Report 4

Part I OWL1 regression

I write the code of OWL1. I obtained an interesting plot. I found this is better way to detect the line.

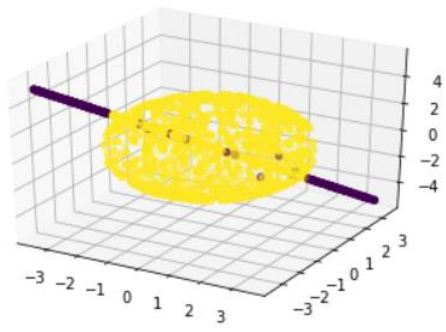

Figure 1. *1000 points of line and 1000 points of sphere. Set weights to be Oscar weights with* $\alpha = 0.0001$ *and* $\beta = 0.01$*. Randomly choose k=100 indexes and do k Owl regressions. Color data points with the second smallest eigenvector of its Laplacian. In this plot, points belonging to line is deeply colored and we could recognize a figure of line.*

Then I set k=200 and do kMeans clustering on Laplacian matrix. Two plots of the result are very interesting.

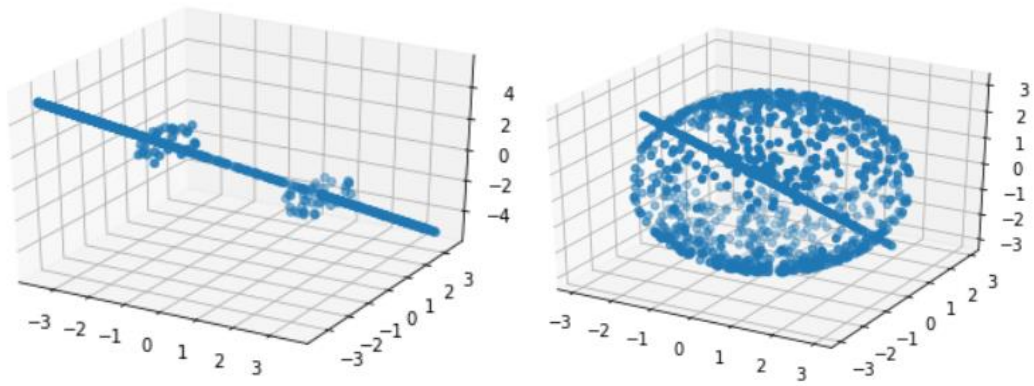

Figure 2. *(a) plot of line and (b) plot of sphere. (a) contains most points on the line except for some points inside sphere and a little proportion of sphere. More iterations did not improve the performance.*

However the set contained a line has some noise from sphere. I would not like to set it as an initial set. I would like to extend set from knn Laplacian matrix.

According to the first stage, we obtained a set from the line. In another word, we obtained a labeled set but relatively small. Let L denote the labeled dataset and U denote unlabeled dataset. Inspired by OWL regularized subspace clustering, we consider a semi-supervised version. For dataset L= $\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$, I do ordinary linear regression and obtain coefficients for m points and set 0 for $\{x_{m+1}, x_{m+2}, \ldots, x_n\}$. For rest data do OWL regression and combine coefficients to build a matrix B. Then according to paper *Scalable Sparse Subspace Clustering via Ordered Weighted l1 Regression*, after obtaining matrix $W = abs(B) + abs(B).T$, do spectral clustering to Laplacian.

However, this method does not work well.

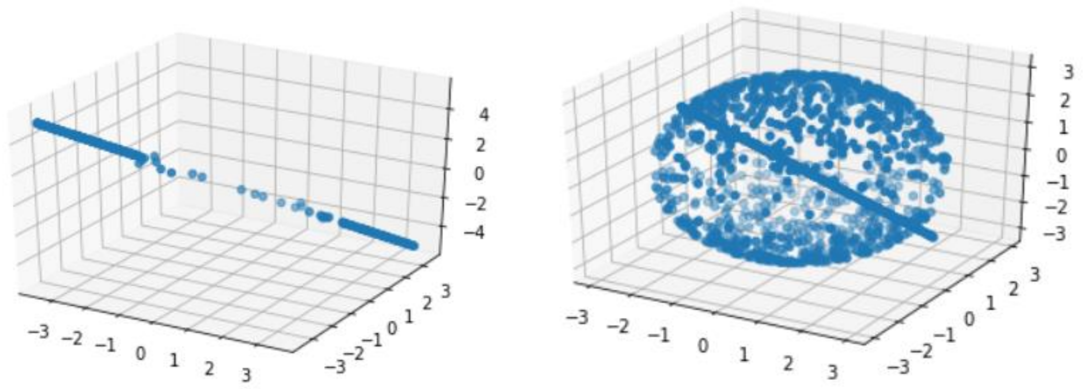

Figure 3. *Every point outside the sphere is classified, but the points inside are not.*

Part II An ideal case for Spectral Clustering

So here comes the question: if line exists inside the sphere, how to detect and classify. For this question, I tried knn Laplacian but the result was bad. Knn Laplacian does not work for this situation. That means the condition inside the sphere is different. However, when I checked the connection matrix for data points, I found the points in line only connected with other points in line. This phenomenon should guarantee the classification of line. I tried again and have this plot below.

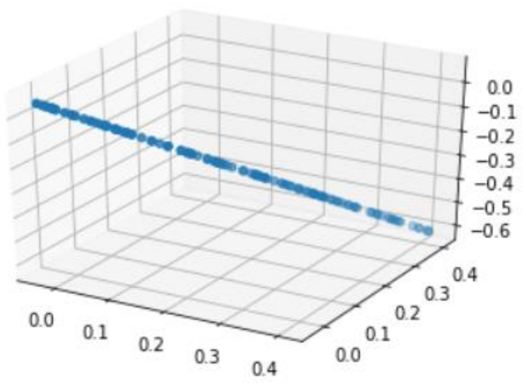

Figure 4. *Obvious figure of a line appeared.*

Then I did nearest-neighbor extension. After 60 iteration, the whole dataset split into two disjointed sets.

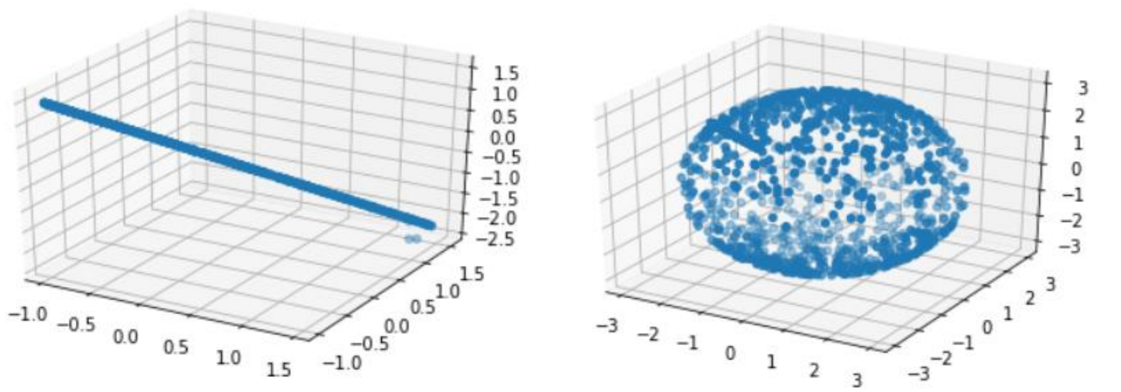

Figure 5. *(a) shows the set of line, (b) shows the rest. We could discover there appears to be some points from the sphere at the end of the line and there still exist a little part of line inside the sphere.*

I don't think this is a good method though we classify the line indeed. First, the line extends to its both ends at the same speed. Usually the initial part of line we detected does not lay at the middle of the whole object. In this case, we could not obtain the entire line. Second, the convergence speed is rather low. Third, this method works well for disjointed or almost disjointed datasets. For those with more complex overlapping relationship, it could not reach our expectation.

Part III Combination of two methods and final Success

After that, I notice that OWL regression could efficiently distinguish the line outside the sphere. After just 3 iterations of OWL regression(procedure in part I), I obtained a nearly ideal dataset for method in part II. Then I applied method in part II to this data set. After just 4 iterations, line and sphere were totally separated. The finally results shows below.

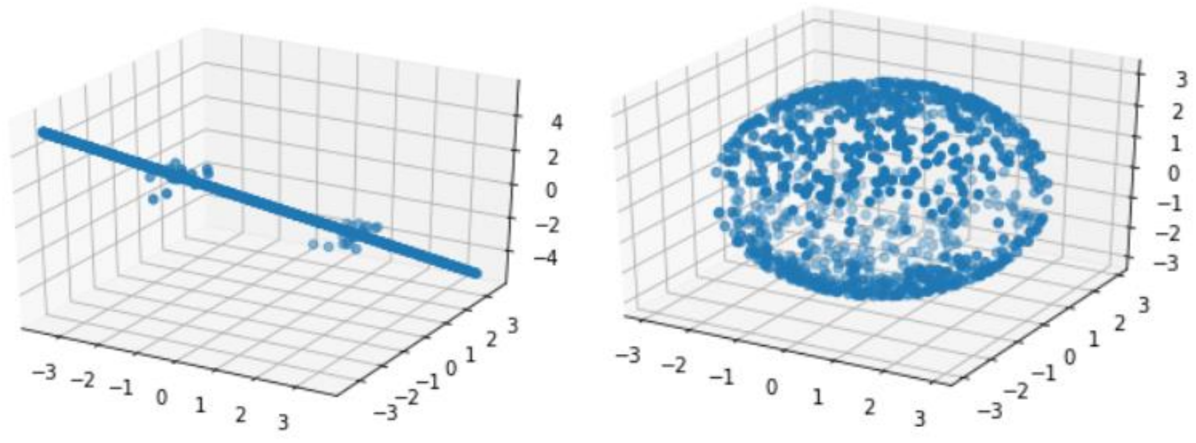

Figure 6. *(a) is the data set for line. There are only few noise points in this data set.*

This results look good. I finally detect the line and split it from the whole data set.

I would summarize this method and try different objects to see how to generalize it. This method is quite simple and only contends two simple algorithms. The computation is simple as well because of only few iterations.