May 2022

A Graph-based Learning Method and Its Capability to Handle Label Noise

Tor Nitayanont TORPONG_NITAYANONT@BERKELEY.EDU
Department of Industrial Engineering and Operations Research

Abstract

Supervised Normalized Cut (SNC) is a graph-based classifier that is based on the idea of Hochbaum's Normalized Cut (HNC). SNC relies on pairwise similarities between samples, with the assumption that samples that are similar tend to share the same label. However, it is often the case that some of the given labels are inaccurate, and SNC is not designed to handle this label noise. In this work, we propose a classifier called Label Confidence Supervised Normalized Cut (LC-SNC), where we introduce two modifications to SNC. The first modification, which directly tackles the issue of noisy labels, is the use of label confidence, which measures how likely each label is noisy. The second modification involves the balance between the two classes in the data. Experimental results show that our model achieves significant better performance than SNC, both with and without the presence of label noise. It also demonstrates decent performance compared to other models that make use of pairwise similarities such as the k-nearest neighbor classifier.

1. Introduction

In the past few decades, the field of machine learning has gained much popularity. One of the data structures that are used often is graph. Data in many domains are intrinsically structured as graph such as social network and transportation network. Most data from domains like computer vision and bioinformatics can also be easily captured by graphical structures. However, the use of graph in learning tasks is not limited to data with graphical structures. In general, any data can be described by a graph or embedded into a graph where each vertex in the graph corresponds to each sample, and the connectivity in the graph can be defined in various ways. In most cases, the connectivity or the weight of the edges between vertices captures the pairwise relationship between samples. In this work, we will investigate graph-based learning methods, for classification task in particular, where graph construction relies on pairwise similarities between samples. Most importantly, we investigate its efficiency in the presence of label noise.

Learning methods, whether it be supervised learning, unsupervised learning or semi-supervised learning, that utilize pairwise similarities vary from commonly used methods like the k-Nearest Neighbor classifier (kNN) and the Kernel Support Vector Machine (kSVM), in which kernel is analogous to pairwise relationship, to methods that make use of concepts in the field of network flow such as the normalized cut. In this work, we propose some development on a network flow or graph-based method called Supervised Normalized Cut (SNC), which we call Label Confidence SNC (LC-SNC) as it makes use of *confidence* measures on the given labels in the data. Our hypothesis is that, with the incorporation of label confidence, the model becomes more robust to noisy labels. We test it against the k-Nearest Neighbor classifier in our experiment.

The Supervised Normalized Cut method or SNC involves solving for a minimum cut on a predefined graph. The resulting cut determines the class of each sample, depending on which side of the cut the node of each sample belongs to. The idea of Supervised Normalized Cut is developed from the Hochbaum's Normalized Cut or HNC (Hochbaum, 2009). In HNC, a single positive-labeled sample and a single negative-labeled sample are preselected so that the corresponding nodes of these two samples are guaranteed to be in the source set and the sink set of the graph cut. In SNC, this idea was developed into a more supervised manner where the labels of all labeled samples are taken into account, rather than just the two preselected nodes. A more detailed explanation of SNC will be provided in Section 2.

Our contribution in this work are the two modifications that we introduce to the SNC method and are the essence of the LC-SNC method. 1) We incorporate the use of label confidence measure into the graph construction so that the method learns to vary its reliance on different labeled samples and 2) SNC method focuses mostly on pairwise similarities among positive samples but we allow the LC-SNC method to learn to adjust weights between the pairwise similarities in both classes.

Our experiments show that the LC-SNC method is more robust to noisy labels than the original SNC method, and many times, it outperforms the k-Nearest Neighbor classifier.

2. Preliminaries

2.1 Preliminaries: Cut and Normalized Cut

First, we explain a graph cut and the minimum cut problem.

Given an undirected graph $G = (V, E)$, and the weight w_{ij} assigned to each edge $[i, j] \in E$, a graph cut is a partition of vertices into two disjoint sets of vertices S and T such that $S \cup T = V$ and $S \cap T = \emptyset$. The capacity of this cut is defined as $C(S, T) = \sum_{i \in S, j \in T} w_{ij}$. In the case of an undirected graph, we consider all edges that connect any node in S and any node in T. However, for a directed graph, we only include the weights of directed edges (or arcs) that go from a node in S to another node in T. The minimum cut problem aims to find a partition of vertices into S and T such that the cut capacity is minimized. The minimum (s, t)-cut problem is similar to the minimum cut problem with the constraints that the two vertices s, often referred to as the source node, and t, or the sink node, must belong to the sets S and T, which are the source set and the sink set, respectively. In this work, when it is obvious that there are designated source and sink node, we would refer to the minimum (s, t)-cut problem as the minimum cut problem.

Before the idea of graph cuts was utilized more broadly in machine learning, it was more common to adopt it in some specific areas such as image segmentation as a graph partitioning problem. The work done by Wu and Leahy (1993) was among the first to apply the concept of graph cuts to data clustering. The data can be represented by an undirected graph of which vertices correspond to samples. Two samples that are considered neighbors are connected by an edge with a weight or a capacity that represents their similarities.

Wu and Leahy suggested that in order to partition samples into groups, we seek for the partition that minimizes the total weights on the cut between the groups, making it a minimum cut problem. According to their experimental results on images, in many images, the min-cut partition usually results in a highly unbalanced partition, with one group significantly smaller than the other one. This result is not surprising. For example, leaving a single sample in one partition can be a way to achieve a small cut capacity. However, in many cases, this type of partitioning is not meaningful.

The concept of normalized cuts was then introduced by Shi and Malik (2000) to mitigate this problem. Shi and Malik's work was also motivated by the problem of image partitioning. They believed that the partition of images had to be done hierarchically, from big picture downward to small regions. Similar to earlier works that adopted hierarchical data structures like trees, Shi and Malik took the graph-based approach where a big network comprised small pixels connecting together. The graph is constructed in a similar way to Wu and Leahy's work. As mentioned earlier, the minimum cut approach usually resulted in the partition that has a small set of vertices on one side. This is problematic because the small cut capacity could come naturally from the fact that there are very few vertices on one side, not always from the difference between the two sets of vertices. To fix this, Shi and Malik introduced the *normalized cut* where the objective function is as follows:

$$Ncut(S,T) = \frac{C(S,T)}{\sum_{i \in S} d_i} + \frac{C(S,T)}{\sum_{i \in T} d_i}$$

where $d_i = \sum_{j \in V} w_{ij}$.

The introduced denominators penalize the separation of only a few vertices. An alternative to the partition that minimizes the normalized interaction weights between the two groups, Shi and Malik also suggested that we could maximize the interaction within groups. It was shown that maximizing the following objective function of normalized intra-interaction is equivalent to minimizing $cut(S,T)$.

$$Nassoc(S,T) = \frac{\sum_{i \in S, j \in S} w_{ij}}{\sum_{i \in S} d_i} + \frac{\sum_{i \in T, j \in T} w_{ij}}{\sum_{i \in T} d_i}$$

However, a major disadvantage of the normalized cut minimization is that it is NP-complete. Shi and Malik only provided an approximate algorithm. Next, we will introduce a variant of normalized cut that can be solved in polynomial time.

One of the variants of normalized cuts that can be solved in polynomial time is the Hochbaum's normalized cut or HNC. In the work of Hochbaum (2009),the formulation of HNC was given as the minimization of

$$\frac{C(S,T)}{C(S,S)}$$

and it was shown in the same work that it is equivalent to minimizing $\frac{C(S,T)}{\sum_{i\in S} d_i}$, which is the first term of the objective function of the standard normalized cut.

$$\begin{aligned}
\arg\min_S \frac{C(S,T)}{C(S,S)} &= \arg\min_S \frac{C(S,T)}{\sum_{i\in S}\sum_{j\in S} w_{ij}} \\
&= \arg\min_S \frac{C(S,T)}{\sum_{i\in S} d_i - \sum_{i\in S}\sum_{j\in T} w_{ij}} \\
&= \arg\min_S \frac{C(S,T)}{\sum_{i\in S} d_i - C(S,T)} \\
&= \arg\max_S \frac{\sum_{i\in S} d_i - C(S,T)}{C(S,T)} \\
&= \arg\min_S \frac{C(S,T)}{\sum_{i\in S} d_i}
\end{aligned}$$

To solve the HNC problem in a polynomial time, it was formulated as a monotone integer programming problem where each inequality has up to three variables. Such integer programming format was proven to be solvable as a minimum cut problem on a relevant graph (Hochbaum, 2002). Note that two nodes (two samples) have to be preselected as *seed* nodes; one for the source set and one for the sink set.

In this paragraph, we will elaborate, to some extent, on how the HNC problem can be formulated as a minimum cut problem because there will be a connection to the Supervised Normalized Cut, which is explained in the next subsection (section 2.2) and is essential to our work. First, the HNC problem can be linearized: finding the minimum value of $\frac{C(S,T)}{\sum_{i\in S} d_i}$ is equivalent to finding the smallest λ such that

$$\min_S C(S,T) - \lambda\, C(S,S) \leq 0$$

The problem of minimizing $C(S,T) - \lambda\, C(S,S)$ is called $HNC(\lambda)$. Since minimizing $\frac{C(S,S)}{C(S,T)}$ is equivalent to minimizing $\frac{C(S,S)}{\sum_{i\in S} d_i}$, the $HNC(\lambda)$ problem can be written in an alternative form, which is to minimize $C(S,T) - \lambda\ \sum_{i\in S} d_i$. Therefore, to solve the HNC problem, we instead find the smallest λ such that the objective value of $HNC(\lambda)$ is non-positive.

First, we note that the $HNC(\lambda)$ problem for each λ can be solved as a graph minimum cut problem, on a directed graph in Figure 1.

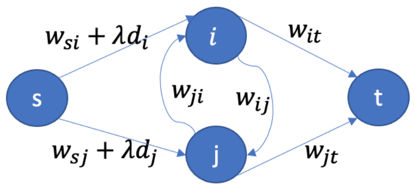

Figure 1: $HNC(\lambda)$ graph

To solve $HNC(\lambda)$, two nodes (or two samples) are preselected to be in different sets. We call them s and t. A graph is constructed as follows: for each $i \in V\backslash\{s,t\}$, there is a directed edge (s, i) with weight λd_i. For each $i, j \in V$ such that $i \neq j$, there are two arcs (i, j) and (j, i), each with weight w_{ij}. Given a graph cut (S, T) of this graph, its capacity is equal to $\sum_{i \in T} \lambda\, d_i + \sum_{i \in S, j \in T} w_{ij}$ Since $\sum_{i \in T} \lambda\, d_i + \sum_{i \in S, j \in T} w_{ij} = \sum_{i \in V} \lambda\, d_i - \sum_{i \in S} \lambda\, d_i + \sum_{i \in S, j \in T} w_{ij} = \text{constant} + \sum_{i \in S, j \in T} w_{ij} - \sum_{i \in S} \lambda\, d_i$, the partition (S, T) that minimizes the cut capacity of the constructed graph also minimizes $\sum_{i \in S, j \in T} w_{ij} - \sum_{i \in S} \lambda\, d_i$, which is the objective function of $HNC(\lambda)$.

The final step is to find λ such that the optimal objective value of the minimum cut problem is non-positive. Instead of searching over a range of values and apply methods like binary search, a parametric pseudoflow algorithm has been proposed to solve this parametric cut problem (Hochbaum, 2008). The complexity of this algorithm is similar to the complexity of solving a single minimum cut problem, which is $O(m^2 \log m)$ where m is the number of arcs in the $HNC(\lambda)$ graph.

The works that we discussed so far only applied the concept of graph partitioning mostly on image data and to solve the image partitioning problem specifically. In the next subsection, we will explore the use of the graph minimum cut in the context of (semi-)supervised learning.

2.2 Preliminaries: Cut as a learning method that relies on pairwise similarities

Aside from the area of image partition, graph cut has also been used in a more general context of machine learning. In particular, it is used as a tool to solve a classification problem, among many other classification methods that share one major characteristic with the graph cut approach, which is the reliance on pairwise similarities between samples. This class of methods include some prominent models such as the k-Nearest Neighbor classifier (kNN) and the kernel support vector machine ($kSVM$).

Before we proceed, let us define some relevant notations for a binary classification task that we will use for the rest of this work. L and U represent the set of labeled samples and the set of unlabeled samples, respectively. $L = \{(x_i, y_i)\}_{i=1}^{|L|}$ and $U = \{(x_i, y_i)\}_{i=|L|}^{|L|+|U|}$ where $x_i \in \mathbb{R}^d$ and $y_i \in \{1, -1\}$ for all $i \in [|L| + |U|]$.

We have seen in section 2.1 how the use of graph min-cut was primarily in partitioning task. It was later extended to the area of binary classification. The work of Blum and Chawla (2001) was among the first.

In the work of Blum and Chawla (2001), the graph can be constructed using vertices that represent samples in L and U, and two additional vertices s and t (the so-called source node and sink node) that represent the positive class and the negative class. Vertices that correspond to positive samples in L, or L^+, are connected to s with edges of infinite weight whereas vertices corresponding to negative samples in L, or L^-, are connected to t, also with edges of infinite weight. Finally, all pairs of vertices in $L \cup U$ are connected with edges whose weights are similarity measures between samples. A minimum cut with designated source and sink vertices s and t, denoted as minimum (s, t) cut, is a cut with minimum capacity among all cuts that have s on one side, which is called the source set S, and t on the other side, which is called the sink set T. To solve a binary classification problem on the constructed graph, we solve for the minimum (s, t) cut. In case that there are multiple minimum (s, t) cuts, we select the one with the smallest source set V^+. We remove all the edges in the cut to obtain two sets of vertices that are not disconnected. Unlabeled samples whose vertices belong to the source set are given positive labels whereas unlabeled samples that belong to the sink set get assigned negative labels.

The naive graph construction suggested that we connect all pairs of samples in $L \cup U$, resulting in a complete graph. Blum and Chawla (2001) provided several alternatives that sparsify the graph. These alternatives include (1) k-nearest neighbors sparsification: connecting each unlabeled sample only to its k nearest neighbors ($k = 3$ in their work) and (2) δ-neighborhood sparsification: connecting two samples if they are within distance δ from each other where δ is a parameter that has to be chosen. A scaling factor α was also introduced to scale the weights of edges that connect two unlabeled samples. It determines how much the classification result relies on the information from the unlabeled samples (Blum and Chawla, 2001). The kNN sparsification has proven to be more effective, and thus, more common, since the choice of ϵ can result in varying performance of the model across different datasets due to the scale of the data. Inappropriate value of ε may reuslt in disconnected graphs or singletons (Wang et al., 2013).

Notice that with kNN sparsification method, we might have a pair of samples (i, j) such that i is among the nearest neighbors of j, but j is not among the nearest neighbors of i. There are two approaches to handle this. (1.1) Symmetric kNN sparsification: construct an edge between samples if one of them is among the k nearest neighbors of the other one. (1.2) Mutual kNN sparsification: construct an edge only if both of them are among the nearest neighbors of each other (Sousa et al., 2013). The asymmetry can be resolved by these two approaches. However, it is still not guaranteed that each node has the same degree. This issue motivates another common approach, which is called (3) b-matching sparsification (Jebara et al., 2009). b-matching sparsification involves solving an optimization problem of edge subset selection where each node is required to have the same number of adjacent edges.

These sparsification methods are important as it has been shown that they improve the efficiency and accuracy of most graph-based methods, as well as robustness to noise (Wang et al., 2013). Moreover, graph sparsification results in a graph structure that reflects the topology of the data more accurately, compared to the complete graph generated without any sparsification.

Regarding the similarity weight, a common function is the Gaussian kernel, also known as the radial basis function kernel: $w_{ij} = exp(-\frac{\|x_i - x_j\|_2^2}{2\sigma^2})$ (Zhu and Goldberg, 2009). σ is a tunable parameter. When one of the sparsification method is used, it is also possible to use a binary weight where $w_{ij} = 1$ if i and j are connected in the sparsified graph, otherwise the weight is zero.

It is important to note that one major weakness of graph-based approach is how the model's performance can vary greatly depending on the construction of the graph as well as the edge weights. This is not limited to the minimum cut approach but also other learning methods in the graph-based class too, as we shall discuss later (Sousa et al., 2013).

Another way to formulate a classification task as a minimum cut problem is to utilize the HNC problem that we described in section 2.1. In HNC, we have one node designated as the source node s and another node as the sink node t before the the minimum (s, t) cut is solved. In the context of semi-supervised learning where labeled examples are given, the HNC problem can be made *supervised*. This is called Supervised Normalized Cut or SNC (Baumann et al., 2019). Given the set of labeled samples L, which is composed of samples with positive labels L^+ and samples with negative labels L^-, and the set of unlabeled samples L^-, in SNC, we would like to have all vertices of L^+ to be on one side of the cut and vertices of L^- on the other side. In other words, instead of having the seed source node s and the seed sink node t, we have two sets of vertices L^+ and L^- instead. To make it compatible with the minimum (s, t) cut problem where we restrict two particular nodes to be on the different partitions of the cut, we merge vertices of samples in L^+ into a single node, s, and merge the vertices of samples in L^- into a single node, t. The graph is illustrated in figure 2 where we have

1. directed edges from the source node s to all unlabeled samples where weight on the directed edge from s to sample i is $w_{si} + \lambda d_i = \sum_{j \in L^+} w_{ij}$
2. directed edges between two unlabaled samples where weights represent pairwise similarities, or w_{ij} for the pair (i, j)
3. directed edges from all unlabeled samples to the sink node t where weight on the directed edge from i to t is $w_{it} = \sum_{j \in L^-} w_{ij}$

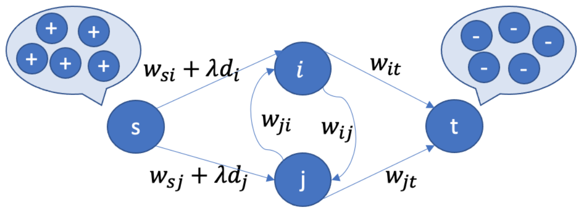

Figure 2: Graph of SNC whose minimum cut provided the predicted labels for the classification task. Nodes i and j represent two of the unlabeled samples.

Before we end this section about cut methods in classification task, it is important to note that in addition to its reliance on pairwise similarities between samples, these cut methods are categorized as *semi-supervised* learning methods. More details about the graph cut approach as a semi-supervised learning method will be elaborated later in Section 2.4 and when we discuss the future directions of this work.

2.3 Preliminaries: Methods that handle label noise

Among many imperfections that can be found in the training data or the labeled set of samples such as redundant features, imbalance between classes and the lack of sample points, label noise is another concerning issue that has a tremendous impact on the outcome of learning methods and receives attention from researchers in the area. In this work where we work on a binary classification task, labels of some samples might be corrupted by external factors. We observe negative labels for the corrupted positive samples and vice versa.

Frénay and Verleysen (2013) divided methods that deal with label noise in classification task into three groups: 1) conventional methods that are inherently robust to label noise such as ensemble methods 2) data cleansing / filtering methods that attempt to detect and either remove or relabel noisy labels such as editing methods in k-Nearest Neighbor classifier and 3) methods that model label noise in the data through embedding data cleansing into the learning algorithm or reduce the influence of label noise. In this work, we provide a brief overview of different techniques that have been used widely.

Methods in the first group are those where neither cleansing nor noise modeling is incorporated. The performance of these methods remain relatively effective when label noise is introduced to the data, yet they are not as robust to noise as methods in the other two groups. Examples of methods in this group include bagging and boosting techniques, ensembles of decision trees

Regarding the second group of which methods "clean" the dataset by removing instances that deem noisy, several methods fall into this category. One common approach is to come up with anomaly measures and remove instances whose values exceed a certain threshold. It is also common to filter out samples based on prediction results. For instance, robust-C4.5 is a method that removed instances that are misclassified by a pruned decision tree. Voting filtering can also be used to prevent excessive number of instances from being removed by using multiple classifiers and apply either consensus vote or majority vote to decide whether each instance is noisy or not. Another subclass of methods under the second group is the subclass of k-NN based methods, called editing methods, as the training set is edited. Edited nearest neighbor, repeated edited nearest neighbor, and All-kNN are editing methods that remove instances that are incorrectly classified. $k = 3$ is a common choice for the number of neighbors of these methods. Condense nearest neighbor is an editing method that results in a training set on which the prediction of all other instances are correct. Reduced nearest neighbor method removes all instances whose removal does not result in incorrect prediction of other samples.

For the methods that model label noise, there are two main classes, which are probabilistic methods and model-based methods. Probabilistic methods rely on the prior information about the distribution of the error rate. Common distributions include Beta and Dirichlet distributions. The probability that each label is noisy is then estimated accordingly, followed by relabeling those with high error estimates.

The method embodied in the Label Confidence Supervised Normalized Cut or LCSNC, which will be explain in the next section, falls into the second group and third group for several reasons. First, it relies on functions that measure how likely each label is noisy. Second, the LC-SNC method attempts to reduce the influence of noisy labels through smaller weight assignments to suspicious instances.

2.4 Preliminaries: Semi-supervised learning

In addition to its characteristic as a method that relies on pairwise similarities, SNC, LC-SNC and other graph cut methods are also considered as semi-supervised learning methods. Semi-supervised learning is a field in machine learning where both labeled samples and unlabeled samples are used to make predictions for the unlabeled samples. Semi-supervised learning methods are particularly applicable to, but not restricted to, the situations where the labels of samples cannot be obtained easily. However, even without the scarcity of labeled samples, semi-supervised learning can still extract useful additional information about the data from the unlabeled samples.

In this subsection, we provide a brief overview of topics in semi-supervised learning that is related to our work. The details in this part are not necessary to understand our proposed models. However, it can be beneficial when we think about the future directions of this work.

The field of semi-supervised classification was grouped in the survey by Van Engelen and Hoos (2020) into two main types: inductive methods and transductive methods. Inductive methods aim to produce a classification function or model that can predict the labels of any unseen samples whereas transductive methods provide labels specifically for the given unlabeled samples (Van Engelen and Hoos, 2020). To describe the two concepts with mathematical notation, we refer to the definition given by Zhu and Goldberg (2009).

Given a set of labeled samples $\{(x_i, y_i)\}_{i=1}^{l}$ and a set of unlabeled samples $\{x_i\}_{i=l+1}^{l+u}$

- an inductive semi-supervised learning learns a classifier $f : \mathcal{X} \to \mathcal{Y}$ that maps any instance from the feature space to the label space
- a transductive semi-supervised learning learns a classifier f that is expected to deliver correct labels specifically for the unlabeled samples $\{x_i\}_{i=l+1}^{l+u}$

Neither of these two types of semi-supervised learning methods is necessarily superior to the other one. While an inductive learner learns a function that is more generalized to the unseen data, the transductive learner is more customized to the unlabeled data that might be more of our interest. Either of them can be appropriate depending on the context.

According to Chapelle et al. (2006), most semi-supervised learning methods rely on some or all of the following assumptions: 1) *Smoothness assumption*: Samples that are close to each other should have the same labels, 2) *Low-density assumption*: Decision boundary should not go through high-density areas and 3) *Manifold assumption*: Samples on the same low-dimensional manifold should have the same labels.

The Supervised Normalized Cut or SNC method, and our proposed model, are semi-supervised transductive methods that mainly rely on the smoothness assumption. According to the survey, all semi-supervised transductive methods are graph-based. Two main components of graph-based methods are graph construction and graph inference.

Graph construction is done in a similar way as presented in section 2.1 where samples are connected by edges with pairwise similarities as weights. Regarding inference in graphs, there are approaches that assign probabilistic labels and methods that are hard label assignment. The minimum cut, SNC and LC-SNC are all hard-label assignment. Examples of methods that involve probabilisitc labels include Markov random fields and Label Propagation, which is an iterative algorithm that propagates labels from labeled samples to unlabaled samples according to the similarity weights, similar to a random walk with transition matrix.

In addition to graph-based methods, there are also other widely used semi-supervised methods in the groups of inductive methods and wrapper methods. We provide several of them in this section. An examples of inductive semi-supervised methods is the semi-supervised support vector machine or S^3VM, introduced by Bennett and Demiriz (1998). The objective of the supervised SVM is to find a decision boundary that separates samples of the two classes such that the distance from the boundary to the closest sample is minimized. This minimum distance marks the *margin* of the decision boundary, in which no samples should lie. A variant of the SVM, which is the soft-margin SVM allows some samples to be in the margin or be on the *incorrect* side of the decision boundary. The number of points that go into the margin and beyond is then penalized in the objective function. In S^3VM, the number of unlabeled samples that lie within the margin is an additional penalty that is added to the objective function.

Regarding the wrapper methods, they are based on a supervised learner of choice. These methods include three main types: co-training, self-training and boosting methods. Self-training method trains a supervised model on labeled samples before predicting the labels of unlabeled samples. It then selects unlabeled samples with the most confident predictions and add them to the labeled set before the process repeats. Co-training methods are similar, except that they are based on multiple supervised classifiers. Diversity of the base classifiers usually enhances the performance of the model. One common approach is the multi-view co-training where two base classifiers are trained on different subsets of features. The two classifiers select unlabeled samples with the most confident predictions separately and add the selected sets to the labeled set on one another. Finally, the boosting methods use mutiple base learners, but without the exchange of pseudo-labelled samples like co-training methods.

3. Label Confidence-Supervised Normalized Cut

3.1 Label Confidence

In this work, we propose a modification to the Supervised Normalized Cut method to handle the presence of noisy labels among the labeled samples. In the Supervised Normalized Cut, labeled samples are not explicitly represented as individual nodes. They are all consolidated with either the source node or the sink node. However, we can work on an equivalent graph where labeled samples are disintegrated from either the source or the sink nodes, depending on their labels, and infinite weights are placed on the edges that connect them to their respective source / sink nodes. As a result, weights on the edges that connect to unlabeled nodes from the source will have to be partially redirected to edges from labeled samples. The resulting graph is shown in Figure 3.

A minimum cut problem on the graph in Figure 3 is equivalent to that of the graph in Figure 2. In contrast to the graph in Figure 2, there are four sets of edges in the new graph.

1. Edges between any two nodes (such as the pairs (i, j),(i, l) and (i, m) in the figure), excluding the source and the sink node, whose weights are pairwise similarities

2. Edges between the source node s and labeled nodes with positive labels with infinite weights

3. Edges between labeled nodes with negative labels and the sink node t with infinite weights

4. Edges between the source node s and unlabeled nodes whose weights are the sums of weighted degrees $d_i = \sum_{j \in V} w_{ij}$ multiplied with λ

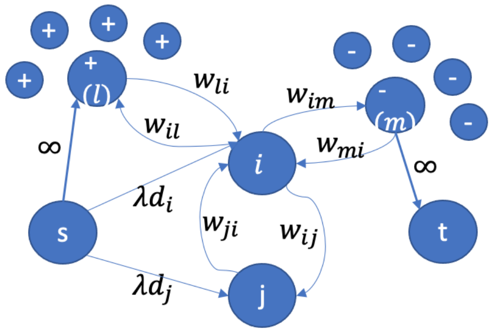

Figure 3: A graph that is equivalent to the SNC graph in Figure 2 where nodes of labeled samples are disintegrated from the source and the sink nodes

Consider the infinite weights that connect the source / sink and the labeled samples. These infinite weights restrict the positive labeled samples and negative labeled samples to belong to the source set and the sink set of the cut, respectively. However, when label noise is present, these constraints need to be relaxed. To handle the presence of noisy labels, we allow some labeled nodes to be on the side that is opposite to their given labels, depending on how likely the given label of each sample is corrupted, by replacing the infinite weights with finite values which we call *label confidence* since they reflect our confidence about the given labels. For a labeled sample $l \in L^+$, we denote the confidence that its positive label is correct by $conf_s(l)$. For a labeled sample with negative label $l \in L^-$, the confidence is denoted by $conf_t(l)$.

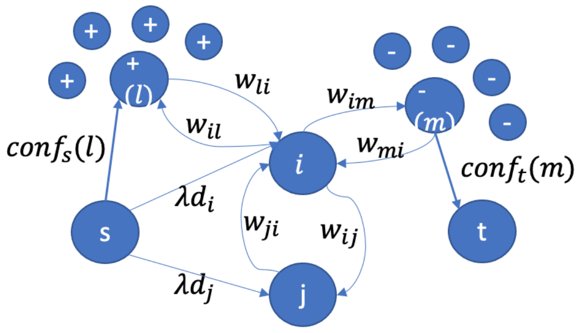

Figure 4: Graph of the LC-SNC method whose minimum cut provides prediction for the classification problem

Regarding the label confidence function, there are many ways to define them. In this work, we use three different label confidence functions as follow

1. Constant function

   $conf_s(l) = 1$ for $l \in L^+$ and $conf_t(l) = 1$ for $l \in L^-$

2. Local mean-based function

   $conf_s(l) = \frac{w(l,\bar{l}_s)}{w(l,\bar{l}_s)+w(l,\bar{l}_t)}$ for $l \in L^+$ and $conf_s(l) = \frac{w(l,\bar{l}_t)}{w(l,\bar{l}_s)+w(l,\bar{l}_t)}$ for $l \in L^-$
   $\bar{l}_s$ and $\bar{l}_t$ are the average vectors of k-nearest positive neighbors and k-nearest negative neighbors of l. $w(\cdot)$ is a weight function that takes in pairwise distance as input. In our experiment, similar to how weights are computed in the graph, $w(x) = exp(-x^2/2\varepsilon^2)$.

3. k neighbors-based function

   $conf_s(l) = \frac{knn^+(l)}{k}$ for $l \in L^+$ and $conf_s(l) = \frac{knn^-(l)}{k}$ for $l \in L^-$
   $knn^+(l)$ and $knn^-(l)$ are the numbers of positive neighbors and negative neighbors among the k nearest neighbors of l.

In the graph, we weigh the label confidence with a factor c, which is a tunable parameter. The local mean-based function is modified from the idea presented in a work on a non-parametric classifier of Mitani and Hamamoto (2006). In general, one can always come up with other reasonable measures of how likely a label is correct, in addition to the three measures that are proposed above.

3.2 Similarities within the positive class and the negative class

In addition to the incorporation of the label confidence that handles the label noise, another modification that we introduce is the enhancement of the symmetry in our model regarding the positive class and the negative class.

We see in Section 2.1 that the motivation behind the Hochbaum's normalized cut is the minimization of the following objective function in a given graph:

$$\frac{C(S,T)}{C(S,S)} \text{ or equivalently } \frac{C(S,T)}{\sum_{i\in S} d_i}$$

We might notice the absence of $C(T,T)$, or $\sum_{i\in T} d_i$ in the objective function above. The rationale behind $C(S,S)$ in the denominator was that, in many contexts, the similarities among the positive samples are more prominent than those of the negative samples. For instance, patients who are diagnosed a certain disease tend to share strong similarity such as poor eating habit, lack of exercise, or living in unhealthy environment, whereas a group people who are negative tend to cover a diverse set of people, and their similarities are not necessarily high. Another example is whether borrowers will default on their loans or not. People who cannot pay back (identified as positive when asked if they will default) do share strong similarities whereas people who can pay back usually cover a more general population.

However, there are also cases where the two classes are not so much different in this respect. We would like the model to handle cases where the similarities among the negative class might be as important, or even more important. Therefore, we consider an alternative objective function for a cut:

$$\frac{C(S,T)}{\lambda_s\, C(S,S) + \lambda_t\, C(T,T)}$$

Again, this can be rewritten in an alternative form:

$$\begin{aligned}
\arg\min_S \frac{C(S,T)}{\lambda_s\, C(S,S) + \lambda_t\, C(T,T)} &= \arg\min_S \frac{C(S,T)}{\lambda_s(\sum_{i\in S} d_i - C(S,T)) + \lambda_t(\sum_{i\in T} d_i - C(T,S))} \\
&= \arg\max_S \frac{\lambda_s \sum_{i\in S} d_i + \lambda_t \sum_{i\in T} d_i - (\lambda_s+\lambda_t)C(S,T)}{C(S,T)} \\
&\quad \textit{(because in our graph/context: } w_{ij} = w_{ji} \\
&\quad \textit{hence, } C(S,T) = \sum_{i\in S}\sum j \in T w_{ij} = \sum_{i\in S}\sum j \in T w_{ji} = C(T,S)) \\
&= \arg\max_S \frac{\lambda_s \sum_{i\in S} d_i + \lambda_t \sum_{i\in T} d_i}{C(S,T)} - (\lambda_s+\lambda_t) \\
&= \arg\min_S \frac{C(S,T)}{\lambda_s \sum_{i\in S} d_i + \lambda_t \sum_{i\in T} d_i}
\end{aligned}$$

The linearized problem then becomes

$$\text{minimize } C(S,T) - \lambda(\lambda_s \sum_{i\in S} d_i + \lambda_t \sum_{i\in T} d_i)$$

For the ease of notation, we replace $\lambda\lambda_s$ and $\lambda\lambda_t$ by just λ_s and λ_t. The objective function that we would like to minimize is $C(S,T) - \lambda_s \sum_{i\in S} d_i - \lambda_t \sum_{i\in T} d_i$. However, this objective function can be simplified further.

$$
\begin{aligned}
&C(S,T) - \lambda_s \sum_{i\in S} d_i - \lambda_t \sum_{i\in T} d_i \\
&= C(S,T) - min(\lambda_s, \lambda_t)\ (\sum_{i\in S} d_i + \sum_{i\in T} d_i) - \mathbb{1}\{\lambda_s \geq \lambda_t\}(\lambda_s - \lambda_t)(\sum_{i\in S} d_i) - \mathbb{1}\{\lambda_t > \lambda_s\}(\lambda_t - \lambda_s)(\sum_{i\in T} d_i)
\end{aligned}
$$

Since the second term is a constant, given λ_s and λ_t, the objective function that we would like to minimize is

$$
f(S) = \begin{cases} C(S,T) - (\lambda_s - \lambda_t)(\sum_{i\in S} d_i) & \text{if } \lambda_s \geq \lambda_t \\ C(S,T) - (\lambda_t - \lambda_s)(\sum_{i\in T} d_i) & \text{otherwise} \end{cases}
$$

Hence, if $\lambda_s \geq \lambda_t$, the graph on which we solve for a minimum cut is similar to the graph in Figure 4, except that λ is replaced by the difference $\lambda_s - \lambda_t$. However, when $\lambda_t > \lambda_s$, we replace the arcs that connect from s to the unlabeled samples to the arcs that connect from the unlabeled samples to the sink node t. Similarly, the weight on the arc from i to t is $(\lambda_t - \lambda_s)d_i$.

In our model, instead of tuning for both λ_s and λ_t, we only need to tune for the difference $\lambda_s - \lambda_t$. By incorporating this modification to the model, the model learns whether it is more appropriate to assign more weights to the source side (or the similarities between the positive samples) or the sink side (or the similarities between negative samples) given the dataset as well as the desired evaluation metric whether it be the accuracy, balanced accuracy, F1 score, or any other binary classification metric.

4. Experiment

4.1 Models and Parameters

The first set of learning methods that we evaluate are 4 variants of the LC-SNC method where in the first three, we use each of the three label confidence function, and the fourth variant is the ensemble method where each prediction is the majority vote from the first three LC-SNC methods. In these four methods, only the first modification is incorporated. The symmetric kNN sparsification method described in Section 2.2 is applied. We denote LC-SNC with constant weight, local mean weight, k-neighbors based weight and the ensemble model by LC-c, LC-lm, LC-knn and LC-ens, respectively.

Along with these 4 methods, we include the SNC method in the experiment for comparison, to see the benefit of the label confidence. We experiment with the SNC method both with and without the sparsification method, denoted by SNC and SNC-full as it is solved on a fully connected graph. In most of the result tables, only the performances of LC are displayed while those of LC-full are omitted since the performance of LC-full is much inferior than LC.

The last set of methods that are based on the normalized cut are the counterparts of the four LC-SNC methods mentioned earlier where the second modification is incorporated. These four counterparts are denoted by LC-c', LC-lm', LC-knn' and LC-ens'.

In addition to the graph-based methods, we also want to compare them to methods that share common characteristics with LC-SNC. One of the common existing learning methods that also rely on pairwise similarities between samples is the k-nearest neighbor classifier. We use a weighted kNN classifier since it yields a better performance than the traditional kNN classifier that simply counts the votes from each neighbor set. The weight used is the Gaussian weight, which is similar to the weight function used in LC-SNC. Moreover, we also test its variants that are robust to noise, which are the edited nearest neighbor and the repeated edited nearest neighbor.

These models involve a set of parameters whose values need to be tuned. We summarize the tunable parameters for each model in Table 1.

models	parameters
SNC	λ, ε, k
LC-SNC	λ, ε, k, c
kNN	ε, k
Edited NN	ε, k

Table 1: Models and relevant parameters

λ is the parameter in the linearized objective function of Hochbaum's Normalized Cut. ε is the parameter in the exponent of the weight function. k is the number of neighbors used in the sparsification and also in the prediction of the kNN classifier and the edited nearest neighbor method. Lastly, c is the coefficient of the label confidence function. Below is the range of parameter values that are tested on in the parameter tuning.

parameters	values
λ	$4^{-4}, 4^{-3}, 4^{-2}, 4^{-1}$
c	$2^{-1}, 2^{0}, 2^{1}, 2^{2}, 2^{3}, 2^{4}$
k (sparsification)	$1, 2, 3, 4, 8, 12, 16, 20$
k (kNN)	$1, 2, 3, 4, 8, 12, ..., 68$
ε	$0.5, 1, 2$

Table 2: Parameter values

4.2 Datasets

We conducted two separate sets of experiments, on real datasets and synthetic datasets. Real datasets are collected from the UC Irvine Machine Learning Repository (Newman et al., 1998). We select data with fewer than 60 features so that the Euclidean distance between them are representative of samples' similarities. Datasets are normalized.

data	name	size	#features	%pos
1	credit1	1500	23	22.12
2	credit2	5000	23	22.12
3	housing	506	13	26.09
4	german	1000	24	30.00
5	fourclass	862	2	35.61
6	obesity	2111	19	46.04
7	letter1	5000	16	49.70
8	letter2	1000	16	49.70
9	raisin	900	7	50.00
10	energy1	768	8	50.13
11	energy2	768	8	51.30
12	banknote	1372	4	55.54
13	maternal	1014	6	59.96
14	spambase	4601	57	60.60
15	wdbc	569	30	62.74
16	blood	748	4	76.20

Table 3: Datasets in ascending order of fraction of positive samples

Even with relatively low dimensionality of the data, we still need to handle possibly noisy features before we proceed with the experiment since we are only testing the model's robustness to noisy labels, not noisy features. Ideally, in the distance computation, we want to give more weights to features that are more indicative of the classes where samples belong. To do so, we fit a random forest on the labeled samples. Each tree then returns a vector called feature importance, which tells us how much each feature brings down the impurity, such as the Gini index, at each node in the tree. Feature importances across all trees are then averaged to get the feature importance of the random forest. The feature importance is normalized to have a unit size. Suppose the number of features is m, and the (normalized) feature importance is denoted by $FI \in \mathbb{R}^m$, the Euclidean distance can be computed as

$$dist(x, y) = (m \cdot \sum_{i=1}^{m} FI_i(x_i - y_i)^2)^{\frac{1}{2}}$$

In addition to the real datasets, we also experiment with synthetic datasets so that we can adjust some features of the data and learn how the performance of the model(s) vary according to these features. The data is generated using the Scikit-learn software library (Pedregosa et al., 2011).

Samples of each class are normally distributed around $n\_clusters$ vertices. These vertices are among the vertices of either an ($n\_features$)-dimensional hypercube or a random polytope. Side lengths of the hypercube is $2 \cdot class\_sep$. We can also control the proportion of samples assigned to each class through the *weights* parameter. Note that in the data generator, we can also set the number of uninformative features and redundant features as well. However, in this experiment, we want all features to be informative features since we would like the Euclidean distance to be representative of the dissimilarities between samples. The topic on noisy features is a separate issue that we do not consider yet in this work.

The configurations for the generation of synthetic datasets are summarized in Table 4.

parameters	values
$n\_samples$	400, 1200
$n\_features$	5, 10
$class\_sep$	$0.5, 1, 2$
$n\_clusters$	1, 2
$weights(\%pos, \%neg)$	(50%, 50%), (70%, 30%), (30%, 70%)
$vertices$	hypercube, polytope

Table 4: Synthetic data configuration

There are 144 different configurations in total. For each configuration, we generate 3 different datasets, resulting in 432 datasets in total.

4.3 Experiment Set-up

Each dataset is randomly divided into a set of labeled samples and a set of unlabeled samples, with a ratio of 60:40, in a way that the proportion between the samples from positive and negative classes are maintained.

Then, for each dataset, we introduce varying degrees of noise, from 0%(no noise at all) to $10\%, 20\%$ and 30%, to the labels in the labeled samples. In order to add noise to the data, we flip their labels from positive to negative and vice versa. For instance, with 10% noise level, we flip the labels of randomly selected 10% of labeled samples with positive labels, as well as randomly selected 10% of labeled samples with negative labels. It is important to note that by doing this, we rely on the assumption that the label corruption on the positive samples has the same distribution as that on the negative samples. Also, the corrupted dataset does not necessarily have the same proportion between the positive and negative classes as the original dataset.

At this point, we may see that the set up of the data embraces some randomness such as the split into labeled and unlabeled set as well as the random selection of labels to be corrupted. Hence, in the experiment on the real data, for each dataset, we do 8 different labeled-unlabeled splits, and for each split, we do 5 different selection of labeled samples to be corrupted, making 40 experiments in total for each dataset and each nonzero noise level. For the zero noise level (the scenario where the dataset is completely uncorrupted), there are 8 experiments for each dataset, which comes from the labeled-unlabeled splits.

Similarly, for each synthetic data, we perform 5 different labeled-unlabeled splits and noise corruptions. Therefore, for each noise level, there are $5(432) = 2160$ experiments.

Regarding the parameter tuning step, we use cross validation (CV). First, we split the labeled set using stratified k-fold which partitions the labeled set into k folds and preserves the percentage of samples from each class in each fold. Then, we apply either the grid search CV or the randomized search CV depending on the complexity of the models. The randomized search CV does not evaluate all permutations of parameter values but randomly selected some of them. In this experiment, the kNN classifier has 60 combinations of parameter values. We apply the grid search CV on the knn classifier and its variants. For our normalized cut models where the numbers of combinations are much higher, we apply the randomized search CV and cap the number of combinations to evaluate at 60.

4.4 Evaluation metrics

4.4.1 Prediction metrics

We evaluate the models using the following three metrics: (1) Accuracy, (2) Balanced accuracy, which is the average accuracy among the two classes. In other words, it is the average of the true positive rate (recall or sensitivity) and the true negative rate (specificity), and (3) F1 Score, which is the harmonic mean of precision and recall. Recall is the true positive rate. Precision is the fraction of the true positive predictions out of all positive predictions.

4.4.2 Statistical test

As we examine the performance of the models regarding these evaluation metrics, we need a tool to determine whether the differences are statistically significant. In this work, we use the Wilcoxon Signed-Ranks test or Wilcoxon test. The Wilcoxon test is argued to be a better alternative to the Paired T-test due to several reasons (Demšar, 2006). First, the T-test might not be a reasonable choice when the performance differences of two classifiers across multiple datasets are not in proportion with each other. For instance, a superior performance of one classifier on a particular dataset can lead to the conclusion that the classifier has a better overall performance if the the gap of the performances on that one particular instance is exceptionally large, even though it exhibits inferior performance on all other datasets. Second, it assumes that the differences are normally distributed . Lastly, the T-test is more sensitive to outliers.

The Wilcoxon test operates on the ranking of the differences between two classifiers. Hence, it still takes the magnitude of the differences into account but only to a moderate degree. With the same reason, it is less affected by the outliers.

4.4.3 Noise detection

Last but not least, since we test our models in the context of their robustness to noisy labels, we also would like to see how accurate the models identify noisy labels. For the LC-SNC models, we can see for each model and each noisy labeled sample whether the model puts them on the correct side of the cut. For the edited nearest neighbor methods, we obtain a set of labeled samples that are flagged as noisy from the beginning. With these information, we compute three measures.

(1) Noise recall: fraction of noisy labels that are detected by the models (2) Noise precision: fraction of labels that are flagged as noisy and are actually noisy and (3) F1 Score of noise detection: the harmonic mean of noise recall and noise precision. Note that the F1 score of noise detection is different from the F1 score of the label prediction in 4.4.1.

4.5 Results

In this subsection, we provide results from both experiments on synthetic data sets and real data sets. Some result tables within this section whereas the rest of the tables are included in the Appendix.

4.5.1 Results on synthetic data

We measure the evaluation metrics of all models on all synthetic datasets and all randomness such as labeled-unlabeled splits, noise levels and the selection of labels to be corrupted. First, we look at the average improvement percentage of different evaluation measures acheieved by the four LC-SNC models and the kNN over those yielded by SNC. The results are displayed in Table 5.

Accuracy					
noise level	LC-c	LC-lm	LC-knn	LC-ens	knn
0	1.37	1.01	1.09	1.49	0.87
10	1.62	1.13	1.66	2.07	0.82
20	1.48	0.89	2.07	2.52	0.68
30	-0.12	-0.58	1.31	2.16	0.10

Balanced Accuracy					
noise level	LC-c	LC-lm	LC-knn	LC-ens	knn
0	1.80	1.14	0.94	1.87	0.88
10	2.03	1.42	1.89	2.55	0.61
20	3.48	2.56	4.05	4.56	1.88
30	2.49	1.86	3.74	4.61	1.99

F1 Score					
noise level	LC-c	LC-lm	LC-knn	LC-ens	knn
0	2.04	1.58	1.58	2.21	0.69
10	2.44	1.46	2.42	2.91	0.56
20	5.12	2.88	5.24	5.71	2.91
30	9.96	4.10	10.14	9.94	9.85

Table 5: Average improvement percentage of three evaluation metrics over SNC across all synthetic datasets at different noise levels

From Table 5, when considering only accuracy and balanced accuracy, we can see that the LC-SNC models improve the accuracy by 1-2 % over SNC. They also outperform the kNN classifier. Among the three LC-SNC models, the one with the k-neighbors weight achieves the best performance in general. The ensemble LC-SNC, which is the majority vote from the three LC-SNC methods, always bring about 2% accuracy improvement. When we compare the outcomes with different noise levels, the improvement seems to reach its highest point at 20% noise. It is possible that when the noise level amounts to 30%, and beyond, the task becomes too difficult and the noise obscures the true signals from the data.

Most importantly, even though the label confidence is introduced mainly to handle label noise, we see that for all evaluation metrics, we obtain decent improvement over the SNC method even when there is no noise (at 0% noise level).

Accuracy					
noise level	HNC	LC-c	LC-lm	LC-knn	LC-ens
0	-0.79	0.52	0.16	0.25	0.64
10	-0.64	0.85	0.35	0.88	1.28
20	-0.35	0.88	0.29	1.45	1.90
30	0.35	-0.09	-0.54	1.34	2.20

Balanced Accuracy					
noise level	HNC	LC-c	LC-lm	LC-knn	LC-ens
0	-0.75	0.97	0.31	0.11	1.04
10	-0.40	1.49	0.88	1.35	2.00
20	-1.36	1.69	0.76	2.21	2.73
30	-1.06	0.72	0.13	1.97	2.83

F1 Score					
noise level	HNC	LC-c	LC-lm	LC-knn	LC-ens
0	-0.55	1.40	0.92	0.92	1.56
10	-0.27	1.96	0.99	1.95	2.43
20	-2.09	2.30	0.16	2.42	2.88
30	-6.90	0.76	-4.19	0.89	0.72

Table 6: Average improvement percentage of three evaluation metrics over kNN across all synthetic datasets at different noise levels

Next, we look at the results in Table 6, which displays a similar result except that it is the average improvement over the kNN. We see that the SNC model achieves lower performance than the kNN. However, by incorporating the label confidence, the models are able to outperform the kNN classifier by about $1 - 2\%$ for all metrics. Again, we see that even when the noise is not present, the use of label confidence can boost the performance over kNN, for all evaluation metrics.

We have seen how label confidence functions in the presence of noisy labels and brings about noticeable improvement of all metrics. Next, we examine the impact of the second modification, which is to include the edges between the unlabeled samples and the sink node, rather than just to the source node. That is, we compare LC-c', LC-lm', LC-knn', LC-ens' to LC-c, LC-lm, LC-knn and LC-ens, as shown in Table 7.

Each column in Table 7 shows average improvement percentage across all data that is attained by the second modification, for the LC-SNC method with each confidence function. We see that the results on F1 score are quite mixed while the results on accuracy are positive yet not as remarkable as expected. However, the improvement on balanced accuracy that the second modification brings in seems to be more noticeable than on other measures. It is important to note that all though these improvement percentages are at most about 1% or 1.5%, we need to be aware that these improvements are not on the baseline SNC method but on the already improved LCSNC methods into which label confidence has been incorporated.

Furthermore, we examine how the models with the flexibility regarding the edges adjacent to unlabeled nodes respond to data with different noise levels and different fractions between the numbers of positive samples and negative samples. In Table 8, for each bal-

Accuracy				
noise level	LC-c	LC-lm	LC-knn	LC-ens
0	0.11	0.26	0.11	0.21
10	0.12	0.09	0.17	0.11
20	0.22	0.63	0.08	0.26
30	1.36	1.72	0.09	0.26

Balanced Accuracy				
noise level	LC-c	LC-lm	LC-knn	LC-ens
0	0.62	0.75	0.76	0.73
10	0.84	0.66	0.84	0.71
20	1.30	1.14	0.86	1.10
30	0.99	0.90	0.60	0.80

F1 Score				
noise level	LC-c	LC-lm	LC-knn	LC-ens
0	-0.16	-0.01	-0.11	-0.08
10	-0.02	0.12	0.01	0.09
20	-0.33	-0.02	0.07	0.22
30	-0.49	-0.12	-0.27	0.65

Table 7: Improvement yielded by the counterpart of each LC-SNC method

Accuracy & LC-c			
	% negative class		
noise level	30	50	70
0	0.6458	0.4444	0.3125
10	0.6753	0.5226	0.3125
20	0.6580	0.4844	0.3351
30	0.5260	0.5156	0.4531

Accuracy & LC-lm			
	% negative class		
noise level	30	50	70
0	0.5556	0.4931	0.3611
10	0.5642	0.5017	0.4149
20	0.5347	0.5087	0.4514
30	0.4844	0.5139	0.5434

Accuracy & LC-knn			
	% negative class		
noise level	30	50	70
0	0.7361	0.4167	0.2500
10	0.7153	0.5017	0.2726
20	0.6753	0.5069	0.3299
30	0.5712	0.5174	0.4167

Balanced Accuracy & LC-c			
	% negative class		
noise level	30	50	70
0	0.8611	0.4444	0.1111
10	0.8385	0.5260	0.1285
20	0.8403	0.4809	0.1267
30	0.7569	0.5052	0.2135

Balanced Accuracy & LC-lm			
	% negative class		
noise level	30	50	70
0	0.7778	0.4722	0.1736
10	0.7899	0.5000	0.2361
20	0.7222	0.4983	0.2431
30	0.6684	0.5139	0.3125

Balanced Accuracy & LC-knn			
	% negative class		
noise level	30	50	70
0	0.8889	0.4236	0.1181
10	0.8472	0.5122	0.1059
20	0.8351	0.5174	0.1354
30	0.7812	0.5052	0.2344

Table 8: Proportion of data where unlabeled nodes are connected to the sink node rather than the source node

ancedness, noise level and model, we count the number of datasets on which the model learns to switch the edges from connected with the source node to the sink node instead.

We see from all subtables that the models learn to adjust the structure of the graph as the underlying balance in the data changes. When there are fewer negative samples, the edges between unlabeled nodes and the sink node are added more often than those between unlabeled nodes and the source node. A possible explanation is that when there are fewer labeled negative samples, without edges between either the source node or the sink node to unlabeled samples, most unlabeled samples will be pulled toward the source set since there are more nodes on that side, and hence, more connectivity from the source set. The cut between the source set and unlabeled samples then becomes expensive. To prevent this type of bias (or in other words, to retain some unlabeled node to the sink set), edges between unlabeled samples and the sink node are added to balance the gravity toward the source set. This explanation is strengthened by the result when evaluating the models on balanced accuracy. We see that the difference between the two extremes is more distinct than when we evaluate the models on the accuracy. When the data is heavily imbalanced, putting all unlabeled samples on one side might attain high accuracy, but the balanced accuracy will be low.

Another observation is that this "signal" or the relationship between the proportion of data with unlabeled samples connected to the sink node and the balancedness of the data is more evident at a lower noise level. This could be partially explained by how we contaminate the data with noisy labels. By flipping the labels of each class with the same proportion, the data becomes less unbalanced. For instance, 20% noise contamination applied on a dataset with 70 positive samples and 30 negative samples results in a less unbalanced data with 62 positive samples and 38 negative samples.

Finally, we look at the performance of the models regarding noise detection capability. Here, we look at the results when the models are trained to optimize for the best balanced accuracy. The results are shown in Table 9.

Noise Recall										
noise level	LC-c	LC-lm	LC-knn	LC-ens	LC-c'	LC-lm'	LC-knn'	LC-ens'	ENN	RENN
10	0.27	0.38	**0.56**	0.39	0.26	0.37	0.55	0.38	**0.86**	**0.89***
20	0.29	0.37	**0.49**	0.36	0.29	0.35	0.48	0.36	**0.80**	**0.85***
30	0.30	0.36	**0.44**	0.35	0.30	0.35	**0.44**	0.34	**0.71**	**0.73***
Noise Precision										
noise level	LC-c	LC-lm	LC-knn	LC-ens	LC-c'	LC-lm'	LC-knn'	LC-ens'	ENN	RENN
10	0.39	0.63	**0.76**	**0.73**	0.37	0.62	**0.77***	0.72	0.44	0.42
20	0.44	0.66	**0.81***	**0.75**	0.43	0.66	**0.81***	**0.75**	0.51	0.50
30	0.46	0.64	**0.78***	**0.71**	0.46	0.63	**0.76**	0.70	0.52	0.52
F1 Score of Noise Detection										
noise level	LC-c	LC-lm	LC-knn	LC-ens	LC-c'	LC-lm'	LC-knn'	LC-ens'	ENN	RENN
10	0.28	0.41	**0.60***	0.44	0.28	0.40	**0.59**	0.43	**0.58**	0.56
20	0.31	0.41	**0.54**	0.42	0.31	0.39	0.53	0.41	**0.62***	**0.62***
30	0.33	0.40	**0.49**	0.41	0.33	0.39	**0.49**	0.40	**0.60***	**0.60***

Table 9: Noise detection performance: noise recall (top), noise precision (middle) and F1 score of noise detection (bottom)

From Table 9, we see that the LC-SNC methods, especially LC-knn, achieve high noise precision of about 75-80%, meaning that 75-80% of the labels that the models put on the opposite side of the cut from the given labels are actually the labels that are corrupted. For the noise-robust variants of kNN, only half of the labels that are filtered out are actually noisy. However, when looking at the noise recall, the variants of kNN attain higher noise recall, meaning that they are able to detect about 70-90% of labels that are corrupted. At the same time, the best that LC-SNC methods can detect is about 50%. As we look deeper into the results, we see that this comes from how the two sets of models behave. LC-SNC methods only mark a few samples as noisy, but with high precision. The kNN-based methods mark so many samples as noisy sample, naturally resulting in high recall but low precision. The last subtable displays the harmonic mean of noise recall and noise precision, showing a slight advantage of ENN and RENN over LC-kNN.

4.5.2 Results on real data

First, we look at the overall accuracy of SNC, LCSNC with three weights, the LC-SNC ensemble and the kNN and its noise-robust variants. The performances of these models are displayed in Table 10 and Table 11.

Accuracy (no noise)

data	SNC	LC-c	LC-lm	LC-knn	LC-ens	kNN	ENN	RENN
1	81.47	81.44	80.58	**81.60**	**81.60**	**81.65***	80.98	81.45
2	81.88	**81.90**	81.59	**81.93**	**82.00***	81.76	80.79	81.38
3	89.32	89.59	89.28	89.80	89.89	**90.38***	**90.24**	**90.11**
4	73.58	**74.39**	73.16	74.32	**74.67**	**74.82***	74.11	73.69
5	99.28	99.83	99.16	99.83	99.83	**99.96***	**99.93**	**99.93**
6	98.86	**99.13**	98.52	**99.19***	**99.18**	98.98	98.86	98.79
7	93.94	**94.98***	94.11	94.41	**94.86**	**94.75**	92.77	92.77
8	83.82	**84.02**	82.67	82.94	**84.07**	**85.47***	82.50	82.50
9	86.36	86.45	86.24	**86.49**	**86.50**	**87.17***	86.12	86.12
10	97.78	97.86	97.62	97.94	**98.03**	**98.18***	**97.99**	97.93
11	**99.39**	**99.45***	98.80	99.00	99.32	**99.36**	98.82	98.82
12	98.38	99.04	98.88	99.29	99.08	**99.92***	**99.78**	**99.78**
13	75.81	**81.15**	80.57	77.27	**80.91**	**82.46***	75.77	74.86
14	91.29	**92.05**	91.57	**92.02**	**92.11***	91.92	91.09	90.99
15	94.94	**95.87**	95.42	95.52	**95.74**	**95.91***	95.65	95.59
16	77.50	78.24	77.78	**78.52**	**78.36**	**78.78***	76.16	76.81

Accuracy (noise level: 20%)

data	SNC	LC-c	LC-lm	LC-knn	LC-ens	kNN	ENN	RENN
1	**79.60**	**79.72**	74.25	**79.60**	**79.89***	79.30	75.85	78.90
2	80.74	**81.07**	80.39	**81.07**	**81.14***	80.88	75.23	78.98
3	87.46	86.75	85.45	**88.03**	**88.22**	**88.43***	85.87	87.43
4	71.64	71.36	69.40	**71.83**	**71.92**	**73.03***	70.17	71.53
5	99.14	**99.30**	98.87	**99.19**	**99.38***	98.53	95.62	97.84
6	95.50	**96.59**	95.72	**96.76**	**96.98***	95.30	93.48	93.48
7	86.92	**88.64**	87.63	**88.58**	**89.05***	87.74	86.60	86.60
8	72.87	74.34	72.85	**75.00**	**75.28**	74.89	**75.43**	**75.43***
9	**85.62**	81.21	85.56	84.85	**85.74**	**86.19***	80.75	80.75
10	97.46	**97.52**	93.22	97.48	**97.67***	**97.64**	94.53	94.53
11	**96.60***	95.82	91.34	95.70	**95.94**	**96.20**	93.18	93.18
12	98.84	98.76	**99.20***	**98.89**	**99.17**	98.78	95.57	95.57
13	71.41	**72.19**	71.87	**72.43**	**72.64***	71.73	70.16	70.22
14	89.35	**90.15**	89.81	**90.04**	**90.43***	88.18	85.37	85.40
15	94.23	94.61	**94.72**	**94.69**	**94.98***	93.83	92.76	93.98
16	75.62	76.48	75.61	**76.49**	**76.57**	**77.31***	69.94	72.74

Table 10: Average accuracy of the models for each dataset when there is no noise (above) and 20% noise (below)

Regarding the accuracy, we can see from Table 10 that the LC-SNC ensemble attains high performance often, and even the highest performance, especially when the noise is present. When there is no noise, the kNN classifier seems to outperform the SNC and all the LC-SNC models in many datasets. Without noise, the performance of LC-c is close to that of kNN. However, when the noise is added to the data, the relative performances of LC-c and LC-knn increase significantly. LC-c and LC-knn appear in the group of the top three models more often than kNN. It is important to note that, first, LC-lm does not yield desirable results, and second, regardless of the existence of noise, the models benefit from

Balanced Accuracy (no noise)								
data	SNC	LC-c	LC-lm	LC-knn	LC-ens	kNN	ENN	RENN
1	**66.56**	**66.67**	66.37	66.50	**67.03***	62.83	63.38	63.02
2	64.14	66.32	**66.57**	**66.58**	**66.65***	64.08	63.31	63.66
3	**84.82***	84.23	**84.76**	83.87	**84.54**	84.10	83.43	82.62
4	**70.03***	68.17	**69.37**	67.76	**68.72**	63.70	63.76	60.69
5	99.44	99.87	99.34	99.87	99.87	**99.95***	**99.91**	**99.91**
6	98.75	**99.09**	98.54	**99.13**	**99.17***	98.98	98.81	98.77
7	93.93	**94.99***	94.11	**94.41**	**94.87**	94.75	92.78	92.78
8	83.68	**83.97**	82.60	83.07	**84.04**	**85.38***	82.44	82.44
9	86.36	86.47	86.28	**86.48**	**86.51**	**87.13***	86.14	86.14
10	97.87	97.86	97.60	97.95	**98.03**	**98.16***	**97.99**	97.93
11	**99.35**	**99.45***	98.89	99.01	**99.33**	99.32	98.79	98.79
12	98.40	98.98	98.82	**99.22**	**99.01**	**99.93***	99.80	99.80
13	76.66	**77.62**	77.31	75.88	**78.08**	**81.97***	75.69	75.19
14	90.44	**91.44**	90.85	**91.32**	**91.48***	90.93	90.17	89.98
15	93.80	**94.71**	**94.81**	94.24	94.63	**94.95***	94.66	94.47
16	61.14	62.35	**66.88***	62.69	**63.35**	**63.69**	61.98	62.09

Balanced Accuracy (noise level: 20%)								
data	SNC	LC-c	LC-lm	LC-knn	LC-ens	kNN	ENN	RENN
1	62.10	**63.42**	61.66	**62.56**	**63.66***	59.04	59.54	59.54
2	62.29	**64.04**	63.41	**63.69***	**64.64***	61.67	60.35	61.85
3	79.88	81.22	**82.07**	**82.16**	**82.65***	81.68	79.80	79.66
4	58.46	**64.64**	**65.50**	64.59	**65.83***	60.96	60.85	59.42
5	**98.88**	98.81	98.53	**99.18**	**99.18***	98.51	95.70	97.88
6	95.34	**96.51**	95.51	**96.65**	**96.92***	95.28	93.57	93.57
7	87.10	**88.64**	87.65	**88.65**	**89.02***	87.74	86.61	86.61
8	72.56	74.31	73.10	74.98	**75.40***	74.97	**75.40***	**75.40***
9	85.62	81.25	**85.64**	84.82	**85.81**	**86.17***	80.79	80.79
10	97.46	**97.52**	93.26	**97.60**	**97.67***	**97.52**	94.55	94.55
11	**96.59***	95.86	91.23	95.50	**95.92**	**96.28**	93.24	93.24
12	98.77	98.70	**99.05**	**99.09**	**99.15***	98.85	95.61	95.61
13	71.14	70.94	71.02	**71.45**	**71.86***	**71.69**	69.79	69.88
14	88.04	**89.13**	**89.07**	88.99	**89.45***	86.69	84.30	84.32
15	92.70	**93.59***	93.10	**93.53**	**93.85***	92.41	91.53	92.85
16	58.80	58.80	**63.78***	59.75	**60.81***	**61.16***	59.14	60.39

Table 11: Average balanced accuracy of the models for each dataset when there is no noise (above) and 20% noise (below)

the label confidence of certain weights as we can see that LC-c and LC-knn attain higher accuracy than SNC most of the time.

In terms of the balanced accuracy, the overall outcome is fairly similar to that for the accuracy. There are still some differences, For instance, kNN becomes less powerful whereas LC-lm becomes more reliable when compared to the results from the accuracy table. The performance of both LC-c and LC-knn are impressive and consistent across all noise levels, with slightly higher relative performance when more noise is added to the data.

The average improvement percentage over the SNC method of all LC-SNC methods are reported in the Appedix. In order to see the significance of the difference between any two classifiers, we apply the Wilcoxon test as explained earlier. We will test each of the proposed LC-SNC methods against each of the two baseline methods, which are SNC and kNN classifier. For each pair of one LC-SNC method and one of the baseline methods, the null hypothesis on which we measure the p-value is that the baseline model has a better

SNC					kNN				
noise level	LC-c	LC-lm	LC-knn	LC-ens	noise level	LC-c	LC-lm	LC-knn	LC-ens
0	**0.0022**	0.3025	**0.0016**	**0.0007**	0	0.9869	0.9998	0.9981	0.9331
10	**0.0010**	0.5819	**0.0006**	**0.0003**	10	0.2507	0.9807	0.1172	**0.0394**
20	0.0738	0.8371	**0.0087**	**0.0010**	20	0.4794	0.9721	0.2190	**0.0394**
30	0.7960	0.9900	0.7810	**0.0440**	30	0.8724	0.9869	0.8495	0.2040

Table 12: P-values of the Wilcoxon test against SNC and kNN on accuracy

SNC					kNN				
noise level	LC-c	LC-lm	LC-knn	LC-ens	noise level	LC-c	LC-lm	LC-knn	LC-ens
0	**0.0170**	0.1760	0.1629	**0.0114**	0	0.5412	0.5412	0.8240	0.4588
10	**0.0075**	0.4181	**0.0056**	**0.0007**	10	0.0604	0.3208	**0.0170**	**0.0087**
20	**0.0352**	0.0894	**0.0036**	**0.0006**	20	0.1897	0.2507	**0.0313**	**0.0031**
30	0.2190	0.3208	0.1172	**0.0016**	30	0.4588	0.4384	0.2190	**0.0193**

Table 13: P-values of the Wilcoxon test against SNC and kNN on balanced accuracy

performance, against the alternative hypothesis that the LC-SNC model performs better. A p-value that is less than 0.05, which is a common threshold, implies that the improvement is significant and we can reject the null hypothesis. However, 0.05 is not strictly defined but a commonly used threshold. P-values of the Wilcoxon tests of pairs of methods on accuracy and balanced accuracy are reported in Table 12 and 13.

First, we examine the p-values from the Wilcoxon test on accuracy. We see that the improvement of LC-SNC methods, except that of LC-lm, over SNC is significant as the p-values are mostly below 0.05. In fact, many of them are as low as 0.01, even 0.005 or below. On the contrary, when compared with kNN, significant improvement only comes from the ensemble method, at the noise level of 10% and 20%. There are other low p-values in the range of 0.1 and 0.2 from LC-c and LC-knn, but they are not small enough to conclude the significance of the improvement.

P-values from the test on balanced accuracy reveals a similar finding. However, the relative performance of the LC-SNC methods are higher in this case. The label confidence weights of LC-c and LC-knn, and the ensemble method LC-ens brings in significant improvement over SNC. LC-lm also exhibits smaller p-values than in the former case. Regarding the comparison with kNN, p-values of the LC-SNC methods are also noticeably smaller. This time, in addition to LC-ens, LC-knn yields p-values that are smaller than 0.05. LC-c also gives p-values in a decently small range.

Furthermore, we look into the improvement that the second modification, or the addition of edges from unlabeled samples to the sink node, brings in. In Table 14 and 15, we compute the change percentage of accuracy and balanced accuracy after we allow unlabeled nodes and the sink node to be connected. Bold numbers are the percent increases that are higher than 0.5%.

Table 14 shows that there is barely any improvement that the second modification contributes to the LC-SNC methods regarding the accuracy. When there is no noise level, the number of positive changes and negative changes are about the same, with only one positive change that is larger than 0.5%. When the noise level goes up to 20%, LC-lm' delivers decent improvement over LC-lm on 5 out of 16 datasets.

However, when it comes to the balanced accuracy, allowing unlabeled nodes to connect to the sink instead of the source results in noticeable improvement to the model on about half of the datasets when there is noise. In Table 15, we see a decent number of model-data pairs where the improvement is higher than 0.5%, especially when 20% of the labels are corrupted.

Accuracy (no noise)					Accuracy (noise level : 20%)				
data	LC-c	LC-lm	LC-knn	LC-ens	data	LC-c	LC-lm	LC-knn	LC-ens
1	0.08	0.07	-0.06	-0.04	1	-0.24	**6.87**	0.44	0.11
2	-0.10	-0.30	-0.11	-0.13	2	-0.17	-0.01	-0.12	0.07
3	-0.17	-0.33	0.18	0.15	3	**1.57**	**0.85**	0.42	0.24
4	-0.19	**0.76**	-0.36	-0.31	4	**0.52**	**2.54**	-0.11	-0.15
5	0.17	0.40	0.17	0.17	5	-0.04	-0.04	0.00	0.07
6	-0.02	0.25	-0.17	-0.08	6	0.15	0.28	0.15	0.29
7	-0.00	0.45	-0.08	0.05	7	-0.26	-0.12	-0.02	-0.03
8	-0.10	0.06	-0.34	0.04	8	-0.18	0.12	0.07	-0.07
9	-0.43	-0.08	0.11	0.22	9	**5.14**	-0.36	-0.48	0.01
10	0.02	-0.32	0.15	-0.13	10	-0.41	**1.90**	-0.23	-0.13
11	-0.04	0.01	-0.15	0.11	11	0.26	**4.85**	0.45	0.48
12	-0.02	0.13	-0.19	-0.05	12	0.16	-0.55	-0.38	-0.06
13	-0.69	-0.10	-0.80	-0.02	13	-0.10	-0.22	-0.42	0.20
14	0.03	0.22	-0.01	0.05	14	0.08	0.35	0.32	0.18
15	-0.09	-0.31	0.29	0.06	15	0.21	0.21	-0.10	-0.10
16	-0.06	-0.31	-0.63	-0.52	16	-0.66	**0.75**	-0.84	-0.07

Table 14: Percentage changes in accuracy after the edge between the unlabeled nodes and the sink node are added

Balanced Accuracy (no noise)					Balanced Accuracy (noise level : 20%)				
data	LC-c	LC-lm	LC-knn	LC-ens	data	LC-c	LC-lm	LC-knn	LC-ens
1	-0.11	0.06	-0.30	-0.06	1	-0.22	-0.28	0.40	0.46
2	**0.83**	**0.70**	0.40	0.47	2	**3.58**	**1.66**	**3.80**	**2.58**
3	-0.26	-1.24	-0.16	0.01	3	**0.92**	-0.72	-0.20	0.11
4	**0.57**	-0.45	**0.51**	0.24	4	**0.52**	0.29	**0.88**	**1.15**
5	0.13	0.03	0.13	0.13	5	0.23	0.34	0.04	0.13
6	-0.02	0.15	-0.14	-0.10	6	0.17	**0.70**	0.01	0.22
7	0.01	0.44	-0.04	0.09	7	-0.20	-0.14	-0.10	0.02
8	0.01	0.03	-0.48	0.09	8	-0.14	-0.26	-0.02	-0.35
9	-0.43	-0.10	0.10	0.18	9	**5.09**	-0.59	-0.43	-0.06
10	0.01	-0.19	0.18	-0.09	10	-0.37	**1.84**	-0.39	-0.09
11	-0.04	-0.09	-0.18	0.08	11	0.21	**3.99**	**0.53**	0.43
12	0.02	0.13	-0.18	-0.04	12	0.20	-0.35	-0.15	-0.02
13	**2.55**	**2.18**	0.59	**2.02**	13	**1.61**	**0.65**	**1.39**	**1.63**
14	0.34	0.46	0.27	0.34	14	-0.44	0.29	**1.04**	**0.70**
15	0.22	-0.43	**0.71**	0.31	15	**0.75**	-1.21	0.42	**0.57**
16	**5.49**	0.00	**4.32**	**4.37**	16	**4.57**	-1.26	**3.35**	**3.57**

Table 15: Percentage changes in balanced accuracy after the edge between the unlabeled nodes and the sink node are added

We also perform a similar analysis on the behavior of the model regarding the second modification. For each dataset, we have 40 different experiments due to the way we set up the experiments. We count the number of experiments where unlabeled nodes are connected to the sink rather than the source, meaning that the modification that we introduced to

Accuracy							Balanced Accuracy						
	no noise			noise level 20%				no noise			noise level 20%		
data	LC-c	LC-lm	LC-knn	LC-c	LC-lm	LC-knn	data	LC-c	LC-lm	LC-knn	LC-c	LC-lm	LC-knn
1	0.45	0.55	0.42	0.42	0.60	0.47	1	0.03	0.35	0.03	0.10	0.30	0.07
2	0.53	0.33	0.55	0.45	0.33	0.60	2	0.82	0.55	0.85	0.93	0.68	0.93
3	0.33	0.53	0.35	0.45	0.68	0.40	3	0.05	0.40	0.05	0.17	0.40	0.15
4	0.25	0.65	0.33	0.35	0.70	0.38	4	0.00	0.30	0.00	0.03	0.42	0.00
5	0.53	0.50	0.50	0.55	0.45	0.42	5	0.55	0.53	0.53	0.47	0.38	0.40
6	0.47	0.60	0.53	0.30	0.38	0.42	6	0.38	0.53	0.53	0.23	0.33	0.40
7	0.53	0.55	0.47	0.50	0.45	0.57	7	0.47	0.53	0.45	0.50	0.45	0.57
8	0.33	0.45	0.57	0.40	0.30	0.47	8	0.30	0.42	0.53	0.38	0.28	0.35
9	0.45	0.35	0.38	0.35	0.40	0.42	9	0.45	0.35	0.38	0.38	0.40	0.42
10	0.70	0.45	0.60	0.55	0.40	0.55	10	0.72	0.47	0.60	0.57	0.40	0.57
11	0.53	0.42	0.47	0.50	0.40	0.45	11	0.50	0.45	0.50	0.50	0.40	0.45
12	0.33	0.35	0.38	0.23	0.33	0.28	12	0.33	0.33	0.35	0.28	0.35	0.40
13	0.50	0.40	0.45	0.33	0.38	0.23	13	0.80	0.62	0.72	0.68	0.57	0.60
14	0.70	0.42	0.57	0.75	0.50	0.75	14	0.97	0.68	0.95	0.90	0.70	0.85
15	0.50	0.53	0.55	0.55	0.53	0.72	15	0.65	0.60	0.65	0.65	0.55	0.88
16	0.65	0.33	0.72	0.30	0.20	0.45	16	1.00	0.57	1.00	0.85	0.72	0.82

Table 16: Proportion of data where unlabeled nodes are connected to the sink node rather than the source node

the model is *activated*. Table 16 shows a similar pattern as in Table 8 from the synthetic experiment when the objective is to maximize the balanced accuracy. At the top of both tables are datasets whose most labels are negative. The ratio between the two classes approaches 1:1 as we get to the middle rows of the table. At the bottom of the tables are datasets with mostly positive samples. The second modification are more *activated* on datasets at the bottom, where positive samples outnumber negative samples. The edges that connect unlabeled samples to the sink prevents the model from predicting most samples, if not all, as positive samples. However, when the model minimizes the prediction accuracy, there is almost no pattern at all.

Finally, we look at the performance of the models regarding noise detection capability when the models are trained to optimize for the best balanced accuracy. The results are shown in Table 17.

We see similar patterns of noise recall and noise precision as we have seen with the synthetic datasets. The noise-robust variants of kNN methods have high noise recall but low noise precision. The LC-SNC methods achieve low noise recall but high noise precisions. The results on the noise detection F1 score are, however, different. In the synthetic experiments, the k-NN based methods outperform consistently but only very slightly. However, on real data, their relative performances are about the same.

5. Conclusion

In this work, we propose a method called Label Confidence-Supervised Normalized Cut or LC-SNC, which is a graph-based classification method that is developed particularly to handle noisy labels. LC-SNC is an extension of the Supervised Normalized Cut, or SNC, with two modifications that we propose in this work. SNC is a classification method that is based on the Hochbaum's Normalized Cut (HNC) that aims to minimize pairwise

Noise Recall

data	LC-c	LC-lm	LC-knn	LC-ens	LC-c	LC-lm	LC-knn	LC-ens	ENN	RENN
1	0.23	0.35	0.36	0.26	0.33	**0.38**	0.33	0.31	**0.70**	**0.80***
2	0.62	0.61	**0.65**	**0.68**	0.51	0.57	0.57	0.56	0.69	**0.81***
3	0.13	**0.57**	0.42	0.33	0.19	**0.57**	0.38	0.34	**0.80**	**0.88***
4	0.05	0.46	0.14	0.11	0.04	**0.47**	0.15	0.13	**0.65**	**0.78***
5	0.75	0.74	**0.90**	0.88	0.84	0.73	**0.95**	0.87	**0.90**	**0.96***
6	0.66	0.60	**0.84**	0.78	0.65	0.54	0.77	0.72	**0.88***	**0.88***
7	0.55	0.11	**0.64**	0.49	0.57	0.13	0.61	0.51	**0.82***	**0.82***
8	0.26	0.16	**0.38**	0.22	0.19	0.19	0.36	0.20	**0.74***	**0.74***
9	0.58	0.70	**0.74***	0.72	0.58	0.72	0.72	0.72	**0.74***	**0.74***
10	0.58	0.70	**0.81**	0.72	0.41	0.71	0.74	0.65	**0.87***	**0.87***
11	0.48	0.64	**0.77**	0.62	0.45	0.59	0.69	0.59	**0.86***	**0.86***
12	0.56	0.64	0.74	0.66	0.56	0.69	**0.81**	0.73	**0.90***	**0.90***
13	0.39	0.43	**0.48**	0.44	0.36	0.35	0.51	0.42	**0.61**	**0.62***
14	0.26	0.59	0.52	0.46	0.31	**0.61**	0.48	0.46	**0.80***	**0.80***
15	0.43	0.81	0.60	0.64	0.37	**0.85**	0.55	0.62	**0.87**	**0.95***
16	0.14	**0.48**	0.35	0.26	0.11	0.46	0.23	0.18	**0.65**	**0.73***

Noise Precision

data	LC-c	LC-lm	LC-knn	LC-ens	LC-c	LC-lm	LC-knn	LC-ens	ENN	RENN
1	0.33	0.37	0.54	**0.55**	0.39	0.36	**0.56**	**0.58***	0.37	0.37
2	0.47	0.48	**0.55**	**0.55**	0.43	0.47	**0.56***	**0.55**	0.36	0.37
3	0.32	0.68	0.87	**0.90**	0.38	0.70	**0.88**	**0.91***	0.51	0.50
4	0.14	0.38	**0.67***	**0.65**	0.17	0.39	0.59	**0.61**	0.32	0.32
5	0.81	0.87	**0.99***	0.94	0.94	0.79	**0.99***	**0.97**	0.68	0.69
6	0.80	0.95	0.95	**0.98***	0.75	**0.97**	**0.96**	**0.96**	0.64	0.64
7	0.73	0.82	0.87	**0.93***	0.78	0.79	**0.88**	**0.89**	0.55	0.55
8	0.44	0.55	**0.66**	**0.65**	0.32	0.54	**0.68***	**0.65**	0.41	0.41
9	0.44	**0.68***	0.64	**0.68***	0.56	0.66	0.66	**0.68***	0.42	0.42
10	0.63	0.72	**0.94***	**0.87**	0.44	0.82	**0.94***	**0.87**	0.64	0.64
11	0.54	0.65	**0.91**	0.79	0.47	**0.82**	**0.93***	0.81	0.62	0.62
12	0.64	0.88	**0.99***	0.94	0.61	0.90	**0.99***	**0.97**	0.68	0.68
13	0.39	0.49	0.51	**0.53**	0.32	**0.53**	0.50	**0.58***	0.39	0.39
14	0.50	0.79	0.85	**0.87***	0.48	0.78	**0.87***	**0.87***	0.52	0.52
15	0.63	0.91	**0.95**	**0.95**	0.55	0.89	**0.95**	**0.96***	0.61	0.61
16	0.29	0.31	**0.51***	**0.51***	0.23	0.31	0.47	**0.48**	0.34	0.34

F1 Score of Noise Detection

data	LC-c	LC-lm	LC-knn	LC-ens	LC-c	LC-lm	LC-knn	LC-ens	ENN	RENN
1	0.22	0.31	**0.37**	0.28	0.31	0.34	0.34	0.32	**0.48**	**0.51***
2	**0.53**	0.51	**0.55**	**0.58***	0.43	0.48	0.49	0.49	0.47	0.50
3	0.15	0.58	0.49	0.42	0.22	**0.60**	0.47	0.43	**0.62**	**0.64***
4	0.06	0.40	0.19	0.17	0.05	**0.41**	0.20	0.19	**0.43**	**0.45***
5	0.76	0.76	**0.93**	**0.90**	0.85	0.75	**0.97***	0.89	0.78	0.80
6	0.69	0.71	**0.87***	**0.85**	0.69	0.67	**0.81**	0.79	0.74	0.74
7	0.61	0.16	**0.71***	0.59	0.64	0.20	**0.68**	0.62	**0.66**	**0.66**
8	0.28	0.20	**0.41**	0.28	0.21	0.24	**0.41**	0.27	**0.53***	**0.53***
9	0.50	**0.67**	**0.67**	**0.68***	0.52	**0.68***	0.64	0.66	0.53	0.53
10	0.58	0.67	**0.83***	0.73	0.40	0.70	**0.78**	0.69	**0.74**	0.72
11	0.46	0.59	**0.80***	0.63	0.44	0.61	**0.74**	0.62	**0.72**	**0.72**
12	0.57	0.69	**0.79**	0.71	0.57	0.74	**0.86***	**0.78**	0.77	0.77
13	0.34	0.40	0.43	0.41	0.31	0.35	**0.44**	0.40	**0.48***	**0.48***
14	0.29	**0.65**	0.57	0.55	0.34	**0.66***	0.56	0.54	**0.63**	**0.63**
15	0.47	**0.84**	0.67	0.72	0.41	**0.86***	0.62	0.70	0.72	**0.74**
16	0.14	**0.37**	0.36	0.30	0.10	0.35	0.25	0.20	**0.45**	**0.46***

Table 17: Noise detection performance on real data: noise recall (top), noise precision (middle) and F1 score of noise detection (bottom)

similarities of samples within the same group while also maximizing pairwise similarities of samples between different groups. In SNC, we construct a graph whose minimum cut solves the linearized version of the objective function of HNC.

In LC-SNC, we suggest the following changes to the graph construction. First, SNC assumes that nodes of positive samples belong to the source set of the cut while those of negative samples belong to the sink set. This is equivalent to having those nodes connected to either the source or the sink node with infinite weights. However, with the presence of noisy labels, this assumption no longer holds. We replace the infinite weights by the so-called "label confidence", which measures how likely it is that the given label of each sample is correct. In this work, we experiment with three label confidence weights: constant weight, local mean weight and k-neighbors weight. These weights take distances between samples as input, as well as other tunable parameters.

The second modification is motivated by the asymmetry between the source set and the sink set in the objective function of the HNC. In the objective function of HNC, the intra-similarity that is maximized is limited to the source set, or the positive class. In this work, we consider a function that also takes into account the pairwise similarities within the sink set. As a result, in the graph on which we solve for a minimum cut, the edges that connect the unlabeled samples to the source node can be connected to the sink node instead. This is incorporated into the model as another parameter to be tuned.

The results from the experiment on synthetic data reveals that by incorporating the label confidence into the model, LC-SNC is less sensitive to noisy labels than SNC. Interestingly, the overall performance of LC-SNC is better than that of SNC even without the presence of label noise. The improvement that the label confidence contributes is particularly large when we evaluate the models on their balanced accuracy and F1 score. When compared to the k-nearest neighbor classifier, LC-SNC methods also show decent improvement. We compare them to the kNN since both models rely on pairwise similarities.

Moreover, the flexibility regarding edges between unlabeled samples and the sink node also enhances the performance of the models, especially when evaluating the balanced accuracy of the prediction, in which we see improvement in the range between 0.6% and 1.3%. This improvement is impressive since it is an improvement from the set of models (LC-SNC with only edges between the source and unlabeled samples) that already improved from the baseline model (SNC). Furthermore, it was shown that, with the second modification, the model learns to adjust the graph structure according to the balance between the two classes, which help maintain the performance of the models, especially when tested on balanced accuracy of the prediction.

However, when testing on the real data, the relative performance of LC-SNC varies to some extent. In comparison with SNC, LC-SNC with label confidence still outperforms in general, for the majority of the datasets, and for all noise levels, with regards to all evaluation metrics. We see from the p-values of Wilcoxon test that the improvement provided by LC-c, LC-knn and LC-ens are statistically significant. Regarding the relative performance with kNN, LC-SNC models yield worse prediction accuracy when there is no noise. When the noise is at 20%, we see slightly better results from LC-knn and LC-ens, but the difference is still not very significant. The improvement is more notable when we look at the balanced accuracy given by the models, especially when the data contains noisy labels.

Regarding the effect of connecting the sink node to unlabeled samples, the improvement is not as evident as with synthetic data. More improvement can be seen when we evaluate the models on their balanced accuracy. We see clearer changes with data that are more unbalanced, although the correspondence remains inconclusive since we might need more data instances to represent each balancedness.

This set of experiment that we conducted in this work and the results, however, rely on several assumptions that are part of the experiment set-up. First, we assume that the balance between the two classes in the labeled set and the unlabeled set are similar. We expect that the improvement that the modification about edges between unlabeled samples and the sink node brings in might be affected when this assumption no longer holds since the inclusion of this set of edges are learnt as a tunable parameter from the cross validation step on the labeled set. Second, we assume that the noise level among the two classes are equal. There are indeed scenarios in practice when one class is more susceptible to being corrupted than the other class.

6. Future Directions

First of all, as mentioned earlier, our experiment relies on several assumptions, mainly the same noise level in the two classes and the same class balances in the labeled and unlabeled sets. It would be interesting to see how the relaxation of these assumptions might affect the performance of the models.

Second, even though the LC-SNC methods and the SNC method, are semi-supervised learning tools, we still rely on a decent amount of information from the labeled samples. The computation of label confidence that leverages more information from the unlabeled samples might be an area to investigate further. This is important particularly for semi-supervised tasks since in many semi-supervised tasks, the availability of labeled samples is quite limited.

In addition to a more intricate information extraction from unlabeled samples, the differences in roles that labeled samples and unlabeled samples play in the model can be made more distinct. For instance, the graph construction around labeled samples and unlabeled samples can be made different. Currently, the graph construction among the labeled samples, not including the sink and source nodes, do not rely on the given labels at all even though they can bring in so much insight about the data.

Furthermore, the LC-SNC methods, as well as the SNC method, solely relies on pairwise similarity between samples. Once the similarity for any pair is computed, the features that are used in the computation essentially becomes unutilized later even though these features might contain a good amount of information that can be helpful to the prediction. Therefore, it can be worthwhile to explore how to use these features to assist our models other than just in the similarity computation.

Last but not least, it could be interesting to make our method more inductive. Currently, our methods only predict the labels of unlabeled samples that are used in the training process. In other words, the prediction is made only on unlabeled samples that are part of the training process. If an unseen unlabeled sample is given, the model will have to be retrained (the minimum cut will have to be resolved) from the beginning. This is unlike inductive methods such as logistic regression, for instance, where new unseen unlabeled samples can be introduced and the model can predict their labels right away without having to be trained again. Regarding this research direction, several works have used graph-based methods to obtain labels of a given batch of unlabeled samples before using all samples to train an inductive supervised classifier. Kveton et al. (2010) applied the minimum cut method to predict the labels of unlabeled samples before training a support vector machine classifier on the combined set of labeled and unlabeled samples, which are now labeled. Another work by Bengio et al. (2006) also uses the minimum cut method to predict the labels of all samples first before predicting the label of an unseen unlabeled data point by comparing the total weights between this new sample and samples that belong to each class. Previously unlabeled samples are now treated as labeled samples with pseudo labels. This research subject is definitely another potential area to explore.

References

Philipp Baumann, Dorit S Hochbaum, and Yan T Yang. A comparative study of the leading machine learning techniques and two new optimization algorithms. *European journal of operational research*, 272(3):1041–1057, 2019.

Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. 11 label propagation and quadratic criterion. 2006.

Kristin Bennett and Ayhan Demiriz. Semi-supervised support vector machines. *Advances in Neural Information processing systems*, 11, 1998.

Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. 2001.

Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning. 2006. *Cambridge, Massachusettes: The MIT Press View Article*, 2, 2006.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2013.

Dorit S Hochbaum. Solving integer programs over monotone inequalities in three variables: A framework for half integrality and good approximations. *European Journal of Operational Research*, 140(2):291–321, 2002.

Dorit S Hochbaum. The pseudoflow algorithm: A new algorithm for the maximum-flow problem. *Operations research*, 56(4):992–1009, 2008.

Dorit S Hochbaum. Polynomial time algorithms for ratio regions and a variant of normalized cut. *IEEE transactions on pattern analysis and machine intelligence*, 32(5):889–898, 2009.

Tony Jebara, Jun Wang, and Shih-Fu Chang. Graph construction and b-matching for semi-supervised learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 441–448, 2009.

Branislav Kveton, Michal Valko, Ali Rahimi, and Ling Huang. Semi-supervised learning with max-margin graph cuts. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 421–428. JMLR Workshop and Conference Proceedings, 2010.

Yoshihiro Mitani and Yoshihiko Hamamoto. A local mean-based nonparametric classifier. *Pattern Recognition Letters*, 27(10):1151–1159, 2006.

D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. Uci repository of machine learning databases, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

Celso André R de Sousa, Solange O Rezende, and Gustavo EAPA Batista. Influence of graph construction on semi-supervised learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 160–175. Springer, 2013.

Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020.

Jun Wang, Tony Jebara, and Shih-Fu Chang. Semi-supervised learning using greedy max-cut. *The Journal of Machine Learning Research*, 14(1):771–800, 2013.

Zhenyu Wu and Richard Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 15(11):1101–1113, 1993.

Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.

7. Appendix: Tables

7.1 Additional results from experiments on synthetic data

pos:neg = 70:30				
noise level	LC-c	LC-lm	LC-knn	LC-ens
0	1.05	0.52	0.59	1.02
10	1.44	0.96	1.26	1.69
20	1.41	1.02	1.75	2.03
30	1.03	0.08	1.76	2.18

pos:neg = 50:50				
noise level	LC-c	LC-lm	LC-knn	LC-ens
0	1.81	1.41	1.55	1.90
10	1.64	1.27	1.97	2.36
20	1.63	1.23	2.28	2.93
30	0.40	0.42	1.97	2.89

pos:neg = 30:70				
noise level	LC-c	LC-lm	LC-knn	LC-ens
0	1.24	1.11	1.14	1.55
10	1.80	1.17	1.75	2.15
20	1.41	0.43	2.17	2.62
30	-1.79	-2.24	0.21	1.42

Table 18: Average accuracy improvement of LC-SNC methods over SNC, grouped by the balance between the two classes

pos:neg = 70:30				
noise level	LC-c	LC-lm	LC-knn	LC-ens
0	1.60	0.48	0.41	1.57
10	2.12	1.46	1.59	2.49
20	2.70	2.04	3.07	3.64
30	2.71	1.92	3.68	4.21

pos:neg = 50:50				
noise level	LC-c	LC-lm	LC-knn	LC-ens
0	1.69	1.29	1.34	1.74
10	1.60	1.24	2.01	2.34
20	1.46	1.00	2.30	2.83
30	0.25	0.41	1.75	2.75

pos:neg = 30:70				
noise level	LC-c	LC-lm	LC-knn	LC-ens
0	2.10	1.64	1.07	2.31
10	2.39	1.55	2.06	2.81
20	6.29	4.64	6.77	7.21
30	4.51	3.24	5.80	6.88

Table 19: Average balanced accuracy improvement of LC-SNC methods over SNC, grouped by the balance between the two classes

7.2 Additional results from experiments on real data

F1 Score (no noise)

data	SNC	LC-c	LC-lm	LC-knn	LC-ens	kNN	ENN	RENN
1	47.88	**48.19**	**48.49**	47.81	**48.66***	41.35	42.45	41.72
2	89.15	**89.18**	88.88	**89.16**	**89.21***	89.09	88.50	88.84
3	77.29	78.15	78.16	78.68	**78.85**	**79.00***	**78.69**	77.71
4	**58.28***	54.95	**57.28**	54.00	**55.96**	47.50	46.23	38.25
5	99.03	99.77	99.13	99.77	99.77	**99.94***	**99.90**	**99.90**
6	98.65	**99.04**	98.38	**99.08**	**99.12***	98.93	98.74	98.68
7	93.93	**94.92***	94.14	94.44	**94.88**	**94.75**	92.81	92.81
8	**84.43**	84.11	83.19	83.12	**84.19**	**85.36***	82.47	82.47
9	85.98	85.92	85.85	**86.22**	**86.12**	**86.60***	85.60	85.60
10	97.88	97.89	97.64	**98.13**	**98.13**	**98.23***	98.06	98.00
11	**99.41**	**99.44***	99.02	99.03	**99.36**	99.34	98.86	98.86
12	98.58	99.08	98.96	99.37	99.13	**99.93***	**99.80**	**99.80**
13	79.78	**85.60***	85.22	81.87	**85.44**	**85.51**	79.05	77.97
14	92.93	**93.52**	93.17	**93.52**	**93.57***	93.46	92.79	92.74
15	96.02	**96.77***	96.46	96.60	**96.71**	**96.71**	96.63	96.51
16	86.60	86.80	86.27	**86.93**	**86.82**	**87.03***	84.97	85.53

F1 Score (noise level: 20%)

data	SNC	LC-c	LC-lm	LC-knn	LC-ens	kNN	ENN	RENN
1	**38.13**	37.82	37.56	**38.01**	**38.95***	33.74	35.70	33.30
2	88.12	**88.54**	88.20	**88.57**	88.52	**88.66***	84.48	87.18
3	63.78	70.11	68.73	**71.72**	**72.59**	71.33	71.38	**72.60***
4	48.95	**51.20**	**53.74***	50.92	**53.31**	43.37	43.16	35.87
5	97.65	**98.23**	96.56	**98.66**	**98.81***	98.01	94.05	97.02
6	94.68	**95.50**	95.29	**95.73**	**96.21***	94.85	93.06	93.06
7	84.17	85.18	**86.64**	85.84	86.45	**87.60***	**86.52**	**86.52**
8	71.37	74.02	72.69	74.02	74.31	**74.75**	**75.01***	**75.01***
9	80.80	80.84	**84.49**	80.33	**82.41**	**85.50***	80.50	80.50
10	96.81	**97.67**	93.61	97.24	**97.69***	**97.52**	94.66	94.66
11	**95.90**	95.55	92.78	95.25	**95.89**	**96.03***	93.29	93.29
12	98.51	98.49	98.73	**98.94**	**99.01***	**98.90**	95.94	95.94
13	77.33	**77.87**	**77.91**	76.78	**77.93***	75.87	74.09	74.11
14	91.39	**91.41**	91.04	**91.51**	**91.87***	90.61	88.20	88.29
15	95.39	95.73	**95.93***	**95.87**	**95.92**	95.34	94.36	95.35
16	**86.53***	**86.51**	86.08	86.44	**86.51**	86.34	80.16	82.26

Table 20: Average F1 Score of the models for each dataset when there is no noise (above) and 20% noise (below)

Accuracy (no noise)					Accuracy (noise level : 20%)				
data	LC-c	LC-lm	LC-knn	LC-ens	data	LC-c	LC-lm	LC-knn	LC-ens
1	-0.0364	-1.0856	0.1595	0.1579	1	0.1706	-6.6984	0.0202	0.3862
2	0.0306	-0.3476	0.0597	0.1416	2	0.4189	-0.4234	0.4271	0.5075
3	0.3179	-0.0170	0.5571	0.6525	3	-0.5560	-2.0312	0.9309	1.1463
4	1.1620	-0.5224	1.0632	1.5463	4	-0.3138	-3.0139	0.3332	0.4599
5	0.5684	-0.1073	0.5684	0.5684	5	0.1709	-0.2557	0.0613	0.2624
6	0.2780	-0.3372	0.3438	0.3320	6	1.1501	0.2366	1.3302	1.5643
7	1.1133	0.1802	0.4986	0.9825	7	2.0043	0.8356	1.9299	2.4649
8	0.2741	-1.3492	-1.0222	0.3223	8	2.1851	0.1112	3.0704	3.4735
9	0.1141	-0.1289	0.1538	0.1701	9	-5.1746	-0.0524	-0.8548	0.1546
10	0.0881	-0.1513	0.1744	0.2652	10	0.1375	-4.2539	0.0948	0.2866
11	0.0671	-0.5865	-0.3888	-0.0703	11	-0.8019	-5.4529	-0.9262	-0.6830
12	0.6799	0.5208	0.9383	0.7252	12	-0.0503	0.3932	0.0796	0.3606
13	7.0892	6.3330	1.9557	6.7734	13	1.2482	0.7816	1.5717	1.8904
14	0.8304	0.3153	0.8047	0.9002	14	0.9022	0.5273	0.7747	1.2220
15	0.9940	0.5211	0.6224	0.8551	15	0.4295	0.5459	0.5113	0.8175
16	0.9914	0.4105	1.3460	1.1412	16	1.2439	0.0823	1.2606	1.3742

Table 21: Average accuracy improvement percentage over SNC across all labeled-unlabeled splits and label noise corruptions of each dataset

Balanced Accuracy (no noise level)					Balanced Accuracy (noise level : 20%)				
data	LC-c	LC-lm	LC-knn	LC-ens	data	LC-c	LC-lm	LC-knn	LC-ens
1	0.2154	-0.2503	-0.0497	0.7647	1	2.3592	-0.4994	0.9839	2.7300
2	3.4136	3.8133	3.8135	3.9333	2	2.8486	1.8807	2.2839	3.8135
3	-0.6351	0.0070	-1.0566	-0.2652	3	1.8991	3.0788	3.1324	3.7519
4	-2.5919	-0.8960	-3.1811	-1.8246	4	11.1106	12.9000	11.0902	13.2484
5	0.4395	-0.0864	0.4395	0.4395	5	-0.0345	-0.2944	0.3947	0.3653
6	0.3406	-0.2154	0.3829	0.4213	6	1.2441	0.1896	1.3847	1.6666
7	1.1396	0.2009	0.5202	1.0117	7	1.7876	0.6491	1.8019	2.2228
8	0.3642	-1.2749	-0.7049	0.4514	8	2.6017	0.8884	3.5212	4.1042
9	0.1303	-0.0809	0.1457	0.1863	9	-5.1242	0.0365	-0.8957	0.2444
10	-0.0072	-0.2797	0.0789	0.1615	10	0.1388	-4.2103	0.2135	0.2883
11	0.1023	-0.4526	-0.3341	-0.0079	11	-0.7488	-5.5525	-1.1245	-0.6918
12	0.6054	0.4438	0.8505	0.6387	12	-0.0292	0.3315	0.3724	0.4287
13	1.3028	0.8924	-0.9871	1.8928	13	-0.0913	-0.0291	0.5987	1.2009
14	1.1053	0.4656	0.9766	1.1559	14	1.2468	1.1819	1.0871	1.6150
15	1.0141	1.1239	0.5055	0.9271	15	1.0044	0.4946	0.9370	1.2871
16	2.3075	9.8293	2.8154	3.9209	16	0.2584	8.9884	1.9134	3.7693

Table 22: Average balanced accuracy improvement percentage over SNC across all labeled-unlabeled splits and label noise corruptions of each dataset

SNC					kNN				
noise level	LC-c	LC-lm	LC-knn	LC-ens	noise level	LC-c	LC-lm	LC-knn	LC-ens
0	0.0604	0.2346	**0.0440**	**0.0100**	0	0.7654	0.9510	0.9106	0.4794
10	**0.0056**	0.4384	**0.0042**	**0.0011**	10	**0.0279**	0.7153	**0.0170**	**0.0008**
20	**0.0031**	0.1388	**0.0219**	**0.0004**	20	0.3782	0.7810	0.1897	**0.0394**
30	**0.0007**	**0.0022**	**0.0005**	**0.0002**	30	0.8927	0.9606	0.8927	0.8371

Table 23: P-values of the Wilcoxon test against SNC and kNN on F1 score