Study of Prediction on Manifolds with Almost No or No Labeled Data

Yuichiro Wada[1]

[1]Department of Computer Science and Mathematical Informatics

August 22, 2019
Supervisor : Professor Masahiko Sakai

Acknowledgements

First of everything, I must express my deep appreciation to my supervisor, Professor. Takafumi Kanamori. My research life was not easy because my topic changed from Master degree to Doctor degree and also lack of my attitude toward research activity. However, under Professor Kanamori supervision, I at least understood how independently and aggressively I had to progress. I am sure that this attitude is forever precious and beneficial for me.

I would also like to thank to my collaborator, Researcher Wataru Kumagai in RIKEN AIP. His knowledge about theory of machine learning always helped me a lot.

Moreover, I would like to thank other students, Professors and secretaries whom I met in Nagoya University and Tokyo Institute of Technology. Especially, thanks to Professor Masahiko Sakai, who helped me a lot during I am in Tokyo.

Finally, I must express my deep thanks to my parents who have supported me a lot.

Abstract

- **Problem Settings**: In this thesis, we investigate the following two fields: Online Graph-Based Semi-Supervised Learning (SSL) and Deep Clustering. In the former field, given a small size labeled dataset, the learning algorithm handles a continuous streamed unlabeled data points. The challenge is to predict the label of newly arrival data point quickly and precisely under sever memory constraints. Note that the distribution of streamed data may change as time goes. In the later field, given a large size unlabeled dataset and the number of clusters, the clustering algorithm estimates the cluster labels. A deep neural network is used to define the statistical model. The unlabeled data points are supposed to be generated from the same distribution independently. The challenge of this clustering is to estimate the cluster labels of given unlabeled dataset precisely as possible as we can.

- **Previous Methods**:

  – Online Graph-Based SSL: Online graph-based SSL is a relatively new filed of SSL studies. Although many online SSL algorithms have been proposed recently, most of them do not consider the processing time and severe memory constraints. Consequently, runtime and memory demands are increasing functions $\Omega(T)$ of streaming size T. On the other hand, an example of a few studies that account for processing time and memory constraints is online Quantized Label Propagation (QLP). This method is at each time, firstly to recompress the data adjacency graph by incorporating a newly arrived data point, then secondly to predict the label of new data point on the graph. The Doubling Algorithm (DA) and Label Propagation (LP) are employed as the graph compressing and label predicting methods, respectively. The other previous methods also take the same strategy, and both of them employ LP as the label predicting method. However, LP is known to not be robust against outliers. The reason is that LP predicts the labels by using all given data points, which often include outliers. In addition, LP is known to not be efficient computationally. As a possible alternate of LP, we can list Geodesic k-Nearest Neighbor (GkNN) algorithm. This method is not only computationally more efficient but also more robust against outliers than LP. The drawback of GkNN is that it does not perform well when the size of given labeled data is small.

  – Deep Clustering: Thanks to the development of deep neural networks, we can now handle large datasets with complicated shapes. Consequently, the studies of clustering using deep neural networks has been proposed. One major direction in the studies is to combine deep AutoEncoders (AE) with classical clustering methods such as k-means. This AE is used to obtain a clustering friendly low dimensional representation. In another major direction, the methods directly group a given unlabeled dataset into the given number clusters in the original input space by employing a deep neural network to their statistical models. Though most of deep clustering methods are built on the following fundamental two assumptions: the smoothness and manifolds assumptions, these methods require additional key conditions where the methods perform well. For an example, CatGAN (Categorical Generative Adversarial Networks) learns discriminative neural network classifiers that maximize mutual information between the input data points and the cluster labels, while en-

forcing the robustness of the classifiers to data points produced by adversarial generative models. Since maximizing mutual information implicitly encourages the cluster-balance distribution of the model to be uniform, if the distribution of cluster-balance with the given unlabeled dataset is not uniform, then CatGAN will not perform well. To the best of our knowledge, like the example, most of their key conditions are not realistic. One of few examples is SpectralNet, whose clustering logic is based on the above two fundamental assumptions. As for the weakness of SpectralNet, the performance is not robust against the existence of outliers. In the learning process, it learns the pairwise similarities over all given data points. Therefore, the existence of outliers disturbs the method to learn the similarities precisely, and thus it returns inaccurate cluster labels.

- **Our Research Ambitions, Proposed Methods and the Numerical Experiments**:
  - Online Graph-Based SSL: Our ambitions are, firstly, to invent a new label predicting offline SSL algorithm which can assist in creating more competitive online graph-based SSL algorithms. Our requirement toward the offline method is that the computational complexity should be as small as GkNN, and the predicting performance should be better than that of LP and GkNN. With the second ambition, after the invention of the offline SSL method, by combining the offline method and a conventional online clustering method, we then propose an online SSL method. As the result we could achieve the ambitions. Our proposed offline and online SSL methods are named Robust Label Prediction (RLP) and online Quantized RLP (online QRLP), respectively. The details of both methods are as follows.
    - RLP: This algorithm consists of three steps. On the basis of the neighbor graph, RLP first selects some unlabeled samples that represent the global structure of the data manifolds. The second step assigns labels to selected unlabeled samples by using LP. The third step predicts the labels on the remaining unlabeled samples by using GkNN. The unlabeled samples selected by the algorithm are collected into the hub dataset, which is denoted as H. The vertices selected for H are those with many neighbors on the data affinity graph G. As for the mechanism of RLP, we can see Fig.9 of this thesis.
    - Online QRLP: This method is obtained by combining DA and Quantized RLP, in which we conduct RLP on the compressed data affinity graph. DA is commonly used online clustering algorithm. In DA, given number of centroids, at each time, the set of centroids is updated. The set can be seen the set of important data points in the original data affinity graph defined up until the time. With regarding the hyperparameter tuning of online QRLP, we can tune them if we could have small size labeled and unlabeled datasets before the stream. The computational cost of this method can be controlled by upper-bounding the number of hub data points to match that of GkNN.
    - Numerical Experiments with online QRLP: Table 8 of this thesis shows the averaged prediction accuracy with the standard deviation on unlabeled data in eight real-world data streams. In each dataset, l denotes the number of labeled data obtained before the arrival of each stream of size T. As you can see, our method outperforms the previous popular online SSL method based on LP.

– Deep Clustering: Our ambition is to invent outlier-robust deep clustering which is built only on the two fundamental assumptions. As mentioned before, the previous methods require the additional key conditions apart from the two assumptions. Therefore, for the unknown unlabeled dataset, their performances are not promised. If we can invent such the method, it can be a good candidate in practical situation. As the result we could achieve the ambitions. The proposed method is named Spectral Embedded Deep Clustering (SEDC). Given an unlabeled dataset and the number of clusters, SEDC directly groups the dataset into the given number clusters in the input space. Our statistical model is the conditional discrete probability distribution, which is defined by a fully connected deep neural network. SEDC does not require key condition except the smoothness and manifold assumptions, and it can be applied to various data domains. Moreover, throughout our numerical experiments, we observed that our method was more robust against outliers than SpectralNet. The procedure of SEDC is composed of two stages. In the first stage, we conduct Spectral Clustering (SC) only on the unlabeled data points selected from high density region by using the geodesic metric to estimate the cluster labels. This selected data points exactly equal to hub data points, which was defined in online SSL study. This special type of SC is named as Selective Geodesic Spectral Clustering (SGSC), which we propose for assisting SEDC as well. Thereafter, we conduct SSL to train the model by using the estimated cluster labels and the remaining unlabeled data points. Note that, in this SSL, we treat the estimated cluster labels of the selected unlabeled data points as the given true cluster labels. At last, by using the trained model, we obtain the estimated cluster labels of all given unlabeled data points. In the following, firstly, let us explain the detail of SGSC, then SEDC.

  * SGSC: The motivation behind SGSC is to assist the semi-supervised learning in SEDC. SGSC conducts SC only on hub (selected unlabeled) data points with the geodesic metric, then returns the estimated cluster labels of hub points. The hub points are defined as data points in high density region, and these data points are approximated by the highest degree nodes on the affinity data graph. The geodesic metric is approximated by the graph shortest path distances on the graph. Empirically speaking, the estimation accuracy of cluster labels with hub points tends to not only be robust against the existence of outliers but also be competitive. This tendency can help SEDC return competitive clustering result. The reason of robustness is that the selection of data points from high density region tends to not be affected by the existence of outliers. The reason to employ the geodesic metric is that the metric is known to be useful to capture the structure of the data manifolds especially when the number of given data points is large. Fig.10 of this thesis is image with mechanism of SGSC.
  * SEDC: Given an unlabeled dataset X and the number of clusters C, SEDC optimizes the objective function of Eq.(35) of this thesis to train the statistical model, which is the conditional discrete probability distribution parameterized by a fully connected deep neural network. In this objective function, the first, second and third loss are the Virtual Adversarial Training (VAT) loss, pseudo empirical loss and the conditional Shannon entropy loss, respectively. With the definitions of symbols in the objective, $\mathbf{x}_{(i)}$, θ and h mean the element of unlabeled dataset, the parameters in the deep neural network and the number of

hub data points. The $q_{(i)}$ means the estimated distribution of cluster labels with hub data point $\mathbf{x}_{(i)}$. This estimated distribution is a part of outputs of SGSC. The minimization of VAT loss imposes the model to follow the smoothness assumption. The minimization of the third term imposes the model to have the large margin between the clusters. After this optimization, we obtain the estimated cluster labels of all given unlabeled data points, which are computed by the trained statistical model.

  - Numerical Experiments with SEDC: Table 16 of this thesis shows the averaged estimation accuracy of cluster labels with all given unlabeled data points. Inside of () shows the standard deviation. Seven times experiments are conducted for the average and the standard deviations. k-means to SEDC are the name of clustering methods. MNIST to TR are the name of datasets. Our proposed method is SEDC. As you can see, our method averagely outperforms the other clustering methods. As for the MNIST dataset, IMSAT performed pretty well since the dataset satisfy the key condition.

- **Conclusions and Future Works**:

  - Online Graph-Based SSL: We proposed a generic graph-based SSL algorithm, called RLP. We confirmed that RLP is robust against noisy data and provides more accurate predictions than LP and GkNN. The computational efficiency of RLP matches that of GkNN. Furthermore, we confirmed the power of RLP as a core technique in the online SSL framework. In the online scenario, the proposed method has two tunable hyperparameters, namely then number of neighbor and hub data points. Future works should focus on the choice of number of hub data points. In this thesis, the upper bound of number of hub points was determined by considering the computational cost. The prediction accuracy when the number is based on other criteria should also be examined. Furthermore, an adaptive method that determines both hyperparameters would be useful for practical online learning.
  - Deep Clustering: In this thesis, we propose a deep clustering method named SEDC. Given an unlabeled dataset and the number of clusters, the method groups the dataset into the given number clusters. Regarding its advantages, it does not require an additional key condition except two fundamental assumptions: smoothness and manifolds assumptions. In this point, only SpectralNet is comparable. In addition, SEDC also can be applied to various data domains since it does not have preferred data domains, as long as raw data is transformed to feature vectors. Furthermore, the performance of SEDC can be robust against existence of outliers unlike SpectralNet. According to these advantages, our proposed method can be expected to averagely perform better than previous deep clustering methods. As a result, this expectation is empirically confirmed by conducting numerical experiments on five commonly used datasets: see Table 16. Therefore, we think our method can be a competitive candidate for users in some practical clustering scenarios where prior knowledge of the given unlabeled dataset is limited. Let us then discuss two limitations of SEDC. On the one hand, since the method needs hyperparameter tuning, if we do not have appropriate labeled source domains to learn them from and transfer, then it may fail. On the other hand, since the method requires the number of clusters, it does not work for datasets where nothing is known on the number of clusters such as

genome datasets. Finally, we discuss about our two future works. The first one is to invent a more noise-robust semi-supervised learning framework and then apply it to SEDC instead of the above objective function. Since some of the estimated cluster labels by SGSC are not perfectly accurate, we need to invent such the framework to stabilize the performance of SEDC. The second one is to modify our method for handling structured data, i.e., graph data or sequential data.

Contents

1 Introduction

1.1 Background

Machine Learning (ML) is an important subject to automatize many simple tasks. Thus, it has the potential to revolutionize human life. One of the most commonly studied scenarios in ML is Supervised Learning (SL). In this scenario, a labeled dataset is used for the learning. Although SL is well studied both practically and theoretically, obtaining labeled dataset is often very expensive. The reason is that the labels are mostly annotated by human labour.

To solve above annotation problem, two alternative frameworks available that are more cost-efficient are Semi-Supervised Learning (SSL) and Unsupervised Learning (UL). In SSL, the given information is a small labeled dataset and a large unlabeled dataset. Let us assume the two datasets follow the same distribution. By using both labeled and unlabeled datasets, we estimate the labels of the given unlabeled dataset or define a classifier which can predict the labels of a newly arrived data. On the other hand, in UL, only the unlabeled dataset is available. The goal of UL depends on the tasks. In one case, the aim is to obtain the better representation than the raw data itself for assisting additional learning. This type of UL is known as representation learning. In other case, the goal is to group the dataset into the clusters based on some measure, which is named as clustering. Even though we have almost no information on the labels in both learnings, we still can propose well performing algorithms. This confidence is based on the following two fundamental assumptions;

- Smoothness assumption: If two data points are close, then so should be the corresponding labels.
- Manifold assumption: The high dimensional data line roughly on a low-dimensional manifold.

By using these two assumptions, many previous popular SSL and UL methods were proposed.

Recently, thanks to the development of Deep Neural Networks (DNNs), we can handle large and non-linear datasets, which we often encounter in ML scenarios. DNNs are not only known to be scalable but also to have strong expression power. The scalability comes from Stochastic Gradient Descent (SGD) methods. The expression power comes from the deeply stacked layers. Therefore, it is natural that huge amount of DNNs based SSL or UL methods have been proposed since then. Note that DNNs does not perform well if the given number of data points are small or the distribution of the dataset will change as time goes.

In this thesis, we investigate the following two fields: *Online Semi-Supervised Learning* (Online SSL) and *Deep Clustering*, and then propose two corresponding methods. Online SSL is a part of SSL studies. The learning algorithms do not handle batch dataset but a continuous streamed dataset. The typical challenge is to predict the label of newly arrival data point quickly and precisely under sever memory constraints. In some cases, the distribution of that streamed dataset changes as time goes. Deep clustering is a part of clustering studies, which is a part of UL. The clustering methods employ DNNs in their statistical models. The challenge of this clustering is to estimate the cluster labels of given unlabeled dataset precisely. The regularization of DNNs by using unlabeled data points is also inevitable challenge. Note that online SSL methods and deep clustering methods can help each other since non-DNNs based online SSL methods can handle steamed dataset whose distribution will change as time goes. Fig.1 shows a

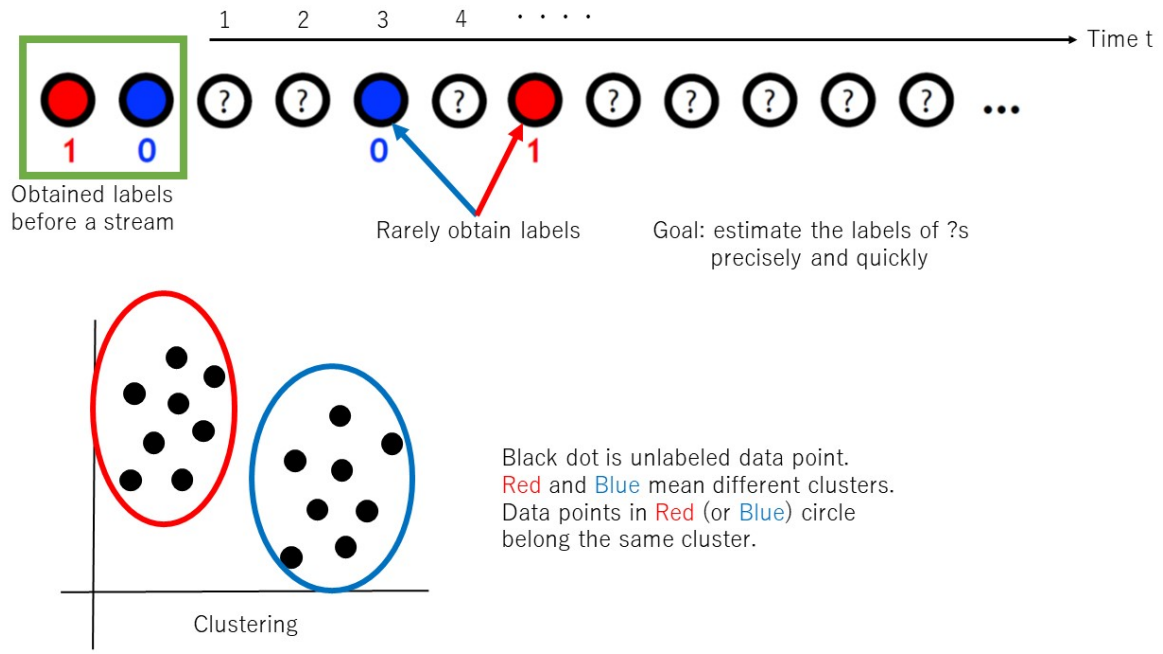

Figure 1: A typical example of online SSL (top) and clustering (bottom). Top picture: At each time t, by using given labeled and unlabeled data points, we try to estimate the label of newly arrived data point $\mathbf{x}_t$. Red and blue colored circles mean differently labeled data points. Question marked circles are unlabeled data points. Bottom picture: By using only unlabeled data points, we try to estimate the cluster labels of the given unlabeled data points, which are expressed by vectors of $\mathbb{R}^2$. It depends on the task whether the number of clusters is given or not. In this case, we are given that number, which is two. Based on the number and some measure, each unlabeled data point is assigned the cluster label.

typical image of online SSL and clustering tasks. In the online SSL task, our goal is to quickly estimate the labels of question marked data points, whcih are unlabeled data points, by using the given labeled and unlabeled data points. We assume the distribution of those data points may change as time goes. In addition, it is also assumed that we can not register all data points in our memory space. Therefore, under sever memory constrain, we have to achieve precise estimation. On the other hand, in the bottom pictured clustering task, by using given unlabeled data points, we simply try to estimate the cluster labels precisely as possible as we can. In this task, we have the number of clusters that is two. It is assumed that all data points are generated from the same distribution independently.

1.2 Main Contributions

In this thesis, we propose two methods. One is an online SSL method, and the other is a deep clustering method. Empirically, both methods could outperform the existing state-of-the-art methods in terms of prediction or estimation accuracy with given unlabeled data points. The key of that success is that we take more advantage of the preceding two fundamental assumptions than the previous methods.

1.2.1 Part I : Robust Label Prediction via Label Propagation and Geodesic k-Nearest Neighbor on Online Semi-Supervised Learning

In this part, we propose a computationally efficient offline SSL algorithm that yields a more accurate prediction than Label Propagation (LP), which is commonly used in online graph-based SSL. This method is named *Robust Label Prediction* (RLP), and is made by modifying the LP algorithm and combines it with the Geodesic k-Nearest Neighbor (GkNN) algorithm. Since this proposed method is an offline method that is intended to assist online graph-based SSL algorithms, we then propose an online graph-based SSL algorithm named *Online Quantized Robust Label Prediction* (Online QRLP). The efficacy of above both offline and online SSL algorithms are demonstrated in numerical experiments.

The remainder with this part is organized as follows. Section 2 and 3 briefly introduces existing SSL studies and our research motivations. Section 4 proposes a learning method for SSL and analyzes its time complexity. Section 5 shows the utility of our algorithm in building an efficient online algorithm for SSL. Section 6 is devoted to numerical experiments. In this section, we show that our method has better prediction accuracy with efficient runtime than some existing methods. Section 7 contains concluding remarks.

1.2.2 Part II : Spectral Embedded Deep Clustering

In this part, we propose a new clustering method named *Spectral Embedded Deep Clustering* (SEDC). Given an unlabeled dataset and the number of clusters, our method directly groups the dataset into the given number clusters in the original space. Our statistical model is the conditional discrete probability distribution, which is defined by a deep neural network. Our clustering strategy is, first to estimate the cluster labels of unlabeled data points selected from high density region, and then to conduct semi-supervised learning to train the model by using the estimated cluster labels and the remaining unlabeled data points. At last, by using the trained model, we obtain the estimated cluster labels of all given unlabeled data points. The advantage of our method is that it does not require key condition. For example, the previous deep neural network based clustering methods require the cluster-balance of given dataset to be uniform. Moreover, it also can be applied to various data domains as long as the data is expressed by a feature vector. In addition, it was observed that our method was robust against outliers. Therefore, the proposed method is expected to averagely perform better than previous methods. This expectation is empirically confirmed by conducting numerical experiments on five commonly used datasets.

In the remainder of this paper, we introduce related works in Section 9. We then introduce our proposed method in Section 10. Finally, we demonstrate the efficiency of our method with numerical experiments in Section 11.

Table 1: Abbreviations and commonly used notations in this thesis.

SSL	Semi-Supervised Learning
RLP	Robust Label Prediction
(Online) QRLP	(Online) Quantized RLP
SGSC	Selective Geodesic Spectral Clustering
SEDC	Spectral Embedded Deep Clustering
IMSAT	Information Maximizing Self-Augmented Training
VAT	Virtual Adversarial Training
SC	Spectral Clustering
$\mathbf{x}, y$	Feature vector and its class or cluster label
l, u	The number of labeled and unlabeled data points.
T	The size of streamed data.
C	The number of classes or clusters
D	The dimension of feature vector
d, d_G	A metric and geodesic metric on graph G
k	The number of neighbors
k_v	#Samples in the majority vote of GkNN
n_c	Maximum number of nodes in compressed graph
$m, \mathbf{m}$	Multiplicity and its vector representation
H, h	Hub set and its size
$\mathbf{W}, \mathbf{L}$	The similarity matrix and its graph Laplacian matrix

Part I
Robust Label Prediction via Label Propagation and Geodesic k-Nearest Neighbor on Online Semi-Supervised Learning

2 Preliminary with Semi-Supervised Learning

SSL involves both labeled and unlabeled data in the learning process. Given that this learning paradigm can solve many real-world problems, it has been intensively studied in recent years [1]. In this section, we introduce both previous offline and online SSL studies.

2.1 Offline Semi-Supervised Learning

Regarding with offline SSL, there are two types: transductive and inductive learning. Though these two types use both labeled and unlabeled dataset, the goals are different. The goal of the former is to only estimate labels of unlabeled dataset precisely, that of the later on the other is to predict the labels of test data points. We here introduce the previous popular transductive and inductive learning. Finally, let us define the shared definitions in both learnings. The labeled dataset and the unlabeled dataset are denoted by $L = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and $U = \{\mathbf{x}_i\}_{i=l+1}^{l+u}$, respectively. $\mathbf{x}_i \in \mathbb{R}^D$ is the feature vector, and $y_i \in \{1, ..., C\}$ is its class label. The total sample size $l + u$ is denoted by n. Let us define the set $L_{\mathbf{x}}$ as the feature vectors in L, i.e., $L_{\mathbf{x}} = \{\mathbf{x}_i\}_{i=1}^{l}$, and $\widehat{U}$ as the estimated unlabeled dataset, i.e., $\widehat{U} = \{(\mathbf{x}_i, \hat{y}_i)\}_{i=l+1}^{l+u}$, where $\hat{y}_i$ is the estimated label of $\mathbf{x}_i$.

2.1.1 Generative Models

Given L and U, by assuming that each class has some distribution such as Gaussian [2], this method finds out the decision boundary. This is categorized as inductive learning. The statistical model is defined as follows:

$$p_\theta(\mathbf{x}, y) = p_\theta(y)p_\theta(\mathbf{x}|y), \tag{1}$$

where θ, $\mathbf{x}$ and y are a set of model parameters, feature vector and its class label, respectively. The decision boundary $f(\mathbf{x})$ is defined by

$$f(\mathbf{x}) = \underset{1\le j\le C}{\operatorname{argmax}} p_\theta(y = j|\mathbf{x}),\ p_\theta(y|\mathbf{x}) = \frac{p_\theta(\mathbf{x}, y)}{\sum_{j'=1}^{C} p_\theta(\mathbf{x}, y' = j')}. \tag{2}$$

The parameters θ are trained based on the following criterion:

$$\underset{\theta}{\operatorname{argmax}} \log \left\{ \prod_{i=1}^{l} p_\theta(\mathbf{x}_i, y_i) \prod_{i=l+1}^{n} p_\theta(\mathbf{x}_i) \right\}. \tag{3}$$

Note that we can rewrite $p_\theta(\mathbf{x}_i) = \sum_{j=1}^{C} p_\theta(y = j)p_\theta(\mathbf{x}|y = j)$. The solution of Eq.(3) is obtained by, for an example, the Expectation-Maximization (EM) algorithm [3]. That solution

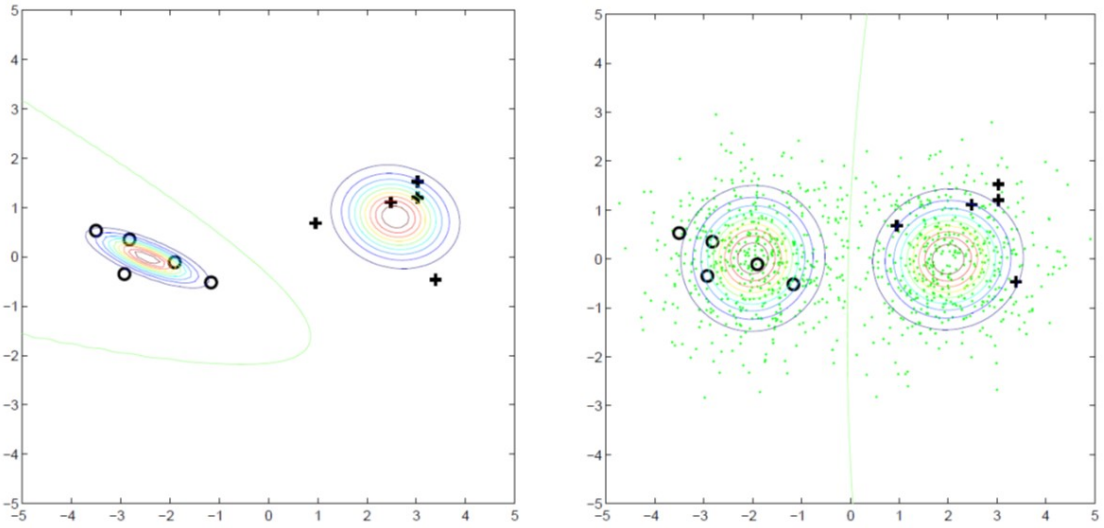

Figure 2: An example of SL (left) vs SSL (right) [4]. In both learnings, the statistical models are defined by Gaussian mixture model whose components are two. Given ten labeled data points (circle and cross symbols), the classifier in left picture is defined by SL. In the right picture, by using both the labeled dataset and given unlabeled dataset, the classifier is defined by SSL.

is guaranteed to be at least a local optimum. Variational approximation also can be helpful to maximize Eq.(3). The right picture of Fig.2 shows the example of trained classifier by SSL where the model is two components Gaussian. Since the true distribution is included in the prameterized model, by incorporating U, the right pictured classifier is improved from the left pictured classifier trained by SL with same model.

The advantages of generative models are that, firstly, it is well-studied framework, then secondly, it can be very effective if the assumed distribution is correct. The disadvantages are that, first, it is difficult to verify the correctness of the model, then secondly, commonly used maximizer named EM algorithm only guarantees the local optima, third, the given unlabeled dataset may degrade the trained model. To avoid such the degrading, the following criterion can be alternative of Eq.(3):

$$\underset{\theta}{\operatorname{argmax}} \log \left\{ \prod_{i=1}^{l} p_\theta(\mathbf{x}_i, y_i) \left(\prod_{i=l+1}^{n} p_\theta(\mathbf{x}_i) \right)^{\lambda} \right\}, \tag{4}$$

where λ is a hyperparameter to down weight with the unlabeled dataset. This hyperparameter ranges 0 to 1.

2.1.2 Semi-Supervised Support Vector Machines

This method is also named S3VMs, and categorized to inductive learning. Inspired by SVMs, it is designed for the output classifier to try to separate the given unlabeled data points from different classes with large margin by using both L and U. To make explanation simplified,

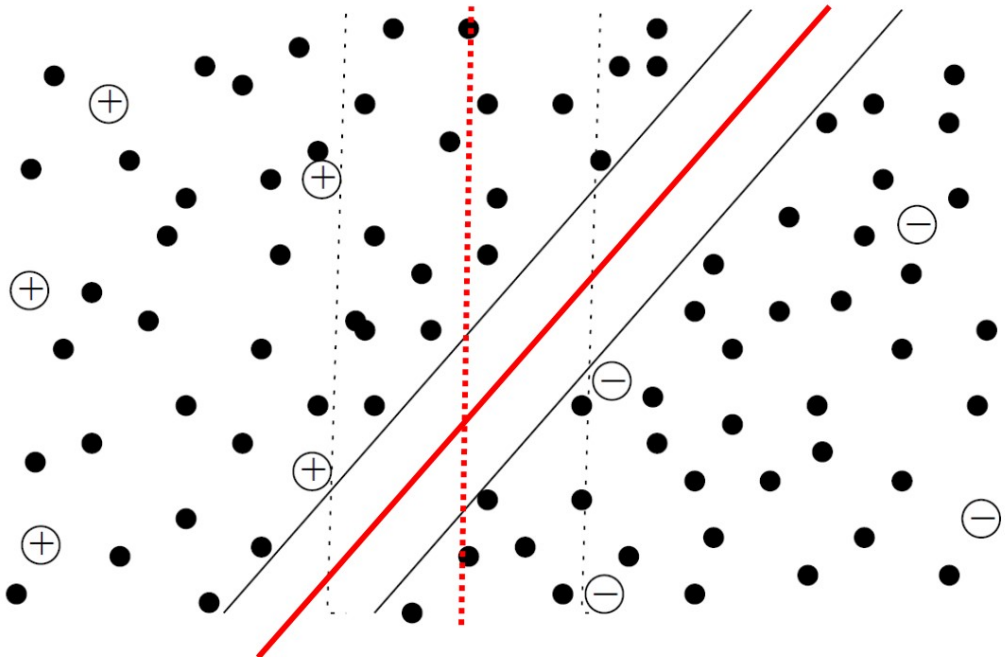

Figure 3: An example of SVM vs S3VM [4]. The black dots mean the given unlabeled data points. The plus and minus symbols mean the given labeled data points. The dotted and solid red lines are classifiers returned by SVM and S3VM, respectively.

let us consider two classes classification problems, i.e., $y \in \{+1, -1\}$. Denote the reproducing Hilbert kernel space induced from a kernel K by $\mathcal{H}_K$. The statistical model is now defined by $f(\mathbf{x}) = h(\mathbf{x}) + b$, where $h \in \mathcal{H}_K$ and $b \in \mathbb{R}$. Then, the objective function to obtain the best hypothesis is defined as follows:

$$\underset{f}{\operatorname{argmin}} \left\{ \frac{1}{l} \sum_{i=1}^{l} (1 - y_i f(\mathbf{x}_i))_+ + \lambda_1 \|h\|^2_{\mathcal{H}_K} + \frac{\lambda_2}{u} \sum_{i=l+1}^{n} (1 - |f(\mathbf{x}_i)|)_+ \right\} \text{ s.t. } \frac{1}{u} \sum_{i=l+1}^{n} f(\mathbf{x}_i) = \frac{1}{l} \sum_{i=1}^{l} y_i, \tag{5}$$

where λ_1 and λ_2 are the hyperparameters ranging $\mathbb{R}_+$. The first term is empirical loss with labeled dataset based on hinge loss function: $(z)_+ = \max\{z, 0\}$. The minimization of second term makes the hypothesis h separate the training dataset with large margin. The third term prefers the unlabeled data points located outside the margin. With respect to the constrain, it is used to avoid the following solutions: most given data points have same label. In other words, the constrain make the hypothesis have the same class balances with the labeled and unlabeled datasets. For the prediction of label with a new data point $\mathbf{x}$, we will use $\hat{y} = \text{sign}(f^*(\mathbf{x}))$ where f^* is the solution of Eq.(5). Fig.3 shows the comparison between SVM and S3VM. The dotted and solid red lines express the classifiers trained by SVM and S3VM, respectively. In the figure, the unlabeled dataset succeeded to help improving the trained classifier from that of SVM.

Regarding with the optimization of Eq.(5), we have faced the challenge since the objective is non-convex. Therefore, there are different optimization approaches such as SVM$^{\text{light}}$ [5], ∇S3VM [6], deterministic annealing [7], CCCP [8], Branch and Bound [9], SDP convex relaxation [10], etc.

As for the advantages of S3VMs are that, first, the applicability wherever SVMs can be

applied, secondly, the clear mathematical framework. The disadvantages are that, firstly, the difficulty of optimization, i.e., only obtaining bad local optima.

2.1.3 Manifold Regularization

This method [11, 12] is designed for the output classifier to satisfy the smoothness assumption while separating data points from different classes with large margin. The method is categorized to inductive learning. Denote the reproducing Hilbert kernel space induced from a kernel K by $\mathcal{H}_K$, agian. The statistical model is defined by $f(\mathbf{x}) \in \mathcal{H}_K$. For the simplicity, let us below consider two classes classification problems, i.e., $y \in \{+1, -1\}$. Then, given L, U, a kernel K and hyperparameters $\lambda_1, \lambda_2 > 0$, the objective function to obtain the best model is defined as follows:

$$f^* = \underset{f \in \mathcal{H}_K}{\operatorname{argmin}} \left\{ \frac{1}{l} \sum_{i=1}^{l} \left(f\left(\mathbf{x}_i\right) - y_i\right)^2 + \lambda_1 \|f\|^2_{\mathcal{H}_K} + \frac{\lambda_2}{n^2} \mathbf{f}^{\mathrm{T}} \mathbf{L} \mathbf{f} \right\}, \tag{6}$$

where $\mathbf{f} = \left[f\left(\mathbf{x}_1\right), \ldots, f\left(\mathbf{x}_n\right)\right]^{\mathrm{T}}$ and $\mathbf{L}$ is the graph Laplacian on the data affinity graph: see the detail in sub-subsection 3.2.1. Regarding with above optimization problem, the representaer theorem gives us the form of the optimal solution as follows:

$$f^*(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i^* K\left(\mathbf{x}_i, \mathbf{x}\right). \tag{7}$$

By using this fact, we will solve the following problem with $\alpha = [\alpha_1, \ldots, \alpha_n]^{\mathrm{T}}$:

$$\alpha^* = \underset{\alpha \in \mathbb{R}^n}{\operatorname{argmin}} \left\{ \frac{1}{l} (\mathbf{Y}_l - \mathbf{J}\mathbf{K}\alpha)^{\mathrm{T}} (\mathbf{Y}_l - \mathbf{J}\mathbf{K}\alpha) + \lambda_1 \alpha^{\mathrm{T}} \mathbf{K} \alpha + \frac{\lambda_2}{n^2} \alpha^{\mathrm{T}} \mathbf{K}\mathbf{L}\mathbf{K}\alpha \right\}, \tag{8}$$

where $\mathbf{Y}_l$, $\mathbf{J}$ and $\mathbf{K}$ is the vector of given labels, $n \times n$ block matrix $\begin{bmatrix} \mathbf{I}_l & \mathbf{0} \\ \mathbf{0} & \mathbf{0}_u \end{bmatrix}$ and the Gram matrix, respectively. The optimize α^* can have the closed form as follows:

$$\alpha^* = \left(\mathbf{J}\mathbf{K} + \lambda_1 l \mathbf{I}_n + \frac{\lambda_2 l}{n^2} \mathbf{L}\mathbf{K} \right)^{-1} \mathbf{Y}_l. \tag{9}$$

For the prediction of label with a new data point $\mathbf{x}$, we will use $\hat{y} = \operatorname{sign}(f^*(\mathbf{x}))$.

Though Eq.(6) employs squares loss function, by replacing the loss with the Hinge loss: $(z)_+ = \max\{z, 0\}$, we can obtain the objective function of Laplacian Support Vector Machines (LapSVMs) as follows:

$$f^* = \underset{f \in \mathcal{H}_K}{\operatorname{argmin}} \left\{ \frac{1}{l} \sum_{i=1}^{l} \left(1 - y_i f(\mathbf{x}_i)\right)_+ + \lambda_1 \|f\|^2_{\mathcal{H}_K} + \frac{\lambda_2}{n^2} \mathbf{f}^{\mathrm{T}} \mathbf{L} \mathbf{f} \right\}. \tag{10}$$

We can again have the closed optimized solution of Eq.(10) by using the representer theorem of Eq.(7). The closed solution of dual problem with Eq.(10) is expressed by

$$\alpha^* = 2 \left(\lambda_1 \mathbf{I}_l + \frac{\lambda_2}{n^2} \mathbf{L}\mathbf{K} \right)^{-1} \mathbf{J}^{\mathrm{T}} \mathbf{Y}_l \beta^*, \tag{11}$$

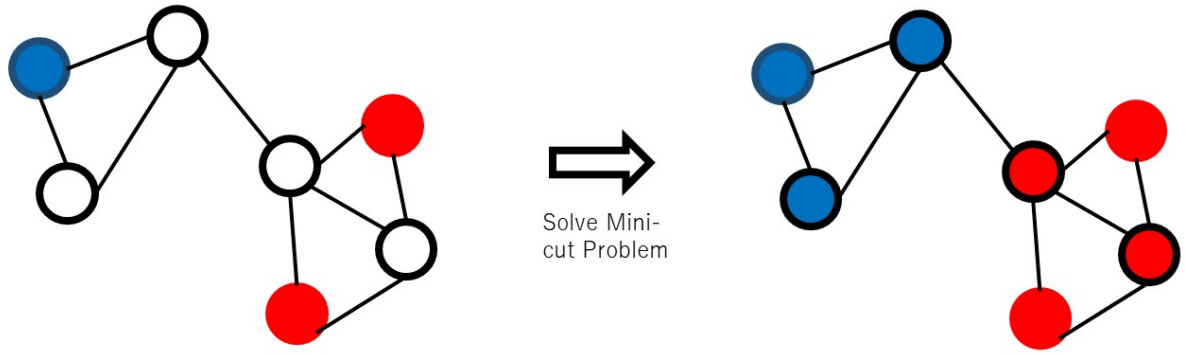

Figure 4: An example of solution by mini-cut method. The left graph represents the neighborhood relationship between given each feature vector. The node represents the feature vector. The red or blue color means the labeled data. No color means the unlabeled data. The undirected edge mean that the two corresponding nodes are in neighbor relationship. Based on this data affinity graph, the labels of unlabeled data points are estimated by minimizing the objective function of mini-cut method. The result of this estimation is shown in the right picture.

where β^* is obtained by solving the following problem:

$$\beta^* = \mathrm{argmax}_{\beta \in \mathbb{R}^l} \left\{ \sum_{i=1}^{l} \beta_i - \frac{1}{2} \beta^{\mathrm{T}} \mathbf{Q} \beta \right\} \quad \text{s.t.} \quad \sum_{i=1}^{l} \beta_i y_i = 0 \ \& \ 0 \leq \beta_i \leq \frac{1}{l} \ (i = 1, \ldots, l), \tag{12}$$

where $\mathbf{Q}$ is defined by

$$\mathbf{Q} = 2\mathbf{Y}_l \mathbf{J} \mathbf{K} \left(\lambda_1 \mathbf{I}_l + \frac{\lambda_2}{n^2} \mathbf{L} \mathbf{K} \right)^{-1} \mathbf{J}^{\mathrm{T}} \mathbf{Y}_l. \tag{13}$$

The advantages of this method are that, first, its clear mathematical framework, then secondly, strong performance if the constructed graph fits the task. The disadvantage is the sensitivity with graph construction. For an example, if the given dataset includes many outliers, then the graph may fail to fit the task.

2.1.4 Transductive Learning

In transductive learning, given L and U, the goal is to estimate the labels of given unlabeled data points, i.e., $\hat{U}$. Most of transductive learning studies employ the data affinity graph, whose node set is $L_{\mathbf{x}} \cup U$. The set of edges is often defined by k-Nearest Neighbor (k-NN). The edges sometimes can be defined between all nodes. The weight on the edges are usually defined

by weight decay with Euclidean distance. We here introduce one popular method named mini-cut [13]. Another popular methods such as Label Propagation and Geodesic k-Nearest Neighbor (GkNN) are introduced in sub-section 3.2.

Let the labels of given data points range 0 or 1. The idea of mini-cut is to minimize $\sum_{1\leq i,j\leq n} w_{ij}\left|y_i - y_j\right|$ while fixing the labels of given labeled data points. This is equivalently solving the following optimization problem:

$$\min_{\hat{\mathbf{Y}}\in\{0,1\}^n}\left\{\infty\sum_{i=1}^{l}(y_i-\hat{y}_i)^2+\sum_{1\leq i,j\leq n}w_{ij}\left(\hat{y}_i-\hat{y}_j\right)^2\right\}, \tag{14}$$

where $\hat{\mathbf{Y}} = [f(\hat{y}_1,\ldots,\hat{y}_n)]^{\mathrm{T}}$. Though this problem is combinatorial problem, there is an algorithm to solve it with the polynominal time. Fig.4 shows the result of mini-cut method. The graph used in the figure is constructed by k-NN manner with $k = 2$, and then the graph is converted into the undirected version. The similarity on the edge is defined by $w_{ij} = \exp\left(-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2/\sigma^2\right)$.

The advantage is that if the constructed graph fits the task, the performance is well. In addition, this framework can potentially handle the streamed dataset whose distribution may change as time goes easier than inductive framework. The disadvantage is that we maybe pay additional graph constructing time cost when handling a newly arrived data point.

2.1.5 Semi-Supervised Learning with Deep Neural Networks

2.2 Online Semi-Supervised Learning

In this learning scenarios, we consider the case in which a few manually labeled data points are provided before the arrival of continuously streamed unlabeled data points. The goal of this study is to predict the label of the newly arrived data point correctly and quickly under severe memory constraints. Note that the data generating distribution may change as time goes. These problems often occur in the real world [15, 16, 14], and are handled by online graph-based SSL algorithms. Fig.5 shows two examples of applications with online graph-based SSL algorithms. The task of top picture is, at each time, to estimate the label of 'car' or 'no-car'. The task of left one is to classify irregular heartbeats on electrocardiogram at each time.

Online graph-based SSL is a relatively new filed of SSL studies. Although many online SSL algorithms have been proposed recently, most of them [17, 18] do not consider the processing time and severe memory constraints. Consequently, runtime and memory demands are increasing functions $\Omega(T)$ of streaming size T. On the other hand, an example of a few studies that account for processing time and memory constraints is online Quantized Label Propagation (QLP) [19]. This method is at each time, firstly to recompress the data adjacency graph by incorporating a newly arrived data point, then secondly to predict the label of new data point on the graph. The Doubling Algorithm (DA) [20] and Label Propagation (LP) [21, 22] are employed as the graph compressing and label predicting methods, respectively. The other methods [23, 14] also take the same strategy, and both of them employ LP as the label predicting method.

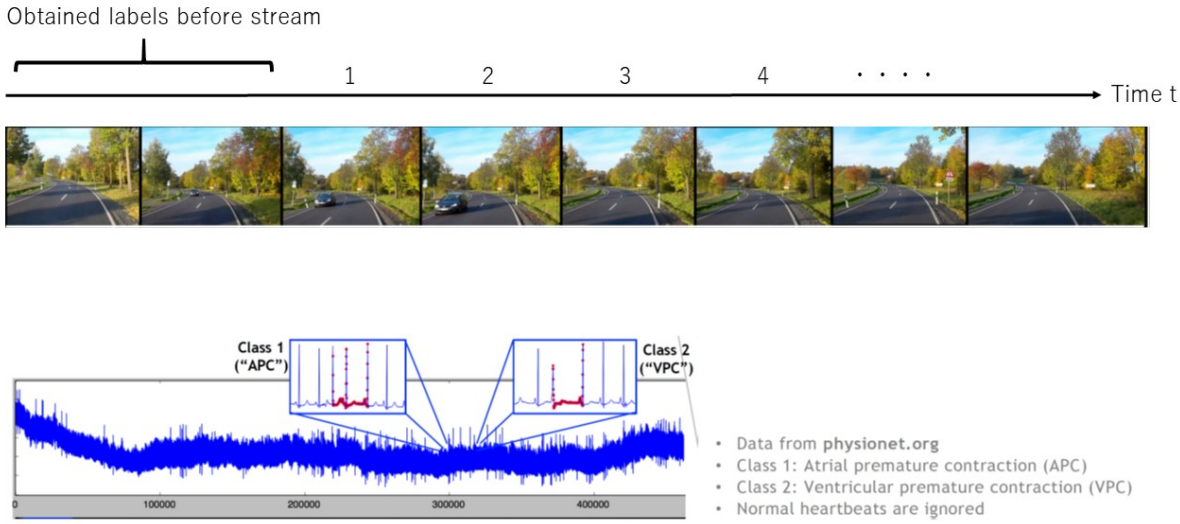

Figure 5: Two examples of applications with online SSL [14]. Top picture: By using small labeled dataset obtained before a streamed data, at each time t, we would like to know whether there exist cars in-vehicle camera or not, i.e., binary classification between 'car' and 'no-car'. The challenge is that the distribution of the streamed unlabeled data points may change as time goes. This type of problem is often handled by online graph-based SSL algorithms since the framework is suitable for data points generated from the time-dependent distribution. Bottom picture: Classification of irregular heartbeats on electrocardiogram. There are two classes: Atrial premature contraction (APC) and Ventricular premature contraction (VPC). By using small labeled dataset obtained before the streamed data, at each time t, we would like to classify the two irregular heartbeats. Note that normal heartbeats are ignored. This problem can be again handled by online graph-based SSL algorithms.

3 Motivation and Related Works with Proposed Methods

3.1 Motivation

Our ambitions are, firstly, to invent a new label predicting offline SSL algorithm which can assist in creating more competitive online graph-based SSL algorithms. As mentioned in subsection 2.2, LP is commonly employed as the core tool in the online graph-based SSL algorithms. However, LP is known to not be robust against outliers. The reason is that LP predicts the labels by using all given data points, which often include outliers. In addition, LP is known to not be efficient computationally. As a possible alternate of LP, we can list GkNN algorithm [24, 25]. This method is not only computationally more efficient but also more robust against noisy data than LP. The drawback of GkNN is that it does not perform well when the size of given labeled data is small. Therefore, our minimum requirement toward the offline method is that the computational complexity should be as small as GkNN, and the predicting performance should be better than that of LP and GkNN. With the second ambition, after the

invention of the offline SSL method, by combining the offline method and a conventional online clustering method, we then propose an online SSL method. The proposed offline and online SSL methods are named Robust Label Prediction (RLP) and online Quantized RLP (online QRLP), respectively.

3.2 Related Works with Proposed Methods

We here introduce the three popular SSL algorithms, i.e., Label Propagation (LP) [21, 22], GkNN [24, 25], and online quantized LP (online QLP) [19]. The first two methods are mainly used in offline settings, whereas the last third method is adopted in online learning. At last, let us show the problem setting of offline SSL again. In this learning, the labeled dataset $L = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and the unlabeled dataset $U = \{\mathbf{x}_i\}_{i=l+1}^{l+u}$ are observed, where $\mathbf{x}_i \in \mathbb{R}^D$ is the feature vector, and $y_i \in \{1, ..., C\}$ is its label. The total sample size $l + u$ is denoted by n. Let us define the set $L_{\mathbf{x}}$ as the feature vectors in L, i.e., $L_{\mathbf{x}} = \{\mathbf{x}_i\}_{i=1}^{l}$, and $\widehat{U}$ as the predicted dataset over U, i.e., $\widehat{U} = \{(\mathbf{x}_i, \hat{y}_i)\}_{i=l+1}^{l+u}$, where $\hat{y}_i$ is the predicted label of $\mathbf{x}_i$. The goal is to obtain the accurate prediction $\widehat{U} = \{(\mathbf{x}_i, \hat{y}_i)\}_{i=l+1}^{l+u}$ over U.

3.2.1 Label Propagation

LP attempts to propagate the label information along with the data adjacency graph, which is often defined by the k-NN graph. Fig.6 shows the example of procedure with LP algorithm. The pseudo algorithm of LP is outlined below.

1. Define the weighted directed graph G, which comprises a set of vertices $L_{\mathbf{x}} \cup U$ and a set of directed edges E defined by k-NN on the basis of a metric d. Suppose that each edge $e_{ij} \in E$ has a non-negative weight w_{ij}.

2. Denote the $n \times n$ graph Laplacian $\mathbf{D} - \mathbf{W}$ on G by $\mathbf{L}$, where $\mathbf{W} = (w_{ij})_{i,j}$, and $\mathbf{D}$ is the diagonal matrix whose entries are given by $d_{ii} = \sum_j w_{ij}$.

3. Compute $\mathbf{Y}_u$ as follows: let $\mathbf{Y}_l$ be the $l \times C$ matrix whose (i, j) element is one for $y_i = j$ and zero otherwise. Then, $\mathbf{Y}_u = (Y_{is})$ is given by

$$\mathbf{Y}_u = \mathbf{L}_{uu}^{-1}\mathbf{W}_{ul}\mathbf{Y}_l, \tag{15}$$

   where the matrices with the subscript u or l denote the block matrix corresponding to unlabeled or labeled data.

4. Estimate the labels of the unlabeled data by $\hat{y}_i = \underset{1 \le s \le C}{\operatorname{argmax}}\ Y_{is}$.

The label prediction $\widehat{U}$ by the LP is denoted by $\widehat{U} = LP(L, U, d, k, w)$. In our numerical experiments, the metric d was assumed as the commonly used Euclidean metric.

Remark 1. *The weight is commonly defined by the Gaussian kernel,*

$$w_{ij} = \begin{cases} \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2), & e_{ij} \in E, \\ 0, & e_{ij} \notin E, \end{cases} \tag{16}$$

The bandwidth σ is selected by the median or mean heuristics [26].

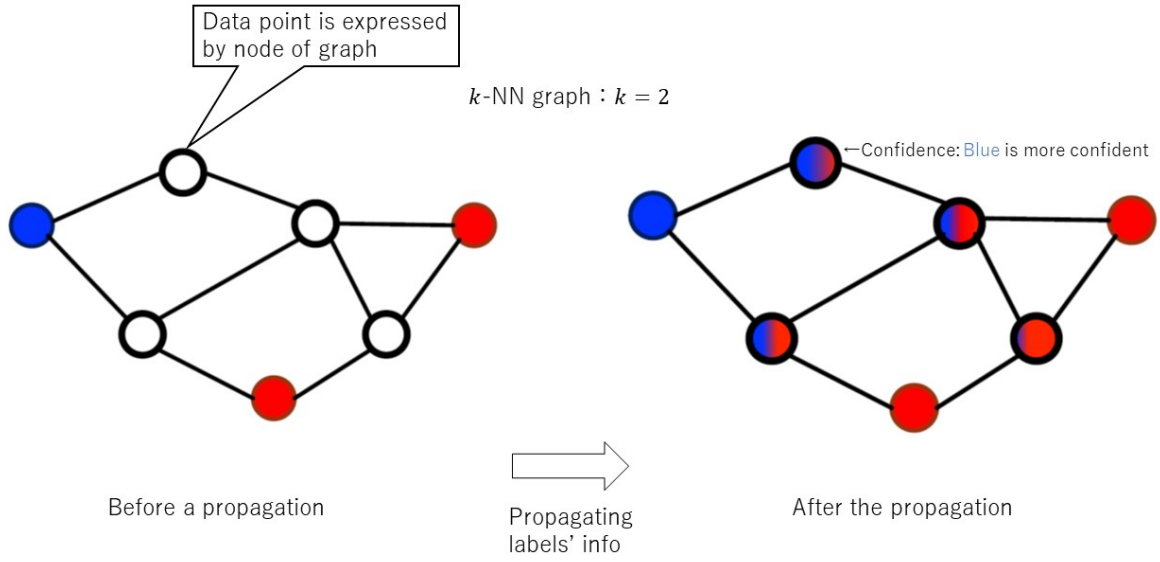

Figure 6: An example with procedure of label propagation algorithm. Left picture: By using given labeled and unlabeled feature vectors, the affinity graph representation is shown. The nodes correspond feature vectors. The colored node means labeled data. The non-colored node means unlabeled data. The edges are defined based on k-NN manner ($k = 2$) with Euclidean metric. Right picture: After the graph construction, the label information of labeled data points are propagated on the graph. As the result, each unlabeled data point have the discrete distribution with the class.

Remark 2. *A class of extended Gaussian kernels is also employed in graph-based methods; see [27, 28].*

Let us consider the time complexity of LP. Suppose that the metric d is the Euclidean metric and a Euclidean distance is computed in $O(D)$ time, where D is the dimension of the feature vector. Thereafter, for the weight w defined by the Gaussian kernel, the time complexity is

$$O\left(Dn^2 + u^3\right). \tag{17}$$

The first and second terms in Eq.(17) are contributed by the brute-force construction of the k-NN graph [29] and the calculation of the $u \times u$ inverse matrix in (15), respectively. The edge weights w_{ij} must be properly defined in LP because they largely affect the final prediction accuracy [30]. In practice, the edge weights are computed by rather simple strategies such as Eq.(28). However, the time complexity given by (17) is a serious issue, particularly when $l \ll u$.

3.2.2 Geodesic k-Nearest Neighbor

In GkNN with $k = k_v$, the label is predicted by majority voting among the geodesic k_v-nearest labeled neighbors on the weighted data adjacency graph, whose vertices are $L_{\mathbf{x}} \cup U$. Given that only the observed data are available, the exact geodesic distance on the true data manifold cannot be computed. Therefore, the geodesic distance should be approximated. Fig.7 shows

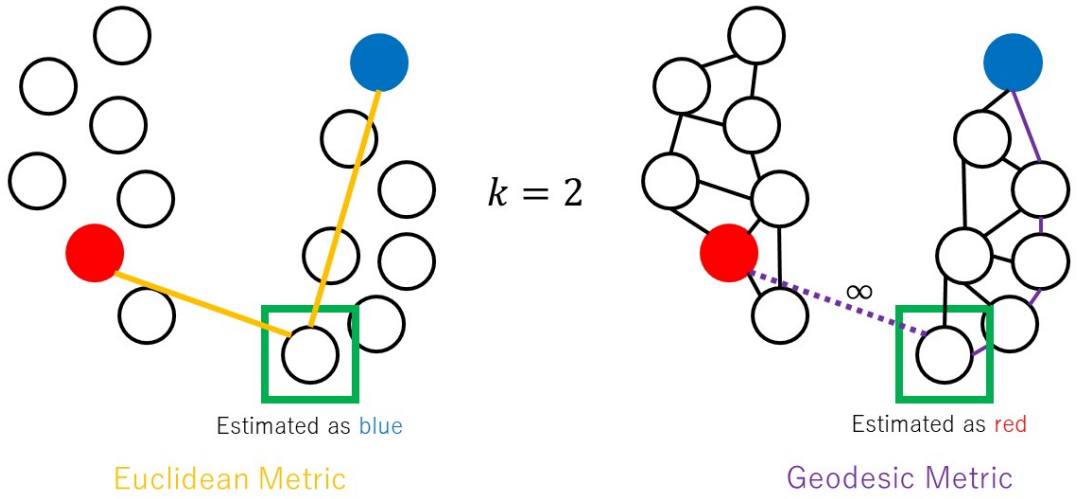

Figure 7: An example of k-NN (left) vs GkNN (right), where $k = 2$ in both methods. Left picture: Let us consider the label estimation problem with green rectangled node. Since the original k-NN employs Euclidean metric, the label of that node is estimated as red label. Right picture: Consider the same estimation problem of left picture. Since GkNN employs the geodesic metric, the label of that node is estimated as blue label. The geodesic metric is approximated by the graph shortest path distance in the data affinity graph.

the comparison between k-NN and GkNN. This figure shows that GkNN performs better than k-NN if the dataset follows smoothness and manifold assumptions.

Let us consider the time complexity of the GkNN. Assume the graph is defined by k_1-NN on the basis of the Euclidean metric. When the geodesic distance is approximated by the shortest path distance on the graph, the time complexity is expressed as follows:

$$O\left(Dn^2 + k_v(\log n + k_1)n\right), \tag{18}$$

where the cost of computing a Euclidean distance is $O(D)$. The first and second terms in (18) are contributed by the brute-force construction of the k_1-NN graph [29] and by algorithm 1 of [25], respectively. In this case, GkNN is computationally more efficient than LP. Moreover, the hyperparameter k_v can be efficiently chosen by heuristics [24, 25].

In our numerical experiments, we employ the k-NN graph and Euclidean metric for computational efficiency.

3.2.3 Online Quantized LP

Online QLP [19] combines label prediction by QLP with an online graph compression method called the doubling algorithm (DA) [20]. The QLP is conducted on the compressed graph.

The purpose of DA is to construct a compressed graph from the original neighbor graph G_0, where each node in G_0 corresponds to a point in Euclidean space. The DA algorithm is detailed

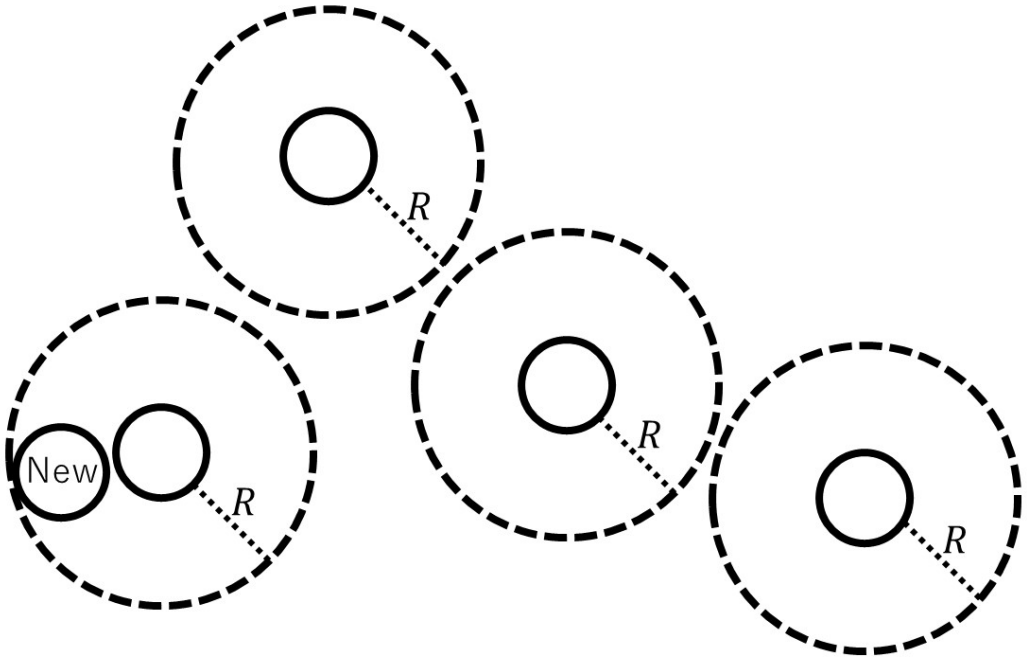

Figure 8: Image of DA algorithm procedure, where $n_c = 4$. At time t, a new data point arrives. If the data point falls within the R-radius hypersphere of one of four centroids, the new data point then is merged to that centroid. Otherwise, the radius R is doubled: $R \leftarrow 2R$. Then, the five data points which are the four centroids and one new data point are checked whether some point can be merged to one of them or not. This will be repeated until some point is merged.

in [20]. Let us now define the compressed graph G of G_0. The node in the compressed graph is called the centroid, and is chosen from the nodes in G_0. The centroid v has a vertex multiplicity that is defined as the number of nodes represented by v. More precisely, the multiplicity is the number of nodes that fall in the ball at center v with a predefined radius R. Each node in G_0 is arranged to contribute to the multiplicity of only one centroid. Hence, the sum of the multiplicities equals the total number of nodes in G_0. This is again mentioned in Definition 2.

Let us consider the online QLP of an online data stream. Let C_{t-1} be the set of centroids at time $t-1$, and $R_{t-1} > 0$ be the radius at time $t-1$. The maximum size of the compressed graph is denoted as n_c. Suppose that a new data point $\mathbf{x}_t$ is observed at time t. The vertex multiplicities at time $t-1$ is compiled into a vector $\mathbf{m}_{t-1}$ with the dimension $|C_{t-1}|$. The DA algorithm is the iterative function

$$(C_t, \mathbf{m}_t, R_t) = DA(\mathbf{x}_t, C_{t-1}, \mathbf{m}_{t-1}, R_{t-1}, n_c).$$

The size of C_t is bounded above by n_c.

When an unlabeled data $\mathbf{x}_t$ is observed at time t, DA provides the compressed graph. The QLP algorithm then predicts the label of $\mathbf{x}_t$ on the compressed graph. In label prediction by QLP, the multiplicities are incorporated into a weight matrix $\hat{\mathbf{W}}$ as detailed in [19].

Let us consider the time complexity of online QLP. At each time, the time complexity of DA [20] based on the Euclidean distance is bounded by

$$O\left(Dn_c \log n_c\right). \tag{19}$$

The time complexity of QLP [27] is the time complexity of LP (defined by (17)) over the compressed graph. Therefore, the worst-case time complexity of QLP is $O\left(Dn^2 + n_c^3\right)$, where $n = l + n_c$. By summing up both time complexities, the time complexity of online QLP at each time is bounded by $O\left(Dn^2 + n_c^3\right)$. Note that when deriving the above complexity, we assumed only unlabeled data in the data stream.

The disadvantage of online QLP is non-robust against outliers [27].

4 Proposed Methods

4.1 Robust Label Prediction

We now propose our label prediction method, which is called *robust label prediction* (RLP). Our algorithm consists of three steps. On the basis of the neighbor graph, RLP first selects some unlabeled samples that represent the global structure of the data manifolds. The second step assigns labels to selected unlabeled samples by using LP. The third step predicts the labels on the remaining unlabeled samples by using GkNN.

The unlabeled samples selected by the algorithm are collected into the *hub* dataset, which is denoted as $H \subset U$. The vertices selected for H are those with many neighbors on the graph G. The hub dataset H is formally defined below.

Definition 1. *Let L and U be the labeled and unlabeled sets. On the graph $G = (L_{\mathbf{x}} \cup U, E)$, let $\mathcal{N}_j$ be the set of adjacent nodes of $\mathbf{x}_j \in L_{\mathbf{x}} \cup U$. Suppose that the multiplicity $m(\mathbf{x}_i)$ is assigned to each node $\mathbf{x}_i$ in G. Let us define $|\mathcal{N}_j|_0$ as the total multiplicity at $\mathbf{x}_j$, i.e.,*

$$|\mathcal{N}_j|_0 = \sum_{\mathbf{x}_i \in \mathcal{N}_j} m(\mathbf{x}_i). \tag{20}$$

For a natural number h, the hub set H is defined as the collection of nodes that ranked in the top-h total multiplicity in U. They are arranged in descending order of $|\mathcal{N}_j|_0$.

In the above definition, generally $|\mathcal{N}_j|_0$ is not the cardinality of $\mathcal{N}_j$ over G, but the cardinality of the neighborhood over the uncompressed graph G_0 in practice.

Algorithm 1 and Fig.9 show the pseudo code and working mechanism of RLP, respectively. The details of the algorithm are given below.

- Line 1: Given L and U, construct a directed weighted graph G by k_1-NN with a Euclidean metric. k_1 is set to a small number such as three, four or five. The graph construction uses all feature vectors, $L_{\mathbf{x}} \cup U$ to capture the global structure of the data manifolds.

- Line 2: Remove outliers prior to LP in line 5. By definition, the hub set H is expected to exclude outliers by the appropriate setting of h. The appropriate number of h is obtained by cross-validation.

- Line 3: When approximating the geodesic distance on true data manifolds by the shortest path distance on G, the directed graph G should be converted to an undirected graph.

- Line 4: Define the geodesic metric d_G. The distances are determined from the Euclidean distances defined on the edges of G. However, among all geodesic distances, we need only to compute the distances required in line 5 and 7. The efficient algorithms are available for this purpose [31, 25].

Algorithm 1 : RLP

Input: Labeled and unlabeled data sets L and U. Number of neighbors k_1, k_2. Size of the hub dataset h. Size of the majority vote k_v.

1: Construct the directed graph $G = (L_{\mathbf{x}} \cup U, E)$, where E is defined by k_1-NN with the Euclidean distance.
2: Build the hub dataset H on the graph G such that $|H| = h$. Thereafter, define $\bar{H}$ by $U \setminus H$.
3: Replace all directed edges in G with undirected ones.
4: Define the geodesic metric d_G by computing the shortest path distance on graph G.
5: Estimate the labels of H as follows: Construct $\widehat{H}$ by LP, by computing $LP(L, H, d_G, k_2, w_2)$, where the weight $w_2(\mathbf{x}_i, \mathbf{x}_j)$ is defined by $\exp(-d_G(\mathbf{x}_i, \mathbf{x}_j)^2/\sigma^2)$.
6: $L \leftarrow L \cup \widehat{H}$.
7: Estimate the labels of $\bar{H}$ by implementing GkNN on G with $k = k_v$ and the labeled set L. Let $\widehat{\bar{H}}$ be the labeled set of $\bar{H}$.

Output: $\widehat{H} \cup \widehat{\bar{H}}$.

- Line 5: Estimate the labels of the non-outliers (i.e., the elements in H) by running LP on $L_{\mathbf{x}} \cup H$. The small k_1 and Euclidean distance are useful for approximating the geodesic distances between nodes in high-density regions, but the nodes in $L_{\mathbf{x}} \cup H$ are sparse on $L_{\mathbf{x}} \cup U$. Thus, we vary the number of neighbors k_2 and the geodesic metric d_G rather than fixing the number of neighbor k_1 and the Euclidean metric. The appropriate number of k_2 is obtained by cross-validation.

- Line 6: Augment the labeled dataset by $L \cup \hat{H}$. Considering that $\hat{H}$ should be a well estimated set, the updated L is expected as a reliable labeled dataset.

- Line 7: Predict the labels on the remaining unlabeled data set $\bar{H}$ by GkNN by using the updated labeled data set $L \cup \widehat{H}$. The updated labeled set is expected to stabilize the prediction performance of GkNN. Recall that the geodesic distance was calculated in line 4. Here, the GkNN is employed for two reasons. First, GkNN runs faster than LP on $\bar{H}$, particularly when $|\bar{H}| \gg 1$. Second, in our observations, GkNN predicted the labels of $\bar{H}$ more robustly than LP.

Remark 3. *For $h = u$ (resp. $h = 0$), RLP is reduced to the LP with geodesic distance d_G (resp. GkNN).*

Remark 4. *In line 1 and 4 of algorithm 1, the geodesic distance d_G is defined by the k-NN graph and the Euclidean metric (because this pair is commonly used). However, when defining d_G, we can apply several graph construction methods with different metrics depending on the given dataset; see [24, 25].*

Remark 5. *GkNN is more robust than LP. The non-hub dataset $\bar{H}$ may include noisy data or outliers. In the LP algorithm, which predicts the label of a point that uses all data points, the outliers may degrade the prediction accuracy of all data in $\bar{H}$. On the contrary, GkNN uses only the k adjacent labeled points in the prediction, thus locally limiting the influence of the outliers.*

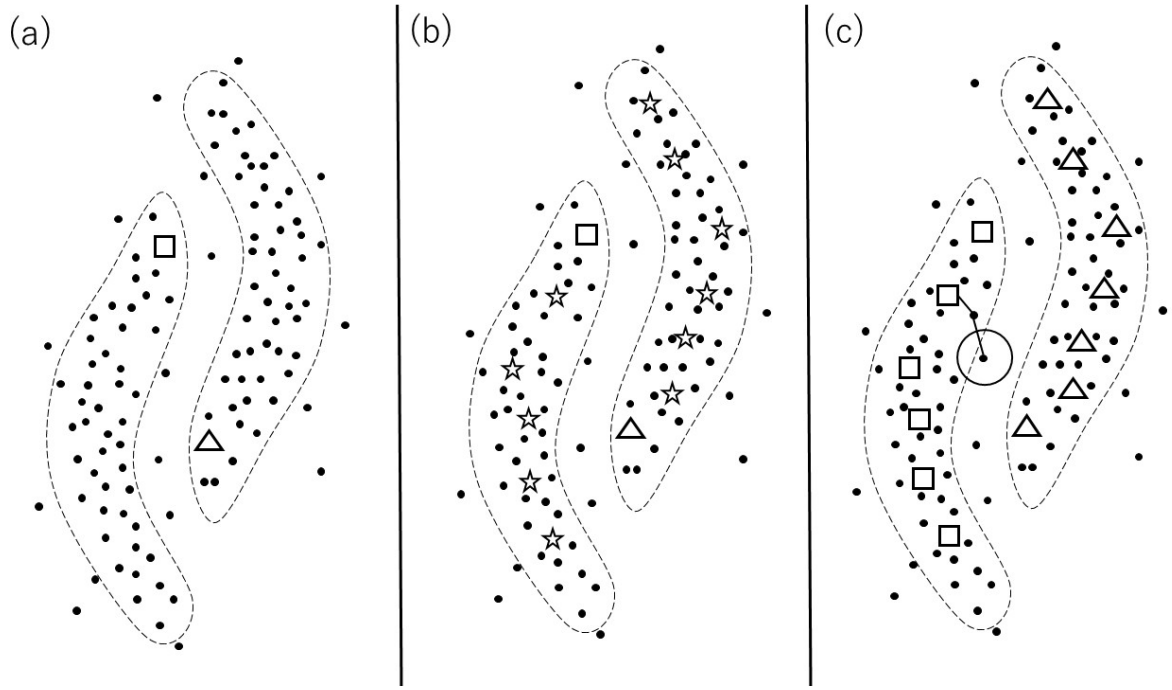

Figure 9: This figure explains the working mechanism of the RLP algorithm. (a) The labeled and unlabeled data (black squares/triangles and black dots, respectively) are given in $\mathbb{R}^2$. The number of classes is two. The two banana shapes delineate the true data manifolds. (b) Line 2 of the RLP algorithm. The star symbols and the black dots denote the hub data and non-hub data, respectively, where $h = 11$. (c) After line 5 of RLP algorithm, the hub data are assigned labels and are considered new labeled data. The labels of $\bar{H}$ are predicted by k-NN ($k = 1$) with the computed geodesic distance of line 4. As an example, the circled black dot is labeled (by GkNN), with the square symbol indicated in the left banana shape.

4.2 Some Properties of RLP

This section discusses the hyperparameters of RLP, and the time and space complexity of the RLP algorithm.

Proposition 1. *Let $|L|$ and $|U|$ be l and u, respectively. For the given k_1, k_2, h and k_v, the time complexity of RLP in algorithm 1 is expressed as follows:*

$$O\left(h^3 + Dn^2 + k(k_1 + \log n)n\right), \tag{21}$$

where $n = l + u$, and $k = \max\{k_2, k_v\}$.

Proof. The time complexity is contributed by three lines in algorithm 1: line 1 computes the Euclidean distance matrix and identifies the k_1-nearest neighbors of each vertex, line 4 computes the geodesic metric, and line 5 labels the hub data. For instance, if the time cost of computing a single Euclidean distance is $O(D)$ time, line 1 consumes $O(Dn^2)$ time. Line 4 consumes $O(k(\log n + k_1)n)$ time, with $k = \max\{k_2, k_v\}$ (see [31, 25] for details), and line 5 requires $O(h^3)$ time for computing the $h \times h$ inverse matrix. □

The space complexity is of order n^2 because the n-node graph G should be maintained.

4.3 Hyperparameter Tuning

The RLP has four hyperparameters, namely, k_1, k_2, h and k_v. Hyperparameters k_1 and k_v in line 1 and 7 of algorithm 1 do not require tuning. We confirmed that k_1-NN with a small k_1 efficiently detects the outliers and obtains an appropriate hub set in our algorithm. The label prediction on $\bar{H}$ with GkNN is based on majority voting among the k_v-nearest labeled samples in terms of geodesic distance. Hence, when k_v is large, GkNN fails to exploit the local structure of the data manifold. We experimentally demonstrated that a small k_v such as three, four or five is a good choice for label prediction.

Let us consider the tuning of hyperparameters k_2 and h. The parameter h ranges from one to u. As h approaches u, the complexity of RLP approaches that of LP, thus rendering our method impractical. To avoid this problem, we upper-bound h by $h^{\max}$.

Corollary 1. *Let $k_2^{\max}$ be the upper bound of k_2 in algorithm 1. Fix k_1, k_v, and k_2 such that $1 \leq k_2 \leq k_2^{\max}$. Thereafter, define the upper bound of h as follows:*

$$h^{\max} = \min\left\{\left(Dn^2 + \kappa(k_1 + \log n)n\right)^{\frac{1}{3}}, u\right\}, \tag{22}$$

where $n = l + u$ and $\kappa = \max\{k_2^{\max}, k_v\}$. In this case, for all h such that $0 \leq h \leq h^{\max}$, the time complexity of RLP is bounded by the following:

$$O\left(Dn^2 + \kappa(k_1 + \log n)n\right). \tag{23}$$

Moreover, suppose that hyperparameters k_2 and h are tuned by K-fold cross-validation in the ranges $1 \leq k_2 \leq k_2^{\max}$ and $0 \leq h \leq h^{\max}$, respectively. Let c be the number of candidates for hyperparameters. The total time complexity of RLP, including the time consumed by cross-validation, is bounded by the following:

$$O\left(KDn^2 + cK\kappa(k_1 + \log n)n\right). \tag{24}$$

Proof. We first consider the worst-case time complexity of (21) in the range $1 \leq k_2 \leq k_2^{\max}$. In this case, the time complexity of RLP is bounded by the following:

$$O\left(h^3 + Dn^2 + \kappa(k_1 + \log n)n\right). \tag{25}$$

By raising both sides of Eq.(22) to the third power, we obtain the following inequality for all h in $0 \leq h \leq h^{\max}$:

$$h^3 \leq (h^{\max})^3 \leq Dn^2 + \kappa(k_1 + \log n)n. \tag{26}$$

By combining (25) and (26), the time complexity (23) is derived. Finally, given that there are $c \times K$ iterations of cross-validation, the time complexity (24) follows from the worst-case time complexity (23). Note that the choice of the hyperparameters h and k_2 does not affect the computation cost of k_1-NN in line 1 of algorithm 1. Hence, the order Dn^2 is kept unchanged. □

In our numerical experiments of RLP, we conduct a five-fold cross-validation by using the $h^{\max}$ of Eq.(22) and the $k_2^{\max}$ set to 20.

Remark 6. *In the numerical experiments of Section 6, we observed that the prediction accuracy of RLP was not negatively affected by limiting the upper bound of h in the cross-validation. The hub set with a large h tended to include noisy data that deteriorated the prediction accuracy on the hub set.*

Remark 7. *Corollary 1 guarantees that the time complexity of RLP with hyperparameter $h^{\max}$ is comparable with that of GkNN. The computational efficiency of RLP can be improved by approximating the k-NN graph. According to [32] and [33], the time complexity of constructing the approximated k-NN graph is $O(Dn \log n)$. Consequently, the computational cost of the RLP with an approximated k-NN is reduced to $O((D + \kappa)n \log n + \kappa k_1 n)$.*

5 Application of RLP to Online Scenario

In this section, we incorporate the RLP into online SSL.

5.1 Quantized RLP

The QLP predicts the labels of the unlabeled samples on compressed graph using LP. The proposed online method replaces LP with RLP, which performs equivalently to QLP on compressed graphs. We refer this RLP as the *quantized RLP* (QRLP). QRLP and RLP differ only by their inclusion and exclusion of the vertex multiplicities, respectively. Thus, the QRLP algorithm is used to rewrite the input, line 5, line 7 of the algorithm 1 as follows.

- Input: The inputs to QRLP are the labeled and unlabeled datasets (L and U, respectively), the vertex multiplicities $\mathbf{m}$ of U, the numbers of neighbors k_1 and k_2, the size of the hub data set h, and the size of the majority vote k_v.
- Line 5: Estimate the labels of H as follows: Conduct QLP on the directed weighted compressed graph $G_1 = (L_{\mathbf{x}} \cup H, E_1, w_1)$, where E_1 is defined by k_2-NN with d_G. The weight $w_1(\mathbf{x}_i, \mathbf{x}_j)$ is defined by $\exp(-d_G(\mathbf{x}_i, \mathbf{x}_j)^2/\sigma^2)$. The estimated hub set is denoted by $\widehat{H}$.
- Line 7: Estimate the labels of $\bar{H}$ by GkNN on G with $k = k_v$ by using the updated labeled set L. Note that, if $\mathbf{x}_i \in H$ is one of the k_v-labeled nearest neighbors with $\mathbf{x}_j \in \bar{H}$, the vote from $\mathbf{x}_i$ to $\mathbf{x}_j$ is counted as $m(\mathbf{x}_i)$ and not as one. Let $\widehat{\bar{H}}$ be the labeled set of $\bar{H}$.

The output $\hat{U}$ of the QRLP algorithm is expressed as follows:

$$\hat{U} = QRLP(L, U, \mathbf{m}, k_1, k_2, h, k_v).$$

Remark 8. *Although we did not rewrite to line 2 of the algorithm 1, the vertex multiplicities are considered when building the hub set; see Eq.*(20) *of definition 2.*

Remark 9. *When* $\mathbf{m} = \mathbf{1}$*, QRLP reverts to RLP.*

The QRLP has four hyperparameters, namely, k_1, k_2, h and k_v. Similar to that in the RLP, k_1 and k_v can be chosen by efficient heuristics. On the compressed graph, the time complexities of QRLP and RLP are the same.

5.2 Online Quantized RLP

The online QRLP is obtained by combining DA and QRLP. The pseudo code is presented in algorithm 2, and the details are presented below.

- *Pretuning and initialization*: If an initial labeled data set L_0 and an unlabeled data set U_0 are obtained before the data stream arrives, pretuning and initialization are desired. As discussed before, QRLP has two tunable hyperparameters, namely, k_2 and h. Moreover, given that the graph is compressed by DA, its size $|C_t|$ may change at each time. Accordingly, k_2 and h should be tuned with $|C_t|$ at each time step; this process is time intensive. Prior to streaming, we prepare appropriate sizes of k_2 and h on the basis of L_0, U_0, and the maximum size of the compressed graph n_c. This preparation maintains the small time complexity of the online QRLP at each time. By setting the cardinality of U_0 to n_c, the pretuning is performed as described below.

  1. Input L_0, U_0, number of neighbors k_1, majority vote size k_v, and maximum size of the compressed graph n_c.
  2. Define the upper bound of the number of neighbors k_2 as $k_2^{\max}$. By using $|L_0|$, $|U_0|$, $k_2^{\max}$, and k_v, calculate $h^{\max}$ by Eq.(22). Given an integer I, we define n_{c_i} by $\lfloor n_c/I \rfloor i$ for $i \in \{1, 2, ..., I\}$, and denote $U_0^{(i)}$ as a set of randomly sampled n_{c_i}-data from U_0. For each i, find the pair (k_2, h) that maximizes the accuracy of the RLP output via the cross-validation by using L_0, $U_0^{(i)}$, k_1 and k_v. The cross validations with k_2 and h ranged through $1 \leq k_2 \leq k_2^{\max}$ and $1 \leq h \leq \min\{h^{\max}, n_{c_i}\}$, respectively. Denote the most accurate pair by $(k_2^{(i)}, h^{(i)})$ for the fixed i. Denote $\{(k_2^{(i)}, h^{(i)}, n_{c_i})\ ; i \in I\}$ by $\mathcal{T}$.
  3. Output $\mathcal{T}$.

  Let us denote the above procedure by

  $$\mathcal{T} = PreTune(L_0, U_0, k_1, k_v, n_c).$$

  To implement the pretuning process, we must acquire unlabeled data before the stream arrives.

  Furthermore, we must initialize the starting set of centroids C_0, the starting vertex multiplicities $\mathbf{m}_0$ of C_0, and the starting radius R_0. Here, we assign $C_0 \leftarrow U_0$, $\mathbf{m}_0 \leftarrow \mathbf{1}$, and $R_0 \leftarrow \epsilon$, where ϵ is a small positive real number.

- Line 1: If the labeled data is observed at time t, the labeled dataset is updated without updating the compressed graph.

- Line 2: If the observed data is unlabeled, the centroid set, its vertex multiplicities, and the radius are updated by DA. Then, QRLP is conducted on the compressed graph with vertices $(L_t)_{\mathbf{x}} \cup C_t$. Note that when the appropriate pair $(k_2^{(i^*)}, h^{(i^*)})$ is chosen from $\mathcal{T}$, the size difference between n_{c_i} and $|C_t|$ is minimized.

Let us now consider the time complexity, space complexity, and choice of the hyperparameters in online QRLP.

Algorithm 2 : Online QRLP

Input : At time $t \geq 1$, a new data point, a labeled data set L_{t-1}, a set of centroids C_{t-1}, the vertex multiplicities $\mathbf{m}_{t-1}$ for C_{t-1}, the radius R_{t-1}, the number of neighbors k_1, the size of majority vote k_v, the maximum size of the compressed graph n_c, and the result of pretuning $\mathcal{T}$ by computing $PreTune(L_0, U_0, k_1, k_v, n_c)$ (See sub-section 5.2 [*pretuning and initialization*])

1: If the new data point is a labeled data $(\mathbf{x}_t, y_t)$, then $C_t \leftarrow C_{t-1}$, $\mathbf{m}_t \leftarrow \mathbf{m}_{t-1}$, $R_t \leftarrow R_{t-1}$, $L_t \leftarrow L_{t-1} \cup \{(\mathbf{x}_t, y_t)\}$.

2: If the new data point is an unlabeled data $\mathbf{x}_t$, first update the labeled data set by $L_t \leftarrow L_{t-1}$. Then, build the $(C_t, \mathbf{m}_t, R_t)$ of a newly compressed graph by computing $DA(\mathbf{x}_t, C_{t-1}, \mathbf{m}_{t-1}, R_{t-1}, n_c)$. Then, estimate the labels of C_t by computing $QRLP(L_t, C_t, \mathbf{m}_t, k_1, k_2^{(i^*)}, h^{(i^*)}, k_v)$, where $i^* = \underset{1 \leq i \leq I}{\operatorname{argmin}} \left| n_{c_i} - |C_t| \right|$ with $\mathcal{T}$. Set the *QRLP* output to $\hat{C}_t$. Finally, determine $\hat{y}_t$ from $\hat{C}_t$.

Output: $\hat{y}_t$.

Proposition 2. *Suppose that algorithm 2 does not detect labeled data. Let L_0 and U_0 be the labeled and unlabeled datasets obtained before the stream, respectively. Fix the number of neighbors k_1, the size of majority vote k_v, and the maximum size of compressed graph n_c. Let the number of neighbors k_2 be upper bounded by $k_2^{\max}$, and run $PreTune(L_0, U_0, k_1, k_v, n_c)$ to obtain $\mathcal{T}$. Then, the time complexity of online QRLP at each time is upper bounded by*

$$O\left(Dn^2 + \kappa(k_1 + \log n)n\right), \tag{27}$$

where $\kappa = \max\{k_2^{\max}, k_v\}$, $n = l + n_c$, $l = |L_0|$.

Proof. The time complexity (27) follows by summing time complexity (19) and (23). □

In practical situations, n_c cannot be large; therefore, if the dimension of the feature vector is excessive, online QRLP will not be faster than online QLP. However, online QRLP runs faster than online QLP on low-dimensional feature vectors. Online QRLP has five hyperparameters. Hyperparameters k_1 and k_v do not require tuning, whereas k_2 and h are tuned in the pretuning stage of algorithm 2. To ensure a fast prediction in the online scenario, n_c must be set to a small value.

Given that the graph on $(L_t)_{\mathbf{x}} \cup C_t$ should be maintained in online QRLP, the space complexity of the algorithm is $O\left(\max\{D(|L_t| + n_c), (|L_t| + n_c)^2\}\right)$. Note that in practical situations, $|L_t|$ cannot be large.

Remark 10. *Our online SSL algorithm combines QRLP with DA. Other online SSL algorithms can be constructed by incorporating online graph compressing methods such as [34] into QRLP.*

6 Numerical Experiments

The performance of our methods was evaluated in offline and online scenarios. Experiments were performed on eight real-world datasets, namely, Yale (Yale Face Database B) [35], ORL, UMNIST [36], COIL (COIL-20) [37], Vowel [38], MNIST [39], optdigits [38], and USPS [36]. Table 12 shows the properties of each dataset.

Table 2: Properties of the data sets, where D, C, and # denote the dimension of feature vector, the number of classes, and the sample size, respectively.

	Yale	ORL	UMNIST	COIL	Vowel	MNIST	optdigits	USPS
D	1024	10304	10304	16384	12	784	64	256
C	15	40	20	20	2	10	2	10
#	165	400	575	1439	1456	70000	5216	11000

6.1 Offline Experiments

We compared the performances of three methods (LP, GkNN, and RLP) on the eight real-world datasets described in Table 12. All experiments were conducted 20 times with different random seeds, and the performances were averaged to obtain the final results. Table 3 shows the prediction accuracy and their standard deviations on the different datasets. Also, the dependency of the prediction accuracy on the unlabeled data size was evaluated on the MNIST dataset. The unlabeled data size varied from 100 to 19900 (in uneven increments), and the size of the labeled data was fixed at 100. Table 4 shows the results. In all tables, the most accurate prediction is highlighted in bold font. The hyperparameters in each method were chosen as described below.

- In RLP, hyperparameters k_1 and k_v were set to four and three, respectively, and k_2 and h were determined by the five-fold cross-validation as mentioned above. Hyperparameter k_2 was chosen from $\{5, 10, 20\}$, and the range of h was $\{h_i \mid 1 \leq i \leq 5, i \in \mathbb{N}\}$, where $h_i = i \times \lfloor h^{\max}/5 \rfloor$; see Eq.(22). The candidates of k_2 were decided through preliminary experiments.

- In LP, the number of neighbors k was set to five, and the weight function w was defined as Gaussian kernel. The bandwidth σ was determined by the mean heuristics.

- In GkNN, the number of neighbors and majority votes were set to five and three, respectively.

As confirmed in Table 3, our method outperformed the other methods on all datasets except Vowel. The Vowel dataset is a two-class dataset with a heavy bias (97% class 1 membership). Accordingly, if all unlabeled data are labeled as class 1, the prediction accuracy is approximately 97%. When only eight labeled data are provided, we can conclude that all methods fail to learn. However, when more labeled data are provided (Table 5), the learning succeeds in RLP but continues to fail in LP and GkNN. Therefore, empirically speaking, the prediction accuracy of our method exceeds those of LP and GkNN. Furthermore, because our RLP method captures the structure of the data manifolds, its accuracy improves with the increasing number of unlabeled data points (Table 4). GkNN exhibits a similar tendency but is less efficient than RLP. Meanwhile, LP cannot capture the structure of the data manifolds even when u is large.

Table 6 and 7 show the averaged runtime with the standard deviation. The problem settings are the same as the experiments in Table 3 and 4, respectively. Both tables indicate that the RLP and GkNN tend to outperform the LP when unlabeled data set gets bigger. This result matches (17) and (23). The GkNN and RLP are comparable in terms of the computational cost. This result agrees to the theoretical analysis in (18) and (23). In Table 7, the difference between the three methods is not large when u is small to medium size. This is because the term pn^2

Table 3: The averaged prediction accuracy with the standard deviation on unlabeled data in eight real-world datasets. In each dataset, l and u denote the number of labeled and unlabeled data with $l = 4C$, respectively.

	(l, u)	RLP	LP	GkNN
Yale	(60,115)	**0.538**(**0.035**)	0.474(0.038)	0.471(0.027)
ORL	(160,240)	**0.915**(**0.021**)	0.894(0.025)	0.832(0.021)
UMNIST	(80,495)	**0.896**(**0.024**)	0.857(0.035)	0.762(0.042)
COIL	(80,1220)	**0.774**(**0.016**)	0.738(0.027)	0.727(0.020)
Vowel	(8,1392)	0.967(0.002)	**0.967**(**0.001**)	0.959(0.030)
MNIST	(40,2960)	**0.712**(**0.040**)	0.466(0.086)	0.670(0.035)
optdigits	(8,2998)	**0.999**(**0.000**)	0.994(0.011)	0.972(0.006)
USPS	(40,3960)	**0.671**(**0.038**)	0.444(0.072)	0.618(0.031)

Table 4: The prediction accuracy with the standard deviation on the MNIST dataset. The number of labeled data is fixed to 100, and the number of unlabeled data u varies from 100 to 19900.

u	RLP	LP	GkNN
100	**0.728**(**0.066**)	0.698(0.055)	0.685(0.036)
200	**0.740**(**0.027**)	0.689(0.046)	0.692(0.039)
400	**0.759**(**0.040**)	0.705(0.036)	0.694(0.023)
900	**0.783**(**0.020**)	0.700(0.039)	0.730(0.026)
1900	**0.801**(**0.020**)	0.703(0.037)	0.747(0.016)
2900	**0.825**(**0.020**)	0.683(0.057)	0.770(0.018)
19900	**0.877**(**0.020**)	0.490(0.080)	0.802(0.017)

including the coefficient is thought to be dominant in the computational cost in this setting. For small datasets, all methods efficiently work, while the RLP maintains high prediction accuracy. Moreover, the RLP outperforms GkNN and LP for large u in terms of both prediction accuracy and computational cost. The prediction accuracy of the LP tends to be degraded for large u due to the noisy unlabeled data. Such phenomenon is commonly observed in semi-supervised learning [40].

6.2 Online Experiments

In this experiment, the performances of online QLP and the proposed online QRLP were compared over the eight real-world datasets. Table 8 shows the averaged prediction accuracy with the standard deviation for two algorithms. The results are the averages of 10 experiments with $n_c = 50$ on different data streams. The labeled dataset was assumed to be given before the stream arrived, and the stream included only unlabeled data. We also assumed that the data distribution remained unchanged during the observation.

The hyperparameters in each method were tuned as follows:

- In online QRLP, hyperparameters k_1 and k_v were set to four and three, respectively. According to the pretuning procedure in sub-section 5.2, $k_2^{\max}$ was set to 20, and $h^{\max}$ was

Table 5: The averaged prediction accuracy with the standard deviation on unlabeled data in the Vowel dataset. In the table, l and u denote the number of labeled and unlabeled data with $l = 4C$, respectively.

(l, u)	RLP	LP	GkNN
(100, 1300)	**0.995(0.005)**	0.968(0.005)	0.874(0.028)

Table 6: The averaged runtime (sec.) with the standard deviation for the experiments in Table 3.

	(l, u)	RLP	LP	GkNN
Yale	(60,115)	**0.090(0.003)**	0.092(0.001)	0.093(0.002)
ORL	(160,240)	5.062(0.034)	**4.971(0.012)**	5.092(0.045)
UMNIST	(80,495)	10.32(0.007)	**10.17(0.027)**	10.33(0.094)
COIL	(80,1220)	88.32(2.321)	89.17(1.575)	**87.83(2.966)**
Vowel	(8,1392)	0.568(0.005)	**0.352(0.004)**	0.519(0.005)
MNIST	(40,2960)	21.72(1.317)	23.00(1.137)	**21.52(0.739)**
optdigits	(8,2998)	4.118(0.086)	4.981(0.018)	**3.979(0.059)**
USPS	(40,3960)	**16.32(0.306)**	20.48(0.668)	17.08(0.533)

computed by Eq.(22). The integer I was set to three. For each i, five-fold cross-validation was conducted in the ranges $k_2 \in \{5, 10, 20\}$ and $h \in \{h_j \mid 1 \leq j \leq I, j \in \mathbb{N}\}$, where $h_j = j \times \lfloor \min\{h^{\max}, n_{c_i}\}/I \rfloor$.

- In online QLP, the number of neighbors k was set to five, and the weight w was defined by the Gaussian kernel. The band width σ was determined by the mean heuristics.

Table 8 shows that overall our method outperformed the online QLP. However, learning in both methods failed on the Vowel dataset. The most accurate predictions of online QRLP and online QLP are highlighted in bold font.

Tables 9 and 10 compare the performance of online QRLP and online QLP on the MNIST and USPS data streams, respectively, on different size n_c. The proposed method outperformed the existing one. Online QRLP with large n_c was the most accurate predictor, though larger n_c is computationally infeasible in the online scenario. Hence, the trade-off between the computational cost and prediction accuracy must be properly balanced when choosing n_c.

6.3 Relationship to Deep Neural Networks

From several viewpoints, we discuss about the relationship between online QRLP and deep neural networks (DNNs) in the online SSL scenario.

In terms of the required memory size in the learning process, online QRLP is more efficient than DNNs. DNNs such as CNNs or ResNets [41, 42] typically have more than a million of parameters, while online QRLP needs only $O(\max\{D(n_c + l), (n_c + l)^2\})$ memory space.

As for the prediction accuracy, online QRLP is thought to be comparable to DNNs when the number of labeled data is small. [43] reported that the prediction accuracy of the CNN was 0.683 when only 100 labeled samples of the MNIST dataset were used. This accuracy was achieved by the intensive search of the network structure out of 7103 candidates. On the other

Table 7: The averaged runtime (sec.) with the standard deviation for the experiments in Table 4.

u	RLP	LP	GkNN
100	0.113(0.000)	**0.100(0.001)**	0.110(0.001)
200	0.238(0.006)	**0.211(0.000)**	0.232(0.005)
400	0.643(0.004)	**0.585(0.010)**	0.640(0.028)
900	2.445(0.022)	**2.248(0.017)**	2.427(0.009)
1900	**9.734(0.050)**	10.66(0.111)	10.81(0.223)
2900	21.46(0.693)	22.23(0.648)	**21.04(0.798)**
19900	**1044.55**(**10.13**)	2005.96(9.442)	1172.37(43.87)

Table 8: The averaged prediction accuracy with the standard deviation on unlabeled data in eight real-world data streams. In each dataset, l denotes the number of labeled data obtained before the arrival of each stream of size T.

	(l, T)	Online QRLP	Online QLP
Yale	(75,85)	**0.528**(**0.046**)	0.471(0.048)
ORL	(80,220)	**0.722**(**0.029**)	0.656(0.058)
UMNIST	(60,240)	**0.637**(**0.028**)	0.544(0.059)
COIL	(80,120)	**0.660**(**0.034**)	0.592(0.062)
Vowel	(100,1300)	**0.969**(**0.004**)	0.966(0.002)
MNIST	(100,1900)	**0.679**(**0.021**)	0.663(0.019)
optdigits	(10,4990)	**0.971**(**0.002**)	0.961(0.031)
USPS	(100,1900)	**0.700**(**0.020**)	0.612(0.050)

hand, the online QRLP achieves 0.679 as shown in Table 9, though the problem setting was slightly different. In our experiments, additional 50 unlabeled samples were available. Note that the online QRLP was not tailored to image classification tasks, unlike CNNs.

Finally, concerning runtime, the learning of DNNs commonly requires much more computational cost than online QRLP, since DNNs have an enormous number of parameters to be learned. The computation time for the prediction depends on the size of DNNs or the number of centroids in online QRLP. The number of unlabeled data in the offline setting corresponds to the number of centroids. Hence, Table 4 and 7 indicate that the online QRLP with a small number of centroids is expected to be computationally efficient and not to occur severe deterioration of the prediction accuracy.

7 Conclusion of Part I

We proposed a generic graph-based SSL algorithm, called RLP. We confirmed that RLP is robust against noisy data and provides more accurate predictions than LP and GkNN. The computational efficiency of RLP matches that of GkNN. Furthermore, we confirmed the power of RLP as a core technique in the online SSL framework. In the online scenario, the proposed method has two tunable hyperparameters, namely k_2 and h. Future works should focus on the choice of hyperparameter h. In this paper, the upper bound $h^{\max}$ of h was determined by con-

Table 9: The averaged prediction accuracy with the standard deviation on the MNIST data stream. The number of labeled data before the stream arrival was set to 100 and the size of the data stream was set to 1900. In the table, n_c denotes the maximum size of the compressed graph.

n_c	Online QRLP	Online QLP
50	**0.679**(**0.021**)	0.663(0.019)
100	**0.714**(**0.015**)	0.684(0.020)
150	**0.720**(**0.017**)	0.679(0.022)

Table 10: The averaged prediction accuracy with standard deviation on the USPS data stream. The number of labeled data before the stream arrival was set to 120, and the size of the data stream was fixed at 1880. In the table, n_c denotes the maximum size of the compressed graph.

n_c	Online QRLP	Online QLP
50	**0.700**(**0.020**)	0.612(0.050)
100	**0.687**(**0.023**)	0.662(0.040)
150	**0.732**(**0.021**)	0.666(0.029)

sidering the computational cost. The prediction accuracy when $h^{\max}$ is based on other criteria should also be examined. Moreover, an adaptive method that determines both hyperparameters would be useful for practical online learning.

Part II
Spectral Embedded Deep Clustering

In this part, we propose a new clustering method based on a deep neural network. Given an unlabeled dataset and the number of clusters, our method directly groups the dataset into the given number clusters in the original space. Our statistical model is the conditional discrete probability distribution, which is defined by a deep neural network. Our clustering strategy is, first to estimate the cluster labels of unlabeled data points selected from high density region, and then to conduct semi-supervised learning to train the model by using the estimated cluster labels and the remaining unlabeled data points. At last, by using the trained model, we obtain the estimated cluster labels of all given unlabeled data points. The advantage of our method is that it does not require key condition. For example, the previous deep neural network based clustering methods require the cluster-balance of given dataset to be uniform. Moreover, it also can be applied to various data domains as long as the data is expressed by a feature vector. In addition, it was observed that our method was robust against outliers. Therefore, the proposed method is expected to averagely perform better than previous methods. This expectation is empirically confirmed by conducting numerical experiments on five commonly used datasets.

8 Introduction of Deep Clustering

Clustering is one of the oldest machine learning fields, where the objective is, given data points, to group them into clusters according to some measure. Many clustering methods have been proposed for a long while [44], and been applied to real-world problems [45].

The most known classical methods are k-means [46] and Gaussian Mixture Model (GMM) clustering [47]. Though those methods are computationally efficient, they can only model convex shapes and are thus applicable in limited cases. The kernel k-means [48], kernel GMM clustering [49] and Spectral Clustering (SC) [50] can capture more complicated shapes than k-means and GMM but are difficult to scale up to large datasets. In recent years, due to technological progress, we can acquire many types of data such as images, texts and genomes in large numbers. Thus, the demand of advanced efficient clustering methods grows even stronger [51].

Thanks to the development of deep neural networks, we can now handle large datasets with complicated shapes [52]. Consequently, the studies of clustering using deep neural networks has been proposed. One major direction in the studies is to combine deep AutoEncoders (AE) [53] with classical clustering methods. This AE is used to obtain a clustering friendly low dimensional representation. Another major direction is directly grouping a given unlabeled dataset into the clusters in the original input space by employing a deep neural network to model the distribution of cluster labels.

With both directions, there exist popular methods. We summarize their applicable data domain and well performing condition in Table 11. For examples, CatGAN (Categorical Generative Adversarial Networks) learns discriminative neural network classifiers that maximize mutual information between the input data points and the cluster labels, while enforcing the robustness of the classifiers to data points produced by adversarial generative models. Since maximizing mutual information implicitly encourages the cluster-balance distribution of the model to be uniform, if the distribution of cluster-balance with the given unlabeled dataset

is uniform, then CatGAN will perform well. JULE (Joint Unsupervised LEarning) learns a clustering friendly low dimensional representation for image datasets by using a convolutional neural network [41]. The assigned cluster labels and low dimensional representation are jointly optimized by updating a $n \times n$ similarity matrix of the representations, where n is the number of data points. Thus, $O(n^2)$ memory space must be secured to conduct the method.

As we can see in Table 11, most of their key conditions are not always realistic since the details of given unlabeled datasets are unknown and their size is large in typical machine learning scenarios. On the other hand, SpectralNet does not require key condition. It only requires the following two fundamental assumptions: the smoothness and manifold assumptions [54]. Note that the other methods in Table 11 also require the two assumptions. As for the weakness of SpectralNet, it is not robust against outliers. In the learning process, it learns the pairwise similarities over all data points. Therefore, the existence of outliers disturbs the method learning the similarities precisely, and thus returns inaccurate clustering results.

In this paper, we propose a deep clustering method named *Spectral Embedded Deep Clustering* (SEDC). Given an unlabeled dataset and the number of clusters, SEDC directly groups the dataset into the given number clusters in the input space. Our statistical model is the conditional discrete probability distribution, which is defined by a fully connected deep neural network. SEDC does not require key condition except the smoothness and manifold assumptions, and it can be applied to various data domains. Moreover, throughout our numerical experiments, we observed that our method was more robust against outliers than SpectralNet. The procedure of SEDC is composed of two stages. In the first stage, we conduct SC only on the unlabeled data points selected from high density region by using the geodesic metric to estimate the cluster labels. This special type of SC is named as *Selective Geodesic Spectral Clustering* (SGSC), which we propose for assisting SEDC as well. Thereafter, we conduct semi-supervised learning to train the model by using the estimated cluster labels and the remaining unlabeled data points. Note that, in this semi-supervised learning, we treat the estimated cluster labels of the selected unlabeled data points as the given true cluster labels. At last, by using the trained model, we obtain the estimated cluster labels of all given unlabeled data points.

In the remainder of this paper, we introduce related works in Section 9. We then introduce our proposed method in Section 10. We demonstrate the efficiency of our method with numerical experiments in Section 11. Finally in Section 12, we conclude the paper with the discussion on future works.

9 Related Works

In this section, we first introduce the existing clustering studies based on deep neural networks. As mentioned in Section 8, there are two major directions in the studies recently, i.e., the deep-AE based clustering and the direct deep clustering. We then introduce the two techniques related to our proposed method, i.e., SC and Virtual Adversarial Training (VAT).

9.1 Existing Clustering Methods Using Deep Neural Network

In deep-AE based clustering, the AE and a classical clustering method such as k-means are combined. The combination strategies are either sequential [55, 56] or simultaneous [57, 58, 59]. In the sequential way, deeply embedded representations of the given dataset are obtained by the deep AE, and then a classical clustering method is applied to the embedded set. In

	Preferred Domain	Need Num of Clusters ?	Need Hyperparameter Tuning ?	**Scalability for Large Dataset**	Computational Complexity	Robustness to Outliers	**Robustness to High dim**	**Additional Condition**
DEC	no	yes	no	yes	low	medium	medium	AE is expected to acquire a good representation.
JULE	Image	yes	yes	no	high	medium	strong	
VaDE	no	yes	no	yes	medium	medium	medium	Representation follows GMM.
IMSAT	no	yes	yes	yes	medium	medium	medium	Cluster-balance is uniform.
CatGAN	no	yes	no	yes	low	medium	medium	Cluster-balance is uniform.
SpectralNet	no	yes	yes	yes	medium	weak	medium	
Proposed	no	yes	yes (5)	yes	medium	strong	medium	

Table 11: Summary of popular deep clustering methods. Three AE based methods and four direct methods are presented, including our method SEDC. A method has a preferred domain when it is specialized for a certain type of data. A method requires hyperparameter tuning if some information required by the method is not learned by itself. Note that all methods require the smoothness and manifold assumptions. Some methods require additional conditions except the above two assumptions. Empty spaces mean that the method does not require it. All methods below require the number of clusters as one of their inputs.

the simultaneous way, the deep representations and their cluster labels are learned jointly by optimizing a single objective function.

As examples of the simultaneous way, we introduce [57] and [58] here. The method of [57] is named Deep Embedded Clustering (DEC). DEC trains a deep neural network by iteratively minimizing the Kullback-Leibler (KL) divergence between a centroid based probability distribution and an auxiliary target distribution. The deep network is used as the AE. They reported the clustering performance of DEC depended on the initial embedded representations and cluster centroids obtained by the AE and k-means. The method of [58] is named Variational Deep Embedding (VaDE). The approach relies on a Variational AutoEncoder (VAE) [60] that utilizes a Gaussian mixture prior. VaDE trains its deep neural network by minimizing the reconstruction error, while enforcing that the low dimensional representations follow the Gaussian mixture model.

With regard to the direct deep clustering, we introduce [61] and [62]. The method of [61] is named Information Maximizing Self-Augmented Training (IMSAT). It is based on data augmentation, where a deep neural network is trained to maximize the mutual information while regularizing the network so that the cluster label assignment of original data will be consistent with the assignment of augmented data. The method of [62] is named SpectralNet. This method is proposed to overcome the scalability and generalization of SC. It uses two deep neural networks. The first network learns the similarities among all given data points. This network is known as Siamese net [63]. Then, the second network learns a dimension reduction mapping which preserves the similarities obtained by the first net. After both are trained, the dimensionality reduced data points obtained by the second network are grouped into clusters by k-means.

9.2 Related Techniques with Proposed Method

SEDC handles the following clustering problem: given a set of unlabeled data points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ ($\mathbf{x}_i \in \mathbb{R}^D$) and the number of clusters C, group $\mathbf{X}$ into C clusters. In SEDC, the estimated cluster label of $\mathbf{x}_i$ is obtained by the trained conditional discrete probability distributional classifier. This classifier is denoted by $p_\theta(y|\mathbf{x})$ where θ, $\mathbf{x}_i$ and y are a set of parameters, a feature vector and a cluster label, respectively. The cluster label y ranges $\{1, ..., C\}$. In addition, the classifier is defined by a fully connected deep neural network whose last layer is the soft-max function. SC and VAT, explained below, take an important role in SEDC: see the detail in Section 10.

9.2.1 Spectral Clustering

SC [50, 64] is a popular classical clustering algorithm. We here introduce the commonly used framework of SC, which is used also in SEDC algorithm. It first embeds the data points in the eigenspace of the Laplacian matrix derived from the pairwise similarities over all given data points. Then, SC applies k-means on the representation to obtain the cluster labels. The SC algorithm is outlined below.

1. Given dataset $\mathbf{X}$, define the weighted undirected graph G which comprises a set of vertices $\mathbf{X}$ and a set of undirected edges E defined by k_1-nearest neighbor (k_1-NN) on the basis of a metric d. Suppose that each edge $e_{ij} \in E$ has a non-negative symmetric weight w_{ij}.

2. Denote the $n \times n$ affinity matrix $\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ on G by $\mathbf{L}$, where $\mathbf{W} = (w_{ij})_{i,j}$, and $\mathbf{D}$ is the diagonal matrix whose entries are given by $d_{ii} = \sum_j w_{ij}$.

3. Given the number of clusters C, compute the largest C eigenvectors $\mathbf{u}_1, ..., \mathbf{u}_C$ of the eigenproblem $\mathbf{Lu} = \lambda\mathbf{u}$. Let $\mathbf{U} \in \mathbb{R}^{n \times C}$ be the matrix containing the vectors $\mathbf{u}_1, ..., \mathbf{u}_C$ as columns. Thereafter, re-normalize each row of $\mathbf{U}$ to have unit length.

4. Cluster n rows of $\mathbf{U}$ as points in $\mathbb{R}^C$ by conducting k-means (k= C). Let $\{\hat{y}_i\}_{i=1}^n$ and $\{\mu_j\}_{j=1}^C$ be the estimated labels of $\mathbf{X}$ and the set of centroids obtained by k-means, respectively.

Now, let us denote the above procedure by $\left(\{\hat{y}_i\}_{i=1}^n, \{\mu_j\}_{j=1}^C, \mathbf{U}\right) = SC(\mathbf{X}, k_1, d, w, C)$. The weight w is often defined by the following similarity function:

$$w_{ij} = \begin{cases} \exp(-d(\mathbf{x}_i, \mathbf{x}_j)^2/\sigma^2), & e_{ij} \in E, \\ 0, & e_{ij} \notin E, \end{cases} \tag{28}$$

The bandwidth σ is selected by the median or mean heuristics [26]. In our proposed method, we employ k-means++ [65] technique since the method uses that SC function.

9.2.2 Virtual Adversarial Training

VAT [66] is a regularization method based on local perturbation. It forces the statistical model $p_\theta(y|\mathbf{x})$ follow the smoothness assumption. VAT is known to empirically perform better than other local perturbation methods such as random perturbation [67] and adversarial training [68] in both semi-supervised and supervised learning scenarios. It can be employed also in unsupervised learning scenarios [61, 69] since VAT only requires the unlabeled data points.

VAT first defines the adversarial point $T_{\text{VAT}}(\mathbf{x})$ with given $\mathbf{x}$ as follows:

$$T_{\text{VAT}}(\mathbf{x}) = \mathbf{x} + \mathbf{r}_{\text{vadv}}, \tag{29}$$

where $\mathbf{r}_{\text{vadv}}$ is ϵ-perturbation to a virtual adversarial direction:

$$\mathbf{r}_{\text{vadv}} = \underset{\mathbf{r}}{\text{argmax}} \left\{ \mathbb{KL} \left[p_{\theta_t}(y|\mathbf{x}) || p_\theta(y|\mathbf{x} + \mathbf{r}) \right] ; ||\mathbf{r}||_2 \leq \epsilon \right\}. \tag{30}$$

In Eq.(30), θ_t is the estimated parameter at t-th iteration, and $\mathbb{KL}$ is Kullback-Leibler divergence [70]. Then, VAT minimizes the following $\mathcal{R}_{\text{VAT}}(\theta; \mathbf{X})$ with respect to θ:

$$\mathcal{R}_{\text{VAT}}(\theta; \mathbf{X}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{KL} \left[p_{\theta_t}(y|\mathbf{x}_i) || p_\theta(y|T_{\text{VAT}}(\mathbf{x}_i)) \right]. \tag{31}$$

The approximation of $\mathbf{r}_{\text{vadv}}$ in Eq.(30) is computed by the following two steps:

$$\mathbf{g} \leftarrow \nabla_{\mathbf{r}} \, \mathbb{KL} \left[p_{\theta_t}(y|\mathbf{x}) || p_{\theta_t}(y|\mathbf{x} + \mathbf{r}) \right] \Big|_{\mathbf{r} = \xi \mathbf{d}}, \tag{32}$$

$$\mathbf{r}_{\text{vadv}} \approx \epsilon \frac{\mathbf{g}}{||\mathbf{g}||_2}, \tag{33}$$

where $\mathbf{d} \in \mathbb{R}^D$ is a random unit vector generated by the standard normal distribution, and $\xi \in \mathbb{R}_+$ is a fixed small positive number. Regarding the logic behind the approximation, see sub-section 3.3 of [66].

Remark 11. *The radius ϵ of Eq.*(30) *is defined for given* $\mathbf{x}$ *as below:*

$$\epsilon(\mathbf{x}) = \alpha ||\mathbf{x} - \mathbf{x}^{(z)}||_2, \tag{34}$$

where α is a scalar and $\mathbf{x}^{(z)}$ is the z-th nearest data point from $\mathbf{x}$*. In [61], $\alpha = 0.4$ and $z = 10$ are used.*

10 Proposed Deep Clustering Method

As we already mentioned in the end of Section 8 and the begining of subsection 9.2, given an unlabeled dataset $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ ($\mathbf{x}_i \in \mathbb{R}^D$) and the number of clusters C, our proposed deep clustering named SEDC groups $\mathbf{X}$ into C clusters. The estimated cluster label of each $\mathbf{x}_i \in \mathbf{X}$ is defined by $\mathrm{argmax}_j p_{\theta^*}(y = j|\mathbf{x}_i)$, where θ^* is the trained set of parameters. In SEDC, the training scheme of the classifier $p_\theta(y|\mathbf{x})$ is as follows: we firstly only estimate the cluster labels of selected unlabeled data points by using SGSC, and then conduct semi-supervised learning to train the classifier. Regarding with this semi-supervised learning, we use the estimated cluster labels of selected unlabeled data points and the remaining unlabeled data points, which are treated as the given true cluster labels and unlabeled data points respectively.

In this section, we first introduce SGSC. Thereafter, we present our main method SEDC.

10.1 Selective Geodesic Spectral Clustering

The motivation behind SGSC is to assist the semi-supervised learning in SEDC. SGSC conducts SC only on selected unlabeled data points with the geodesic metric, then returns the estimated cluster labels of selected points. The selected points are defined as data points in high density region, and these data points are approximated by the highest degree nodes on the affinity data graph. The geodesic metric is approximated by the graph shortest path distances on the graph. Empirically speaking, the estimation accuracy of cluster labels with selected points tends to not only be robust against the existence of outliers but also be competitive. This tendency can help SEDC return competitive clustering result. The reason of robustness is that the selection of data points from high density region tends to not be affected by the existence of outliers [71]. The reason to employ the geodesic metric is that the metric is known to be useful to capture the structure of the data manifolds especially when the number of given data points is large [25].

Now, let us refer the selected data points as *hub* data points, and define the set of hub data points by $H \subset \mathbf{X}$. The data points selected for H are those with the most neighbors on the graph G. The hub dataset H is formally defined below.

Definition 2. *Let* $\mathbf{X}$ *be the given unlabeled dataset. On the graph* $G = (\mathbf{X}, E)$*, let* $\mathcal{N}_j$ *be the set of adjacent nodes of* $\mathbf{x}_j \in \mathbf{X}$*. For a natural number h, the hub set H is defined as the collection of nodes that ranked in the top-h cardinality of* $\mathcal{N}_j$ *in* $\mathbf{X}$*. They are arranged in the descending order.*

Algorithm 3 and Fig.10 show the pseudo code of SGSC and the mechanism of SGSC, respectively. The detail of this algorithm is explained below.

- Line 1: Given an unlabeled dataset $\mathbf{X}$, the undirected graph G_0 is constructed in the k_0-nearest neighbor (k_0-NN) manner with a Euclidean metric. G_0 is used not only for defining the hub set H but also for approximating the geodesic distance on the manifolds shaped by $\mathbf{X}$. We consider k_0 as a hyperparameter.

- Line 2: Given the number of hub points h and G_0, the algorithm defines the hub set H based on definition 2. By the appropriate setting of h, H can exclude outliers. In this algorithm, h is considered as a hyperparameter.

Algorithm 3 : $\mathbf{Q} = SGSC(\mathbf{X}, k_0, k_1, h, C)$

Input: Unlabeled dataset $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$. Number of neighbors k_0, k_1. Number of hub data points h. Number of clusters C.

Output: The estimated conditional discrete probability distributions with hub data points, $\mathbf{Q}$.

1: Construct the undirected graph $G_0 = (\mathbf{X}, E)$, where the edge set E is defined by k_0-NN with the Euclidean distance.

2: Build the hub dataset H on graph G_0 such that $|H| = h$. Denote the element of H by $\mathbf{x}_{(i)}$ $(i = 1, ..., h)$.

3: Define the geodesic metric d_{G_0} as the shortest path distance on the graph G_0.

4: Define $\{\mu_j\}_{j=1}^C$ and $\mathbf{U}$ as the two outputs of $SC(H, k_1, d_{G_0}, w, C)$, where the weight $w(\mathbf{x}_i, \mathbf{x}_j)$ is defined by $\exp(-d_{G_0}(\mathbf{x}_i, \mathbf{x}_j)^2/\sigma^2)$. Then, compute the conditional cluster probability $q_{j|(i)}$ with each hub data point $\mathbf{x}_{(i)}$ in H as follows:

$$q_{j|(i)} = \frac{\left(1 + \left\|\tilde{\mathbf{x}}_{(i)} - \mu_j\right\|_2^2/\gamma\right)^{-\frac{\gamma+1}{2}}}{\sum_{j'=1}^C \left(1 + \left\|\tilde{\mathbf{x}}_{(i)} - \mu_{j'}\right\|_2^2/\gamma\right)^{-\frac{\gamma+1}{2}}},$$

where γ is a small positive number and $\tilde{\mathbf{x}}_{(i)}$ is i-th row of $\mathbf{U}$.

5: Let $q_{(i)}$ and $\mathbf{Q}$ be $(q_{1|(i)}, ..., q_{C|(i)})$ and $h \times C$ matrix, respectively. The i-th row of $\mathbf{Q}$ is defined by $q_{(i)}$.

- Line 3: The geodesic distance is determined from the Euclidean distances defined on the undirected edges of G_0. Since we only use the geodesic distances between the data points of H in line 4, we compute the required distances. Efficient algorithms are available for this purpose [31, 25].

- Line 4: Given the number of clusters C, we here estimate the conditional discrete probability distribution $p(y|\mathbf{x}_{(i)})$ for each $\mathbf{x}_{(i)} \in H$, where y is the cluster label ranging $\{1, ..., C\}$. Let us denote the estimated $p(y|\mathbf{x}_{(i)})$ by $q_{(i)} = (q_{1|(i)}, ..., q_{C|(i)})$. This estimation relies on conducting SC with d_{G_0} metric only on H. The definition of the weight w in this SC follows Eq.(28). The key to succeed the estimation is to employ the combination of a different number of neighbors k_1 from k_0 and the geodesic metric d_{G_0} to a SC. Typically, given data points that are dense in the input space, the combination of a small number of neighbors and the Euclidean metric makes a SC perform well. However, we here consider H, which is sparse in the input space. This is why we employ the combination. We here consider k_1 as a hyperparameter as well. Following [72], we compute $q_{j|(i)}$ by using the outputs $\{\mu_j\}_{j=1}^C$ and $\mathbf{U}$ of $SC(H, k_1, d_{G_0}, w, C)$. Note that $q_{j|(i)}$ can be considered as the probability that $\tilde{\mathbf{x}}_i$ belongs to the cluster j, where $\tilde{\mathbf{x}}_{(i)}$ is the low dimensional representation of $\mathbf{x}_{(i)}$ according to the property of SC [64]. As for γ, we set 10^{-60} to it.

Remark 12. *Though we say we estimate the "cluster labels" of hub data points by SGSC, it actually outputs the estimated conditional probability distributions with hub data points. The reason is that, throughout our preliminary experiments, we observed that employing* $\mathbf{Q}$ *of line 5 made SEDC perform better than employing the one-hot vector. This one-hot vector, for in-*

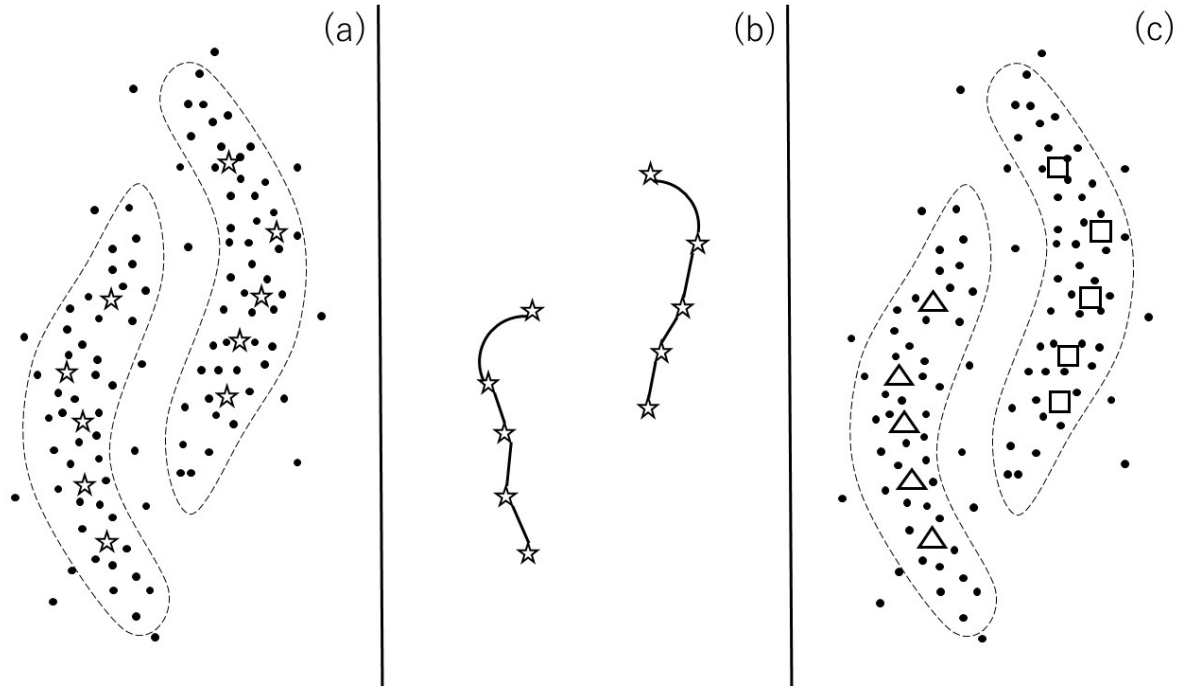

Figure 10: This figure explains how SGSC works. (a):Given unlabeled data points, in line 2 of Algorithm 3, SGSC computes the hub data points. The hub data points are expressed by star symbols, and the number of hub data points h is ten in this case. (b):In line 4 of the algorithm, SGSC focuses only on the hub data points, then conducts SC with the geodesic metric on those hub points, where we set one to k_1. (c):As the results, we obtain the cluster labels of hub points. The triangle and square symbols mean different labels. Note that an actual output of SGSC is the estimated conditional discrete probability distributions with hub data points, but we can obtain the estimated cluster labels from the distributions.

stance, can be defined by using the estimated cluster labels $\{\hat{y}_{(i)}\}_{i=1}^{h}$ *which is one of the outputs of* $SC(H, k_1, d_{G_0}, w, C)$.

10.2 Spectral Embedded Deep Clustering

SEDC is a deep clustering method. Given an unlabeled dataset $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{n}$ and the number of clusters C, it groups $\mathbf{X}$ into C clusters. As mentioned in the begining of this section, this method employs the conditional discrete probability distribution $p_\theta(y|\mathbf{x})$ as the statistical model, which is defined by a fully connected deep neural network. By using the trained model, we obtain the estimated cluster label of each $\mathbf{x}_i$. This method does not require an additional condition except two fundamental assumptions: the smoothness and manifold assumptions. Therefore, among the methods of Table 11, only SpectralNet is comparable to SEDC in this point. In addition, our method can be applied to various data domains once the raw data is transformed to the feature vector. Furthermore, empirically speaking, the performance of SEDC can be robust against outliers due to the robustness of SGSC against them. The pseudo code of SEDC is shown in Algorithm 4. The explanation is below.

The procedure of SEDC is composed of two stages. In the first stage, we estimate the conditional discrete probability distributions $\mathbf{Q}$ with hub data points. In the second stage, by treating $\mathbf{Q}$ as the given true distributions of hub data points, we conduct semi-supervised learning where

Algorithm 4 : $\{\hat{y}_i\}_{i=1}^n = SEDC(\mathbf{X}, k_0, k_1, h, C, \lambda_1, \lambda_2)$

Input: Unlabeled dataset $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$. Number of neighbors k_0, k_1. Number of hub data points h. Number of clusters C. Regularization parameters $\lambda_1, \lambda_2 > 0$.

Output: The estimated cluster labels of $\mathbf{X}$, $\{\hat{y}_i\}_{i=1}^n$.

1: Obtain the $h \times C$ matrix of estimated conditional discrete probability distributions with hub data points $\mathbf{Q}$ by computing $SGSC(\mathbf{X}, k_0, k_1, h, C)$ of Algorithm 3. Denote i-th row of $\mathbf{Q}$ by $q_{(i)}$, which means the estimated cluster label probability distribution of hub data point $x_{(i)}$. The index i ranges $\{1, ..., h\}$.

2: Let $p_\theta(y|\mathbf{x})$ be a statistical model, which is the cluster label probability distribution with given data point $\mathbf{x}$. The cluster label ranges $\{1, ..., C\}$. Define the objective of Eq.(35) by using $p_\theta(y|\mathbf{x})$, $\{q_{(i)}\}_{i=1}^h$ and given λ_1, λ_2. Then, minimize the objective with θ in stochastic gradient descent fashion. Denote the optimized parameter by θ^*.

3: Obtain the estimated cluster labels of all data points in $\mathbf{X}$ by using the trained classifier $p_{\theta^*}(y|\mathbf{x})$. Denote $p_{\theta^*}(y = j|\mathbf{x}_i)$ by $p^*_{j|i}$. Then, for all data point index i, compute $\hat{y}_i$ by $\hat{y}_i = \underset{j}{\text{argmax}}\ p^*_{j|i}$.

$\mathbf{Q}$ and the remaining unlabeled data points are used, to train the statistical model $p_\theta(y|\mathbf{x})$. After this training, SEDC returns the estimated cluster labels of each $\mathbf{x}_i \in \mathbf{X}$ by $\text{argmax}_j p_{\theta^*}(y = j|\mathbf{x}_i)$, where θ^* is the trained set of parameters and $j \in \{1, ..., C\}$. The estimated cluster labels of $\mathbf{x}_i$ is denoted by $\hat{y}_i$. Note that the estimated cluster labels of hub data points might be updated at the end of SEDC procedure.

Let us explain the details of the second stage. Suppose that, given a natural number h, we already finished the first stage. Now, let us denote the hub data points and their estimated conditional discrete probability distributions by $\mathbf{x}_{(i)} \in H$ and $q_{(i)}$ $(i = 1, ..., h)$, respectively. Then, by using all $q_{(i)}$, we conduct semi-supervised learning to train our statistical model $p_\theta(y|\mathbf{x})$. Recall that the model $p_\theta(y|\mathbf{x})$ is defined by the deep neural network whose last layer is soft-max function. The number of neurons of the first and last layer are the dimension of feature vector D and the number of clusters C, respectively. In this semi-supervised learning, we minimize the following loss with respect to θ:

$$\mathcal{R}_{\text{VAT}}(\theta; \mathbf{X}) + \frac{\lambda_1}{h} \sum_{i=1}^{h} \mathbb{KL}\left[p_\theta(y|\mathbf{x}_{(i)}) \| q_{(i)}\right] + \lambda_2 \mathbb{H}\left(\mathbf{Y}|\mathbf{X}\right), \tag{35}$$

where λ_1 and λ_2 are hyperparameters that range over positive numbers. In Eq.(35), the first and second terms express VAT loss of Eq.(31) and the pseudo empirical loss with estimated cluster probabilities of hub data points, respectively. The third term is the conditional Shannon entropy [70] averaged over $\mathbf{X}$, and it is defined as follows:

$$\mathbb{H}\left(\mathbf{Y}|\mathbf{X}\right) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{C} p_\theta(y = j|\mathbf{x}_i) \log p_\theta(y = j|\mathbf{x}_i).$$

We use the Adam optimizer [73] for the minimization. After minimizing Eq.(35), we estimate the labels of $\mathbf{X}$ by using the trained parameter θ^*. Let us denote the estimated cluster labels of $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ by $\{\hat{y}_i\}_{i=1}^n$. The labels are computed as follows: $\hat{y}_i = \text{argmax}_j\ p_{\theta^*}(y = j|\mathbf{x}_i)$.

As mentioned in sub-section 9.2, the minimization of the VAT loss encourages $p_\theta(y|\mathbf{x})$ to follow the smoothness assumption. In addition, that of entropy loss helps the model to follow

the cluster assumption [54]. The cluster assumption says that true decision boundary is not located in regions of the input space that are densely populated with data points. The entropy loss is commonly used in many studies [74, 75, 61, 66]. Note that the entropy loss is defined only by using the unlabeled data points, like the VAT loss. With regard to the pseudo empirical loss, we can consider other candidates such as the cross entropy. The reason why we chose the KL-divergence is that we observed that the KL-divergence made SEDC perform better than other candidates in our preliminary experiments.

10.3 Computational and Space Complexity of SEDC

The procedure of SEDC is composed of two stages. The first and second stages correspond line 1 of Algorithm 4 and line 2 of Algorithm 4, respectively. Let us now discuss about the computational complexity of this algorithm. Since the line 1 of Algorithm 4 is equivalent to Algorithm 3 itself, we investigate the complexity of Algorithm 3. Suppose that $h, k_0, k_1 \ll n$. In line 1 of Algorithm 3, we consume $O(Dn^2)$ to construct k_0-NN graph [76], where D is the dimension of feature vector. Then, in the line 3, we need to compute the graph shortest path distances between the data points in H. Thus, we consume $O\left(h(\log n + k_0)n\right)$ for computing the distances: see algorithm 1 of [25]. Thereafter, we need to solve the eigenvector problem with Laplacian matrix where $O(h^3)$ complexity is consumed. In line 2 of Algorithm 4, we consume $O(Dmn_{\text{itr}})$ where m and n_{itr} are the mini-batch size and the number of iteration in the training.

As for the memory complexity, since the dominant factors are to save k_0-NN graph and the model, SEDC needs $O\left(\max\{k_0 n, |\theta|\}\right)$ where θ is the set of parameters in a deep neural network.

Remark 13. *For most of deep clustering methods relying on k-NN graph construction, the dominant factor with their total computational complexity is k-NN graph construction, i.e., we need $O(Dn^2)$. However, according to [32, 33], by constructing the approximated k-NN graph, we only need $O(Dn \log n)$ for the construction.*

11 Numerical Experiments

In this section, we show the results of our numerical experiments. First, we show how accurately SGSC can estimate the labels of hub data points on five datasets. Then, we show the clustering results on the same five datasets by SEDC. With regards to the clustering experiments, we compare our proposed method with five popular methods: k-means [46], SC [50], DEC [57], IMSAT [61] and SpectralNet [62].

11.1 Datasets and Evaluation Metric

We conducted experiments on five datasets. Two of them are real-world datasets named MNIST [39] and Reuters-10k [77]. The other three are synthetic datasets named Four-Clusters (FC), Two-Moons (TM) and Three-Rings (TR). A summary of the dataset statistics is described in Table 12.

MNIST is a collection of 28×28 gray-scale images of handwritten digits. It is composed of 60,000 training and 10,000 test sets. The digits are transformed to 784 dimensional feature vectors. Then, they are centered, and the size is normalized. In this experiment, we use all 70,000 data points. Reuters-10k is a dataset of English news stories labeled with a category tree [77]. Following [57], we used the same four root labels. The news stories are transformed

Table 12: Summary of dataset statistics. #Points is the number of data points used for the training of each clustering method. #Clusters is the given number of clusters. Dimension is the dimension of given feature vector. %Largest cluster means the percentage of the largest cluster size to each size of dataset.

Dataset	#Points	#Clusters	Dimension	%Largest cluster
MNIST	70000	10	784	11%
Reuters-10k	10000	4	2000	43%
FC	10000	4	2	50%
TM	10000	2	2	50%
TR	10000	3	2	50%

to feature vectors by computing the TF-IDF features on the two-thousand most frequent words. This dataset contains 685,071 documents. In this experiment, random subsets of 10,000 samples from the full dataset are drawn. As for the synthetic datasets, we show the generated examples in Fig.11. An example of FC is the left picture in the figure. This dataset is composed of simple four clusters with some outliers. The cluster-balance is biased. The ratio of biggest cluster is 50% to the dataset, then 20%, 20% and 10%. The four clusters are close to each other. An example of TM is middle picture in the table. This dataset includes some outliers and its decision boundary is non-linear. The cluster-balance is uniform. The example of TR is the right picture in Table 11. This dataset is composed of three concentric rings. The ratio of outer ring to the whole size is 50%, and the middle and inner ones are 33% and 17%, respectively.

Regarding the evaluation metric, since we are in unsupervised learning scenario, we adopt the standard metric for evaluating clustering performance [57, 61], which measures how close the estimated clusters are to the ground truth. For an unlabeled dataset $\{\mathbf{x}_i\}_{i=1}^n$, let $\{y_i\}_{i=1}^n$ and $\{\hat{y}_i\}_{i=1}^n$ be its true label set and estimated label set, respectively. n is the number of data points. The clustering accuracy (ACC) is defined as follows:

$$\mathrm{ACC} = \max_{\tau} \frac{\sum_{i=1}^n \mathbf{1}[y_i = \tau(\hat{y}_i)]}{n}, \tag{36}$$

where τ ranges over all permutations between clusters and labels. The optimal assignment of τ can be computed using the Kuhn-Munkres algorithm [78].

11.2 Performance Evaluation of SGSC

We here show how accurately SGSC could estimate the cluster labels of five datasets in Table 12. Strictly speaking, SGSC of algorithm 3 has five hyperparameters, which are k_0, σ of Eq.28, k_1, h and γ. In this experiment, we nevertheless do not consider σ and γ as the hyperparameters. The reason is, when median heuristics and 10^{-60} were employed to σ and γ respectively, SGSC performed well throughout all datasets of Table 12 in our preliminary experiments. Thus, we consider the rest three as the hyperparameters.

Generally speaking, in unsupervised learning, it is difficult to tune hyperparameters because we cannot conduct cross-validation. However, by using the idea of transfer learning, we can ease the difficulty. Following [61], we tune the three aforementioned hyperparameters. Let Λ be the triplet (k_0, k_1, h). The best one is denoted by $\Lambda^* = (k_0^*, k_1^*, h^*)$. Now, Λ^* is defined as

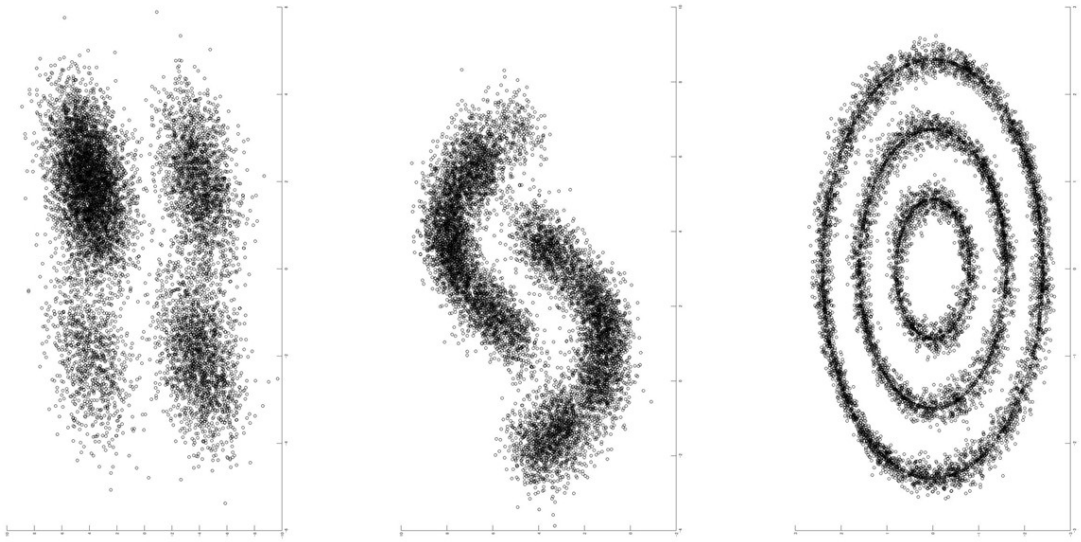

Figure 11: The generated examples of three commonly used synthetic datasets. The left, middle and right pictures correspond the examples of FC, TM and TR, respectively.

follows:

$$\Lambda^* = \underset{\Lambda}{\operatorname{argmax}} \sum_i \frac{\mathrm{ACC}(\Lambda, \mathrm{dataset}_i)}{\mathrm{ACC}(\Lambda^*_{\mathrm{dataset}_i}, \mathrm{dataset}_i)}, \tag{37}$$

where $\mathrm{dataset}_i$ is i-th source domain. $\Lambda^*_{\mathrm{dataset}_i}$ is the best hyperparameter for the $\mathrm{dataset}_i$. $\mathrm{ACC}(\Lambda, \mathrm{dataset}_i)$ is the clustering accuracy of Eq.(36) when the hyperparameter Λ is selected for $\mathrm{dataset}_i$. Let us call a given dataset by target domain. The source domain of each given dataset is shown in Table 13. USPS [36] is a dataset of hand-written digit. The number of clusters is ten. 20news [61] is a dataset of newsgroup documents, which is partitioned nearly evenly across 20 different newsgroups. The source domain of FC is FC, TM and TR. Note that FC, TM and TR used as sources have slightly different generating rules compared to target ones. The same things goes for the target TM and TR datasets. By using these source domains, we tune the hyperparameters.

As for the ranges of candidates with three hyperparameters, we define $k_0 \in \{10 \times i_0 \mid i_0 = 1, ..., 10\}$, $k_1 \in \{5 \times i_1 \mid i_1 = 1, ..., 5\}$ and $h \in \{100 \times i_2 \mid i_2 = 2, ..., 5\}$. By conducting tuning technique of Eq.(37) on each source domain, we obtained the best hyperparameters for given datasets of Table 12 as follows: The best ones for MNIST, Reuters-10k, FC, TM, and TR are $(k_0^*, k_1^*, h^*) = (10, 10, 500)$, $(100, 20, 200)$, $(10, 10, 500)$, $(10, 10, 500)$ and $(10, 10, 500)$ respectively. For an example, when $k_0 = 10$, we obtained the accuracy matrix for both MNIST and USPS shown in Table 14. According to this table, the best pair $(k_1^*, h^*) = (10, 500)$ of USPS is transferred to MNIST. Finally, based on the above best pair of hyperparameters, we conduct SGSC on each target dataset. The results are shown in Table 15.

Table 13: The source domain is a dataset we can use to tune our hyperparameters. Dimension means the dimension of feature vector with source dataset. #Points means the number of data points with source dataset. The target domain is a dataset which we want to cluster using knowledge from the source.

Source Domain	Dimension	#Points	Target Domain
USPS	256	11000	MNIST
20news	2000	10^4	Reuters-10k
(FC, TM, TR)	(2, 2, 2)	(10^4, 10^4, 10^4)	FC
(FC, TM, TR)	(2, 2, 2)	(10^4, 10^4, 10^4)	TM
(FC, TM, TR)	(2, 2, 2)	(10^4, 10^4, 10^4)	TR

Table 14: The accuracies of estimated labels with hub data points for both MNIST and USPS are shown. The labels are estimated by SGSC. k_1 and h mean the number of neighbors and the number of hub data points, respectively. Each accuracy is computed by the output $\{\hat{y}_i\}_{i=1}^n$ of SGSC using corresponding the pair (k_1, h). Another number of neighbors k_0 used in the SGSC is fixed to ten. The bold font below means the best accuracy for each dataset. Note that, since we use all data points of both MNIST and USPS for the estimation, no standard deviation is shown.

Dataset	h \ k_1	5	10	15	20	25
Target : **MNIST**	200	0.88	0.78	0.76	0.75	0.79
	300	0.45	**0.92**	0.91	0.90	0.89
	400	0.79	0.83	0.86	0.90	0.88
	500	0.39	0.88	0.86	0.86	0.89
Source : **USPS**	200	0.59	0.49	0.51	0.44	0.46
	300	0.49	0.65	0.49	0.52	0.48
	400	0.65	0.61	0.63	0.48	0.50
	500	0.68	**0.69**	0.64	0.65	0.53

11.3 Performance Evaluation of SEDC

We here show how accurately SEDC can cluster the five given datasets of Table 12. First of all, let us discuss the implementation. Across all the datasets, we define the network structure of $p_\theta(y|\mathbf{x})$ by D-1200-1200-C, where D and C are the dimension of feature vector and the number of clusters, respectively. We used ReLU [79] for all the activation functions, and employed batch normalization technique [80] on each layer. For the Adam optimizer, we set the learning rate to 0.002. Following [81], we initialized the bias term and the weights of directed edges in the deep neural network as follows: each weight is initialized by the value of a Gaussian distribution with a mean of 0, and standard deviation of $\delta \times \sqrt{2/\mathrm{f_{in}}}$, where $\mathrm{f_{in}}$ is the number of input neurons. We set the δ to 10^{-1}-10^{-1}-10^{-4} for weight matrices from the input to the output. The all bias terms were initialized to 0. The number of epoch is fixed to 25, and each epoch is made by 100 iterations. As for the mini-batch with Eq.(35), in each iteration, we sample $h/10$ and $(n-h)/100$ data points from the pseudo labeled dataset and the unlabeled

Table 15: The mean accuracy and standard deviation obtained by SGSC using best hyperparameters are shown. These numbers are averaged number over seven times experiments. Since we use all samples of MNIST for the estimation, no standard deviation is shown.

MNIST	Reuters-10k	FC	TM	TR
0.88	0.78(0.06)	0.94(0.04)	0.98(0.01)	0.97(0.02)

Table 16: The mean clustering accuracy (ACC) of Eq.(36) and standard deviation are shown. Five popular clustering methods and our proposed method were tested on five datasets. For each method, Average means the averaged ACC over the five datasets. The experiments were conducted seven times on each pair of method and dataset.

Method	MNIST	Reuters-10k	FC	TM	TR	Average
k-means	0.53	0.53(0.04)	0.60(0.05)	0.64(0.04)	0.35(0.03)	0.53
SC	0.72	0.62(0.03)	0.80(0.04)	0.85(0.03)	0.96(0.03)	0.79
IMSAT	**0.98**	0.71(0.05)	0.70(0.04)	0.66(0.05)	0.34(0.01)	0.68
DEC	0.84	**0.72(0.05)**	0.72(0.04)	0.67(0.03)	0.48(0.04)	0.69
SpectralNet	0.83	0.67(0.03)	0.79(0.03)	0.87(0.02)	**0.99(0.01)**	0.83
SEDC	0.89	**0.73(0.05)**	**0.95(0.03)**	**0.96(0.02)**	**0.99(0.00)**	**0.90**

dataset, respectively. The pseudo labeled samples are used for approximating the second term of Eq.(35), and both pseudo labeled and unlabeled samples are used for approximating the first and third terms of Eq.(35). Moreover, in VAT, we set ξ to ten in Eq.(32). The radius ϵ is defined by the same way as [61]: see Remark 11.

With respect to the selection of hyperparameters in SEDC, since we have already finished to tune k_0, k_1 and h in previous sub-section, we only focus on the remaining hyperparameters λ_1 and λ_2. The tuning tactic is also based on Eq.(37), and the source domains of Table 13 are used for the tuning. The ranges of candidates are defined by as follows: $\lambda_1 \in \{0.1 \times j_1 \mid j_1 = 1, ..., 10\}$, $\lambda_2 \in \{0.1 \times j_2 \mid j_1 = 1, ..., 10\}$. After the tuning, we obtained $\lambda_1 = \lambda_2 = 1$ as the best ones for all datasets. By using the tuned hyperparameters and computing the SEDC, we get the results shown in Table 16.

As we can see in Table 16, our method averagely performs better than other methods. One of the reason is that we do not require key conditions to SEDC. For an example, IMSAT does not perform well for datasets with non-uniform cluster-balance such as TR. In addition to the wide applicability, another reason lies on the robustness against outliers. When we see the performance of SEDC on FC which includes outliers, the method is more robust against outliers compared to SpectralNet. In fact, SpectralNet suffered from the two datasets which include outliers. Note that the hyperparameter tuning of SEDC hugely contributes to the robustness. On the other hand, if we see some columns of Tabel 16 such as MNIST and Reuters-10k, SEDC does not outperform IMSAT. The clustering accuracy of IMSAT with MNIST is known to be one of the best results among several deep clustering methods [82]. In addition to VAT regularization, the almost perfect prior knowledge of cluster balance with MNIST seems to greatly help IMSAT achieve such the result. Though our method employs similar objective

Table 17: The corresponding mean runtime (seconds) and standard deviation of Table 16 are shown. Five popular clustering methods and our proposed method were tested on five datasets. For each method, Average means the averaged runtimes over the five datasets. The experiments were conducted seven times on each pair of method and dataset. Note that the runtimes of SC, IMSAT, SpectralNet and SEDC do not include the runtimes of hyperparameter tuning. As for DEC, the runtime does not include the runtime of pre-training. The bold font means the fastest runtime among six methods.

Method	MNIST	Reuters-10k	FC	TM	TR	Average
k-means	**136**	**30(2.3)**	**0.04(0.0)**	**0.05(0.0)**	**0.03(0.0)**	**33.2**
SC	96362	473(5.7)	417(4.3)	408(5.1)	413(4.9)	19614
IMSAT	5097	749(8.2)	429(5.0)	424(4.7)	428(3.8)	1425
DEC	1274	258(3.1)	121(4.9)	135(5.2)	153(6.6)	388
SpectralNet	2239	232(2.4)	122(2.0)	110(2.3)	117(2.5)	564
SEDC	3466	440(4.4)	226(1.9)	233(3.0)	237(3.4)	920

Table 18: The clustering accuracy obtained by SEDC using the tuned hyperparameters. The top row is the number of unlabeled data points used for SEDC. The MNIST dataset is used for all this experiments. The bottom row is the clustering accuracy.

10000	20000	30000	40000	50000	60000
0.786	0.823	0.848	0.853	0.870	0.880

function with IMSAT, since we use the estimated cluster labels, the estimation error degraded the performance of SEDC. As for Reuters-10k, we can list two reasons why SEDC does not outperform IMSAT well. The first one is that the number of given data points is not enough since the geodesic metric is approximated by the graph shortest path distance in SEDC. In fact, we observed that, by using 20000 unlabeled data points with Reuters, the clustering accuracies of SEDC and IMSAT were 0.739 (0.06) and 0.696 (0.05), respectively. These accuracies were the average of seven times experiments. The second one is that the source domain of Reuters-10k might be not appropriate since the cluster balances of 20new and Reuters are uniform and non-uniform, respectively.

Moreover, our method has the following favorable property: If the number of unlabeled data points increases, the clustering accuracy also increases. This property is important since, first, not many clustering methods seem to not have this property and, second, we have higher possibility to obtain the unlabeled data point than obtain the labeled ones. As we can see from Table 18, more unlabeled data points are used in the learning, higher accuracy we can get.

In Table 17, we show the corresponding mean runtime and standard deviation of Table 16. As we can expect, k-means is the fastest clustering method among the six. The runtime of SC is quite long especially when the size of dataset is large. The most heavy computation is in solving the eigenvalue problem. IMSAT is the second slowest method in the six. Since this method require the k-NN graph for defining the adaptive radius with VAT: see Remark 11, this computation is dominant in the whole procedure. In addition to the time of graph construction,

relatively larger number of epochs (50 epochs) for training the deep neural network also affected the total runtime. DEC is the fastest method among the deep clustering methods. After the pre-training, DEC simply trains the deep neural network by using mini-batchs until the convergence. Note that the shown runtimes of DEC does not include the runtimes consumed on the pre-training. If we combine the runtime of that pre-training with shown ones, the total runtimes of DEC will be much longer: see sub-section 4.3 of [57]. SpectralNet also computes k-NN graph. Therefore, the dominant part of computation is the graph construction. In addition to this constructing time, since the number of epochs for training the two deep neural networks are not large, the total runtime of SpectralNet is relatively fast among the four deep clustering methods. Regarding with SEDC, as we already mentioned in sub-section 10.3, the dominant part is computing k-NN graph. In addition to this computing, since we set 25 epochs to the training, the total runtimes is medium.

12 Conclusion of Part II

In this paper, we propose a deep clustering method named SEDC. Given an unlabeled dataset and the number of clusters, the method groups the dataset into the given number clusters. Regarding its advantages, it does not require an additional condition except two fundamental assumptions: smoothness and manifolds assumptions. In this point, only SpectralNet of Table 11 is comparable. In addition, SEDC also can be applied to various data domains since it does not have preferred data domains, as long as raw data is transformed to feature vectors. Furthermore, the performance of SEDC can be robust against existence of outliers unlike SpectralNet. According to these advantages, our proposed method can be expected to averagely perform better than previous deep clustering methods. As a result, this expectation is empirically confirmed by conducting numerical experiments on five commonly used datasets: see Table 16. Therefore, we think our method can be a competitive candidate for users in some practical clustering scenarios where prior knowledge of the given unlabeled dataset is limited.

Let us then discuss two limitations of SEDC. On the one hand, since the method needs hyperparameter tuning, if we do not have appropriate labeled source domains to learn them from and transfer, then it may fail. On the other hand, since the method requires the number of clusters, it does not work for datasets where nothing is known on the number of clusters such as genome datasets.

Finally, we discuss about our two future works. The first one is to invent a more noise-robust semi-supervised learning framework and then apply it to SEDC instead of Eq.(35). Since some of the estimated cluster labels by SGSC are not perfectly accurate, we need to invent such the framework to stabilize the performance of SEDC. The second one is to modify our method for handling structured data, i.e., graph data or sequential data.

13 Concluding Remarks

Throughout of this thesis, we investigate the online graph-based SSL and deep clustering. In the former learning, we could succeed to propose the promising offline graph-based SSL method named RLP which can assist in creating a new online graph-based SSL algorithm by combining a conventional online clustering method such as doubling algorithm. From the numerical experimental point of view, the proposed online graph-based SSL method named online QRLP outperformed the previous method since RLP can be more robust against outliers than LP. In the later, we could succeed to propose the widely applicable deep clustering method named SEDC which does not require additional key conditions. This applicability makes SEDC become the competitive candidate in practical situation. From the numerical experimental point of view, our deep clustering method averagely performs better than previous popular deep clustering methods.

Surely we can be pleased at this moment since our proposed method outperformed previous popular methods. However, we should know that there are still many problems left. With the proposed online graph-based SSL study, the biggest weakness is maybe the application to the real-world problem where the quite high prediction accuracy is required such as automation driving problems. Regarding these types of problems, we may have take completely different approach to earn much higher accuracy than the graph-based SSL. Possible approach is life-long learning [83] combined with active learning [84] and deep neural networks.

As for the proposed deep clustering, still we could not solve the curse-of-dimension problem like other methods. Probably, as long as we use the naively defined feature vectors or take the k-NN based strategy, we can not solve it. One possible hope is the representation learning [85]. This learning tries to extract core feature of raw data, perhaps, like human beings, and at this moment, for the image dataset, the machine learning community seems to get some success. We guess this filed will be more intense in the near future since we can obtain tons of unlabeled dataset. If we human can succeed the representation learning in several domains, by using simple classical clustering algorithm, we may get several competitive results.

References

[1] X. Zhu, "Semi-supervised learning literature survey," *Computer Science, University of Wisconsin-Madison*, vol. 2, no. 3, p. 4, 2006.

[2] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using em," *Machine Learning*, vol. 39, no. 2-3, pp. 103–134, 2000.

[3] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[4] X. J. Zhu, "Semi-supervised learning literature survey," tech. rep., University of Wisconsin-Madison Department of Computer Sciences, 2005.

[5] T. Joachims, "Transductive inference for text classification using support vector machines," in *International Conference on Machine Learning*, vol. 99, pp. 200–209, 1999.

[6] O. Chapelle and A. Zien, "Semi-supervised classification by low density separation.," in *AISTATS*, vol. 2005, pp. 57–64, 2005.

[7] V. Sindhwani, S. S. Keerthi, and O. Chapelle, "Deterministic annealing for semi-supervised kernel machines," in *Proceedings of the 23rd International Conference on Machine Learning*, pp. 841–848, ACM, 2006.

[8] R. Collobert, F. Sinz, J. Weston, and L. Bottou, "Trading convexity for scalability," in *Proceedings of the 23rd International Conference on Machine Learning*, pp. 201–208, ACM, 2006.

[9] O. Chapelle, V. Sindhwani, and S. S. Keerthi, "Branch and bound for semi-supervised support vector machines," in *Advances in Neural Information Processing Systems*, pp. 217–224, 2007.

[10] L. Xu and D. Schuurmans, "Unsupervised and semi-supervised multi-class support vector machines," in *AAAI*, vol. 5, p. 13, 2005.

[11] M. Belkin, I. Matveeva, and P. Niyogi, "Regularization and semi-supervised learning on large graphs," in *International Conference on Computational Learning Theory*, pp. 624–638, Springer, 2004.

[12] M. Belkin, P. Niyogi, and V. Sindhwani, "On manifold regularization.," in *AISTATS*, p. 1, 2005.

[13] A. Blum and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts," 2001.

[14] T. Wagner, S. Guha, S. P. Kasiviswanathan, and N. Mishra, "Semi-supervised learning on data streams via temporal label propagation," in *International Conference on Machine Learning*, pp. 5082–5091, 2018.

[15] A. B. Goldberg, M. Li, and X. Zhu, "Online manifold regularization: A new learning setting and empirical study," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 393–407, Springer, 2008.

[16] G. Krempl, I. Zliobaite, D. Brzeziński, E. Hüllermeier, M. Last, V. Lemaire, T. Noack, A. Shaker, S. Sievi, M. Spiliopoulou, *et al.*, "Open challenges for data stream mining research," *ACM SIGKDD Explorations Newsletter*, vol. 16, no. 1, pp. 1–10, 2014.

[17] L. Huang, X. Liu, B. Ma, and B. Lang, "Online semi-supervised annotation via proxy-based local consistency propagation," *Neurocomputing*, vol. 149, pp. 1573–1586, 2015.

[18] S. Ravi and Q. Diao, "Large scale distributed semi-supervised learning using streaming approximation," in *Artificial Intelligence and Statistics*, pp. 519–528, 2016.

[19] M. Valko, B. Kveton, L. Huang, and D. Ting, "Online semi-supervised learning on quantized graphs," in *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, UAI'10, (Arlington, Virginia, United States), pp. 606–614, AUAI Press, 2010.

[20] M. Charikar, C. Chekuri, T. Feder, and R. Motwani, "Incremental clustering and dynamic information retrieval," *SIAM Journal on Computing*, vol. 33, no. 6, pp. 1417–1440, 2004.

[21] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," 2002.

[22] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the 20th International Conference on Machine Learning*, pp. 912–919, 2003.

[23] Y. Tao, R. Triebel, and D. Cremers, "Semi-supervised online learning for efficient classification of objects in 3d data streams," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2904–2910, IEEE, 2015.

[24] A. S. Bijral, N. Ratliff, and N. Srebro, "Semi-supervised learning with density based distances," *arXiv preprint arXiv:1202.3702*, 2012.

[25] A. Moscovich, A. Jaffe, and B. Nadler, "Minimax-optimal semi-supervised regression on unknown manifolds," *arXiv preprint arXiv:1611.02221*, 2016.

[26] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[27] B. Kveton, M. Philipose, M. Valko, and L. Huang, "Online semi-supervised perception: Real-time learning without explicit feedback," in *IEEE Conference on Computer Vision and Pattern Recognition Workshop*, pp. 15–21, IEEE Computer Society, 2010.

[28] X. Zhu, A. B. Goldberg, and T. Khot, "Some new directions in graph-based semi-supervised learning," in *IEEE International Conference on Multimedia and Expo*, pp. 1504–1507, IEEE, 2009.

[29] W. Dong, C. Moses, and K. Li, "Efficient k-nearest neighbor graph construction for generic similarity measures," in *Proceedings of the 20th International Conference on World Wide Web*, pp. 577–586, ACM, 2011.

[30] M. Karasuyama and H. Mamitsuka, "Manifold-based similarity adaptation for label propagation," in *Advances in Neural Information Processing Systems*, pp. 1547–1555, 2013.

[31] S. Har-Peled, "Computing the k nearest-neighbors for all vertices via dijkstra," *arXiv preprint arXiv:1607.07818*, 2016.

[32] D. Wang, L. Shi, and J. Cao, "Fast algorithm for approximate k-nearest neighbor graph construction," in *2013 IEEE 13th International Conference on Data Mining Workshops*, pp. 349–356, IEEE, 2013.

[33] Y. Zhang, K. Huang, G. Geng, and C. Liu, "Fast knn graph construction with locality sensitive hashing," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 660–674, Springer, 2013.

[34] J. A. Aslam, E. Pelekhov, and D. Rus, "The star clustering algorithm for static and dynamic information organization," *Journal of Graph Algorithms and Applications*, vol. 8, pp. 95–129, 2004.

[35] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 643–660, 2001.

[36] H. Wechsler, J. P. Phillips, V. Bruce, F. F. Soulie, and T. S. Huang, *Face Recognition: From Theory to Applications*, vol. 163. Springer Science & Business Media, 2012.

[37] S. A. Nene, S. K. Nayar, H. Murase, *et al.*, "Columbia object image library (coil-20)," 1996.

[38] A. Asuncion and D. J. Newman, "Uci machine learning repository [http://www. ics. uci. edu/˜ mlearn/mlrepository. html]. irvine, ca: University of california," *School of Information and Computer Science*, vol. 12, 2007.

[39] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[40] Y. Yamaguchi and K. Hayashi, "When does label propagation fail? a view from a network generative model.," in *IJCAI*, pp. 3224–3230, 2017.

[41] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[43] R. N. D'souza, P.-Y. Huang, and F.-C. Yeh, "Small data challenge: Structural analysis and optimization of convolutional neural networks with a small sample size." Cold Spring Harbor Laboratory, 2018. BioRxiv.

[44] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping Multidimensional Data*, pp. 25–71, Springer, 2006.

[45] R. Xu and D. C. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on Neural Network*, vol. 16, pp. 645–678, 2005.

[46] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–297, Oakland, CA, USA, 1967.

[47] N. E. Day, "Estimating the components of a mixture of normal distributions," *Biometrika*, vol. 56, no. 3, pp. 463–474, 1969.

[48] M. Girolami, "Mercer kernel-based clustering in feature space," *IEEE Transactions on Neural Networks*, vol. 13, no. 3, pp. 780–784, 2002.

[49] J. Wang, J. Lee, and C. Zhang, "Kernel trick embedded gaussian mixture model," in *International Conference on Algorithmic Learning Theory*, pp. 159–174, Springer, 2003.

[50] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems*, pp. 849–856, 2002.

[51] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[52] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[53] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[54] O. Chapelle and A. Zien, "Semi-supervised classification by low density separation.," in *AISTATS*, pp. 57–64, 2005.

[55] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering," in *28th AAAI Conference on Artificial Intelligence*, 2014.

[56] P. Huang, Y. Huang, W. Wang, and L. Wang, "Deep embedding network for clustering," in *22nd International Conference on Pattern Recognition*, pp. 1532–1537, IEEE, 2014.

[57] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International Conference on Machine Learning*, pp. 478–487, 2016.

[58] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational deep embedding: An unsupervised and generative approach to clustering," *arXiv preprint arXiv:1611.05148*, 2016.

[59] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5147–5156, 2016.

[60] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[61] W. Hu, T. Miyato, S. Tokui, E. Matsumoto, and M. Sugiyama, "Learning discrete representations via information maximizing self-augmented training," in *International Conference on Machine Learning*, pp. 1558–1567, 2017.

[62] U. Shaham, K. Stanton, H. Li, B. Nadler, R. Basri, and Y. Kluger, "Spectralnet: Spectral clustering using deep neural networks," *arXiv preprint arXiv:1801.01587*, 2018.

[63] U. Shaham and R. R. Lederman, "Learning by coincidence: Siamese networks and common variable learning," *Pattern Recognition*, vol. 74, pp. 52–63, 2018.

[64] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[65] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete algorithms*, pp. 1027–1035, Society for Industrial and Applied Mathematics, 2007.

[66] T. Miyato, S. Maeda, S. Ishii, and M. Koyama, "Virtual adversarial training: a regularization method for supervised and semi-supervised learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[67] P. Bachman, O. Alsharif, and D. Precup, "Learning with pseudo-ensembles," in *Advances in Neural Information Processing Systems*, pp. 3365–3373, 2014.

[68] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[69] Y. Tao, K. Takagi, and K. Nakata, "Rdec: Integrating regularization into deep embedded clustering for imbalanced datasets," in *Asian Conference on Machine Learning*, pp. 49–64, 2018.

[70] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley & Sons, 2012.

[71] S. Dasgupta and S. Kpotufe, "Optimal rates for k-nn density and mode estimation," in *Advances in Neural Information Processing Systems*, pp. 2555–2563, 2014.

[72] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 11, pp. 2579–2605, 2008.

[73] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[74] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Advances in Neural Information Processing Systems*, pp. 529–536, 2005.

[75] A. Krause, P. Perona, and R. G. Gomes, "Discriminative clustering by regularized information maximization," in *Advances in Neural Information Processing Systems*, pp. 775–783, 2010.

[76] W. Dong, C. Moses, and K. Li, "Efficient k-nearest neighbor graph construction for generic similarity measures," in *Proceedings of the 20th International Conference on World Wide Web*, pp. 577–586, ACM, 2011.

[77] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "Rcv1: A new benchmark collection for text categorization research," *Journal of Machine Learning Research*, vol. 5, no. 4, pp. 361–397, 2004.

[78] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[79] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning*, pp. 807–814, 2010.

[80] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[81] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015.

[82] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long, "A survey of clustering with deep learning: From the perspective of network architecture," *IEEE Access*, vol. 6, pp. 39501–39514, 2018.

[83] Z. Chen and B. Liu, "Lifelong machine learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 10, no. 3, pp. 1–145, 2016.

[84] B. Settles, "Active learning literature survey," tech. rep., University of Wisconsin-Madison Department of Computer Sciences, 2009.

[85] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," *arXiv preprint arXiv:1808.06670*, 2018.