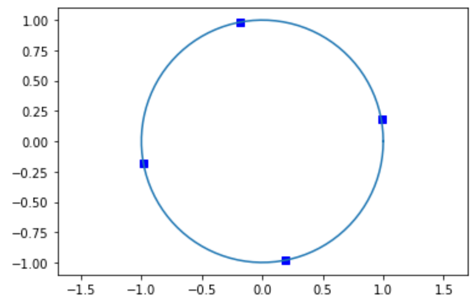
Jared Kelnhofer

Machine Learning Project 6

April 21, 2010

To start my project, I went through the entire notebook to ensure that I had a proper understanding of what was going on. I enjoyed learning about the five platonic solids, and how they all have their vertices on the unit sphere. Two areas that took a while for me to comprehend were the sphere_term and repel_term methods.

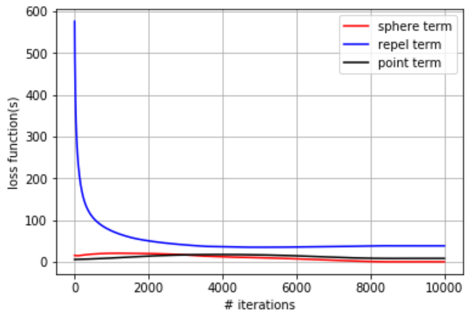
When I worked on the first part of the assignment, which asked for a new term in the cost function that would cause a point to gravitate towards [..., 1] I decided to cause the first point in the array supplied to be penalized for being far from that point. This ended up having a huge effect on the other points, because they all "repelled" each other. Here are the statistics for my two-dimensional run of this part of the assignment:



2D visualization after 10,000 runs with new "desirable point" term added to loss function.

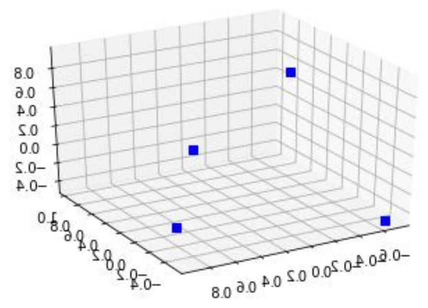
```
tf_array is:
[[ 0.98535286  0.18659521]
 [-0.18524838  0.9855357 ]
 [-0.983537   -0.18504064]
 [ 0.18714326 -0.9828854 ]]

the norm is.....[1.00286491 1.00279489 1.00079222 1.000543   ]
```

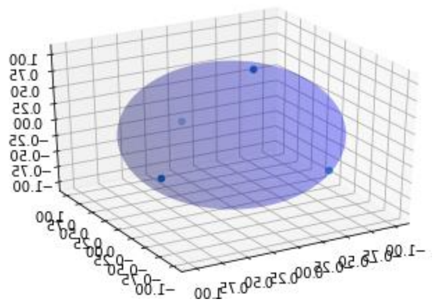
Information about the values of the 2D points and their norms (distances from the origin) after the addition of the new term.



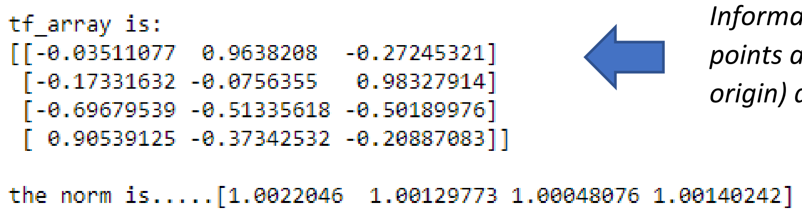
The values of different loss function terms plotted against the number of iterations.

I decided to scale the creation of the "desirable" point dynamically alongside the number of dimensions. Because of this, I was able to change the NUM_DIMS variable and quickly run my 3-dimensional test:



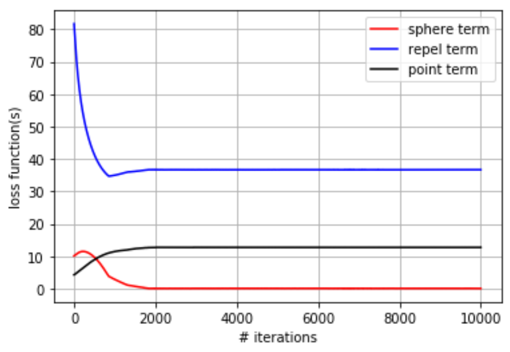


3D visualizations after 10,000 runs with new "desirable point" term added to loss function.



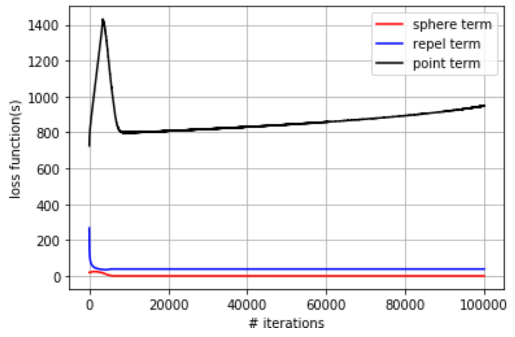
```
tf_array is:
[[-0.03511077  0.9638208  -0.27245321]
 [-0.17331632 -0.0756355   0.98327914]
 [-0.69679539 -0.51335618 -0.50189976]
 [ 0.90539125 -0.37342532 -0.20887083]]

the norm is.....[1.0022046  1.00129773 1.00048076 1.00140242]
```

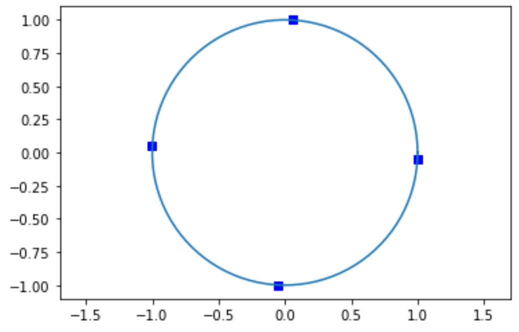
Information about the values of the 3D points and their norms (distances from the origin) after the addition of the new term.



The values of different loss function terms plotted against the number of iterations.

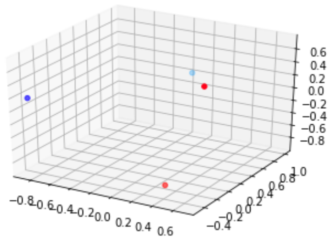
When I had to do some troubleshooting, I decided to alter the Alpha, Beta, and Gamma coefficients in the loss function, and hope to gain some insights. I decided to give the Gamma coefficient a higher value because once one point was “locked in” the other bits of the function would be able to fall into place. It would be much harder to do it the other way around though. These are the results of my increasing the Gamma coefficient by a factor of 10, and also doing 100,000 iterations. I think that, despite the higher training time, this approach yielded results that are worth it:



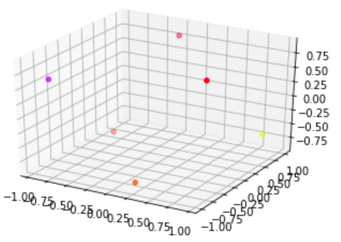


For part two of the project, I did a little research, and learned that the five platonic solids are all composed of different numbers of points that lie on the unit sphere and are as far from each other as possible. The numbers of vertices that the platonic solids have are, (4, 6, 8, 12, 20.) Because I had a list of the numbers of vertices I would need, I simply changed the NUM_DIMS variable to 3, and the NUM_PTS variable to each of those numbers in order, and observed the resulting solids:

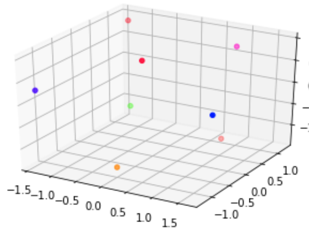
Tetrahedron (4 vertices):



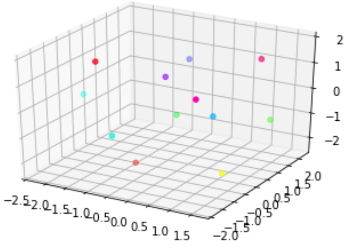
Octahedron (6 vertices):



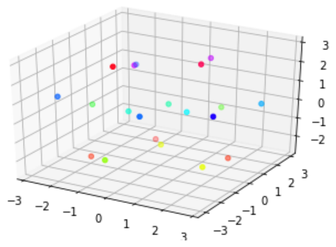
Cube (8 vertices):



Icosahedron (12 vertices):



Dodecahedron (20 vertices):



The thing that I liked most about this project was the fact that it made me ask questions. This is the first time that I haver really dived into what you could call “math world” and played around with shapes, points, and angles without considering real-world applications. It was very fun to experience that. This project also showed me just how much I still can learn regarding libraries such as TensorFlow and Matplotlib.

I also really liked tweaking things like the number of dimensions to work in, the number of points to use, the weights of different loss function terms, and the number of iterations. It seems like a lot of these examples needed far fewer than the default 10000 iterations, because their loss coefficient graphs usually ended up going flat rather quickly.

I think the most valuable thing that I learned from this assignment was what to prioritize as far a learning. There were many tempting rabbit holes to follow, and it was a struggle to sort out the most essential concepts were for me to learn, so that I could get it done on time. I still have questions, and lots of things that I would like to continue working on for this assignment, such as automation, and prettifying my plots, but overall I would say that this assignment has been the most informative for me so far.