Flexible Hochbaum's Normalized Cut

A graph-based learning method that relies on pairwise similarities and its capability to handle label noise

Tor Nitayanont

UC Berkeley, IEOR department

January 2022

Outline

Learning methods that rely on pairwise similarities

Hochbaum's Normalized Cut (HNC)

Flexible Hochbaum's Normalized Cut (Flexible HNC)

Experiment

Learning methods that rely on pairwise similarities

- Several common machine learning methods are based on pairwise similarities (or dissimilarities) between samples such as the *k*-nearest neighbor classifier or the support vector machine.
- Hochbaum's normalized cut or HNC is another learning method that builds a prediction model based on similarities between samples.
- HNC is a graph-based approach where pairwise similarities are used as weights on arcs that connect vertices, which represent samples. We will discuss this later.

Graph Cut

- Given a directed graph $G = (V, A)$ with arc weights $W = \{w_a \text{ for } a \in A\}$
- A **cut** is a partition of vertices into two disjoint sets S and T such that $S \cup T = V$ and $S \cap T = \emptyset$.
- The **capacity** of the cut that partitions V into S and T is equal to $C(S, T) = \sum_{i \in S, j \in T} w_{ij}$
- In the **minimum cut** problem, we seek for a cut with the smallest capacity.
- It has been used widely as a clustering method, mostly in image segmentation.
- **Normalized cut** is another well-known cut problem. It was proposed in [1].

Hochbaum's Normalized Cut (HNC)

- **Hochbaum's normalized cut** was introduced in [2] as an alternative to the normalized cut problem, which is NP-hard.
- The goal of the **HNC** problem is to find a partition (S, T) that minimizes:

$$\underset{\emptyset \subset S \subset V}{minimize} \frac{C(S,T)}{\sum_{i \in S} d_i}$$

  where $d_i = \sum_{j \in V} w_{ij}$
- Note that there could be two designated seed sets V_S and V_T where vertices in V_S are required to be in S and vertices in V_T are required to not be in S (required to be in T).
- The problem of minimizing the ratio above is equivalent to finding the smallest $\lambda > 0$ such that the optimal objective function of the following linearized problem (HNC(λ)) is non-positive.

$$\underset{\emptyset \subset S \subset V}{minimize}\ C(S,T) - \lambda \cdot \sum_{i \in S} d_i$$

HNC

- It has been shown that solving a minimum cut on the following graph gives the solution to $HNC(\lambda)$ [2], [3].

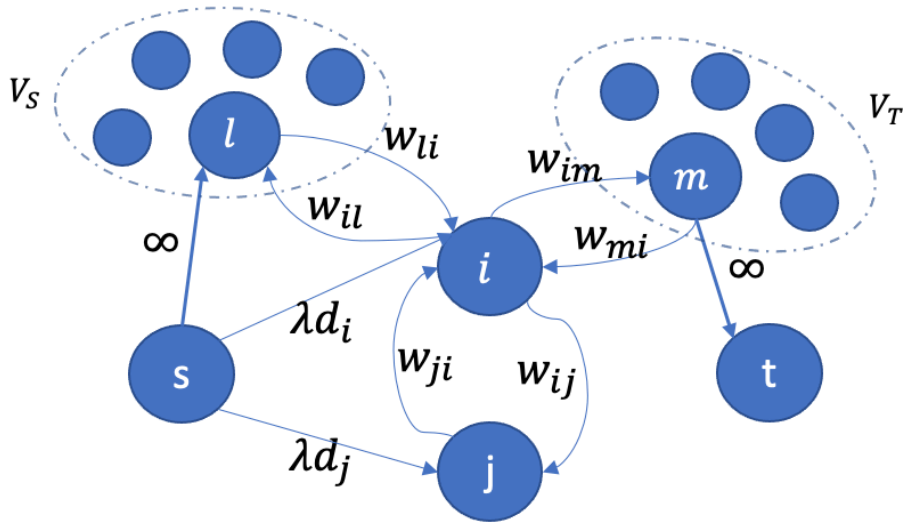


Figure 1: graph on which we solve the minimum cut

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

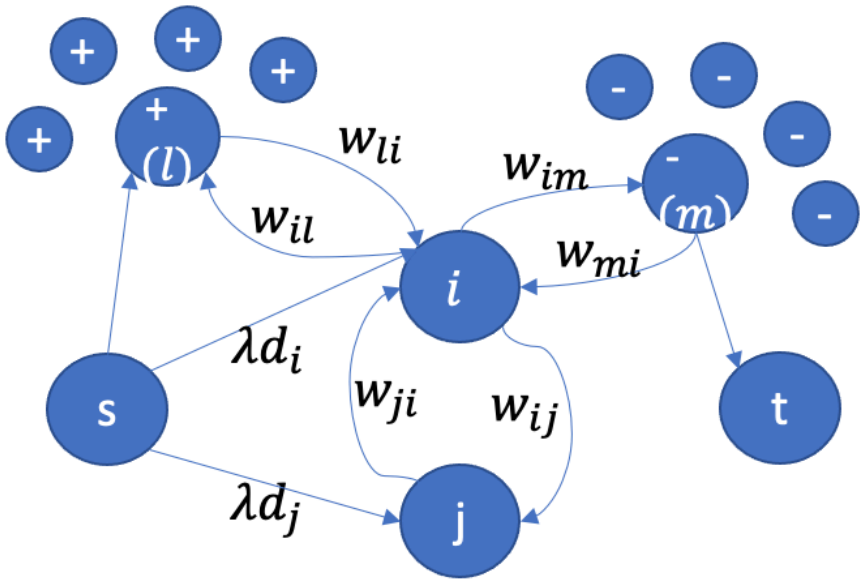
HNC

- In the context of (semi-)supervised learning, we consider a binary classification problem where each sample belongs to either positive class or negative class.
- Given a set of labeled samples $L = L^+ \cup L^-$ where
  - L^+ is the set of samples with positive labels
  - L^- is the set of samples with negative labels

  and the set of unlabeled samples U
- L^+ is equivalent to the seed set V_S. L^- is equivalent to the seed set V_T.
- Samples in the unlabeled set U are not constrained by the infinite arcs to be in either the source set S or the sink set T.
- Weights w between pairs of samples represents their similarities. For instance, we can use the Gaussian similarity weight $w_{ij} = exp(-\frac{|x_i - x_j|^2}{2\epsilon^2})$ to represent the similarity between sample i and sample j.

Flexible HNC

- We want our graph to take into account the possibility that some of the given labels of the labeled samples might be inaccurate.
- The infinite weights on the arcs that connect *s* to the positive samples and connect the negative labels to *t* are replaced with some finite weights. This is to allow the labels of the labeled samples to change.



Flexible HNC

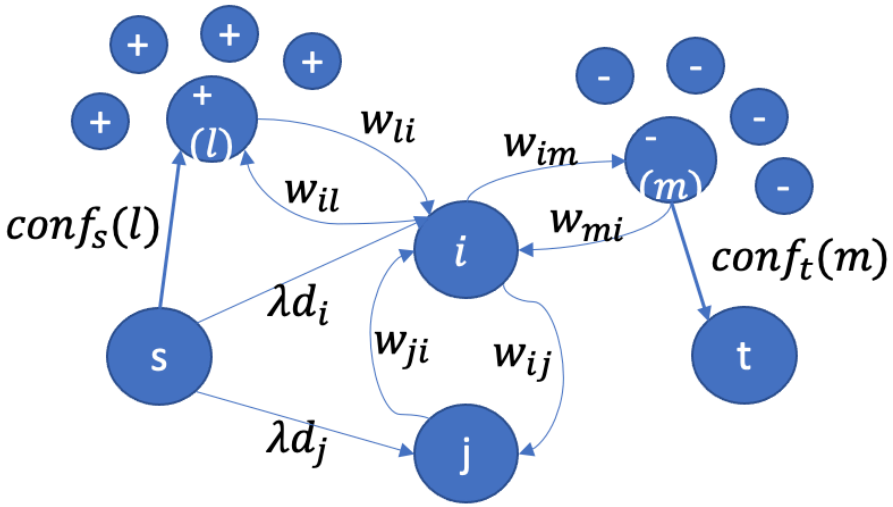


Figure 2: graph on which we solve the minimum cut to obtain the solution of Flexible HNC

- $conf_s(l)$ is our "confidence" about the positive label of $l \in L^+$.
- $conf_t(m)$ is our "confidence" about the negative label of $m \in L^-$.

Confidence functions

In this work, we experiment with three confidence functions:

- **constant confidence weight**: $conf_s(l) = \theta$ and $conf_t(m) = \theta$ for $s \in L^+$ and $m \in L^-$
- **local mean-based confidence weight**:
  - For sample l with positive label, we find its k nearest positive neighbors $L_k^+(l)$ and k nearest negative neighbors $L_k^-(l)$.
  - We compute the distance from l to the average feature vectors of $L_k^+(l)$ and $L_k^-(l)$; denote them by $d_k^+(l)$ and $d_k^-(l)$.
  - $conf_s(l) = \theta \cdot \frac{exp(-\frac{d_k^+(l)^2}{2\epsilon^2})}{exp(-\frac{d_k^+(l)^2}{2\epsilon^2}) + exp(-\frac{d_k^-(l)^2}{2\epsilon^2})}$ and $conf_t(m) = \theta \cdot \frac{exp(-\frac{d_k^-(m)^2}{2\epsilon^2})}{exp(-\frac{d_k^+(m)^2}{2\epsilon^2}) + exp(-\frac{d_k^-(m)^2}{2\epsilon^2})}$
- **k-nearest neighbor confidence weight**
  - $conf_s(l) = \theta \cdot \frac{\text{number of positive samples among } k \text{ nearest neighbors}}{k}$
  - $conf_t(m) = \theta \cdot \frac{\text{number of negative samples among } k \text{ nearest neighbors}}{k}$

Graph sparsification

- In this work, we also apply a graph sparsification method to our HNC graph. Instead of connecting each sample to all other samples, we only connect it to its *k* nearest neighbors.
- This idea was first introduced in [4] but it has never been applied to HNC.
- There are also other sparsification methods such as the sparse computation framework introduced in [5].

Experiment

We test HNC and Flexible HNC models along with three other similarity-based models

1. HNC on a fully connected graph (**HNC**)
2. HNC with kNN sparsification (**HNC sp**)
3. Flexible HNC with constant confidence weight (**FHNC-const**)
4. Flexible HNC with local mean confidence weight (**FHNC-LM**)
5. Flexible HNC with k-nearest neighbor confidence weight (**FHNC-kNN**)
6. k-nearest neighbor classifier (**kNNC**)
7. k-nearest neighbor classifier with repeated edited nearest neighbors method (**kNNC-RENN**)
8. k-nearest neighbor classifier with AllkNN method (**kNNC-AllkNN**)

- Methods applied to kNN classifier in 7) and 8) attempt to detect noisy labels and remove them before the classification step.
- Note that graph sparsification is also applied to 3), 4) and 5).

Experiment

Noise detection techniques that are applied to the k nearest neighbor classifier

- RENN: Repeated Edited Nearest Neighbors
  - Each sample is flagged as noisy and is removed if the label of the majority of its k nearest neighbors (use $k = 5$) is different from its labels.
- All kNN
  - Similar to RENN, but after each iteration, the number of neighbors that we consider is increased.
  - The procedure is repeated until no noisy samples are detected.

Both techniques were proposed in [6] and have been used and referred to consistently in numerous works related to noise detection such as in [7, 8, 9, 10].

Experiment Setup

- We test 8 models on 13 datasets. For each dataset, we test the models on 5 different train-test partitions.
- In each train-test split, 60% of the samples are labeled and 40% are unlabeled.
- We perform the experiments on two scenarios.
  1. we make no changes to the given labeled samples
  2. we flip 25% of the labels of the labeled samples
- Parameters are tuned using 10-fold shuffle split cross validation.
- Evaluation metrics:
  1. accuracy of the label prediction
  2. F1 score of the label prediction
  3. noise recall: fraction of noisy labels that are detected
  4. noise precision: fraction of detected samples that are actually noisy
- Note that metrics 3 and 4 are evaluated only in the second scenario where some labels are flipped.

Parameters

1. λ: the parameter in HNC
2. θ: the weights of the confidence function on the arcs that connect the labeled samples to the source/set node
3. k: the number of neighbors used in both the sparsification of HNC and the prediciton of the k nearest neighbor classifier
4. ϵ: the parameter in the Gaussian similarity

	HNC	HNC sp	FHNC-const	FHNC-LM	FHNC-kNN	kNNC	kNNC-RENN	kNNC-AllkNN
λ	✓	✓	✓	✓	✓			
θ			✓	✓	✓			
k		✓	✓	✓	✓	✓	✓	✓
ϵ	✓	✓	✓	✓	✓	✓	✓	✓

λ	$2^{-8}, 2^{-7}, 2^{-6}, ..., 2^{-1}, 2^{0}$
θ	$2^{-3}, 2^{-2}, 2^{-1}, ...2^{2}, 2^{3}$
k	$4, 8, 12, ..., 76, 80$
ϵ	$2^{-3}, 2^{-2}, 2^{-1}, ...2^{2}, 2^{3}$

Datasets

dataset	# samples	# features	%positive labels
credit	1500	23	22.12%
credit	5000	23	22.12%
german	1000	24	30%
letter	1000	16	49.7%
letter	5000	16	49.7%
fourclass	862	2	35.61%
australian	690	14	44.49%
wdbc	569	30	37.26%
redwine	1599	11	96.06%
blood	748	4	23.8%
banknote	1372	4	44.46%
abalone	4177	10	49.82%
housing	506	13	26.09%

Results: Average accuracy (without label noise)

Average accuracy of the models (no noise)

data	HNC	HNC sp	FHNC-const	FHNC-LM	FHNC-kNN	kNNC	kNNC-RENN	kNNC-AllkNN
0	0.777667	0.799667	0.801	0.795667	0.799	0.796333	0.800667	0.798333
1	0.7894	0.8103	0.8098	0.8099	0.8099	0.8088	0.8102	0.8066
2	0.7	0.707	0.7205	0.711	0.703	0.72	0.703	0.701
3	0.7775	0.822	0.8295	0.8295	0.8265	0.8345	0.803	0.805
4	0.9096	0.9414	0.944	0.9455	0.9423	0.9414	0.9161	0.9244
5	1	0.99942	1	1	1	0.99942	0.997681	0.998261
6	0.84058	0.847826	0.855072	0.844203	0.845652	0.842029	0.846377	0.844203
7	0.922807	0.964912	0.953509	0.953509	0.952632	0.959649	0.941228	0.942105
8	0.960625	0.960313	0.956875	0.957188	0.96	0.96	0.960938	0.960313
9	0.766	0.766	0.773333	0.777333	0.774667	0.788	0.76	0.756667
10	0.999636	0.997086	0.998543	0.998543	0.998543	0.998543	0.998543	0.99745
11	0.743746	0.781807	0.786475	0.778695	0.776541	0.790425	0.777379	0.775105
12	0.837438	0.879803	0.883744	0.881773	0.862069	0.900493	0.862069	0.855172
mean	0.848077	0.867503	0.870181	0.867909	0.865446	0.872276	0.859783	0.858816

Results: Accuracy ranking (without label noise)

Accuracy ranking of the models (no noise)

data	HNC	HNC sp	FHNC-const	FHNC-LM	FHNC-kNN	kNNC	kNNC-RENN	kNNC-AllkNN
0	8	3	1	7	4	6	2	5
1	8	1	5	3.5	3.5	6	2	7
2	8	4	1	3	5.5	2	5.5	7
3	8	5	2.5	2.5	4	1	7	6
4	8	4.5	2	1	3	4.5	7	6
5	2.5	5.5	2.5	2.5	2.5	5.5	8	7
6	8	2	1	5.5	4	7	3	5.5
7	8	1	3.5	3.5	5	2	7	6
8	2	3.5	8	7	5.5	5.5	1	3.5
9	6	5	4	2	3	1	7	8
10	1	8	4	4	4	4	4	7
11	8	3	2	4	6	1	5	7
12	8	4	2	3	6	1	5	7
mean	6.42308	3.80769	2.96154	3.73077	4.30769	3.57692	4.88462	6.30769

Berkeley
UNIVERSITY OF CALIFORNIA

Results: Accuracy compared to HNC (without label noise)

Relative accuracy compared to HNC (no noise)

data	HNC	HNC sp	FHNC-const	FHNC-LM	FHNC-kNN	kNNC	kNNC-RENN	kNNC-AllkNN
0	0	2.82898	3.00043	2.31462	2.74325	2.40034	2.95757	2.65752
1	0	2.64758	2.58424	2.59691	2.59691	2.45756	2.63491	2.17887
2	0	1	2.92857	1.57143	0.428571	2.85714	0.428571	0.142857
3	0	5.72347	6.6881	6.6881	6.30225	7.33119	3.27974	3.53698
4	0	3.49604	3.78188	3.94679	3.59499	3.49604	0.7146	1.62709
5	0	-0.057971	0	0	0	-0.057971	-0.231884	-0.173913
6	0	0.862069	1.72414	0.431034	0.603448	0.172414	0.689655	0.431034
7	0	4.56274	3.327	3.327	3.23194	3.9924	1.9962	2.09125
8	0	-0.0325309	-0.390371	-0.35784	-0.0650618	-0.0650618	0.0325309	-0.0325309
9	0	1.44938e-14	0.957354	1.47955	1.13142	2.87206	-0.78329	-1.21845
10	0	-0.255102	-0.109329	-0.109329	-0.109329	-0.109329	-0.109329	-0.218659
11	0	5.11748	5.74509	4.69907	4.4094	6.27615	4.52205	4.21629
12	0	5.05882	5.52941	5.29412	2.94118	7.52941	2.94118	2.11765
mean	0	2.38089	2.75127	2.45242	2.13915	3.01172	1.46712	1.33508

Results: Average F1 score (without label noise)

Average F1 score of the models (no noise)

data	HNC	HNC sp	FHNC-const	FHNC-LM	FHNC-kNN	kNNC	kNNC-RENN	kNNC-AllkNN
0	0.413394	0.414574	0.447917	0.445219	0.43636	0.38577	0.308712	0.259385
1	0.415581	0.463847	0.483564	0.480167	0.476253	0.39215	0.380325	0.34241
2	0.479383	0.50686	0.528269	0.522145	0.492498	0.437619	0.131753	0.0852968
3	0.782248	0.814758	0.822539	0.825866	0.82751	0.832615	0.799275	0.803021
4	0.914227	0.941788	0.94397	0.945299	0.943092	0.94162	0.916862	0.924593
5	1	0.99919	0.99919	0.99919	0.99919	0.998374	0.996741	0.997558
6	0.807348	0.834729	0.83628	0.836972	0.837822	0.816791	0.818942	0.810819
7	0.885927	0.950234	0.942592	0.942592	0.93369	0.943184	0.914921	0.919918
8	0.97991	0.979741	0.97794	0.978102	0.979585	0.979422	0.98008	0.979754
9	0.471884	0.470364	0.470237	0.473766	0.484474	0.386266	0.365788	0.279819
10	0.999591	0.996763	0.998367	0.998367	0.998367	0.998367	0.998367	0.99713
11	0.774115	0.798646	0.796869	0.795008	0.79248	0.796179	0.779881	0.77686
12	0.663799	0.719752	0.743493	0.738203	0.704025	0.775288	0.665085	0.635985
mean	0.737493	0.760865	0.768556	0.767761	0.76195	0.744896	0.696672	0.677888

Results: F1 score ranking (without label noise)

F1 score ranking of the models (no noise)

data	HNC	HNC sp	FHNC-const	FHNC-LM	FHNC-kNN	kNNC	kNNC-RENN	kNNC-AllkNN
0	5	4	1	2	3	6	7	8
1	5	4	1	2	3	6	7	8
2	5	3	1	2	4	6	7	8
3	8	5	4	3	2	1	7	6
4	8	4	2	1	3	5	7	6
5	1	3.5	3.5	3.5	3.5	6	8	7
6	8	4	3	2	1	6	5	7
7	8	1	3.5	3.5	5	2	7	6
8	2	4	8	7	5	6	1	3
9	3	4	5	2	1	6	7	8
10	1	8	4	4	4	4	4	7
11	8	1	2	4	5	3	6	7
12	7	4	2	3	5	1	6	8
mean	5.30769	3.80769	3.07692	3	3.42308	4.46154	6.07692	6.84615

Results: F1 score compared to HNC (without label noise)

Relative F1 score compared to HNC (no noise)

data	HNC	HNC sp	FHNC-const	FHNC-LM	FHNC-kNN	kNNC	kNNC-RENN	kNNC-AllkNN
0	0	0.285316	8.35118	7.69836	5.55532	-6.68234	-25.3226	-37.2548
1	0	11.6141	16.3584	15.5411	14.5993	-5.63818	-8.48345	-17.6069
2	0	5.73189	10.1977	8.92029	2.7359	-8.71208	-72.5161	-82.207
3	0	4.15591	5.15061	5.57592	5.78614	6.43872	2.17661	2.65549
4	0	3.01477	3.25337	3.39882	3.15734	2.99632	0.288274	1.13392
5	0	-0.0809717	-0.0809717	-0.0809717	-0.0809717	-0.162604	-0.325867	-0.244234
6	0	3.39143	3.58354	3.66932	3.77457	1.1696	1.43603	0.429935
7	0	7.25879	6.39612	6.39612	5.39133	6.46294	3.27276	3.83684
8	0	-0.0172491	-0.201105	-0.184504	-0.0332539	-0.049827	0.0172762	-0.0159507
9	0	-0.322143	-0.348999	0.398712	2.66803	-18.1439	-22.4836	-40.7018
10	0	-0.282944	-0.122416	-0.122416	-0.122416	-0.122416	-0.122416	-0.246178
11	0	3.16888	2.93931	2.69897	2.37232	2.85026	0.744772	0.354584
12	0	8.4292	12.0057	11.2087	6.05991	16.7955	0.193596	-4.19022
mean	0	3.56515	5.19096	5.00911	3.9895	-0.215232	-9.31728	-13.3889

Results conclusion for the case where there is no label noise

- Accuracy
  - The kNN classifier without noise detection attains the highest accuracy (87.22%). The accuracies of the flexible HNC models are quite close, especially the one with constant confidence weight (87.01%).
  - In terms of the average rank, the flexible HNC with constant confidence weight has the best performance (2.96). The second and the third best are the kNN classfier (3.58) and the flexible HNC with local mean confidence weight (3.73).
- F1-score
  - All the HNC models, especially the flexible HNC (76.86%, 76.77%,76.20%), perform better than kNN classifiers (74.49%).
  - The flexible HNC models achieve higher F1 score than both the standard HNC (73.75%) and the sparsified version of the standard HNC (76.09%).
  - Constant confidence weights and local mean-based confidence weights give higher F1-score than kNN confidence weights.

Results: Average accuracy (with 25% label noise)

Average accuracy of the models (25% noise)

data	HNC	HNC sp	FHNC-const	FHNC-LM	FHNC-kNN	kNNC	kNNC-RENN	kNNC-AllkNN
0	0.776333	0.784667	0.792667	0.775333	0.784333	0.778333	0.776	0.787333
1	0.7774	0.801	0.8038	0.8026	0.8013	0.7941	0.7853	0.7935
2	0.6995	0.686	0.6895	0.682	0.6785	0.673	0.694	0.638
3	0.6585	0.7045	0.723	0.727	0.7045	0.708	0.726	0.6945
4	0.7656	0.8441	0.8433	0.8433	0.8488	0.8437	0.8351	0.8
5	0.994203	0.976812	0.998841	0.998841	0.99942	0.957681	0.976812	0.90029
6	0.792029	0.842754	0.850725	0.835507	0.842029	0.831884	0.819565	0.77029
7	0.89386	0.928947	0.936842	0.939474	0.935965	0.903509	0.896491	0.867544
8	0.959688	0.960313	0.959688	0.959687	0.958125	0.954375	0.960938	0.936875
9	0.754667	0.754	0.766	0.766	0.762	0.771333	0.710667	0.658
10	0.983971	0.98725	0.985792	0.985792	0.988707	0.983971	0.961384	0.95847
11	0.729982	0.751526	0.745063	0.745302	0.748175	0.766128	0.717175	0.690245
12	0.772414	0.812808	0.80197	0.795074	0.778325	0.837438	0.813793	0.799015
mean	0.812165	0.833437	0.838245	0.83507	0.833091	0.831035	0.821017	0.791851

Results: Accuracy ranking (with 25% label noise)

Accuracy ranking of the models (25% noise)

data	HNC	HNC sp	FHNC-const	FHNC-LM	FHNC-kNN	kNNC	kNNC-RENN	kNNC-AllkNN
0	6	3	1	8	4	5	7	2
1	8	4	1	2	3	5	7	6
2	1	4	3	5	6	7	2	8
3	8	5.5	3	1	5.5	4	2	7
4	8	2	4.5	4.5	1	3	6	7
5	4	5.5	2.5	2.5	1	7	5.5	8
6	7	2	1	4	3	5	6	8
7	7	4	2	1	3	5	6	8
8	3.5	2	3.5	5	6	7	1	8
9	5	6	2.5	2.5	4	1	7	8
10	5.5	2	3.5	3.5	1	5.5	7	8
11	6	2	5	4	3	1	7	8
12	8	3	4	6	7	1	2	5
mean	5.92308	3.46154	2.80769	3.76923	3.65385	4.34615	5.03846	7

Results: Accuracy compared to HNC (with 25% label noise)

Relative accuracy compared to HNC (25% noise)

data	HNC	HNC sp	FHNC-const	FHNC-LM	FHNC-kNN	kNNC	kNNC-RENN	kNNC-AllkNN
0	0	1.07342	2.10391	-0.128811	1.03049	0.257621	-0.0429369	1.41692
1	0	3.03576	3.39594	3.24157	3.07435	2.14819	1.01621	2.07101
2	0	-1.92995	-1.42959	-2.50179	-3.00214	-3.78842	-0.786276	-8.79199
3	0	6.98557	9.79499	10.4024	6.98557	7.51708	10.2506	5.46697
4	0	10.2534	10.1489	10.1489	10.8673	10.2011	9.07785	4.49321
5	0	-1.74927	0.466472	0.466472	0.524781	-3.67347	-1.74927	-9.44606
6	0	6.40439	7.4108	5.48948	6.3129	5.03202	3.47667	-2.74474
7	0	3.92542	4.80864	5.10304	4.7105	1.07949	0.294406	-2.94406
8	0	0.0651254	0	-1.15686e-14	-0.162813	-0.553566	0.130251	-2.37708
9	0	-0.0883392	1.50177	1.50177	0.971731	2.20848	-5.83039	-12.8092
10	0	0.33321	0.185117	0.185117	0.481303	0	-2.29545	-2.59163
11	0	2.9513	2.06591	2.0987	2.49221	4.95163	-1.75439	-5.44352
12	0	5.22959	3.82653	2.93367	0.765306	8.41837	5.35714	3.44388
mean	0	2.80689	3.40611	2.99543	2.69627	2.59989	1.3188	-2.32741

Results: Noise detection - fraction of samples that are flagged as noisy (with 25% label noise)

Average fractions of samples that are flagged as noisy (25% noise)

data	FHNC-const	FHNC-LM	FHNC-kNN	kNNC-RENN	kNNC-AllkNN
0	0.323111	0.321778	0.321111	0.481111	0.584222
1	0.328467	0.3374	0.338067	0.472667	0.5746
2	0.252667	0.321	0.346333	0.522	0.468
3	0.111	0.217	0.252667	0.42	0.428
4	0.227333	0.240867	0.2378	0.3448	0.3838
5	0.247969	0.248356	0.24913	0.313733	0.366731
6	0.266667	0.271981	0.270531	0.376329	0.433816
7	0.259238	0.254545	0.26217	0.340176	0.397654
8	0.259437	0.221272	0.265068	0.360792	0.521168
9	0.35	0.349554	0.365625	0.479464	0.509375
10	0.25322	0.25322	0.253706	0.299635	0.361847
11	0.325698	0.355227	0.338627	0.417079	0.444932
12	0.188119	0.256106	0.276568	0.430363	0.522112
mean	0.260994	0.280639	0.290569	0.404473	0.461251

Results: Noise detection - Recall (with 25% label noise)

Recall = the number of detected noisy labels / the number of all noisy labels

Average noise recall of the models (25% noise)

data	FHNC-const	FHNC-LM	FHNC-kNN	kNNC-RENN	kNNC-AllkNN
0	0.753778	0.724444	0.744	0.792	0.846222
1	0.775467	0.784533	0.7832	0.822933	0.8656
2	0.44	0.564	0.612	0.777333	0.608
3	0.253333	0.46	0.529333	0.726667	0.678667
4	0.6848	0.711733	0.7096	0.8024	0.699467
5	0.993798	0.995349	0.99845	0.955039	0.731783
6	0.792308	0.755769	0.744231	0.740385	0.665385
7	0.916279	0.913953	0.927907	0.848837	0.727907
8	0.929167	0.790833	0.945	0.966667	0.953333
9	0.732143	0.733929	0.755357	0.753571	0.689286
10	0.981553	0.981553	0.98835	0.896117	0.729126
11	0.679872	0.725559	0.714058	0.673163	0.603514
12	0.536842	0.692105	0.686842	0.881579	0.892105
mean	0.728411	0.756443	0.779871	0.818207	0.745415

Noise recall ranking of the models (25% noise)

data	FHNC-const	FHNC-LM	FHNC-kNN	kNNC-RENN	kNNC-AllkNN
0	3	5	4	2	1
1	5	3	4	2	1
2	5	4	2	1	3
3	5	4	3	1	2
4	5	2	3	1	4
5	3	2	1	4	5
6	1	2	3	4	5
7	2	3	1	4	5
8	4	5	3	1	2
9	4	3	1	2	5
10	2.5	2.5	1	4	5
11	3	1	2	4	5
12	5	3	4	2	1
mean	3.65385	3.03846	2.46154	2.46154	3.38462

Results: Noise detection - Precision (with 25% label noise)

Precision = the number of flagged samples that are actually noisy / the number of all flagged samples

Average noise precision of the models (25% noise)

data	FHNC-const	FHNC-LM	FHNC-kNN	kNNC-RENN	kNNC-AllkNN
0	0.584807	0.56573	0.579165	0.412093	0.362346
1	0.590244	0.581369	0.579206	0.435478	0.376629
2	0.448285	0.447286	0.446735	0.372147	0.324715
3	0.669827	0.570752	0.556654	0.432617	0.396478
4	0.761243	0.742708	0.756176	0.581889	0.455688
5	1	1	1	0.759583	0.499394
6	0.746672	0.700564	0.696541	0.49583	0.387579
7	0.89276	0.906096	0.893894	0.631131	0.463164
8	0.896727	0.900154	0.892214	0.67162	0.457825
9	0.524328	0.526427	0.516506	0.393604	0.336226
10	0.97027	0.97027	0.975162	0.748571	0.504117
11	0.52357	0.510259	0.527277	0.403204	0.338874
12	0.734426	0.679186	0.637403	0.517426	0.429387
mean	0.718705	0.700062	0.696687	0.527323	0.410186

Noise precision ranking of the models (25% noise)

data	FHNC-const	FHNC-LM	FHNC-kNN	kNNC-RENN	kNNC-AllkNN
0	1	3	2	4	5
1	1	2	3	4	5
2	1	2	3	4	5
3	1	2	3	4	5
4	1	3	2	4	5
5	2	2	2	4	5
6	1	2	3	4	5
7	3	1	2	4	5
8	2	1	3	4	5
9	2	1	3	4	5
10	2.5	2.5	1	4	5
11	2	3	1	4	5
12	1	2	3	4	5
mean	1.57692	2.03846	2.38462	4	5

Results: Noise detection - F1 score (with 25% label noise)

F1 score = the harmonic mean of recall and precision

Average F1 score of noise detection (25% noise)

data	FHNC-const	FHNC-LM	FHNC-kNN	kNNC-RENN	kNNC-AllkNN
0	0.657599	0.63325	0.651029	0.542013	0.507307
1	0.670244	0.667819	0.665913	0.569436	0.524868
2	0.410636	0.490031	0.513779	0.503313	0.423241
3	0.3211	0.472551	0.52272	0.542236	0.500347
4	0.715855	0.725046	0.727631	0.674507	0.5518
5	0.996875	0.997659	0.999222	0.846081	0.592755
6	0.768521	0.724993	0.714186	0.592412	0.489135
7	0.903905	0.909753	0.910197	0.722915	0.565456
8	0.91214	0.791034	0.9178	0.792266	0.618537
9	0.609897	0.611859	0.61349	0.516916	0.451514
10	0.975847	0.975847	0.981686	0.815663	0.595691
11	0.588688	0.599118	0.605983	0.504279	0.433981
12	0.608781	0.68493	0.655277	0.651294	0.579481
mean	0.703084	0.714145	0.729147	0.63641	0.525701

Ranking of F1 score of noise detection (25% noise)

data	FHNC-const	FHNC-LM	FHNC-kNN	kNNC-RENN	kNNC-AllkNN
0	1	3	2	4	5
1	1	2	3	4	5
2	5	3	1	2	4
3	5	4	2	1	3
4	3	2	1	4	5
5	3	2	1	4	5
6	1	2	3	4	5
7	3	2	1	4	5
8	2	4	1	3	5
9	3	2	1	4	5
10	2.5	2.5	1	4	5
11	3	2	1	4	5
12	4	1	2	3	5
mean	2.80769	2.42308	1.53846	3.46154	4.76923

Results conclusion for the case where 25% of the labels of training samples are corrupted

- Accuracy
  - All the flexible HNC models (83.82%, 83.51%, 83.31%) have higher accuracy than kNN classifiers (83.10%).
  - The sparsified HNC (83.34%), even without noise detection, also has higher accuracy then kNN classifiers.
  - The three flexible HNC models and the sparsified HNC are the four models with the best ranking. The flexible HNC model with the constant confidence weight has the best rank.
- Noise detection
  - The kNN classifier with RENN has the highest noise recall (i.e. the fraction of noisy labels that are detected). This partly comes from the fact that kNN with RENN and AllkNN flag the labels of so many samples as noisy.
  - Regarding the noise precision (i.e. the fraction of noisy labels out of all samples that are flagged as noisy), all the flexible HNC models have better performance.
  - The flexible HNC models with confidence weights (70.31%, 71.41%, 72.19 %) have higher F1 score of noise detection than the kNN classifiers (63.64%, 52.57%).

Future directions

- other confidence functions
  - confidence function at each node could be dependent on (or scaled by) the weights of other arcs that are adjacent to the node
- graph sparsification
  - other existing sparsification methods such as sparse computation framework and regular graph structure
  - the sparsification (graph structure) among the nodes of labeled samples and the nodes of unlabeled samples can be different
- more flexibility
  - probabilistic labels rather than hard labels (binary labels)
- varying noise level
  - vary the noise level that we add to the datasets
  - corrupt the two classes with unequal noise levels

References I


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


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.


Dorit S Hochbaum.
A polynomial time algorithm for rayleigh ratio on discrete variables: Replacing spectral techniques for expander ratio, normalized cut, and cheeger constant.
Operations Research, 61(1):184–198, 2013.


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

References II


Dorit S Hochbaum and Philipp Baumann.
Sparse computation for large-scale data mining.
IEEE Transactions on Big Data, 2(2):151–174, 2016.


Ivan Tomek et al.
An experiment with the edited nearest-nieghbor rule.
1976.


Donghai Guan, Weiwei Yuan, Young-Koo Lee, and Sungyoung Lee.
Nearest neighbor editing aided by unlabeled data.
Information Sciences, 179(13):2273–2282, 2009.


Michael R Smith and Tony Martinez.
Improving classification accuracy by identifying and removing instances that should be misclassified.
In *The 2011 International Joint Conference on Neural Networks*, pages 2690–2697. IEEE, 2011.

References III

Salvador Garcia, Joaquin Derrac, Jose Cano, and Francisco Herrera.
Prototype selection for nearest neighbor classification: Taxonomy and empirical study.
IEEE transactions on pattern analysis and machine intelligence, 34(3):417–435, 2012.

Nordiana Mukahar and Bakhtiar Affendi Rosdi.
Performance comparison of prototype selection based on edition search for nearest neighbor classification.
In *Proceedings of the 2018 7th International Conference on Software and Computer Applications*, pages 143–146, 2018.