of the 1.0 impulse and see how the activity of the neuron cases. Here is an example of the dirac input:



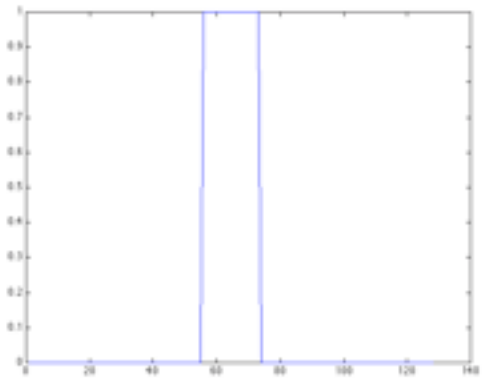
You will want to repeat this for both the gaussian and “mexican hat” receptive field (center-surround) and plot the results as a function of the position of the input pulse (i.e., for all 128 inputs there is a unique output, measure this and plot it).

Step 3: Probe your neuron for various width but centered pulses
Our last inputs were tiny impulse that were only “on” at a particular space in the input vector. Now we want to put a pulse of various widths in the center of the field.



Again, You will want to repeat this for both the gaussian and “mexican hat” receptive field (center-surround).

Step 4: Probe your neuron for a fixed width pulse but that “slides” across the input space.
Our last inputs were wide step impulse that was centered in the middle of the neuron’s RF but grew in size to be larger or smaller. Now we want to choose a pulse of a fixed width and see what happens when it slides over.



Again, You will want to repeat this for both the gaussian and “mexican hat” receptive field (center-surround).

Next Week: We will extend our system to build a simple network of neurons that together “perceive” a particular input! If you don’t finish playing with this in class today try it for homework.