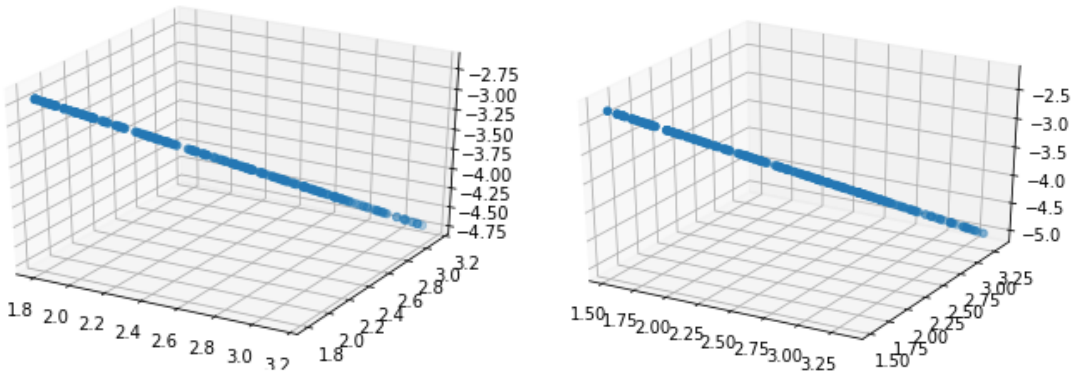
Part 1

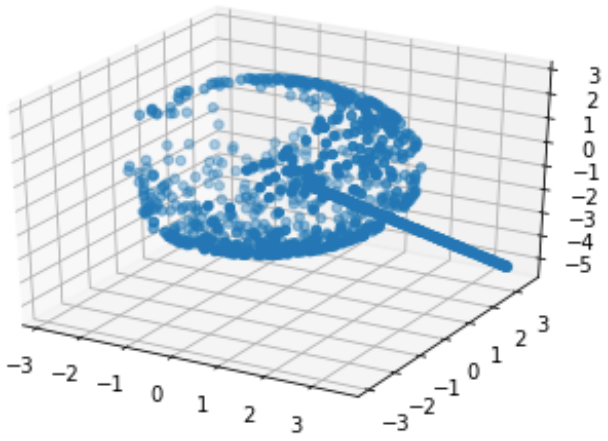
According to last comments, I adjusted my methods. I just used unnormalized graph Laplacian and still detected the line. Then I tried to extend the line. My major idea was to add neighbors of points in line to the set of line. I tried k-nearest neighbors and did the iteration. At the first 5 to 10 iterations, the method worked good and I got the line extended.



(a) (b)

Figure 1.1 (a) is the original detected line and (b) is the extended line after 5 iterations. The line is obviously extended, thus this method is useful.

But after 10 iterations, not only the line, the sphere was also extended. Like this



Obviously, the shape of sphere could be recognized. And I found that the sphere was

recovered faster than the line did. The cause is obvious. When I extended the line, I could not avoid adding points on the sphere to the set and then I would finally recover the whole dataset. Then I thought of distinguishing different points during the adding process. I tried few approaches.

1. applying denoising procedure after each iteration

I considered the points from the sphere as noise data. And every time I added points to the line, I denoised the data and decrease the points from sphere. However, when the iteration increases, the performance was not good. My denoising approach was to delete the neighbors which had large distance to the corresponding points. I checked the plot and found I could not delete all points from the sphere, thus this approach did not work well.

2. clustering during iteration

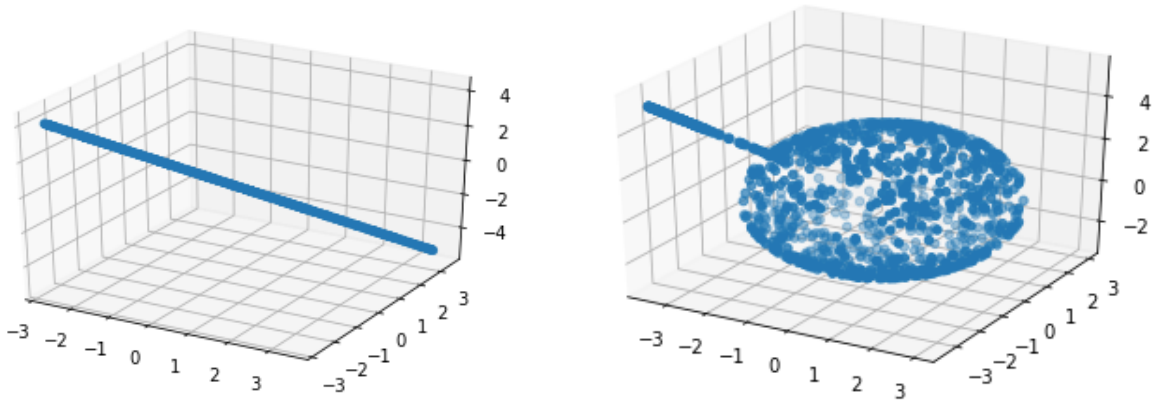
I tried to do clustering after a iteration, but the result showed me the points could not be clustered. This method failed.

3. Using direction vector

My goal was to extend only the line. My line had a very important property, direction vector. The direction vector of the line was a constant vector while that of the sphere kept changing. Thus I could distinguish the neighbors which has the same direction vector when comparing with corresponding points. This method worked well and I did extend the full line.

But I still found some problems in method 3. I checked the length of the dataset of line and the plot of the complement of this set (it should be the sphere). After 200 iterations,

the length is 838, and 162 points were missing, and the plot of the complement set is:



(a) (b)

Figure 1.2 (a) was the extended line after 200 iterations, and (b) is the plot of rest points. Very obviously, the full shape of the line was recovered but there were still some points left in complement set. I found that during first 20-30 iterations, the extending speed was fast and then the speed was getting slower. Only 120 points were added to the line during 100^{th} to 200^{th} iteration.

And I was confused about why not all points were added to the line. From the plot, the left line was absolutely neighbors of the existed line, but they were not included in the extended set. I am still correcting the code to make the procedure faster.

Part 2

In Part 1, the method based on direction vector worked well on lines. I tried other combination of objects like plane and line. Lines could be easily detected and extended. But I thought this method was still too specific and restricted. I was still looking for a more general method.

To generalize the method, I think of this procedure:

1. detect objects with graph Laplace and clustering;

2. find unique properties for the objects we would like to extend. Especially, the properties could be checked by points.

3. using these properties to distinguish the points and add appropriate points to my target set.

But I found it is quite hard to find the properties. Thus I read the other paper you sent me *The Laplace-Beltrami operator: a ubiquitous tool for image and shape processing* and tried find general methods. For I was not familiar with manifold and this operator, I find a book *Introduction to Manifold* and learned about differential geometry.

I was trying to apply Laplace-Beltrami operator on discrete data. And I have a problem, is this Laplace-Beltrami operator the same as graph Laplace or this is a different operator? The paper considers 2D dimensional manifold embedded in higher dimensional space. This specification coincides the dataset I simulated and the detecting problem. I was still focusing on this paper. I was trying to write some code to do the same thing in this paper. This paper was a little bit complicated for me. If I discover something, I will report to you.

I will add error to my dataset, and try more things about denoising and look for other ways of extending. I have to admit, extending the specific object is not easy.