

# Positive-Unlabeled Learning Using Pairwise Similarity and Parametric Minimum Cuts

Torpong Nitayanont<sup>a</sup> and Dorit S. Hochbaum<sup>b</sup>

Department of Industrial Engineering and Operations Research, University of California, Berkeley, CA, U.S.A.  
torpong\_nitayanont@berkeley.edu, dhochbaum@berkeley.edu

**Keywords:** Positive-Unlabeled Learning, Binary Classification, Pairwise Similarity, Parametric Minimum Cut.

**Abstract:** Positive-unlabeled (PU) learning is a binary classification problem where the labeled set contains only positive class samples. Most PU learning methods involve using a prior  $\pi$  on the true fraction of positive samples. We propose here a method based on Hochbaum’s Normalized Cut (HNC), a network flow-based method, that partitions samples, both labeled and unlabeled, into two sets to achieve high intra-similarity and low inter-similarity, with a tradeoff parameter to balance these two goals. HNC is solved, for all tradeoff values, as a parametric minimum cut problem on an associated graph producing multiple optimal partitions, which are nested for increasing tradeoff values. Our PU learning method, called *2-HNC*, runs in two stages. Stage 1 identifies optimal data partitions for all tradeoff values, using only positive labeled samples. Stage 2 first ranks unlabeled samples by their likelihood of being negative, according to the sequential order of partitions from stage 1, and then uses the likely-negative along with positive samples to run HNC. Among all generated partitions in both stages, the partition whose positive fraction is closest to the prior  $\pi$  is selected. An experimental study demonstrates that *2-HNC* is highly competitive compared to state-of-the-art methods.

## 1 INTRODUCTION


Positive-unlabeled (PU) learning is a variant of binary classification where labeled samples only come from the positive class. Each unlabeled sample could either belong to the positive or negative class. PU learning is related to the one-class learning problem in which the model is trained solely on the positive labeled set, but unlabeled samples are not utilized (Khan and Madden, 2014). PU learning is also related to semi-supervised learning, where unlabeled samples are used in addition to the labeled set of samples from both classes, giving better performances than one-class learning methods (Lee and Liu, 2003; Li et al., 2010). PU learning is a special case of semi-supervised learning where no negative labeled samples are provided.


PU learning arises in contexts where negative samples are difficult to verify or obtain, and when the absence of positive label does not always imply that the sample is negative. In personalized advertising (Yi et al., 2017; Bekker and Davis, 2020), each advertisement that is clicked is a positive sample. However, an

unclicked advertisement is regarded as unlabeled as it could either be uninteresting (*negative*) or interesting but overlooked (*positive*). In the identification of malignant genes (Yang et al., 2012; Yang et al., 2014a), a limited set of genes have been verified to cause diseases (*positive*) while many other genes have not been evaluated (*unlabeled*). Other domains include fake reviews detection (Li et al., 2014; Ren et al., 2014) and remote sensing (Li et al., 2010).

A natural way to deal with the absence of negative labeled samples is to identify unlabeled samples that are likely negative, and train a traditional classifier using the positive labeled set and the likely-negative unlabeled set (Liu et al., 2002; Li and Liu, 2003). Another common approach is to train a classifier on a modified risk estimator, in which each unlabeled sample can be regarded as positive and negative with different weights. This idea has been adopted in different learning methods such as neural network models (Du Plessis et al., 2014; Du Plessis et al., 2015; Kiryo et al., 2017), and random forest (Wilton et al., 2022) with a modified impurity function. Most of these methods rely on the prior information of the fraction of positive samples,  $\pi$ , in the dataset.

The method that we propose here is based on a network flow-based method called Hochbaum’s Nor-

<sup>a</sup>  <https://orcid.org/0009-0002-6976-1951>

<sup>b</sup>  <https://orcid.org/0000-0002-2498-0512>

malized Cut (HNC) (Hochbaum, 2010). HNC partitions samples into two sets to achieve high intra-similarity within sets and low inter-similarity between the two, with a tradeoff parameter that balances the two goals. The problem was shown in (Hochbaum, 2010) to be solved, for all values of the tradeoff parameter, as a minimum cut problem on an associated graph. This method was previously used as in binary classification where both positive and negative labeled samples are available (Yang et al., 2014b; Baumann et al., 2019). HNC is applicable in PU learning since it does not require labeled samples from both classes. Moreover, it makes use of unlabeled samples through their similarities with labeled samples and among themselves, making it advantageous when labeled data is limited.

As a transductive method, HNC predicts labels only for the given unlabeled samples. This is different from inductive methods that make predictions for any unlabeled samples, whether they are the given unlabeled samples, or unseen, separate set of unlabeled samples. Indeed, HNC can be extended and used as an inductive classifier.

The main contribution of this work is a new method for PU learning that utilizes the unique features of HNC in two stages, called *2-HNC*. Stage 1 generates multiple partitions of data samples, corresponding to different tradeoff values, efficiently, with a parametric cut procedure. We infer from the sequence of partitions in stage 1 the likelihood of unlabeled samples to be negatively labeled. Based on this, stage 2 generates a set of likely-negative unlabeled samples and apply HNC using both the positive samples and the likely-negative samples. Among all partitions generated in both stages, the one whose fraction of positive samples is closest to the given prior  $\pi$  is selected as the prediction for unlabeled samples.

Additional and independent contribution here is the method of extracting likely-negative samples from the unlabeled set using results of stage 1. This method has potential uses in settings other than PU learning. Another contribution of this work is the consideration of the intra-similarities of both positive and negative prediction sets, in data partitioning. This is in contrast to past uses of HNC, such as in (Baumann et al., 2019; Spaen et al., 2019; Asín Achá et al., 2020), where the scenario considered was to maximize the intra-similarity of the positive prediction set only.

We show via experiments on real data that *2-HNC* outperforms leading methods, which include two standard benchmarks, *uPU* (Du Plessis et al., 2014; Du Plessis et al., 2015) and *nmPU* (Kiryo et al., 2017), as well as a recent state-of-the-art tree-based method, *PU ET* (Wilton et al., 2022).

## 2 RELATED WORKS

The main challenge of PU learning is the lack of negative labeled samples. A number of methods utilize a preprocessing step to identify a set of unlabeled samples that are likely to be negative prior to training a traditional binary classifier. For instance, the Spies technique (Liu et al., 2002) selects a few positive labeled samples as *spies* and include them in the unlabeled set, all of which are treated as negative. With a binary classifier trained on this data, unlabeled samples with lower posterior probability than the spies are considered likely to be negative. The Rocchio method (Li and Liu, 2003) marks unlabeled samples that are closer to the centroid of unlabeled samples than that of positive labeled samples as likely negative. (Lu and Bai, 2010) used Rocchio to also expand the positive labeled set when a small labeled set is given.

Another common approach in recent works is to train a model based on an empirical risk estimator, modified in the context of PU learning. (Du Plessis et al., 2014; Du Plessis et al., 2015) proposed *uPU*, an unbiased risk estimator for PU data on which neural network models are trained. (Kiryo et al., 2017) mitigates the overfitting nature of *uPU* via a non-negative risk estimator in their state-of-the-art method known as *nmPU*. There are also works on other classifiers, besides deep learning models, that apply this similar idea such as a random forest model called *PU ET* (Wilton et al., 2022), in which the impurity function is modified for PU data. *PU ET* gives competitive results, especially on tabular data type where deep learning PU methods are not always effective.

There are methods, other than the above, which rely on pairwise similarities between samples. In label propagation method of (Carnevali et al., 2021), a graph representation of the data is constructed with edge weights that reflect pairwise similarities. The likelihood of being negative for each unlabeled sample is inferred based on its shortest path distance on the graph to the positive labeled set. Labels are then propagated from the positive and likely-negative unlabeled samples to the remaining unlabeled ones. (Zhang et al., 2019) presented a maximum margin-based method that penalizes similar samples that are classified differently. While methods like (Carnevali et al., 2021; Zhang et al., 2019) utilize graph representation of the data as well as pairwise similarities, a network-flow based approach, which is a closely related area, has never been utilized in PU learning.

Hochbaum’s Normalized Cut or HNC (Hochbaum, 2010) has been used in binary classification, where labeled samples from both classes are given. It was shown to be competitive in many

applications (Baumann et al., 2019; Spaen et al., 2019; Yang et al., 2014b). In this work, we devise a variant of HNC for PU learning, called *2-HNC*. We compare 2-HNC to the following benchmarks: *uPU* (Du Plessis et al., 2014), *nnPU* (Kiryo et al., 2017) and *PU ET* (Wilton et al., 2022). *uPU* and *nnPU* are selected as standard PU learning benchmarks. *nnPU* exhibited competitive performance consistently, mostly on image and text data. *PU ET*, a recent state-of-the-art method, demonstrated leading performance, particularly on tabular data where it outperformed deep learning models. Similar to most PU methods, the fraction of positive samples in the data, or  $\pi$ , is given as a prior information for 2-HNC and the benchmark methods.

### 3 PRELIMINARIES, NOTATION AND HNC

#### 3.1 Notations

Given a dataset  $V$  with a set of positive labeled samples  $L^+$  and a set of unlabeled samples  $U$ , which is a mixture of positive and negative samples, the goal is to predict the label, or class, of each sample in  $U$ . We formalize the PU-learning task as a graph problem.

Let the directed graph  $G = (V, A)$  represent the data with  $V$ , the set of vertices that corresponds to samples in the data, and  $A = \{(i, j) | i, j \in V, i \neq j\}$  the set of arcs that connect each sample pair. Arcs  $(i, j)$  and  $(j, i)$  that connect  $i$  and  $j$  carry the same capacity weight  $w_{ij}$ , which reflects the symmetry of pairwise similarity of  $i$  and  $j$ .

#### 3.2 Hochbaum’s Normalized Cut (HNC)

Given a dataset, with the set of samples  $V$  and pairwise similarities  $w_{ij}$  for  $i, j \in V$ , the goal of HNC is to find a partition of  $V$  to two non-empty sets  $S$  and  $\bar{S}$  that optimizes the tradeoff between two objectives: high *intra*-similarity within the set  $S$  and small *inter*-similarity between  $S$  and its complement  $\bar{S}$ . We denote their inter-similarity by  $C(S, \bar{S})$ , defined as  $\sum_{i \in S, j \in \bar{S}} w_{ij}$ . The intra-similarity within  $S$  is defined as  $C(S, S) = \sum_{i, j \in S, i < j} w_{ij}$ . HNC, with a tradeoff parameter  $\mu \geq 0$ , is the following problem:

$$(HNC+) \quad \underset{\emptyset \subset S \subset V}{\text{minimize}} \quad C(S, \bar{S}) - \mu C(S, S) \quad (1)$$

Because of the symmetry between  $S$  and  $\bar{S}$ , the problem can be alternatively presented for the trade-

off between the intra-similarity within  $\bar{S}$  and the inter-similarity between it and its complement.

$$(HNC-) \quad \underset{\emptyset \subset S \subset V}{\text{minimize}} \quad C(S, \bar{S}) - \mu C(\bar{S}, \bar{S}) \quad (2)$$

One might consider a variant of HNC that incorporates both intra-similarities,  $C(S, S)$  and  $C(\bar{S}, \bar{S})$ , as a more generalized version of both HNC+ (1) and HNC- (2). This variant, with two tradeoff weights  $\alpha \geq 0$  and  $\beta \geq 0$ , is given as problem (3) below.

$$\underset{\emptyset \subset S \subset V}{\text{minimize}} \quad C(S, \bar{S}) - \alpha C(S, S) - \beta C(\bar{S}, \bar{S}) \quad (3)$$

However, as proved in the next lemma, problem (3) is equivalent to either HNC+ or HNC-, depending on the relative values of  $\alpha$  and  $\beta$ .

**Lemma 3.1.** *Problem (3) is equivalent to HNC+ (1) when  $\alpha \geq \beta$  for  $\mu = \frac{\alpha - \beta}{1 + \beta}$ , and is equivalent to HNC- (2) when  $\alpha < \beta$  for  $\mu = \frac{\beta - \alpha}{1 + \alpha}$ .*

*Proof.*  $C(V, V)$  is a constant, which we denote by  $W_V$ , and is equal to  $C(S, \bar{S}) + C(S, S) + C(\bar{S}, \bar{S})$  for any nonempty  $S \subset V$ . Hence, the objective function of (3) can be written as  $C(S, \bar{S}) - \alpha C(S, S) - \beta(W_V - C(S, \bar{S}) - C(S, S)) = (1 + \beta)(C(S, \bar{S}) - \frac{\alpha - \beta}{1 + \beta} C(S, S)) - \beta W_V$ . Minimizing this function is equivalent to solving (1) with the tradeoff  $\mu = \frac{\alpha - \beta}{1 + \beta} \geq 0$  when  $\alpha \geq \beta$ .

Alternatively, the objective function of (3) can be written as  $C(S, \bar{S}) - \alpha(W_V - C(S, \bar{S}) - C(\bar{S}, \bar{S})) - \beta C(\bar{S}, \bar{S}) = (1 + \alpha)(C(S, \bar{S}) - \frac{\beta - \alpha}{1 + \alpha} C(\bar{S}, \bar{S})) - \alpha W_V$ . Hence, minimizing this objective is equivalent to solving (2) with  $\mu = \frac{\beta - \alpha}{1 + \alpha} \geq 0$  when  $\alpha < \beta$ .  $\square$

Therefore, instead of solving (3) where the two intra-similarities are shown explicitly, it is sufficient to consider either HNC+ or HNC- depending on whether we put more weight on the intra-similarity of  $S$ , or of  $\bar{S}$ . We note that in prior applications of HNC to binary classification, e.g. (Yang et al., 2014b; Baumann et al., 2019), the model was the one that considered the intra-similarity in  $S$  only, as in HNC+.

Applying HNC in binary classification, when labeled samples from both classes are given, the goal is to partition a data that consists of the positive and negative labeled sets,  $L^+$  and  $L^-$ , as well as the unlabeled set,  $U$ , into  $S$  and  $\bar{S}$ , and predict the labels of unlabeled samples in  $U$  accordingly. In previous works, e.g. (Yang et al., 2014b; Baumann et al., 2019), the labeled sets are used as *seeds* and either HNC+ or HNC- is solved with the restriction that  $L^+ \subseteq S \subseteq V \setminus L^-$ . Unlabeled samples in the optimal  $S^*$  and  $\bar{S}^*$  are predicted positive and negative, respectively.

## 4 2-HNC: A TWO-STAGE METHOD FOR PU LEARNING

In this section, we describe the 2-HNC method where HNC is applied in two stages in PU learning where only the positive labeled set  $L^+$  and the unlabeled set  $U$  are given. We then show how the optimization problems in 2-HNC are solved as parametric minimum cut problems on associated graphs.

### 4.1 2-HNC for PU Learning

The 2-HNC method consists of two stages. In stage 1, we solve HNC- using only the given positive labeled set. In stage 2, we utilize the likely-negative samples extracted from the unlabeled set based on the result of the first stage, prior to solving HNC+ using both the positive labeled set and the likely-negative set. The output solution is the one data partition, among those that were generated in both stages, that has the fraction of positive samples closest to the ratio  $\pi$ , given as prior.

#### 4.1.1 Stage 1: Solving HNC- with Positive Labeled Samples

The given positive labeled set  $L^+$  is used as the seed set for the set  $S$  in HNC+ and HNC-. Since no negative labeled samples are provided,  $L^- = \emptyset$ , that is, no seed sample is required to be in  $\bar{S}$ . The seed set constraint imposed on HNC+ and HNC- is then  $L^+ \subseteq S$ .

Without a seed set for  $\bar{S}$ , HNC+ is not well defined: the optimal solution to HNC+ is always  $(S^*, \bar{S}^*) = (V, \emptyset)$  for any tradeoff  $\mu \geq 0$ . That is, HNC+ has only the trivial solution in which all unlabeled samples are predicted to be positive. HNC+, however, will be used in stage 2 when likely-negative samples are available.

On the other hand, HNC-, with only positive labeled samples, gives non-trivial data partitions for various values of the tradeoff parameter.

The optimal data partition for HNC- is dependent on the tradeoff  $\mu$ . We solve HNC-, under the constraint  $L^+ \subseteq S$ , for all tradeoff  $\mu \geq 0$  as a parametric minimum cut problem on an associated parametric graph. For  $\mu = 0$ , the optimal partition  $(S^*, \bar{S}^*)$  is  $(V, \emptyset)$ . As  $\mu$  increases, the optimal partition gradually changes, for some  $\mu$ , until  $\mu$  reaches a sufficiently large value, at which  $(S^*, \bar{S}^*)$  is  $(L^+, V \setminus L^+)$ . The result of the associated parametric minimum cut problem is a sequence of data partitions:  $(S_1^*, \bar{S}_1^*), (S_2^*, \bar{S}_2^*), \dots, (S_q^*, \bar{S}_q^*)$ , that correspond to increasing values of  $\mu$ . Here,  $q$  is the number of different partitions in the parametric minimum cut so-

lution, and can be different for different data. This sequence of partitions for increasing values of  $\mu$ , in fact, is nested. That is,  $\bar{S}_1^* \subseteq \bar{S}_2^* \subseteq \dots \subseteq \bar{S}_q^*$ . We discuss the procedure of solving HNC- as a parametric minimum cut problem, as well as *the nested cut property* in Section 4.2. Stage 1 ends here with the data partition sequence, that is the optimal solution to HNC- for different tradeoff values, as an output.

#### 4.1.2 Stage 2: Solving HNC+ with Positive Labeled Samples and Likely-Negative Unlabeled Samples

Solving HNC- in stage 1 does not require negative labeled samples and gives us, for each tradeoff  $\mu$ , a partition of data samples into the positive prediction set  $S^*$  and the negative prediction set  $\bar{S}^*$ . However, HNC- only considers the scenario where the intra-similarity of the negative prediction set  $\bar{S}$  is given higher importance than that of the positive prediction set  $S$ . Here, we consider HNC+, before combining the results from both stages as a final step of 2-HNC.

To handle the issue of HNC+ being ill-defined in the absence of the negative labeled set, as discussed in Section 4.1.1, we add to the problem the seeds for  $\bar{S}$ . We select the set of samples that are likely to be negative, or  $L^N$ , from the unlabeled set  $U$  as the seed set for  $\bar{S}$ . The random sampling procedure to form  $L^N$ , called *SelectNeg*, is based on the results of stage 1.

SelectNeg takes as input the sequence of optimal data partitions  $(S_1^*, \bar{S}_1^*), (S_2^*, \bar{S}_2^*), \dots, (S_q^*, \bar{S}_q^*)$ , which are the results of solving HNC- for all  $\mu \geq 0$ , for increasing values of  $\mu$ , in stage 1. The nested sequence  $\bar{S}_1^* \subseteq \bar{S}_2^* \subseteq \dots \subseteq \bar{S}_q^*$  starts from  $\bar{S}_1^* = \emptyset$  and expands until  $\bar{S}_q^* = V \setminus L^+$ , which is the largest possible since we require  $L^+$  to be in  $S_q^*$ . The implication of the nestedness is that, for an unlabeled sample that is predicted negative for a particular  $\mu$ , it is also predicted negative for any larger value of  $\mu$ .

We consider unlabeled samples that belong to the negative prediction set  $\bar{S}^*$  for small  $\mu$  as likely to be negative. As  $\mu$  increases from zero, these samples are predicted negative before other unlabeled samples. Formally, for an unlabeled sample  $i \in U$ , we denote  $q_i = \max\{\gamma \mid i \in S_\gamma^*\}$  as the index of the last partition in the sequence where sample  $i$  is still in the positive prediction set.  $\eta(i) = |\bar{S}_{q_i}^*|$  is the number of samples that are predicted negative at the same or larger values of tradeoff  $\mu$ . A large  $\eta(i)$  implies that sample  $i$  is more likely to be predicted negative than a large number of samples. In our sampling method SelectNeg, the probability that unlabeled sample  $i$  is selected as likely-negative is  $\eta(i) / \sum_{j \in U} \eta(j)$ . The number of likely-negative samples to be selected, or the

size of the set  $L^N$ , is chosen in this work to be equal to the number of the positive labeled samples,  $|L^+|$ .

Once the likely-negative set  $L^N$  is formed, we use the positive labeled set  $L^+$  and the likely-negative set  $L^N$  as the seed sets for  $S$  and  $\bar{S}$ , respectively, and solve HNC+ with the seed set constraint  $L^+ \subseteq S \subseteq V \setminus L^N$ . As a result, the output from stage 2 is another sequence of data partitions, which are the optimal solutions to HNC+ for all nonnegative tradeoff  $\mu$ .

#### 4.1.3 Combining Results from Both Stages

Among all data partitions generated in both stages, we select the partition whose positive fraction, computed as  $\frac{|S^*|}{|V|}$  for a partition  $(S^*, \bar{S}^*)$ , is closest to the prior  $\pi$ . Unlabeled samples in  $S^*$  of the selected partition are predicted positive, and those in  $\bar{S}^*$  negative.

## 4.2 Solving Parametric Minimum Cut Problems in 2-HNC

We mentioned in the previous subsection that 2-HNC involves solving HNC+ and HNC- as parametric minimum cut problems on associated graphs. We first explain how the two problems are solved for a single tradeoff  $\mu \geq 0$  as minimum cut problems, in Subsection 4.2.1. In 2-HNC, we solve them for all tradeoffs  $\mu \geq 0$ , prior to selecting one partition from all that are generated. We describe how this is done as parametric minimum cut problems in Section 4.2.2. The nested cut property of the partition sequence as a result of stage 1 is also discussed here.

### 4.2.1 Solving HNC+ and HNC- for a Tradeoff Parameter $\mu$ as a Minimum Cut Problem

HNC+ and HNC- are special cases of *monotone integer programs*, (Hochbaum, 2002; Hochbaum, 2021), and as such can be solved as a minimum cut problem on an associated graph, which is a mapping from the integer programming formulation of both problems (Hochbaum, 2010). This is because any monotone integer programming problem can be solved as a minimum cut problem on an associated graph, the construction of which is a mapping from the formulation (Hochbaum, 2002; Hochbaum, 2021).

Using the standard formulations of HNC+ and HNC-, in the associated graph, there is a node for each sample, and a node for each pair of samples. As a result, the size of this graph is quadratic in the size of the data. However, there are alternative formulations that are ‘‘compact’’, (Hochbaum, 2010), in that the associated graph has number of nodes equal to the number of samples,  $|V|$ , only. The alternative formulations are

shown for HNC+ and HNC- in the following lemma.

**Lemma 4.1.** *HNC+ is equivalent to the following problem:*

$$\underset{\emptyset \subset S \subset V}{\text{minimize}} \quad C(S, \bar{S}) - \lambda \sum_{i \in S} d_i \quad (4)$$

and HNC- is equivalent to

$$\underset{\emptyset \subset S \subset V}{\text{minimize}} \quad C(S, \bar{S}) - \lambda \sum_{i \in \bar{S}} d_i \quad (5)$$

where  $\lambda = \frac{\mu}{\mu+2}$  and  $d_i = \sum_{j \in V \setminus \{i\}} w_{ij}$  for  $i \in V$ .

*Proof.*  $C(S, S) = \sum_{i,j \in S, i < j} w_{ij} = \frac{1}{2} \sum_{i \in S} \sum_{j \in S \setminus \{i\}} w_{ij}$  since  $w_{ij} = w_{ji}$ .  $\sum_{i \in S} \sum_{j \in S \setminus \{i\}} w_{ij} = \sum_{i \in S} (\sum_{j \in V \setminus \{i\}} w_{ij} - \sum_{j \in \bar{S}} w_{ij}) = \sum_{i \in S} d_i - C(S, \bar{S})$ . Hence,  $C(S, S) = \frac{1}{2} (\sum_{i \in S} d_i - C(S, \bar{S}))$

We rewrite the objective of HNC+ as  $C(S, \bar{S}) - \frac{\mu}{2} (\sum_{i \in S} d_i - C(S, \bar{S})) = (1 + \frac{\mu}{2}) C(S, \bar{S}) - \frac{\mu}{\mu+2} \sum_{i \in S} d_i$ . Hence, HNC+ can be solved by minimizing (4):  $C(S, \bar{S}) - \lambda \sum_{i \in S} d_i$ , with  $\lambda = \frac{\mu}{\mu+2}$ . The equivalence of HNC- and (5) can be shown similarly by rewriting  $C(\bar{S}, S)$  in HNC- as  $\frac{1}{2} (\sum_{i \in \bar{S}} d_i - C(S, \bar{S}))$ .  $\square$

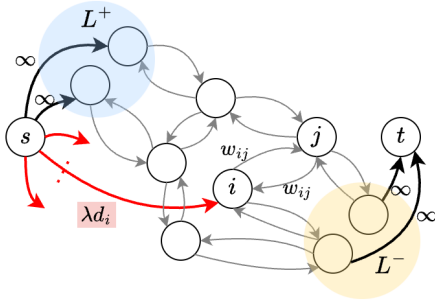
When both labeled sets  $L^+$  and  $L^-$  are given, the seed set constraint is  $L^+ \subseteq S \subseteq V \setminus L^-$ . Under this constraint, the solution to HNC+ for a tradeoff  $\mu$ , which is now solved via (4) with a tradeoff  $\lambda = \frac{\mu}{\mu+2}$ , is obtained from the minimum cut solution of the associated graph,  $G_{st}^+(\lambda)$ . Let  $(\{s\} \cup S^*, \{t\} \cup \bar{S}^*)$  denote the minimum cut solution of  $G_{st}^+(\lambda)$ . Then,  $(S^*, \bar{S}^*)$  is the optimal solution to HNC+. The proof provided in (Hochbaum, 2010) is omitted here.

The construction of  $G_{st}^+(\lambda)$  for (4), with the constraint  $L^+ \subseteq S \subseteq V \setminus L^-$ , is illustrated in Figure 1a and described as follows: We add to graph  $G$ , described in Section 3.1, source node  $s$  and sink node  $t$ , and connect  $s$  to all nodes of samples in  $L^+$  with arcs of infinite capacity. Similarly, nodes in  $L^-$  are connected to  $t$  with arcs of infinite capacity. In addition, all unlabeled sample nodes,  $i \in V \setminus (L^+ \cup L^-)$ , or equivalently  $i \in U$ , have arcs from  $s$  to  $i$  of capacity  $\lambda d_i$ .

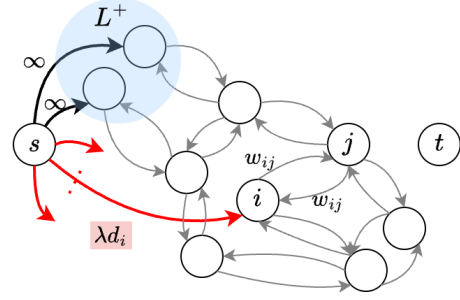
Let  $(\{s\} \cup S^*, \{t\} \cup \bar{S}^*)$  be the minimum cut solution of  $G_{st}^+(\lambda)$ , then we predict unlabeled samples in  $S^*$  are positive, and those in  $\bar{S}^*$  negative.

HNC- may also be used for binary classification and can be solved similarly, via (5) for a tradeoff  $\lambda = \frac{\mu}{\mu+2}$ , as a minimum cut problem on the associated graph,  $G_{st}^-(\lambda)$ , illustrated in Figure 1b. The only difference between  $G_{st}^+(\lambda)$  and  $G_{st}^-(\lambda)$  is that, in the latter, each  $i \in V \setminus (L^+ \cup L^-)$  is connected to  $t$ , rather than  $s$ , with capacity of  $\lambda d_i$ .

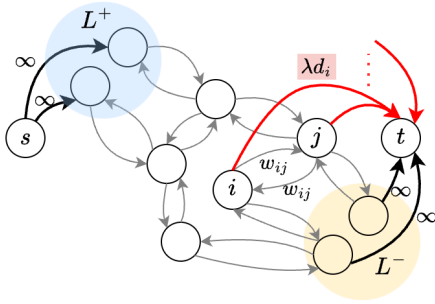
In PU learning, negative labeled samples are not given and therefore  $L^- = \emptyset$ . HNC+ and HNC- in



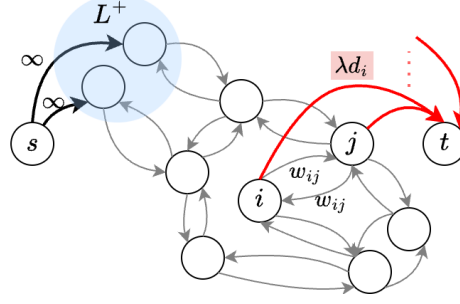
(a) Graph  $G_{st}^+(\lambda)$  for solving HNC+ with the constraint  $L^+ \subseteq S \subseteq V \setminus L^-$ .



(a) Graph  $G_{st}^+(\lambda)$  for solving HNC+ with the constraint  $L^+ \subseteq S$ , when  $L^- = \emptyset$ .



(b) Graph  $G_{st}^-(\lambda)$  for solving HNC- with the constraint  $L^+ \subseteq S \subseteq V \setminus L^-$ .



(b) Graph  $G_{st}^-(\lambda)$  for solving HNC- with the constraint  $L^+ \subseteq S$ , when  $L^- = \emptyset$ .

Figure 1: Associated graphs with HNC+ and HNC- formulations, when labeled samples from both classes are given. Nodes in the middle, outside the blue and yellow shaded areas, correspond to unlabeled samples in  $U$ .

this context are then solved, for a tradeoff  $\lambda$ , as minimum cut problems on the graphs in Figure 2a and 2b, which are  $G_{st}^+(\lambda)$  and  $G_{st}^-(\lambda)$  where  $L^- = \emptyset$ . As explained in Section 4.1.1, HNC+, with  $L^- = \emptyset$ , has a trivial solution for all  $\lambda \geq 0$ . This is also reflected in the minimum cut of  $G_{st}^+(\lambda)$  (Figure 2a) with  $L^- = \emptyset$ , which is  $(\{s\} \cup V, \{t\})$ , as  $t$  is disconnected from other nodes. Hence, in stage 1, we solve only HNC- using the graph  $G_{st}^-(\lambda)$  in Figure 2b. Once the likely-negative samples are used as seed samples in stage 2 (Section 4.1.2), HNC+ can be solved using the graph  $G_{st}^+(\lambda)$  in Figure 1a.

#### 4.2.2 Solving HNC+ and HNC- for All Tradeoff Values with a Parametric Minimum Cut Procedure

Graphs  $G_{st}^+(\lambda)$  and  $G_{st}^-(\lambda)$  are *parametric flow networks* in that the capacities of source-adjacent and sink-adjacent arcs ( $(s, i)$  and  $(i, t)$  for  $i \in V$ ) are monotone non-increasing and non-decreasing with the parameter value ( $\lambda$ ), or vice versa. For instance,  $G_{st}^+(\lambda)$  in Figure 1a, has source-adjacent capacities that can only increase with  $\lambda$ , and sink-adjacent capacities that are fixed. The minimum cuts in a parametric flow

Figure 2: Graphs on which we solve HNC+ and HNC- as minimum cut problems, in PU learning where negative labeled samples are not provided.

network are solved for all values of the parameter in the complexity of a single minimum cut procedure using the parametric cut (flow) algorithm, (Gallo et al., 1989; Hochbaum, 1998; Hochbaum, 2008). The first is based on the push-relabel algorithm, and the latter two on the HPF (pseudoflow) algorithm.

For our method, 2-HNC, HNC- with no negative seed for  $\bar{S}$  ( $L^- = \emptyset$ ) and HNC+ with the seed set  $L^- = L^N$  for  $\bar{S}$  are solved for *all* nonnegative tradeoff  $\lambda$  in stage 1 and 2, respectively, with a parametric cut procedure.

In stage 1, HNC- with  $L^- = \emptyset$  is solved on the parametric graph  $G_{st}^-(\lambda)$  in Figure 2b. As explained in Section 4.1.1, the result is a sequence of minimum cuts, or data partitions, for increasing values of  $\mu$  (and also  $\lambda$ ), with the *nestedness* property that motivates how we select likely-negative samples.

**Nested Cut Property.** (Gallo et al., 1989; Hochbaum, 1998; Hochbaum, 2008): *Given a parametric flow graph  $G(\lambda)$ , where, as the parameter  $\lambda$  increases, the capacities of the source-adjacent, sink-adjacent and other arcs are non-increasing, non-decreasing and constants, respectively, and a sequence of values  $\lambda_1 < \lambda_2 \dots < \lambda_q$ , then the corresponding minimum cut partitions,  $(S_1^*, S_1^*), (S_2^*, S_2^*), \dots, (S_q^*, S_q^*)$ ,*

satisfy  $\bar{S}_1^* \subseteq \bar{S}_2^* \subseteq \dots \subseteq \bar{S}_q^*$ .

Since the parametric graph  $G_{st}^-(\lambda)$ , in Figure 2b, is a parametric flow graph, it follows that the nested cut property applies. Let the sequence of partitions according to the parametric minimum cut of  $G_{st}^-(\lambda)$  for increasing  $\lambda$ ,  $\lambda_1 < \lambda_2 \dots < \lambda_q$ , be  $(S_1^*, \bar{S}_1^*)$ ,  $(S_2^*, \bar{S}_2^*)$ ,  $\dots$ ,  $(S_q^*, \bar{S}_q^*)$ . It follows that  $\bar{S}_1^* \subseteq \bar{S}_2^* \subseteq \dots \subseteq \bar{S}_q^*$ . This nested data partitions sequence, that is the output of stage 1, is then used in stage 2 (Section 4.1.2) to find the likely-negative set,  $L^N$ . An example of the nested sequence is shown in Figure 3.

At the end of stage 2, we obtain the predictions of unlabeled samples by selecting one partition, from all partitions that are generated in stage 1 and 2, whose positive fraction is closest to the prior  $\pi$ .

A general drawback of using minimum cut in very dense graphs is that the solution tends to not favor “balanced” partitions. In a balanced partition, there is a constant fraction  $f < 1$  of nodes on one side, and the number of edges between the two sets in the partition is  $fn(1-f)n$ , which is  $O(n^2)$ . In that case, even if many edges in the partition have small capacities, their sheer number makes the capacity of such cuts much higher than cuts that contain a small number of nodes on one side. In the graphs we study, all pairwise similarities are evaluated. Therefore, such graphs are complete and dense. The standard approach to obtaining meaningful cut partitions is to apply **graph sparsification**. There are many approaches for graph sparsification in the context of semi-supervised learning, as studied by (de Sousa et al., 2013). Among the approaches evaluated therein, we select the method that was shown to give the best performance, which is the  $k$ -nearest neighbor (kNN) sparsification (Blum and Chawla, 2001) where samples  $i$  and  $j$  are connected only if  $i$  is among the  $k$  nearest neighbors of  $j$ , or vice versa. This results in a graph representation  $G = (V, E)$  where  $E$  is the set of similar samples according to the kNN sparsification.

## 5 IMPLEMENTATION OF 2-HNC

This section includes the specification of several implementation details. First, we give a brief description of the parametric minimum cut solver used in this work. Second, we describe the choice of  $k$  in the  $k$ -nearest neighbor graph sparsification method, as mentioned in the previous section. Finally, we specify the pairwise similarity measure between pairs of samples.

### 5.1 Parametric Minimum Cut Solver

Solving HNC, via (4) and (5), for all nonnegative tradeoff  $\lambda$  as a parametric minimum cut problem can be done using the pseudoflow algorithm, given by (Hochbaum, 2008) as a *fully* parametric minimum cut solver that identifies all tradeoff values where optimal partitions change as the tradeoff increases. In this work, we use an implementation<sup>1</sup> of the pseudoflow algorithm that is a *simple* parametric minimum cut solver. It takes as input the list of values of  $\lambda$  for which we solve for the minimum cut of  $G_{st}^+(\lambda)$  and  $G_{st}^-(\lambda)$ . The  $\lambda$  values we use are  $\{0, 0.001, 0.002, \dots, 0.500\}$ . The simple parametric minimum cut solver finds the minimum cuts for all the listed  $\lambda$  values, efficiently, in the complexity of a single minimum cut procedure.

### 5.2 Graph Sparsification

As described in Section 4.2.2, we apply the kNN sparsification to  $G_{st}^+(\lambda)$  and  $G_{st}^-(\lambda)$  on which we solve the parametric minimum cut problem. For each data, we use multiple values of  $k$  and find the partitions for all of them prior to selecting one for the prediction. For data of size less than 10000, we use  $k \in \{5, 10, 15, 20, 25\}$ . For larger data, we use  $k \in \{5, 10\}$ .

The procedure to select a data partition from those generated by all  $k$ 's is as follows: For each  $k$ , we find the parametric minimum cut on the kNN-sparsified graph and select the partition whose positive fraction is closest to  $\pi$  as the *candidate* partition. Among the candidate partitions from all  $k$ , we choose the one with the largest  $k$  that has its positive fraction within 2% from  $\pi$ . Larger  $k$  is preferred since it maintains more pairwise information. If no candidate partition has positive fraction within 2% from  $\pi$ , we choose the one with the fraction closest to  $\pi$ .

Here, only smaller values of  $k$  are evaluated on large data. This is because, as discussed in Section 4.2.2, large datasets, with dense graph representation, often have highly unbalanced cuts. These large datasets benefit from a higher degree of sparsification. Hence, smaller  $k$ 's are applied.

### 5.3 Pairwise Similarity Computation

Given  $H$ -dimensional vector representations of samples  $i$  and  $j$ ,  $x_i, x_j \in \mathbb{R}^H$ , we compute their distance  $d_{ij}$  as a Euclidean distance between  $x_i$  and  $x_j$ . The pairwise similarity  $w_{ij}$  is then computed using the

<sup>1</sup><https://riot.ieor.berkeley.edu/Applications/Pseudoflow/parametric.html>



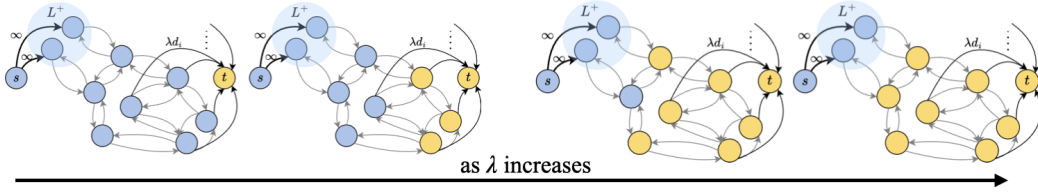


Figure 3: An example of a nested sequence of data partitions as a result of solving the parametric minimum cut problem in stage 1 of 2-HNC, illustrated on the graph  $G_{st}^-(\lambda)$ . The sets of nodes in blue and yellow are the sets of positive and negative predictions, respectively, for increasing tradeoff values  $\lambda$ .

Gaussian kernel, which is commonly used in methods that rely on pairwise similarities (Jebara et al., 2009; de Sousa et al., 2013; Baumann et al., 2019), as  $w_{ij} = \exp(-d_{ij}^2/2\sigma^2)$ . We use  $\sigma = 0.75$  for data with less than 10000 samples. For larger datasets, we use  $\sigma = 0.25$ . Again, large datasets require a higher degree of graph sparsification. Hence, a smaller  $\sigma$  is applied so that similarities of distant pairs are brought closer to zero, for the same effect as the sparsification technique discussed in Section 4.2.2 and 5.2.

In addition to the standard Euclidean distance, we also use a weighted Euclidean distance as an alternative:  $d_{ij} = \sqrt{\sum_{h=1}^H \rho_h (x_{ih} - x_{jh})^2}$  where  $\rho = [\rho_1, \dots, \rho_H]$  is the weight for the feature vector of size  $H$ .  $\rho$  is scaled so that  $\sum_{h=1}^H \rho_h = H$ . We use the feature importance from a random forest-based PU learning method (Wilton et al., 2022) as the weight  $\rho$ . Features with high importance contribute to high impurity reduction at tree node splits in the random forest.

We refer to 2-HNC with the unweighted Euclidean distance as 2-HNC(EU) and the variant with feature importance as 2-HNC(FI).

## 6 TIME COMPLEXITY ANALYSIS

Let  $N$  denote the data size, that is,  $N = |L^+| + |U|$ . Scikit-Learn implementation using the k-d tree data structure for kNN sparsification and distance computation runs in  $O(N \log N)$  (Pedregosa et al., 2011). The similarity weights computation takes  $O(N)$  time since there are  $O(N)$  pairs remain after sparsification.

The pseudoflow algorithm, known as HPF or Hochbaum’s PseudoFlow, solves the parametric minimum cut problem in the complexity of a single minimum cut procedure (Hochbaum, 2008). The complexity of HPF on a graph with  $n$  nodes and  $m$  arcs, denoted by  $T(n, m)$ , depends on the implementation. For instance, (Hochbaum and Orlin, 2013) provides a version of HPF that runs in  $O(mn \log(\frac{n^2}{m}))$ . Since the number of nodes in the graphs of both stages are

at most  $N$ . The numbers of arcs are at least  $2kN$  and at most  $4kN$  due to the kNN sparsification. Hence, solving HNC in both stages runs in  $O(N^2 \log N)$ . This runtime dominates other steps. Therefore, the time complexity of 2-HNC is  $O(N^2 \log N)$ .

## 7 EXPERIMENTS

We evaluate 2-HNC with benchmark methods on real data. The test for the methods’ robustness against the misspecification of the prior  $\pi$  is also included.

### 7.1 Datasets

Datasets are listed in Table 1, with the number of all samples, labeled and unlabeled samples, the number of features and the fraction of positive samples ( $\pi$ ) of each data. All datasets are from the UCI ML Repository (Kelly et al., ), except for *CIFAR10* from Keras (Chollet et al., 2015), and *20News* and *MNIST* from Scikit-learn (Pedregosa et al., 2011). Samples in each dataset are assigned labels (positive vs negative) as follow: *Vote*: {Democrat} vs {Republican}, *Obesity*: {Obesity Type I, II and III} vs {Insufficient, Normal, Overweight}, *Mushroom*: {Edible} vs {Poisonous}, *20News*: {alt., comp., misc., rec.} vs {sci., soc., talk.}, *Letter*: {A-M} vs {N-Z}, *CIFAR10*: {bird, cat, deer, dog, frog, horse} vs {airplane, automobile, ship, truck}, *MNIST*: {1,3,5,7,9} vs {0,2,4,6,8}. Following (Kiryo et al., 2017; Wilton et al., 2022), we use a pre-trained GloVe word embedding (Pennington et al., 2014) to map each document in *20News* to a 300-dimension vector.

For each dataset, except for *Vote*, we randomly sample 10% of the positive samples (with the number rounded to the nearest hundred) as the positive labeled set  $L^+$ . All the remaining samples are used as unlabeled samples, or the set  $U$ . For *Vote*, as a small dataset, we randomly select 40 samples as the positive labeled set. We run the experiments 5 times, with different sampling of labeled samples.

As described in the introduction, 2-HNC is a



Table 1: Datasets: 10% of positive samples are randomly selected as labeled samples. The unlabeled set consists of negative samples and the remaining 90% of positive samples.  $\pi$  is the fraction of positive samples in each dataset.

Name	# Samples	# Labeled	# Unlabeled	# Feature	$\pi$
Vote	435	40	395	16	0.61
Obesity	2111	100	2011	19	0.46
Mushroom	8124	400	7724	112	0.52
20News	18846	1000	17846	300	0.56
Letter	20000	1000	19000	16	0.50
CIFAR10	60000	3600	56400	3072	0.60
MNIST	70000	3500	66500	784	0.51

Table 2: Classification accuracy (%) average (and standard error) across 5 runs of both variants of 2-HNC and benchmark methods. Number in bold for each data is the highest accuracy.

Data	uPU	nnPU	PU ET	2-HNC(EU)	2-HNC(FI)
Vote	51.60 (1.42)	84.08 (6.80)	92.51 (2.93)	90.33 (0.46)	<b>94.99</b> (1.22)
Obesity	85.92 (7.23)	91.92 (1.23)	92.74 (0.66)	89.64 (1.78)	<b>96.54</b> (1.18)
Mushroom	87.92 (4.54)	98.94 (0.64)	99.35 (0.66)	99.67 (0.18)	<b>99.85</b> (0.09)
20News	58.83 (1.23)	70.90 (0.73)	84.89 (0.41)	76.63 (0.54)	<b>86.03</b> (1.46)
Letter	81.33 (1.97)	87.50 (0.76)	86.21 (0.55)	<b>88.88</b> (1.51)	87.92 (2.09)
CIFAR10	43.00 (0.01)	<b>87.98</b> (0.65)	81.55 (0.11)	78.46 (0.85)	77.81 (0.41)
MNIST	72.65 (1.85)	94.25 (0.91)	95.30 (0.12)	<b>96.44</b> (0.07)	94.87 (1.17)

Table 3: F1 score (%) average (and standard error) across 5 runs of both variants of 2-HNC and benchmark methods. Number in bold for each data is the highest F1 score.

Data	uPU	nnPU	PU ET	2-HNC(EU)	2-HNC(FI)
Vote	26.26 (11.81)	88.85 (4.67)	93.20 (3.22)	91.54 (0.40)	<b>95.63</b> (1.00)
Obesity	74.59 (13.70)	86.26 (7.75)	91.04 (0.88)	87.94 (2.24)	<b>94.63</b> (2.80)
Mushroom	85.19 (6.93)	98.91 (0.66)	99.34 (0.69)	99.67 (0.19)	<b>99.85</b> (0.09)
20News	20.34 (4.38)	70.97 (3.39)	85.81 (0.23)	78.55 (0.61)	<b>87.11</b> (1.28)
Letter	74.97 (3.26)	85.72 (1.82)	83.49 (0.73)	<b>88.34</b> (1.64)	87.42 (2.14)
CIFAR10	21.02 (10.11)	<b>89.44</b> (1.19)	83.99 (0.11)	81.05 (0.77)	80.92 (0.49)
MNIST	30.75 (7.96)	93.27 (2.13)	95.11 (0.16)	<b>96.27</b> (0.07)	94.63 (1.09)

transductive method that predicts specifically for samples in the given unlabeled set. Hence, we evaluate the models on their predictions of unlabeled samples in  $U$  that the models are trained on. The metrics that we use are the classification accuracy and F1 score, averaged over 5 experiments on each dataset.

## 7.2 Benchmark Methods

2-HNC is compared against the following benchmarks: uPU (Du Plessis et al., 2014; Du Plessis et al., 2015), nnPU (Kiryo et al., 2017) and PU ET (Wilton et al., 2022), as discussed in Section 2.

The choices of neural networks of uPU and nnPU are similar to (Kiryo et al., 2017): a 6-layer MLP with Softsign activation function for *20News*, a 13-layer CNN with a ReLU final layer for *CIFAR10* and *MNIST*, and a 6-layer MLP with ReLU for other datasets. For PU ET, we use the default hyperparameters as suggested in (Wilton et al., 2022). We use the

available implementations<sup>2</sup> of these methods.

As explained in Section 5.3, we use two variants of 2-HNC: 2-HNC(EU) and 2-HNC(FI) that use unweighted and feature importance-weighted Euclidean distance, respectively.

## 7.3 Results

The accuracy and F1 score of both variants of 2-HNC and benchmark models are reported in Table 2 and 3. 2-HNC(FI) yields the best result on tabular data (Vote, Obesity, Mushroom) and the text data (20News). 2-HNC(EU) outperforms all methods on Letter and MNIST. However, nnPU has the best performance for CIFAR10. The relative performance of the models are similar for both accuracy and F1 score.

We also test the statistical significance of the out-performance of 2-HNC over other methods. The best

<sup>2</sup>uPU, nnPU: <https://github.com/kiryor/nnPUlearning>, PU ET: <https://github.com/jonathanwilton/PUExtraTrees>

Table 4: P-values for the t-test on the performances of 2-HNC and the best benchmarks. (\*) denotes p-values where HNC outperforms with high statistical significance ( $\alpha = 0.05$ ). P-values on CIFAR10 are not shown as 2-HNC does not give the highest performance on CIFAR10.

Data	Best 2-HNC variant	Best benchmark	P-values: accuracy	P-values: f1 score
Vote	2-HNC(FI)	PU ET	0.0903	0.0875
Obesity	2-HNC(FI)	PU ET	0.0017*	0.0119*
Mushroom	2-HNC(FI)	PU ET	0.0312*	0.0302*
20News	2-HNC(FI)	PU ET	0.1025	0.0465*
Letter	2-HNC(EU)	nnPU	0.0251*	0.0269*
MNIST	2-HNC(EU)	PU ET	1.2584e-5*	8.4961e-5*

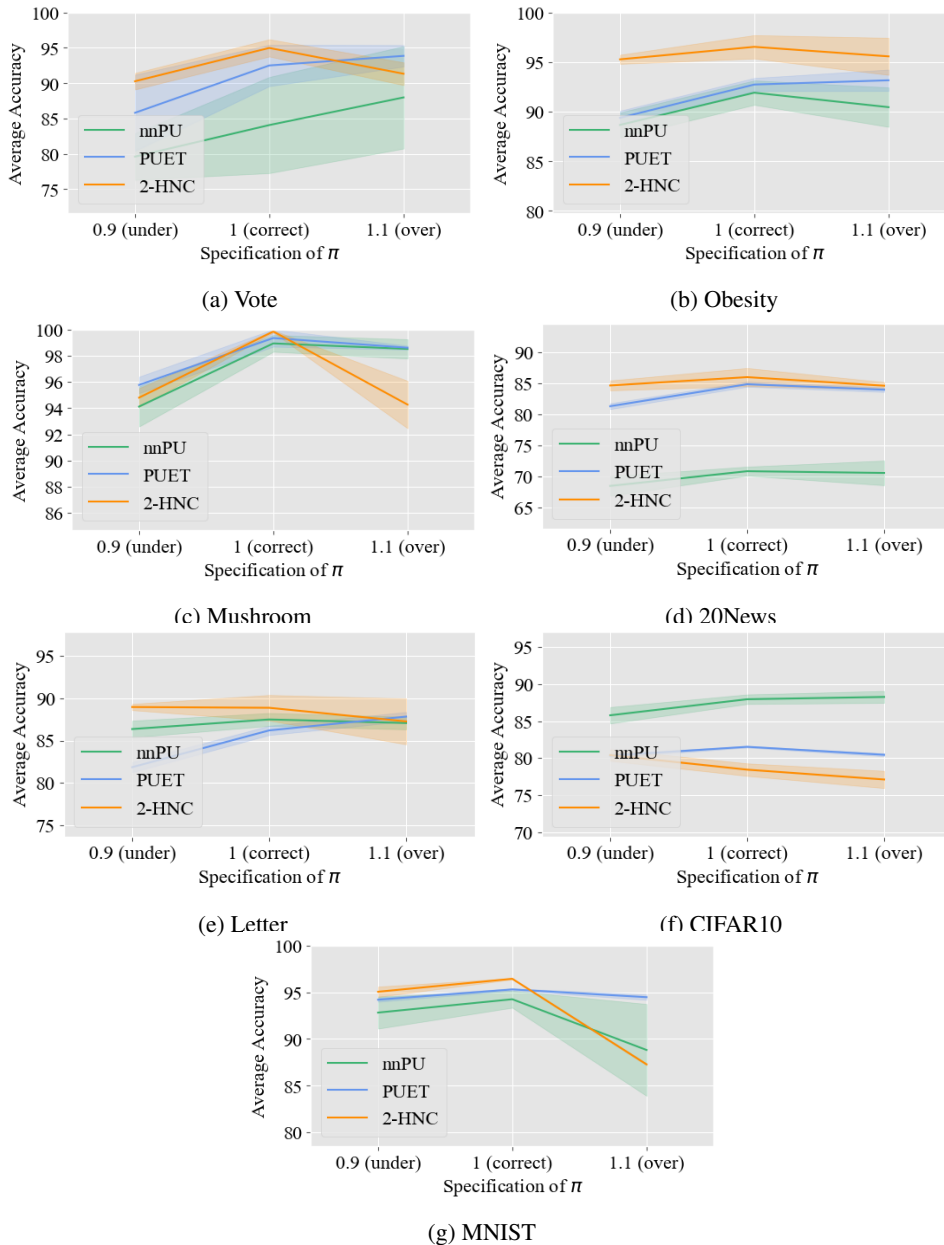


Figure 4: Average accuracy (with the shaded regions as error bars) of each PU learning method when the prior of the positive fraction  $\pi$  is misspecified, compared to the results when the correct  $\pi$  is provided.

variant between 2-HNC(EU) and 2-HNC(FI), is compared to the best among the three benchmarks for each dataset. P-values of the paired t-tests are reported in Table 4. 2-HNC outperforms other methods with high statistical significance (significance level of 0.05) on most data for both metrics. The exceptions are accuracy and F1 score on Vote, where p-values are 0.0903 and 0.0875, and accuracy on 20News, with p-values of 0.1025. Despite that, these p-values still demonstrate the statistical significance level of around 0.1.

## 7.4 Sensitivity Analysis

We evaluate the models' sensitivity to the misspecification of the prior of positive fraction  $\pi$ . For each dataset, with true positive fraction  $\pi_0$ , we over-specify and under-specify the prior using  $\pi = 1.1\pi_0$  and  $\pi = 0.9\pi_0$ , respectively. Results are shown in Figure 4 with *uPU* omitted for clarity of the plots as *uPU* achieves lowest accuracy in all cases. In this analysis, we use the better variant of 2-HNC for each data, according to the result from the previous subsection.

As shown in Figure 4, 2-HNC exhibits higher robustness than other methods when  $\pi$  is under-specified, for all datasets except Mushroom and CIFAR10. For CIFAR10, 2-HNC yields similar performance as PUET, where the two methods have the accuracy of  $80.43 \pm 0.77$  and  $80.35 \pm 0.25$ , respectively. Moreover, the rate of accuracy decline for 2-HNC is lower than that of nnPU and PUET on many datasets such as Vote, Obesity, 20News and Letter.

When  $\pi$  is over-specified, 2-HNC is not as robust as other methods. On data such as Obesity and 20News, the improvements become smaller.

## 8 CONCLUSIONS

Our PU learning method called 2-HNC is a two-stage variant of a network flow-based Hochbaum's Normalized Cut that was previously used in binary classification with labeled samples of both classes. The output of 2-HNC is the partition of samples into the positive and negative prediction sets.

Both stages of 2-HNC generate nested sequences of data partitions for varying tradeoffs between the inter-similarity of the positive and negative prediction sets, and the intra-similarity within sets, solved as parametric minimum cut problems. Stage 1 puts more weights on the intra-similarity of the negative prediction set, whereas stage 2 emphasizes on the positive one. Stage 2 utilizes the set of likely-negative unlabeled samples, determined by the order in which unlabeled samples enter the negative prediction set in

the nested sequence of stage 1. A partition whose positive fraction approximates the prior  $\pi$  most closely is selected as the predictions for unlabeled samples.

Experiments on real datasets demonstrate that 2-HNC outperforms benchmark methods in terms of accuracy and F1 scores, as well as better robustness to the under-specification of the prior  $\pi$ .

Future research directions include methods that learn accurate pairwise similarities measure based on the PU data as the current similarity measure is unsupervised. Another potential direction is the selection of likely-negative samples from the unlabeled set. While an approach based on the nested partition sequence is employed in this work, other techniques are also worth further investigation.

## ACKNOWLEDGEMENTS

This research was supported in part by the AI Institute NSF Award 2112533. We would also like to thank William Pham for his help with the literature review.

## REFERENCES

- Asín Achá, R., Hochbaum, D. S., and Spaen, Q. (2020). Hnccorr: combinatorial optimization for neuron identification. *Annals of Operations Research*, 289:5–32.
- Baumann, P., Hochbaum, D. S., and Yang, Y. T. (2019). A comparative study of the leading machine learning techniques and two new optimization algorithms. *European journal of operational research*, 272(3):1041–1057.
- Bekker, J. and Davis, J. (2020). Learning from positive and unlabeled data: A survey. *Machine Learning*, 109(4):719–760.
- Blum, A. and Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts.
- Carnevali, J. C., Rossi, R. G., Milios, E., and de Andrade Lopes, A. (2021). A graph-based approach for positive and unlabeled learning. *Information Sciences*, 580:655–672.
- Chollet, F. et al. (2015). Keras. <https://keras.io>.
- de Sousa, C. A. R., Rezende, S. O., and Batista, G. E. (2013). Influence of graph construction on semi-supervised learning. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13*, pages 160–175. Springer.
- Du Plessis, M., Niu, G., and Sugiyama, M. (2015). Convex formulation for learning from positive and unlabeled data. In *International conference on machine learning*, pages 1386–1394. PMLR.

- Du Plessis, M. C., Niu, G., and Sugiyama, M. (2014). Analysis of learning from positive and unlabeled data. *Advances in neural information processing systems*, 27.
- Gallo, G., Grigoriadis, M. D., and Tarjan, R. E. (1989). A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing*, 18(1):30–55.
- Hochbaum, D. S. (1998). The pseudoflow algorithm and the pseudoflow-based simplex for the maximum flow problem. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 325–337. Springer.
- Hochbaum, D. S. (2002). 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.
- Hochbaum, D. S. (2008). The pseudoflow algorithm: A new algorithm for the maximum-flow problem. *Operations research*, 56(4):992–1009.
- Hochbaum, D. S. (2010). Polynomial time algorithms for ratio regions and a variant of normalized cut. *IEEE transactions on pattern analysis and machine intelligence*, 32(5):889–898.
- Hochbaum, D. S. (2021). Applications and efficient algorithms for integer programming problems on monotone constraints. *Networks*, 77(1):21–49.
- Hochbaum, D. S. and Orlin, J. B. (2013). Simplifications and speedups of the pseudoflow algorithm. *Networks*, 61(1):40–57.
- Jebara, T., Wang, J., and Chang, S.-F. (2009). Graph construction and b-matching for semi-supervised learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 441–448.
- Kelly, M., Longjohn, R., and Nottingham, K.
- Khan, S. S. and Madden, M. G. (2014). One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(3):345–374.
- Kiryó, R., Niu, G., Du Plessis, M. C., and Sugiyama, M. (2017). Positive-unlabeled learning with non-negative risk estimator. *Advances in neural information processing systems*, 30.
- Lee, W. S. and Liu, B. (2003). Learning with positive and unlabeled examples using weighted logistic regression. In *ICML*, volume 3, pages 448–455.
- Li, H., Chen, Z., Liu, B., Wei, X., and Shao, J. (2014). Spotting fake reviews via collective positive-unlabeled learning. In *2014 IEEE international conference on data mining*, pages 899–904. IEEE.
- Li, W., Guo, Q., and Elkan, C. (2010). A positive and unlabeled learning algorithm for one-class classification of remote-sensing data. *IEEE transactions on geoscience and remote sensing*, 49(2):717–725.
- Li, X. and Liu, B. (2003). Learning to classify texts using positive and unlabeled data. In *IJCAI*, volume 3, pages 587–592. Citeseer.
- Liu, B., Lee, W. S., Yu, P. S., and Li, X. (2002). Partially supervised classification of text documents. In *ICML*, volume 2, pages 387–394. Sydney, NSW.
- Lu, F. and Bai, Q. (2010). Semi-supervised text categorization with only a few positive and unlabeled documents. In *2010 3rd International conference on biomedical engineering and informatics*, volume 7, pages 3075–3079. IEEE.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.
- Ren, Y., Ji, D., and Zhang, H. (2014). Positive unlabeled learning for deceptive reviews detection. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 488–498.
- Spaen, Q., Asín-Achá, R., Chettih, S. N., Minderer, M., Harvey, C., and Hochbaum, D. S. (2019). Hnccorr: A novel combinatorial approach for cell identification in calcium-imaging movies. *eneuro*, 6(2).
- Wilton, J., Koay, A., Ko, R., Xu, M., and Ye, N. (2022). Positive-unlabeled learning using random forests via recursive greedy risk minimization. *Advances in Neural Information Processing Systems*, 35:24060–24071.
- Yang, P., Li, X., Chua, H.-N., Kwoh, C.-K., and Ng, S.-K. (2014a). Ensemble positive unlabeled learning for disease gene identification. *PloS one*, 9(5):e97079.
- Yang, P., Li, X.-L., Mei, J.-P., Kwoh, C.-K., and Ng, S.-K. (2012). Positive-unlabeled learning for disease gene identification. *Bioinformatics*, 28(20):2640–2647.
- Yang, Y. T., Fishbain, B., Hochbaum, D. S., Norman, E. B., and Swanberg, E. (2014b). The supervised normalized cut method for detecting, classifying, and identifying special nuclear materials. *INFORMS Journal on Computing*, 26(1):45–58.
- Yi, J., Hsieh, C.-J., Varshney, K. R., Zhang, L., and Li, Y. (2017). Scalable demand-aware recommendation. *Advances in neural information processing systems*, 30.
- Zhang, C., Ren, D., Liu, T., Yang, J., and Gong, C. (2019). Positive and unlabeled learning with label disambiguation. In *IJCAI*, pages 4250–4256.