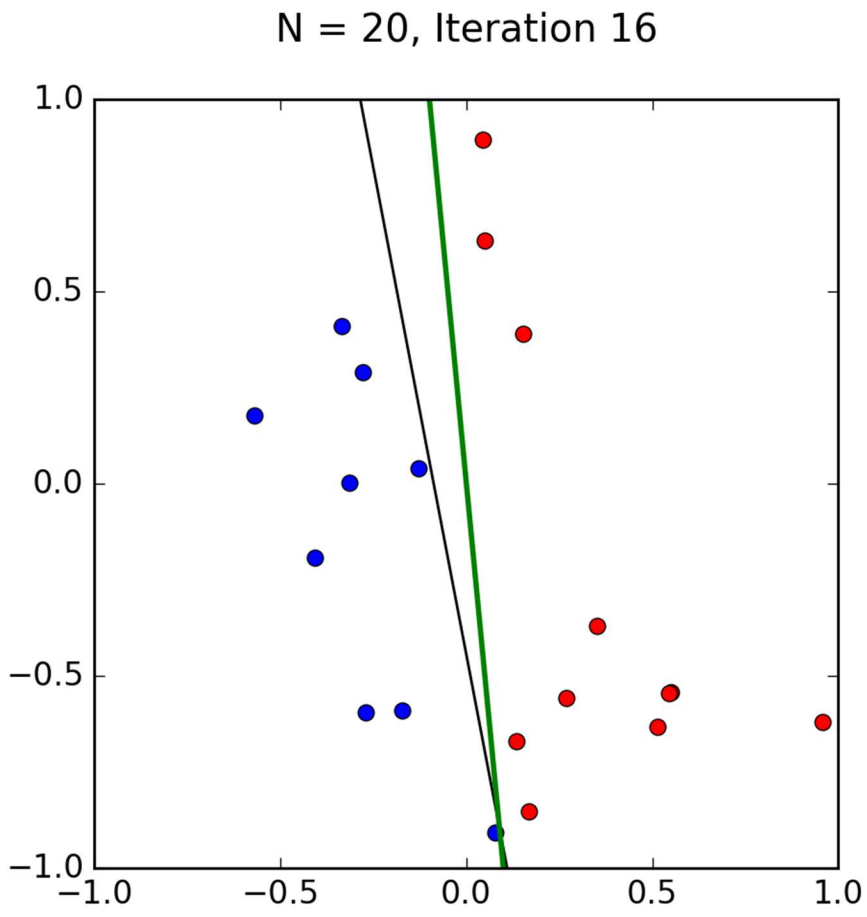
Jack Barry

AI CMPT 404 HW1

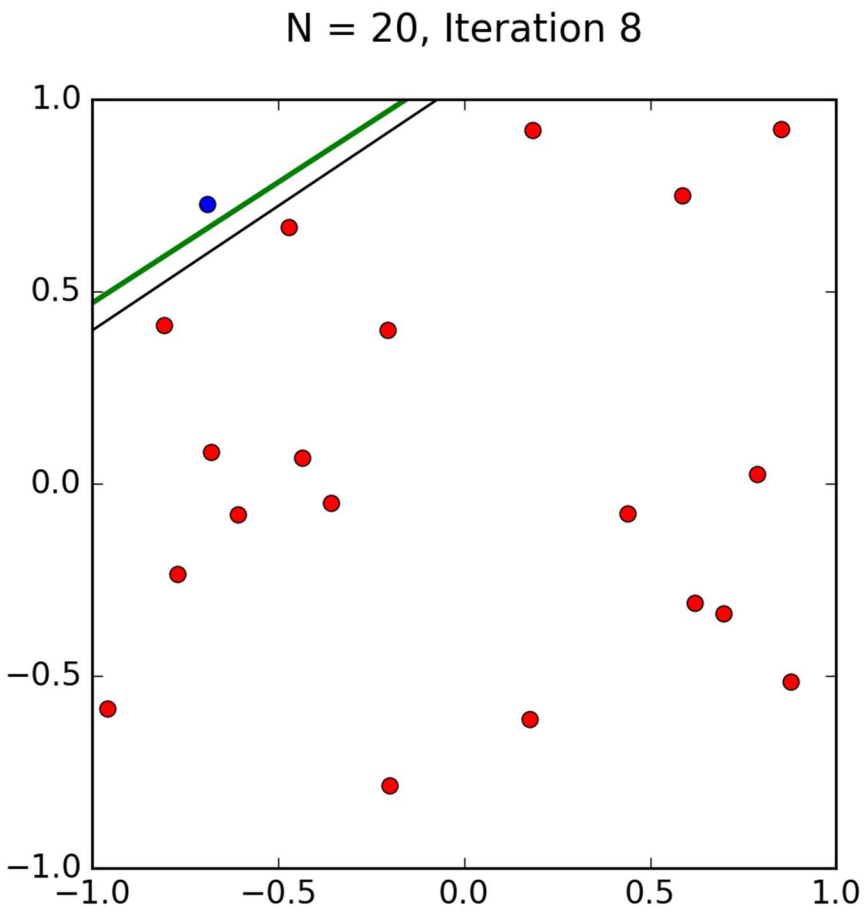
9/20/2016

a- This part of the problem was solved by the code that we used from the data science blog, we had to make minor updates by installing extensions to packages so that the code could compile.

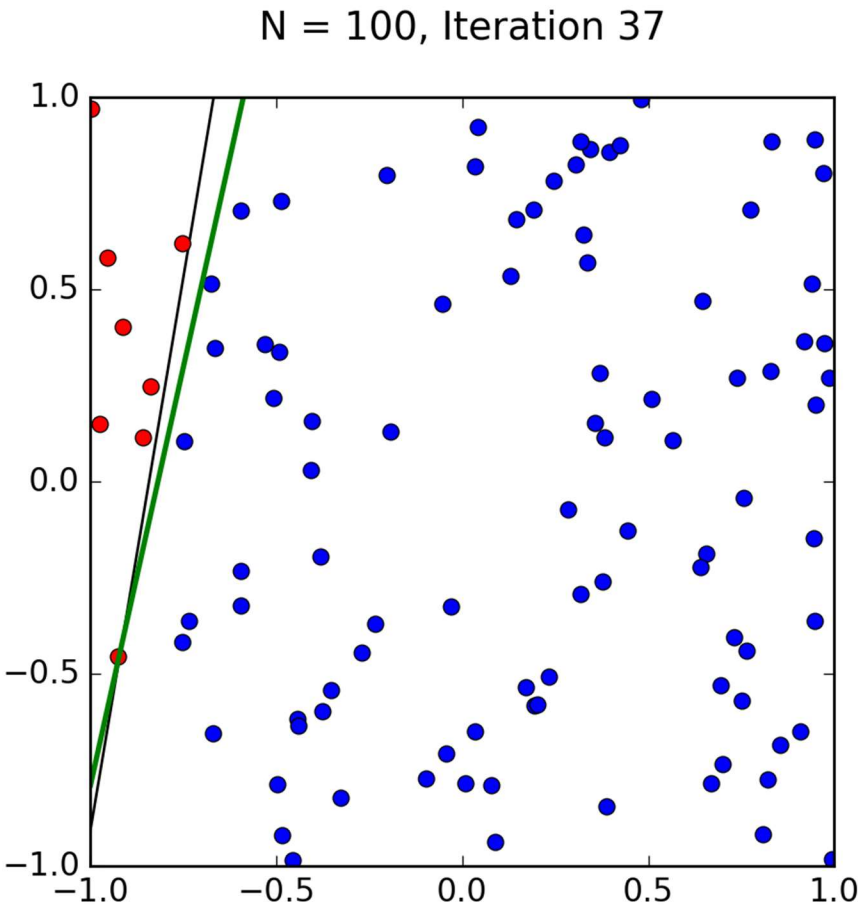
b- This is the first run of the pla with n=20. We can see that it took 16 iterations to converge. g is relatively close to f however it diverges from f as we go up the y axis.

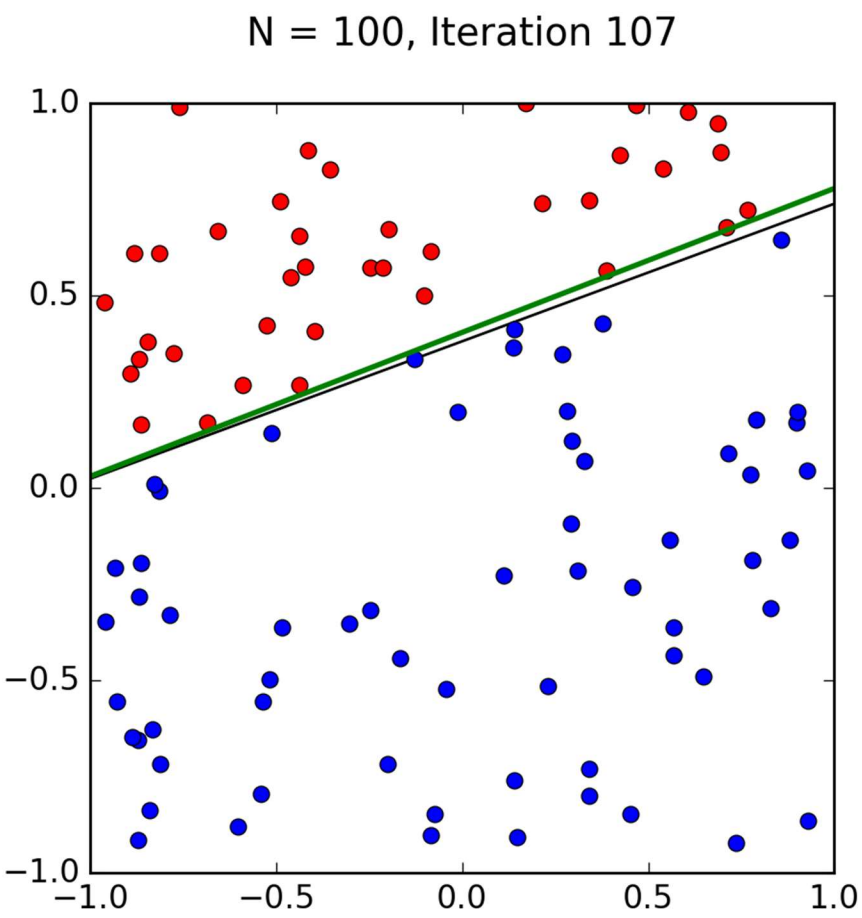


c- This is the second run of the pla with input of n=20. We notice here that the pla took half the amount of iterations to find a solution when compared to the previous example. It is also interesting to note that the data is split 1 to 19 which is much less even than in the previous run. In this case g is mirroring f in slope but the y intercept is moved up slightly.



d- When we increase the size of the input n=100 we notice that the number of iterations increases to 37. This is more than double the number of iterations in case b.

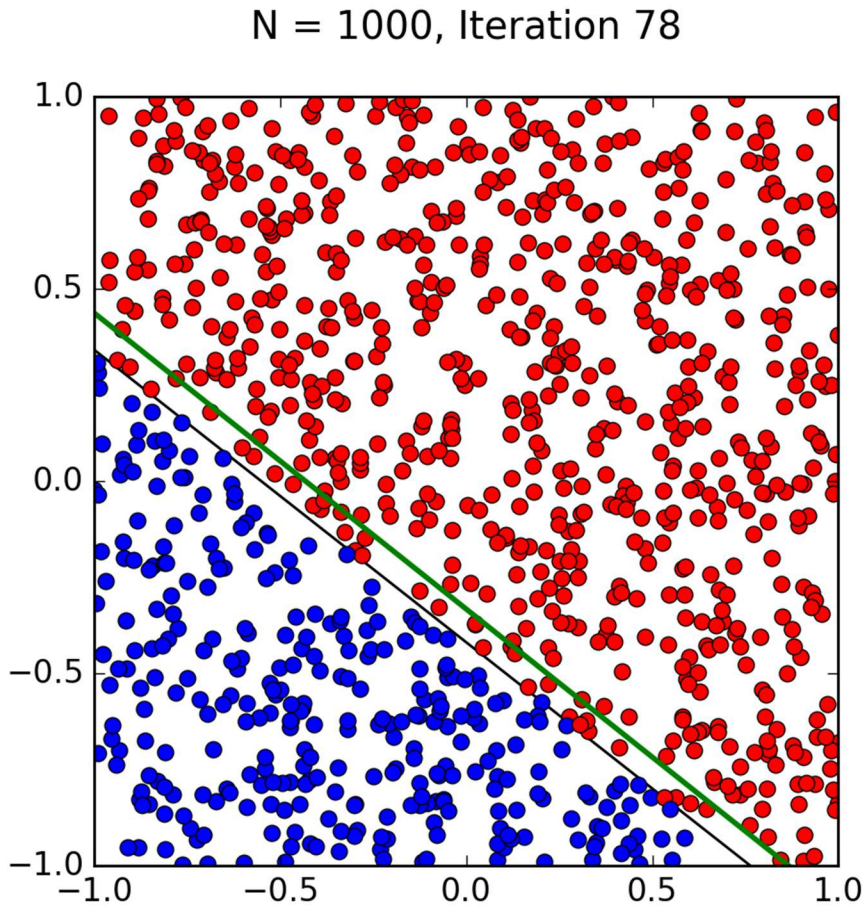


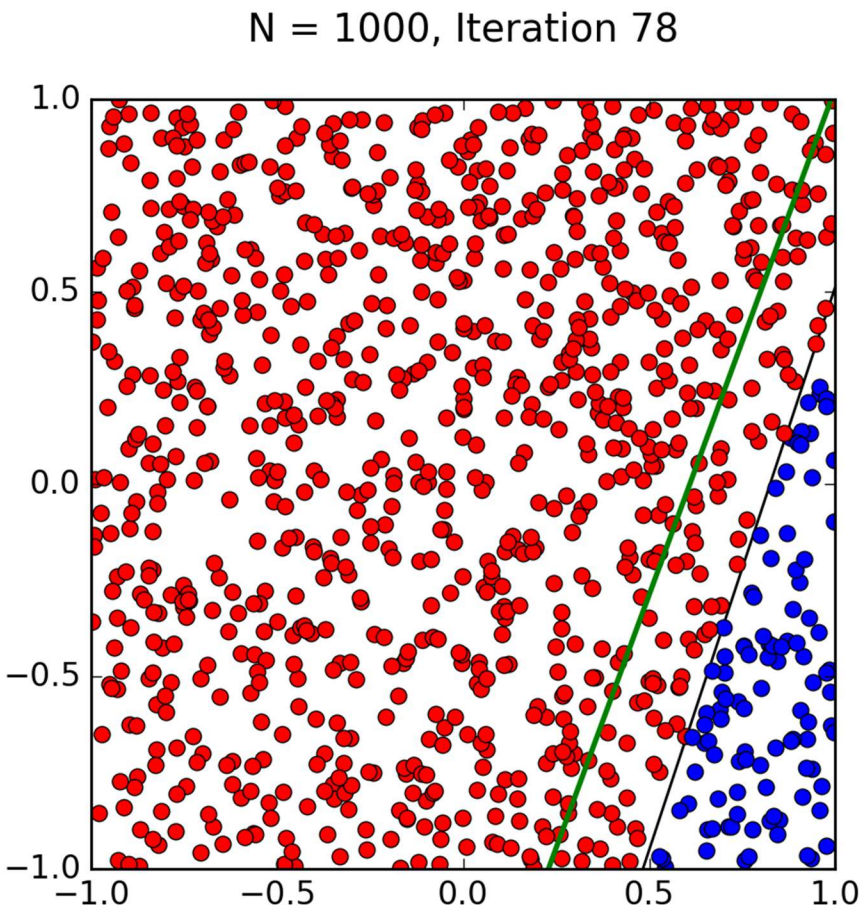


When we run the pla again with n=100 we see a significant increase in the number of iterations from the previous run with n=100.

e- When we run the pla with n=1000 we encounter a memory error. This is likely due to the low capacity and processing power of my laptop. If this were to run a server I would expect better results. I tried to run this twice and as seen below both times the algorithm failed at the 78th iteration and gave the same error below.

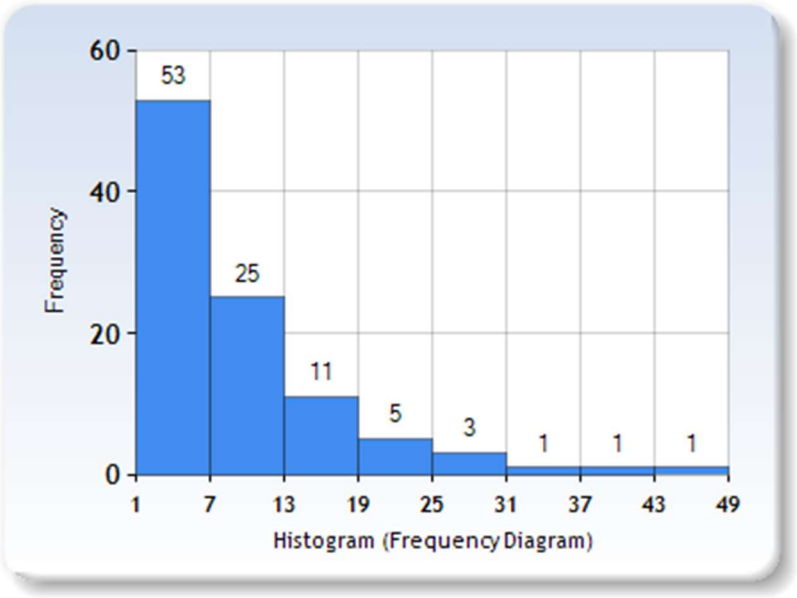
```
  File "C:\Python27\lib\site-packages\matplotlib\backends\backend_agg.py", line 469, in draw
    self.renderer = self.get_renderer(cleared=True)
  File "C:\Python27\lib\site-packages\matplotlib\backends\backend_agg.py", line 486, in get_renderer
    self.renderer = RendererAgg(w, h, self.figure.dpi)
  File "C:\Python27\lib\site-packages\matplotlib\backends\backend_agg.py", line 93, in __init__
    self._renderer = _RendererAgg(int(width), int(height), dpi, debug=False)
MemoryError: In RendererAgg: Out of memory

Process finished with exit code 1
```





f- When the problem is increased to 10 dimensions the output is not able to be visualized so we begin to record the number of iterations. The algorithm breaks due to memory problems when n=1000 in this case again. It stops at the 20th iteration instead of the 78th iteration in the previous examples when we encountered memory issues.

g- The histogram can be seen below. The output vector for the number of iterations was [19, 5, 16, 11, 7, 7, 6, 9, 5, 17, 2, 9, 11, 2, 2, 3, 2, 9, 4, 1, 12, 7, 19, 7, 3, 4, 13, 5, 7, 7, 3, 15, 4, 11, 3, 2, 1, 13, 13, 1, 4, 13, 1, 2, 8, 25, 31, 7, 18, 3, 3, 6, 3, 4, 5, 2, 3, 2, 40, 5, 1, 11, 3, 17, 1, 5, 19, 13, 5, 6, 5, 10, 14, 1, 11, 2, 1, 4, 12, 2, 4, 5, 20, 45, 6, 7, 22, 29, 26, 7, 4, 11, 1, 4, 9, 4, 1, 2, 7, 7]. I classified these into groups to make the visualization readable.



h- In summary I think that due to the variations I encountered in the number of iterations there are two factors that affect the pla. The first one being the input size and the second being the nature of the data. If the data is very skewed towards one set then the pla converges faster than if it is more evenly distributed. And as the input data increases the algorithm takes more iterations to converge which is standard with most algorithms. I suspect that this algorithm is running at around $O(n^2)$.