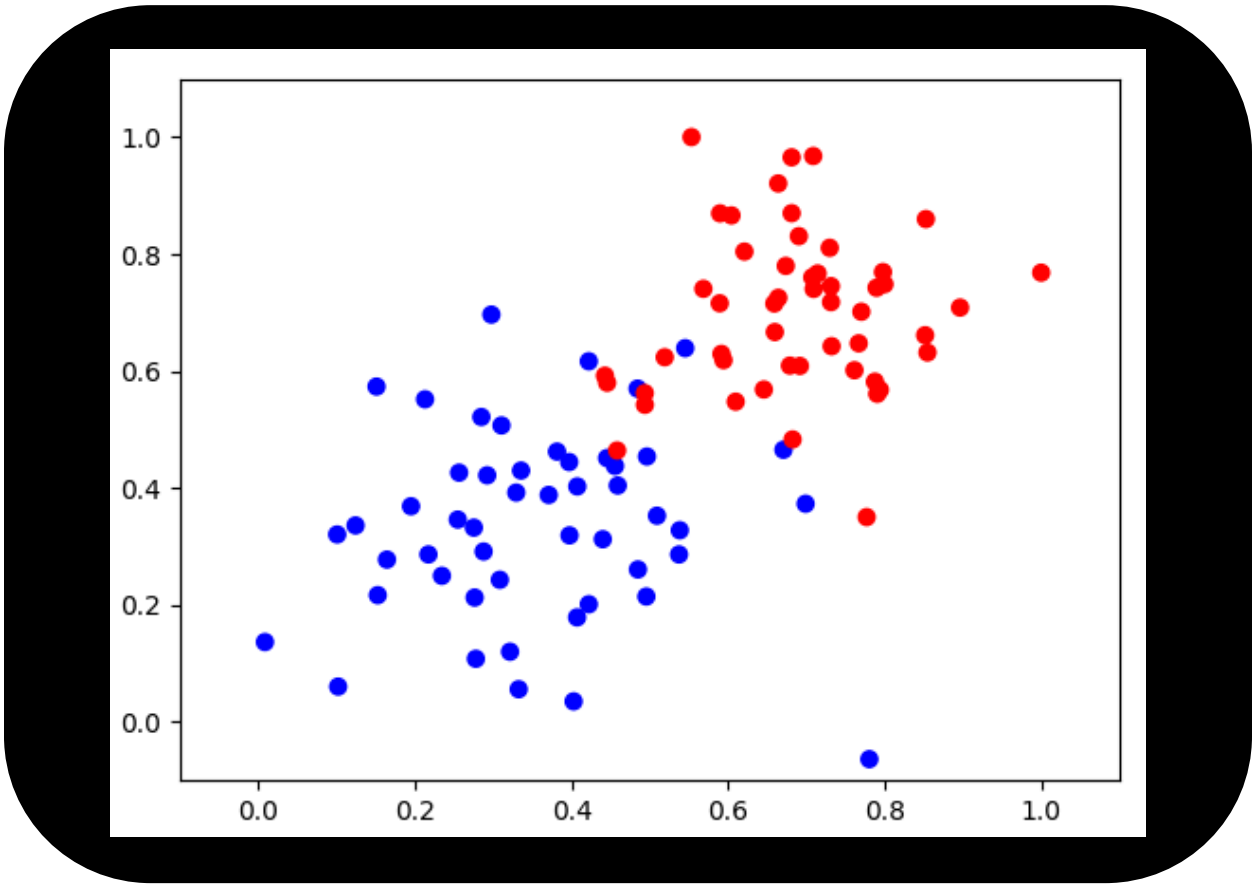
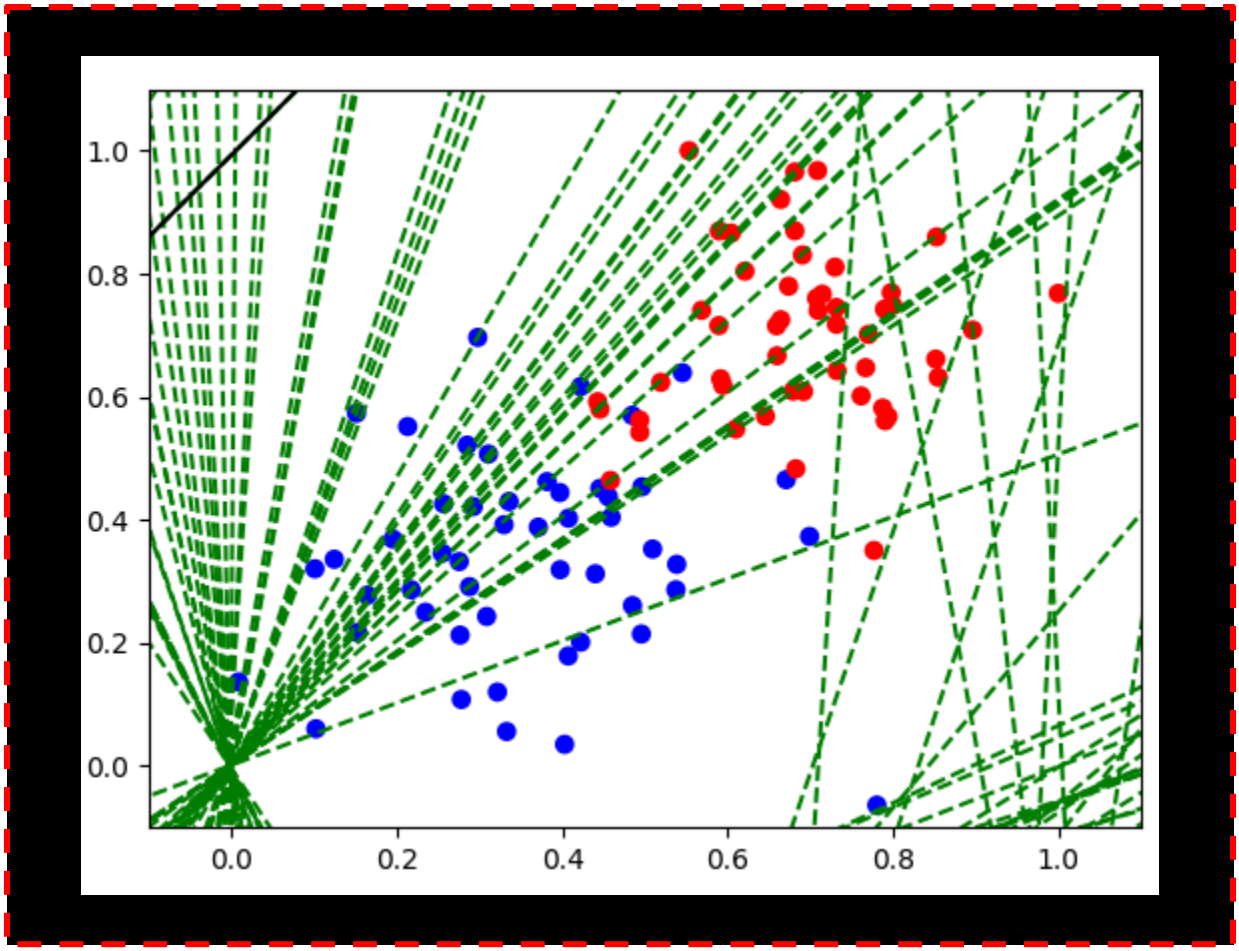
Assignment#4

Joel Collazo



Here is the data for the graph this will be the same for the two algorithms that were used in the Home work and they are separated by color the 0 are RED and the 1 are BLUE

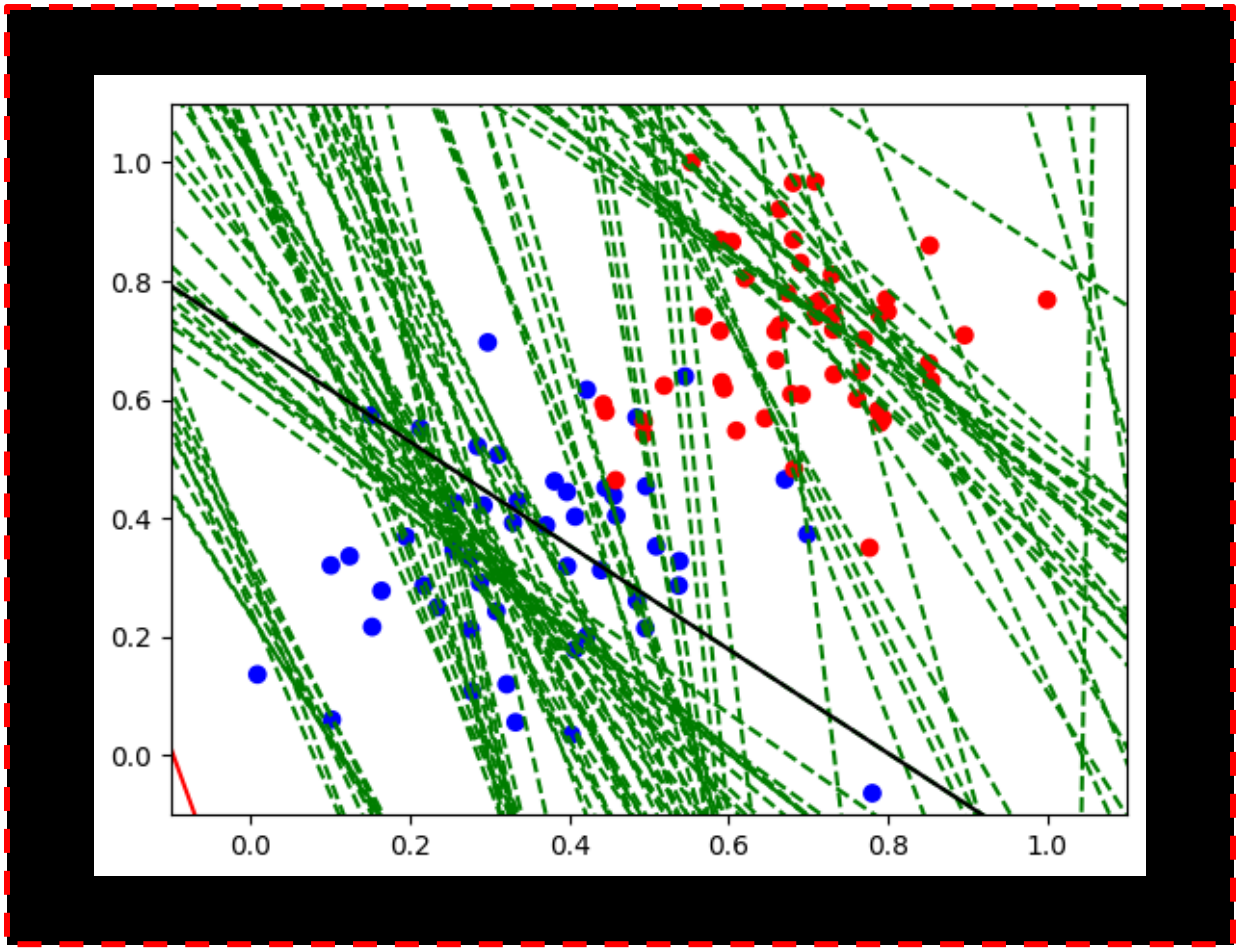
When you run the “make_graph.py” this will be made and then shown up in the window should look just like this image since this is from Python



100 iterations **Rate** of change at 1

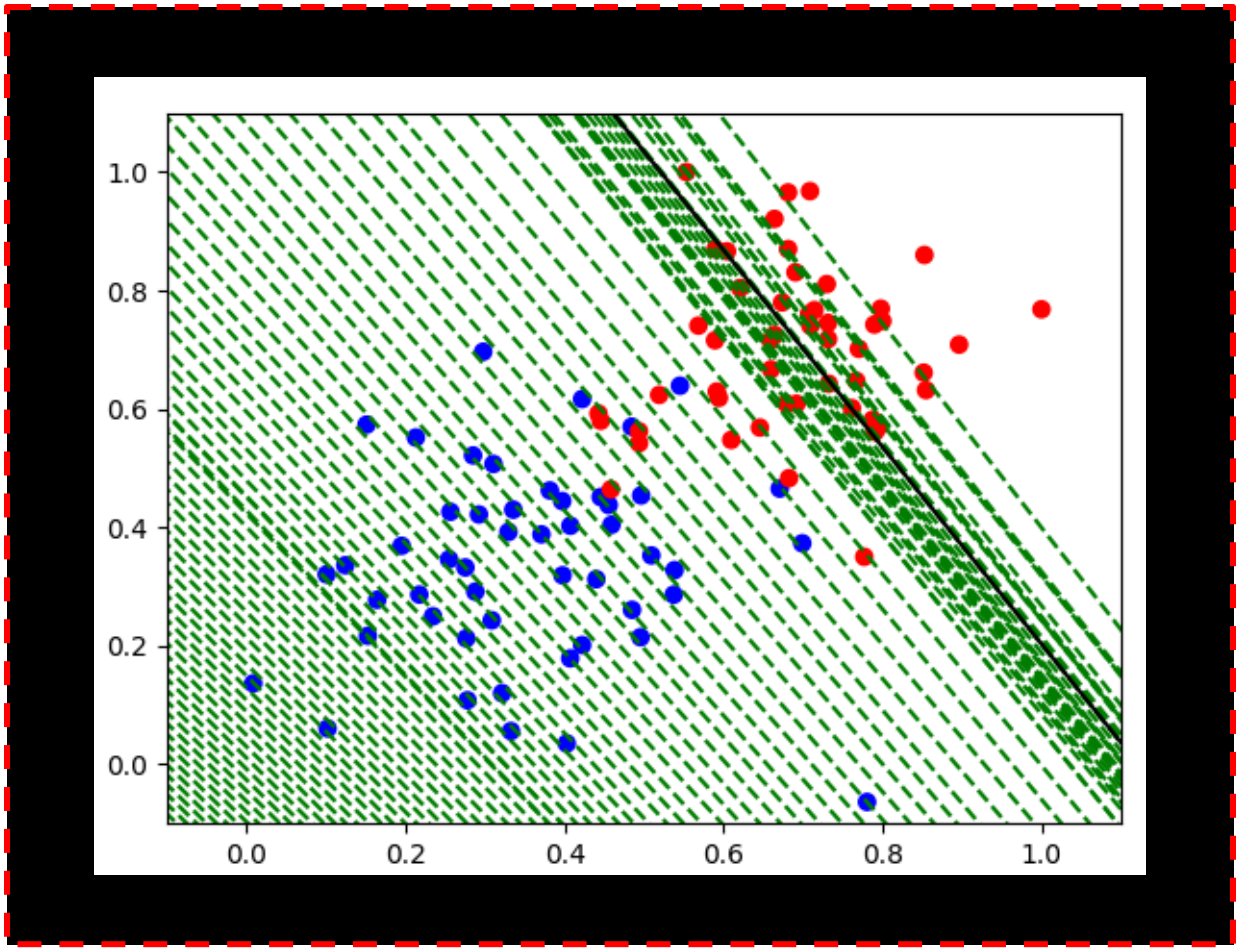
Here is a test of the “**perceptron_heuristic.py”** program and does the heuristic approach that was requested that we did from here we can see that the rate of change is 1 this will make the jumps up or down by 1 this is large so there is alot more jumping around to find the line that can separate the most data

It was here that i did realize that the data can NOT be split there is a few points that just don't make that possible so i just did 100 iterations to see some lines being made and not just a mess of lines since there is a large rate of change



100 iterations **Rate** of change at 0.1

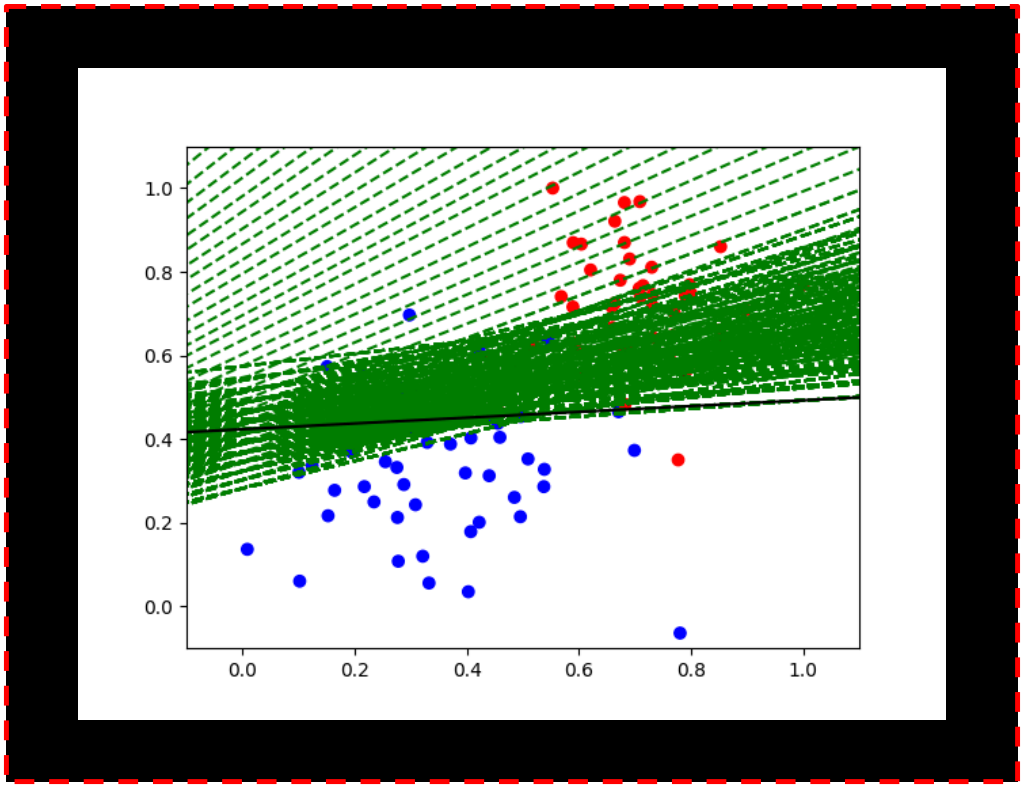
Here is another test for the "**perceptron_heuristic.py"** program here i did change the rate and it does seem to have changed it a bit but there is not much a difference at a glance but it does make sense since we are doing a rate of change that is about ten times smaller than with this there is still clumping to find the line that can have the points that can be separated by this line



100 iterations **Rate** of change at 0.01

Here is another test for the “**perceptron_heuristic.py”** where i changed the change even smaller this time and this one you can see how it really does take it's time to really get in there and look for a point that can be the solution

This however is really really slow you can see that it has not covered that many other angles or places where the line can be so this is much more thought to find the right line and angle but there is one problem it is much slower to get there but the chance of missing that point is lower



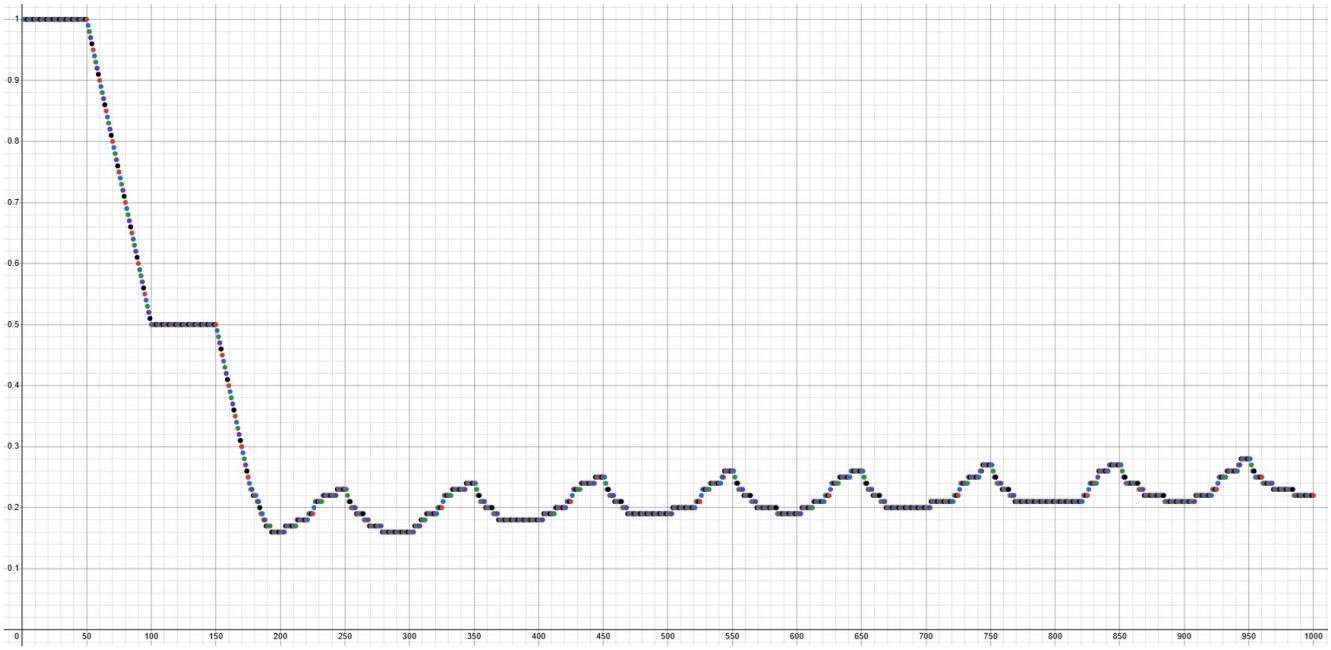
1000 iterations **Rate** of change at 0.01

perceptron_gradient_descent.py

Here i am using another appellation i made to do this logic for the gadient deasect that was in the slides where there is some things that are similar to the but the way you add the values are a bit different

There is as well as a low rate of change so then we can keep going down to find that point where we can stop i did many many iterations to find low ponts and in this one where was a good over all movement to find this lowest point but it was then going up witch i thought it was interesting and then remind me of class when we went over the true lowest point

I did print out the points for this chart to see what this curve looks like when we are looking for the lowest point



It starts at one then goes down very well looking for the lowest point
Then it does slowly start to go up from there while it tries to find that point

The shape is over all very interesting and can see that it's really good at the beginning trying to find that low point but as it goes on then it does start to loose where it is going