Spectral Clustering Based on Local PCA and Applications

Jia Guo
Department of Mathematics
guojia@umich.edu

Masaya Tsukamoto
Department of Mathematics
masayats@umich.edu

Zihan Wang
Department of Climate and Space Science and Engineering
wzihan@umich.edu

Guangting Yu
Department of Mathematics
yugtmath@umich.edu

Abstract

We study a spectral clustering method based on local principal component analysis (PCA) (**?**). Instead of dealing with pair-wise distances between points like the previous algorithms do, this algorithm is able to resolve intersections. Especially, the new setting requires the surfaces to be smooth. Within this smoothness assumption, there is a mathematical theorem supporting the new algorithms. Furthermore, we apply this algorithm to autograding arithmetic problems.

1 Introduction

The recognition of handwritten digit strings is relevant in a variety of applications in our daily life. In post office, handwritten ZIP code can be recognized automatically. At primary school, the grading of arithmetic problems can be completed in one second with a smart phone application. However, there are still several challenges in this field. One bottleneck is related with the segmentation module, which segments a string of characters into individual one. The reason behind is that strings sometimes are not neatly written, e.g. overlapping, touching, and intersecting (**??**). Many algorithms have been proposed to deal with this problem. They can mainly be divided into two categories (**?**). The first one is segmentation-recognition, which means segmentation is completed before recognition. The second one is recognition based, where the algorithm yields a list of segmentation hypotheses and then assesses each of them through the recognition process. This method will take the information of the foreground and background into account. Recently, segmentation-free methods are also introduced (**?**).

Figure 1: Variability of connections between handwritten digits.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

In this project, the main goal is to deal with the intersections in the handwritten digit strings. Here we will focus on segmentation-recognition implementing the spectral clustering based on local principle component analysis (PCA) (**?**). A straightforward idea for segmentation is to utilize spectral clustering (**?**). However, a fatal drawback of spectral clustering is that it cannot separate two intersecting clusters. Thus, it cannot deal with the segmentation in handwritten digits. **?** claim that a multiway affinity is needed to capture complex structure in data (e.g. intersection) beyond proximity attributes. **?** first implemented subspace clustering based on local PCA. Later, **?** developed a spectral clustering method within a semi-supervised learning framework. Continuing this line of work, (**?**) suggest a spectral clustering method based on the estimation of the local linear structure (tangent bundle) via local PCA. There are also two concurrent publications (**??**) with quite similar methods.

The rest of the report is organized as follows. In Section 2, we will show authors' algorithms to implement spectral clustering based on local PCA. The rationale behind the algorithm will be briefly discussed. In Section 3, we will analyze the results of the algorithm. In Section 4, we will show one application of this algorithm. In Section 5, we will present the conclusion.

2 Methodology

There are four different algorithms on spectral clustering provided in the paper (**?**). Algorithm 1 gives the sketch of the standard spectral graph partitioning with given affinity matrix W. Different variants, Algorithm 2 and 3 are also proposed. One of the biggest defference is the affinity matrix $W_{ij} = \mathbf{1}_{\{||\boldsymbol{y}_i - \boldsymbol{y}_j|| \leq \epsilon\}} \mathbf{1}_{\{||\boldsymbol{C}_i - \boldsymbol{C}_j|| \leq \eta r^2\}}$ in Algorithm 2 and $W_{ij} = \mathbf{1}_{\{||\boldsymbol{y}_i - \boldsymbol{y}_j|| \leq \epsilon\}} \mathbf{1}_{\{||\boldsymbol{Q}_i - \boldsymbol{Q}_j|| \leq \eta\}}$ in Algorithm 3. They are kinds of "hard" versions and more tractable, where Theorem 1 introduced below holds. However, the author found Algorithm 4 performs better via numerical experiments. So we just provide Algorithm 4 below.

Input:
Data points $\boldsymbol{x}_1, ..., \boldsymbol{x}_n \in \mathbb{R}^D$; neighborhood radius $r > 0$; spatial scale $\epsilon > 0$; projection scale $\eta > 0$; intrinsic dimension d; number of clusters K.

Steps: (called "Algorithm 4" in the paper)

1. Pick one point $\boldsymbol{y}_1$ at random from the data. Pick another point $\boldsymbol{y}_2$ among the data points not included in neighborhood $N_r(\boldsymbol{y}_1)$, and repeat the process, selecting centers $\boldsymbol{y}_1, ..., \boldsymbol{y}_{n_0}$. Here we define the neighborhood $N_r(\boldsymbol{x}) = \{\boldsymbol{x}_j : ||\boldsymbol{x} - \boldsymbol{x}_j|| \leq r\}$ for any point $\boldsymbol{x} \in \mathbb{R}^D$ and $r > 0$, given a data set $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$.
2. For each i = 1, ..., n_0, compute the sample covariance matrix $\boldsymbol{C}_i$ of $N_r(\boldsymbol{y}_i)$. Let $\boldsymbol{Q}_i$ denote the orthogonal projection onto the space spanned by the top d eigenvectors of $\boldsymbol{C}_i$.
3. Compute the following affinities between center pairs:
$$W_{ij} = \exp\left(-\frac{||\boldsymbol{y}_i - \boldsymbol{y}_j||^2}{\epsilon^2}\right) \cdot \exp\left(-\frac{||\boldsymbol{Q}_i - \boldsymbol{Q}_j||^2}{\eta^2}\right) \tag{1}$$
4. (Spectral Graph Partitioning) Compute $\boldsymbol{Z} = (Z_{ij})$ according to $Z_{ij} = W_{ij}/\sqrt{\Delta_i \Delta_j}$, with $\Delta_i = \Sigma_{j=1}^n W_{ij}$. Extract the top K eigenvectors of Z. Renormalize each row of the resulting $n \times K$ matrix. Apply K-means to the row vectors.
5. The data points are clustered according to the closest center in Euclidean distance.

The main idea is that we first divide the original data points into small circles (or spheres) $N(\boldsymbol{y}_i)$ with the radius r. Then we grab the information about the direction of the local data arrangement in each circle through the sample covariance matrix $\boldsymbol{C}_i$ and extract the information as the projection matrix $\boldsymbol{Q}_i$. The affinities W_{ij} are set so that circles are classified as the same cluster if they are geometrically close and the local directions of the data arrangement are similar.

?? is the simplest example of the clustering by this algorithm ($K = 2$, $r = 15.0$, $\epsilon = 15.0$, $\eta = 0.5$). The cross-shaped data is seccessfully separated by the intersection. The overview of this algorithm is described intuitively in **??**.

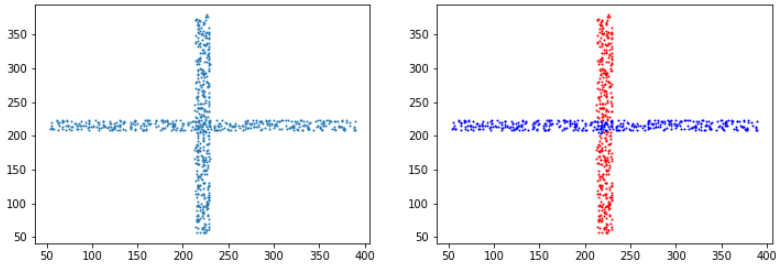


Figure 2: Clustering example

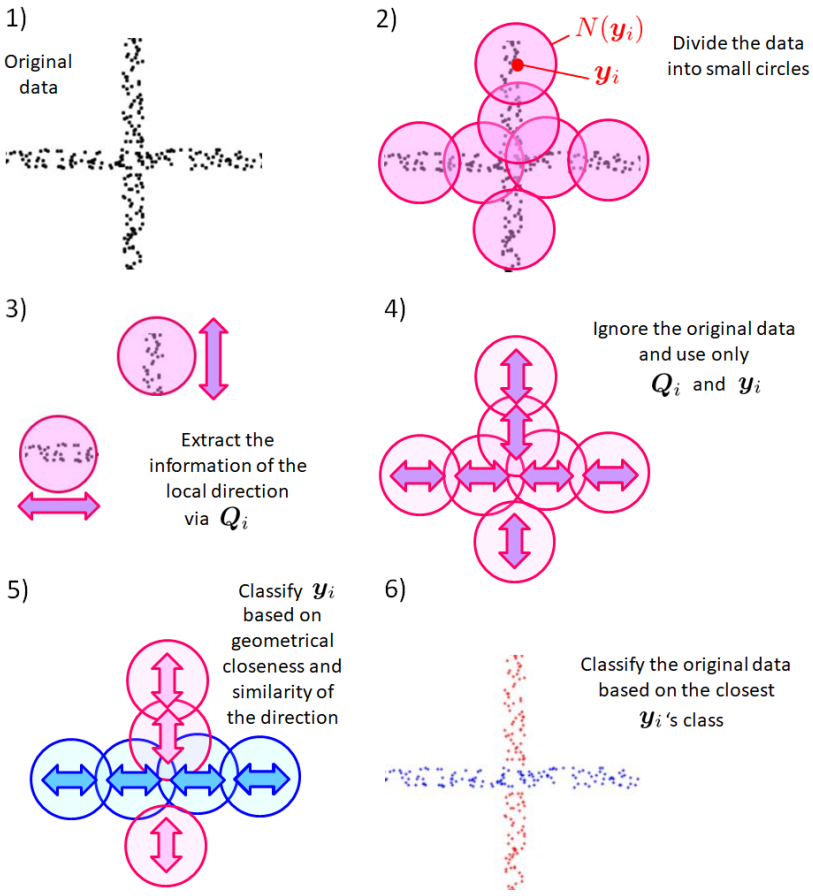


Figure 3: Overview of the Algorithm 4

3 Theoretical Analysis and Examples

While the analysis of Algorithm 4 seems within reach, there are some complications due to the fact that points near the intersection may form a cluster of their own. See **??**. The vertical line and ∞-shaped line are sccessfully separated by Algorithm 4 ($K = 2$, $r = 15.0$, $\epsilon = 15.0$, $\eta = 0.3$). But points near the intersections are fully classified as “Blue" group.

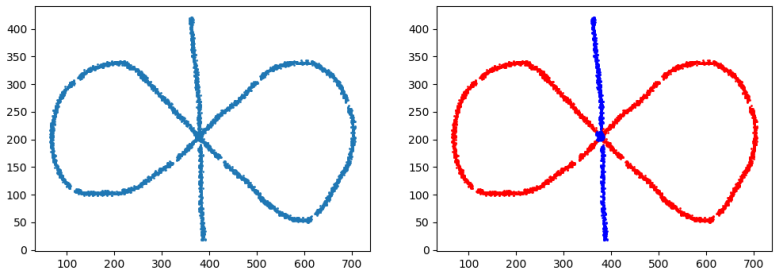


Figure 4: Example of intersection cluster

Instead, the authors mainly focus on some simpler variants described in Algorithm 2 and Algorithm 3, and then give a comment on the analysis of Algorithm 4. More theoretical results on intersecting manifolds refer to **?**,**?**,**?**. The generative model is a natural mathematical framework for multi-manifold learning where points are sampled in the vicinity of smooth surfaces embedded in Euclidean space. One can see in the numerical experiments that the Algorithm 4 fails to deal with phenomenon that the intersection with a sharp corner. See **??**. This is obtained by Algorithm 4 ($K = 3$, $r = 10.0$, $\epsilon = 10.0$, $\eta = 0.5$), and is a typical result of this algorithm, where “2" is separated by the bottomleft sharp corner.

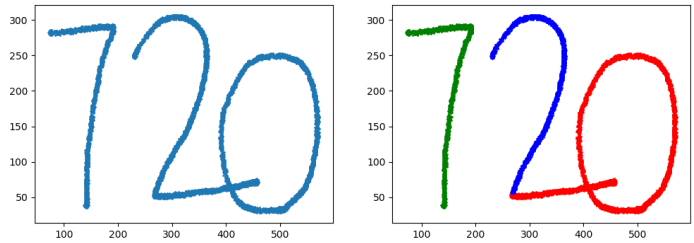


Figure 5: Failure at the corner

To clarify the characteristics of the algorithm, additional “720"-shaped data which were written intentionally smoothly are applied below. See **??**, both of which are obtained by Algorithm 4 with the parameters $K = 3$, $r = 10.0$, $\epsilon = 10.0$, $\eta = 0.5$. Due to smoothness, “720"s are successfully separated into 3 digits.

Precisely, $K = 2$ and each surface is a connected, C^2 and compact submanifold without boundary and of dimension d embedded in $\mathbb{R}^D$. Any such surface has a positive reach, which is what we use to quantify smoothness. The clusters are generated as follows. Each data point x_i is drawn according to

$$x_i = s_i + z_i$$

where s_i is from the uniform distribution over $S_1 \cup S_2$ and z_i is an additive noise term satisfying $||z_i|| \leq \tau$, where S_1 and S_2 represent two different clusters, and $\tau \geq 0$ is a non-negative constant. The main theorem of the paper is given below.

Theorem 1 *Consider two connected, compact, twice continuously differentiable submanifolds without boundary, of same dimension d, intersecting at a strictly positive angle, with the intersection set having strictly positive reach. Assume the parameters are set so that*

$$\tau \leq \frac{r\eta}{C}, \quad r \leq \frac{\epsilon}{C}, \quad \epsilon \leq \frac{\eta}{C}, \quad \eta \leq \frac{1}{C}$$

for large constant $C \geq 1$. Then with probability at least $1 - Cn\exp(-nr^d\eta^2/C)$:

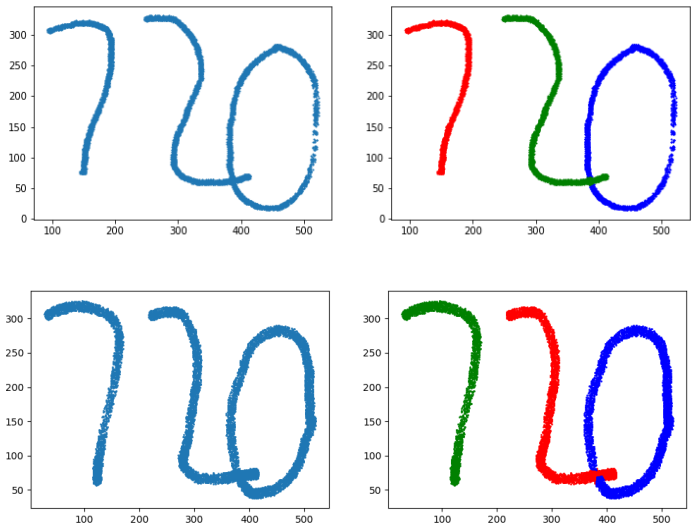


Figure 6: Successful results with smooth data

- *Algorithm 2 returns exactly two groups such that two points from different clusters are not grouped together unless one of them is within distance $C \cdot r$ from the intersection.*
- *Algorithm 3 returns at least two groups, and such that two points from different clusters are not grouped together unless one of them is within distance $C \cdot r$ from the intersection.*

Proof We give a sketch of proof of algorithm 2 and refers to the paper for more details. The most important notation is I^*, which indexes the points whose neighborhoods do not contain points from the other cluster.

1. Assume $\tau = 0$, and then explain for how things change for $\tau > 0$. Deriving a concentration inequality for local covariances with large constant C.
2. Then show that among points away from the intersection, those that are in the same cluster are neighbors in the graph if they are within distance ϵ, while those in different clusters cannot be neighbors in the graph.
3. Confirming that step 2 in algorithm 2 can eliminate all points which are not included in I^*, and the points removed are within distance $C \cdot r$ of the intersection, furthermore, points removed are within distance $C \cdot r$ of the intersection.
4. Concluding that in step 4, algorithm 2 returns two graphs, and each group covers an entire cluster except for points within distance $O(r)$ of the intersection.

■

To conclude, the key point of the approach is using principal component analysis to study the local behavior of data, and then use the information about affinity to generate a new local neighborhood graph. However, the selection of the parameters is quite sensitive to the final output, it is still a quite acceptable approach in many applications and perform well in particular example.

For comparison, results with different parameters are described in **??**, **??**, and **??**. If not explicitly mentioned, parameters $K = 3$, $r = 10.0$, $\epsilon = 10.0$, $\eta = 0.5$ are used.

When η is small, local directions of data tend to have a more important role in clustering than geometric closeness between data. In fact, in **??** [Left], only horizontal parts of “7" and “2" are labeled as green. On the other hand, in **??** [Right], the orthogonal intersection doesn’t work as a separating point between the vertical and the horizontal, and then a part of “2" is absorbed in “0".

r should be chosen so that small circles with radius r can appropriately cover the thickness of lines or surfaces of data points. If r is too small, the small circles may be buried in lines or surfaces. If r is

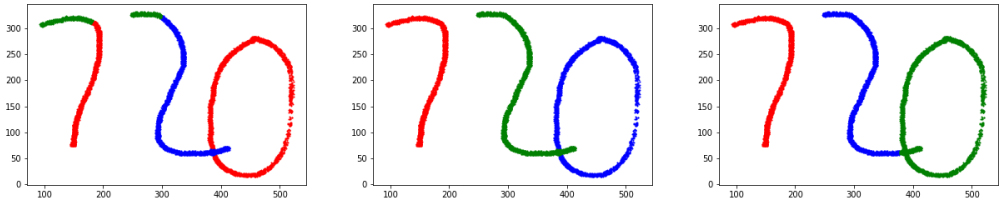


Figure 7: [Left]: $\eta = 0.1$, [Middle]: $\eta = 0.5$, [Right]: $\eta = 1.0$

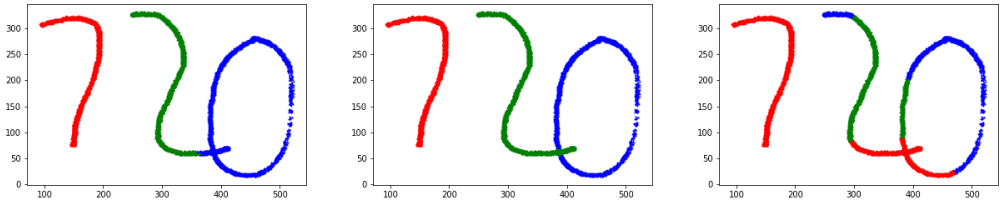


Figure 8: [Left]: $r = 3.0$, [Middle]: $r = 10.0$, [Right]: $r = 40.0$

too large, some circle may totally wrap around an intersection. In both cases, the classification will fail as described in **??** [Left] and [Right]. **??** shows an enlarged view of the boundary between the blue and green areas in **??** [Left]. A small green circle buried in the line can be seen here.

When ϵ is small, the affinity based on geometric distance decays immediately. Therefore it is natural that **??** [Left] has a mottled pattern. Conversely, when ϵ is large, the affinity remains higher over a wide distance.

4 Application: Arithmetics autograder

In the application of arithmetics autograder, the input image is ususally very large (typically around 3000×2000 pixels), among which 10% are black pixels. This means if we apply "Algorithm 4" directly, the input list of coordinates typically have length 10^5 to 10^6, which is too long for the algorithm to return the results in reasonable period of time. A more promising approach is to split the procedure into the following steps:

1. cluster the worksheet into equations
2. cluster the numbers and operators in each equation from step 1.
3. handwritten recognition

4.1 Equation clustering

One observation is that step 1 does not need accuracy in the numbers and operators, so that we can resample the input image so that the input for Algorithm 4 is significantly reduced (to around 10^3 coordinates of black pixels). We can see from **??** is blurred compared with the input **??**, and most of the equations are correctly clustered.

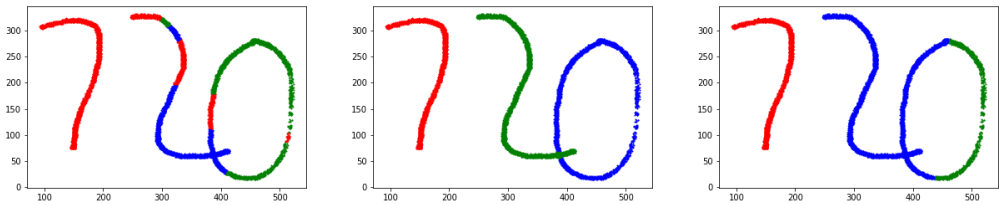
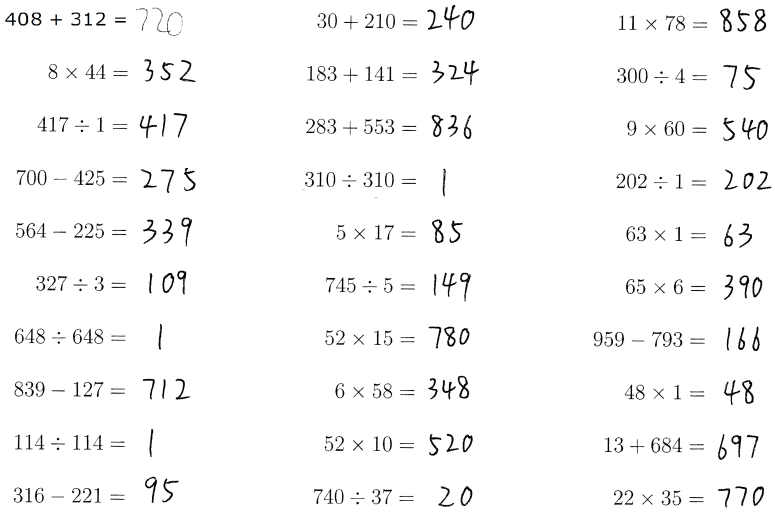
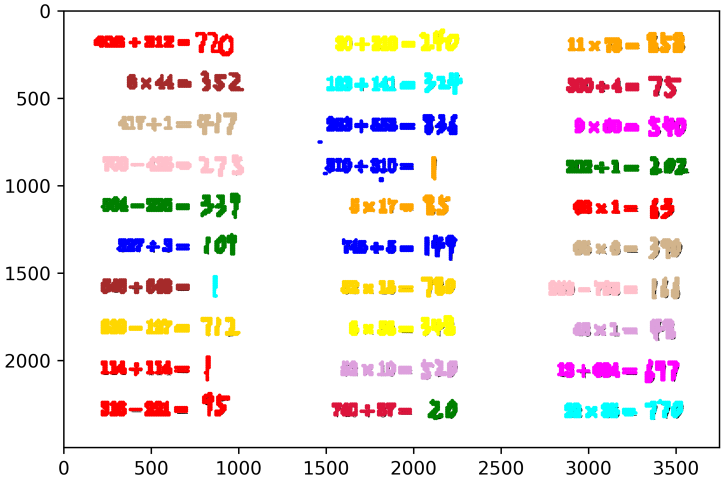


Figure 9: [Left]: $\epsilon = 3.0$, [Middle]: $\epsilon = 10.0$, [Right]: $\epsilon = 40.0$

Figure 10: Small green circle buried in the line



(a) Input image



(b) Equation clustering effect

Figure 11: Equation clustering by Algorithm 4

4.2 Numbers and operators clustering

We take the first equation “408 + 312 = 720" as an example and apply Algorithm 4 again. The main goal in this section is to divide the equation into separete digits and symbols. Especially we deal with the intersection between handwritten “2" and handwritten “0". See **??**.

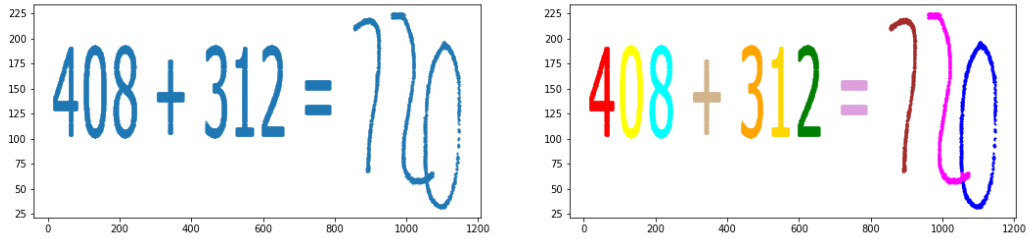


Figure 12: Equation clustering by Algorithm 4

4.3 Handwritten recognition

We implement handwritten recognition by using PCA. The results shows that we successfully recognize handwritten “7" and “2" but failed in handwritten “0". However, it quite depends on the database we use, MNIST. Handwritten digit recognition is a well-studied problem. Readers can use existing packages from open platform to handle this problem. Since this is an irrelevant problem of our main project, we leave it as a future work.

5 Conclusion

In this project, we study new algorithms based on local PCA. The main contribution behind this paper is that dealing with the situations where the surfaces intersect and a strong mathematical theorem guarantees the algorithms with high probability to succeed. However, smoothness assumption is crucial to the algorithm and does not always hold in applications. Another drawback behind the algorithm is that the points near the intersections may generate a new cluster of their own. But it is still a competitive method in spectral clustering, and we give an application in the real world to demonstrate the ability to handle the intersections.

6 Code and data

The Python codes and figure data in this report are uploaded in the below GitHub.

`https://github.com/masayats/EECS545_Fall2018_code_and_data/`

These codes are implemented from scratch by the team members. The main algorithm is written in “spectral_clustering_lpca.py", which is called in “generate_figures.ipynb", “generate_figures.py", and “application.py". In all_except_for_fig11 folder, “generate_figures.ipynb" and “generate_figures.py" are doing almost the same thing.

7 Acknowlegements

Jia Guo reads the proofs of theorems in the paper and conduct mathematical analysis on our implementation. **Masaya Tsukamoto** write the code of the algorithm and designed some testcases, including hand-written digit generation and smooth intersecting manifolds for separation. **Zihan Wang** writes the introduction and additional algorithm in the paper we study. **Guangting Yu** implements the arithmetic autograding based on the algorithm and compare with other existing algorithms. Every group member writes the report together and contributes to the project equally.