PAPER

Robust Label Prediction via Label Propagation and Geodesic *k*-Nearest Neighbor in Online Semi-Supervised Learning

Yuichiro WADA[†], ***Student Member***, **Siqiang SU**[††], **Wataru KUMAGAI**[†††], ***Nonmembers***, ***and*** **Takafumi KANAMORI**[†††,††††a)], ***Member***

SUMMARY This paper proposes a computationally efficient offline semi-supervised algorithm that yields a more accurate prediction than the label propagation algorithm, which is commonly used in online graph-based semi-supervised learning (SSL). Our proposed method is an offline method that is intended to assist online graph-based SSL algorithms. The efficacy of the tool in creating new learning algorithms of this type is demonstrated in numerical experiments.

key words: *semi-supervised learning, label propagation, manifold learning, online learning, geodesic distance*

1. Introduction

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 [28].

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. These problems often occur in the real world [8], [13], [23], and are known as online graph-based SSL problems.

Online graph-based SSL is a relatively new alternative to traditional SSL. Although many online SSL algorithms have been proposed, most of them [11], [19] do not consider the processing time and severe memory constraints. Consequently, runtime and memory demands are increasing functions $\Omega(T)$ of streaming size T. Studies that account for processing time and memory constraints [21]–[23] adopt a similar overall strategy. At each time, the data adjacency graph is recompressed after incorporating a newly arrived data point. Thereafter, the new data point on the graph is assigned a predicted label. The first and second steps are called graph compression and offline graph-based SSL, respectively. The second step employs the *label propagation* (LP) algorithm [29], [30].

We propose an offline graph-based SSL algorithm as a core tool for building online algorithms. Our approach modifies the LP algorithm and combines it with the *geodesic k-nearest neighbor* (G*k*NN) algorithm [3], [17]. The proposed method yields a more accurate prediction with less computational cost then LP. Consequently, we can construct more sophisticated online graph-based SSL algorithms than existing studies.

The remainder of this paper is organized as follows. Section 2 briefly introduces existing SSL studies. Section 3 proposes a learning method for SSL and analyzes its time complexity. Section 4 shows the utility of our algorithm in building an efficient online algorithm for SSL. Section 5 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 6 contains concluding remarks.

2. Related Works

We introduce the three popular SSL algorithms, i.e., Label Propagation (LP) [29], [30], G*k*NN [3], [17], and online quantized LP (online QLP) [22]. The first two methods are mainly used in offline settings, whereas the last third method is adopted in online learning.

In the offline scenario of the SSL algorithm, the labeled data set $L = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and the unlabeled data set $U = \{\mathbf{x}_i\}_{i=l+1}^{l+u}$ are observed, where $\mathbf{x}_i \in \mathbb{R}^p$ is the feature vector, and $y_i \in \{1, \ldots, S\}$ 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. Table 1 shows the other notations used throughout this paper.

2.1 Label Propagation

LP attempts to propagate the label information along with the data adjacency graph, which is often defined by the k-nearest neighbor (k-NN) graph. The LP algorithm 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

Manuscript received December 13, 2018.
Manuscript revised March 18, 2019.
Manuscript publicized April 26, 2019.

[†]The author is with Nagoya University, Nagoya-shi, 464–8601 Japan.
[††]The author is with The Hong Kong Polytechnic University, Hung Hom, Kowloon, 999077, Hong Kong.
[†††]The authors are with RIKEN AIP, Tokyo, 103–0027 Japan.
[††††]The author is with Tokyo Institute of Technology, Tokyo, 152–8552 Japan.
a) E-mail: kanamori@c.titech.ac.jp

DOI: 10.1587/transinf.2018EDP7424

Table 1 Abbreviations and notations.

LP	Label propagation
GkNN	k-Nearest neighbor with geodesic distance
(Online) QLP	(Online) Quantized LP
RLP	Robust label prediction
(Online) QRLP	(Online) Quantized robust label prediction
d	A metric
d_G	Geodesic distance on graph G
k_v	#Samples in the majority vote of GkNN in RLP
n_c	Maximum number of nodes in compressed graph
m, $\mathbf{m}$	Multiplicity and its vector representation
H, h	Hub set and its size

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 $l \times S$ matrix whose (i, j) element is one for $y_i = j$ and zero otherwise. Then, $\mathbf{Y}_u = (Y_{is})$ is given by

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

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 $\hat{y}_i = \operatorname{argmax}_{1 \le s \le S} Y_{is}$.

The label prediction $\widehat{U}$ by the LP is denoted by $\widehat{U} = \mathsf{LP}(L, U, d, k, w)$. 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,*

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

The bandwidth σ is selected by the median or mean heuristics [20].

Remark 2. *A class of extended Gaussian kernels is also employed in graph-based methods; see [15], [31].*

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

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

The first and second terms in Eq. (3) are contributed by the brute-force construction of the k-NN graph [5] and the calculation of the $u \times u$ inverse matrix in (1), respectively. The edge weights w_{ij} must be properly defined in LP because they largely affect the final prediction accuracy [12]. In practice, the edge weights are computed by rather simple strategies such as Eq. (2). However, the time complexity given by (3) is a serious issue, particularly when $l \ll u$.

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.

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:

$$O\left(pn^2 + k_v(\log n + k_1)n\right), \tag{4}$$

where the cost of computing a Euclidean distance is $O(p)$. The first and second terms in (4) are contributed by the brute-force construction of the k_1-NN graph [5] and by algorithm 1 of [17], respectively. In this case, GkNN is computationally more efficient than LP. Moreover, the hyperparameter k_v can be efficiently chosen by heuristics [3], [17].

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

2.3 Online Quantized LP

Online QLP [22] combines label prediction by QLP with an online graph compression method called the doubling algorithm (DA) [4]. 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 in [4]. 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 1.

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

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

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 [22].

Let us consider the time complexity of online QLP. At each time, the time complexity of DA [4] based on the Euclidean distance is bounded by

$$O\left(pn_c \log n_c\right). \tag{5}$$

The time complexity of QLP [15] is the time complexity of LP (defined by (3)) over the compressed graph. Therefore, the worst-case time complexity of QLP is $O\left(pn^2 + n_c^3\right)$, where $n = l + n_c$. By summing up both time complexities, the time complexity of online QLP at each time is bounded by $O\left(pn^2 + n_c^3\right)$. 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 [15].

3. Robust Label Prediction Methods

3.1 Proposed Algorithm

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.,*

$$|\mathcal{N}_j|_0 = \sum_{\mathbf{x}_i \in \mathcal{N}_j} m(\mathbf{x}_i). \tag{6}$$

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

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 $\mathsf{LP}(L, H, d_G, k_2, w_2)$, 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: $\widehat{H} \cup \widehat{\bar{H}}$.

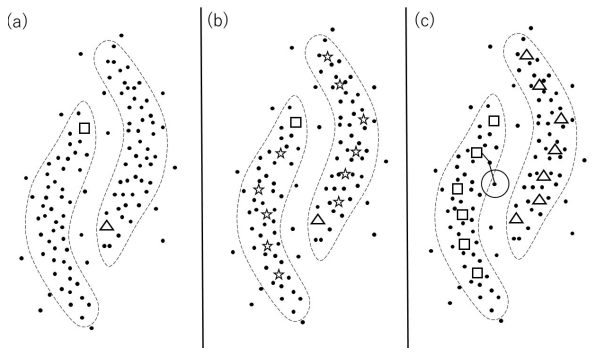

Fig. 1 This figure explains the working mechanism of the RLP algorithm. (a) The labeled and unlabeled data (black squares/triangles and black dots, respectively) are given in $\mathbb{R}^2$. The number of classes is two. The two banana shapes delineate the true data manifolds. (b) Line 2 of the RLP algorithm. 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.

- 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 [9], [17].

- 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 [3], [17].*

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.*

3.2 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:*

$$O\left(h^3 + pn^2 + k(k_1 + \log n)n\right), \tag{7}$$

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 $O(p)$ time, line 1 consumes $O(pn^2)$ time. Line 4 consumes $O(k(\log n + k_1)n)$ time, with $k = \max\{k_2, k_v\}$ (see [9], [17] 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.

3.3 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 $1 \le k_2 \le k_2^{\max}$. Thereafter, define the upper bound of h as follows:*

$$h^{\max} = \min\left\{\left(pn^2 + \kappa(k_1 + \log n)n\right)^{\frac{1}{3}}, u\right\}, \tag{8}$$

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

$$O\left(pn^2 + \kappa(k_1 + \log n)n\right). \tag{9}$$

Moreover, suppose that hyperparameters k_2 and h are tuned by K-fold cross-validation in the ranges $1 \le k_2 \le k_2^{\max}$ and $0 \le h \le h^{\max}$, 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:

$$O\left(Kpn^2 + cK\kappa(k_1 + \log n)n\right). \tag{10}$$

Proof. We first consider the worst-case time complexity of (7) in the range $1 \le k_2 \le k_2^{\max}$. In this case, the time complexity of RLP is bounded by the following:

$$O\left(h^3 + pn^2 + \kappa(k_1 + \log n)n\right). \tag{11}$$

By raising both sides of Eq. (8) to the third power, we obtain the following inequality for all h in $0 \le h \le h^{\max}$:

$$h^3 \le (h^{\max})^3 \le pn^2 + \kappa(k_1 + \log n)n. \tag{12}$$

By combining (11) and (12), the time complexity (9) is derived. Finally, given that there are $c \times K$ iterations of cross-validation, the time complexity (10) follows from the worst-case time complexity (9). 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 pn^2 is kept unchanged. □

In our numerical experiments of RLP, we conduct a five-fold cross-validation by using the $h^{\max}$ of Eq. (8) and the $k_2^{\max}$ set to 20.

Remark 6. *In the numerical experiments of Sect. 5, 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 [24] and [27], the time complexity of constructing the approximated k-NN graph is $O(pn \log n)$. Consequently, the computational cost of the RLP with an approximated k-NN is reduced to $O((p+\kappa)n \log n + \kappa k_1 n)$.*

4. Application of RLP to Online Scenario

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

4.1 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:

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

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.* (6) *of definition 1.*

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.

4.2 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. (8). Given an integer I, we define n_{c_i} by $\lfloor n_c/I \rfloor i$ for $i \in \{1, 2, \ldots, I\}$, 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 $1 \le k_2 \le k_2^{\max}$ and $1 \le h \le \min\{h^{\max}, n_{c_i}\}$, 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

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

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.

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*

$$O\left(pn^2 + \kappa(k_1 + \log n)n\right), \tag{13}$$

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

Proof. The time complexity (13) follows by summing time complexity (5) and (9). □

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 $O\left(\max\{p(|L_t| + n_c), (|L_t| + n_c)^2\}\right)$. 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 [1] into 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 Sect. 4.2 [*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: $\hat{y}_t$.

5. 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) [7], ORL, UMNIST [25], COIL (COIL-20) [18], Vowel [2], MNIST [16], optdigits [2], and USPS [25]. Table 2 shows the properties of each dataset.

5.1 Offline Experiments

We compared the performances of three methods (LP, GkNN, and RLP) on the eight real-world datasets described in Table 2. 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

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

	Yale	ORL	UMNIST	COIL
p	1024	10304	10304	16384
S	15	40	20	20
#	165	400	575	1439

	Vowel	MNIST	optdigits	USPS
p	12	784	64	256
S	2	10	2	10
#	1456	70000	5216	11000

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 $l = 4S$, 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)

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 $l = 4S$, respectively.

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

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 $\{h_i \mid 1 \le i \le 5, i \in \mathbb{N}\}$, where $h_i = i \times \lfloor h^{\max}/5 \rfloor$; see Eq. (8). 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.

Tables 6 and 7 show the averaged runtime with the standard deviation. The problem settings are the same as the experiments in Tables 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 (3) and (9). The GkNN and RLP are comparable in terms of the computational cost. This result agrees to the theoretical analysis in (4) and (9). 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 including the coefficient is

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)

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)

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 [26].

5.2 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:

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)

- In online QRLP, hyperparameters k_1 and k_v were set to four and three, respectively. According to the pretuning procedure in Sect. 4.2, $k_2^{\max}$ was set to 20, and $h^{\max}$ was computed by Eq. (8). 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.

5.3 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 [10], [14] typically have more than a million of parameters, while online QRLP needs only $O(\max\{p(n_c + l), (n_c + l)^2\})$ memory space.

As for the prediction accuracy, online QRLP is thought to be comparable to DNNs when the number of labeled data is small. [6] 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 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, Tables 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.

6. Conclusion

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

References

[1] J.A. Aslam, E. Pelekhov, and D. Rus, "The star clustering algorithm for static and dynamic information organization," J. Graph Algorithms Appl., vol.8, no.1, pp.95–129, 2004.
[2] A. Asuncion and D.J. Newman, "Uci machine learning repository," Irvine, CA: University of California, School of Information and Computer Science, 12, 2007.
[3] A.S. Bijral, N. Ratliff, and N. Srebro, "Semi-supervised learning with density based distances," Proc. Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, pp.43–50, 2011.
[4] 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.
[5] W. Dong, C. Moses, and K. Li, "Efficient k-nearest neighbor graph construction for generic similarity measures," Proc. 20th international conference on World wide web, pp.577–586, ACM, 2011.
[6] 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.
[7] 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 Trans. Pattern Anal. Mach. Intell., vol.23, no.6, pp.643–660, 2001.
[8] A.B. Goldberg, M. Li, and X. Zhu, "Online manifold regularization:

A new learning setting and empirical study," Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp.393–407, 2008.
[9] S. Har-Peled, "Computing the k nearest-neighbors for all vertices via dijkstra," arXiv preprint arXiv:1607.07818, 2016.
[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Proc. IEEE conference on computer vision and pattern recognition, pp.770–778, 2016.
[11] 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.
[12] M. Karasuyama and H. Mamitsuka, "Manifold-based similarity adaptation for label propagation," Advances in Neural Information Processing Systems, pp.1547–1555, 2013.
[13] G. Krempl, I. Žliobaite, D. Brzeziński, E. Hüllermeier, M. Last, V. Lemaire, T. Noack, A. Shaker, S. Sievi, M. Spiliopoulou, and J. Stefanowski, "Open challenges for data stream mining research," ACM SIGKDD explorations newsletter, vol.16, no.1, pp.1–10, 2014.
[14] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "Imagenet classification with deep convolutional neural networks," Advances in Neural Information Processing Systems, pp.1097–1105, 2012.
[15] B. Kveton, M. Philipose, M. Valko, and L. Huang, "Online semi-supervised perception: Real-time learning without explicit feedback," CVPR Workshops 2010, pp.15–21, IEEE Computer Society, 2010.
[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE, vol.86, no.11, pp.2278–2324, 1998.
[17] A. Moscovich, A. Jaffe, and N. Boaz, "Minimax-optimal semi-supervised regression on unknown manifolds," Proc. 20th International Conference on Artificial Intelligence and Statistics, pp.933–942, 2017.
[18] S.A. Nene, S.K. Nayar, H. Murase, et al., Columbia object image library (coil-20), 1996.
[19] S. Ravi and Q. Diao, "Large scale distributed semi-supervised learning using streaming approximation," Artificial Intelligence and Statistics, pp.519–528, 2016.
[20] B. Schölkopf and A.J. Smola, Learning with Kernels, MIT Press, Cambridge, MA, 2002.
[21] Y. Tao, R. Triebel, and D. Cremers, "Semi-supervised online learning for efficient classification of objects in 3d data streams," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp.2904–2910, 2015.
[22] M. Valko, B. Kveton, L. Huang, and D. Ting, "Online semi-supervised learning on quantized graphs," Proc. Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, pp.606–614, 2010.
[23] T. Wagner, S. Guha, S.P. Kasiviswanathan, and N. Mishra, "Semi-supervised learning on data streams via temporal label propagation," International Conference on Machine Learning, pp.5082–5091, 2018.
[24] D. Wang, L. Shi, and J. Cao, "Fast algorithm for approximate k-nearest neighbor graph construction," 13th IEEE International Conference on Data Mining Workshops, ICDM Workshops, TX, USA, Dec. 7-10, 2013, pp.349–356, IEEE Computer Society, 2013.
[25] 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.
[26] Y. Yamaguchi and K. Hayashi, "When does label propagation fail? A view from a network generative model," C. Sierra, ed., Proc. Twenty-Sixth International Joint Conference on Artificial Intelligence, pp.3224–3230, 2017.
[27] Y.-M. Zhang, K. Huang, G. Geng, and C.-L. Liu, "Fast knn graph construction with locality sensitive hashing," Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp.660–674, Springer, 2013.
[28] X. Zhu, "Semi-supervised learning literature survey," Computer Science, University of Wisconsin-Madison, vol.2, no.3, p.4, 2006.
[29] X. Zhu and Z. Ghahramani, "Learning from Labeled and Unlabeled Data with Label Propagation," CMU CALD tech report CMU-CALD-02-107, 2002.
[30] X. Zhu, Z. Ghahramani, and J.D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," Proc. 20th International conference on Machine learning (ICML-03), pp.912–919, 2003.
[31] X. Zhu, A.B. Goldberg, and T. Khot, "Some new directions in graph-based semi-supervised learning," IEEE International Conference on Multimedia and Expo, pp.1504–1507, 2009.

Yuichiro Wada is a Ph.D student from department of computer science and mathematical informatics of Nagoya University. He received his B.S. and M.S. degrees from Tokyo Institute of Technology, Japan, in 2009 and 2011, respectively. He is currently interested in machine learning.

Siqiang Su is a Year-4 bachelor student from department of applied mathematics of The Hong Kong Polytechnic University.

Wataru Kumagai is a scientific researcher at RIKEN Center for Advanced Intelligence Project. He received his B.S., M.S. and Ph.D. degrees from Tohoku University, Japan, in 2009, 2011 and 2013, respectively. His current research areas include machine learning, optimization and artificial intelligence.

Takafumi Kanamori is professor at Tokyo Institute of Technology, and team leader at RIKEN AIP. He received his B.S. and M.S. degrees from the University of Tokyo, and Ph.D. degrees from the Graduate University for Advanced Studies, Japan, in 1994, 1996 and 1999, respectively. He is currently working on mathematical statistics, machine learning and optimization.