Non-parametric Semi-supervised Learning by Bayesian Label Distribution Propagation

Jonatan Møller Nuutinen Gøttcke[1](✉), Arthur Zimek[1](✉), and Ricardo J. G. B. Campello[2](✉)

[1] Institute of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark
{goettcke,zimek}@imada.sdu.dk

[2] School of Mathematical and Physical Sciences, University of Newcastle, Callaghan, Australia
Ricardo.Campello@newcastle.edu.au

Abstract. Semi-supervised classification methods are specialized to use a very limited amount of labelled data for training and ultimately for assigning labels to the vast majority of unlabelled data. Label propagation is such a technique that assigns labels to those parts of unlabelled data that are in some sense close to labelled examples and then uses these predicted labels in turn to predict labels of more remote data. Here we propose to not propagate an immediate label decision to neighbors but to propagate the label probability distribution. This way we keep more information and take into account the remaining uncertainty of the classifier. We employ a Bayesian schema that is simpler and more straightforward than existing methods. As a consequence we avoid to propagate errors by decisions taken too early. A crisp decision can be derived from the propagated label distributions at will. We implement and test this strategy with a probabilistic k-nearest neighbor classifier, proving competitive with several state-of-the-art competitors in quality and more efficient in terms of computational resources.

Keywords: Semi-supervised classification · k-Nearest neighbor classification · Transductive learning · Label propagation

1 Introduction

While easily collectable unlabelled data become more abundant, and labelled data continue to be a scarce resource, semi-supervised learning remains relevant. Semi-supervised learning falls between supervised learning (learning from labelled data) and unsupervised learning (learning from unlabelled data) [5]. It is used for improving unsupervised learning by taking advantage of information traditionally used in supervised learning and for improving the performance of supervised learning methods by using unlabelled instances.

© Springer Nature Switzerland AG 2021

In semi-supervised classification we have to distinguish between assigning labels to unlabelled data during training and the application of the resulting semi-supervised classifier to new unseen data during testing, where the classifier is built on the complete training data consisting of labelled and unlabelled instances. The labelling of unlabelled training instances is known as 'transduction' [20,21] or as 'label propagation' since it is often done by propagating labels from the labelled training instances to the unlabelled training instances. After label propagation, all training data can be used for inducing labels of new unseen query instances as it is conventionally done in supervised learning. This is therefore also known as 'induction'. Applying a learner for transduction therefore yields a training set with then all instances labelled that could be used by any other classifier for induction, i.e., conventional classification [27]. The classification model used for induction could therefore be different from the semi-supervised classifier used for transduction, but it could also be the same method that is used beyond transduction on the training data also for induction on new, unseen data. Besides describing different phases, tasks, or scenarios in the context of semi-supervised learning, the exact relationship between 'transduction', 'induction', and 'semi-supervised learning' remains debatable [4]. In this paper we evaluate methods in a transductive setting as it is common practice [8,9,12,23–26].

We argue that it might be important to account for uncertainties during transduction and to keep information on uncertain decisions possibly also beyond the transduction phase, if induction is treated separately and the classification algorithm employed for induction can make use of uncertain label information or label probability distributions. For a simple illustration, consider the one-dimensional distribution of classes in one attribute (sepal length) of the well-known Iris data, plotted in Fig. 1. Some classification model might be based on the estimated probability density distribution and decide for the maximum likelihood class at any given point in the data space. Considering the example of the figure, if we have a sepal length, say, between 5 and 6 cm, we can decide on a clear decision boundary but a high probability would remain to have chosen the wrong class. If just the resulting label is propagated and used for ensuing decisions, these later decisions are necessarily oblivious of a possibly considerable level of uncertainty that actually affects also these later decisions. To account for this uncertainty, we suggest that, instead of propagating a label, the label *probability distribution* should be propagated and would thus also be available for later decisions. We could see this as an attempt to keep as much information as possible as long as potentially useful for the classification of new instances.

This idea has been employed in some specific graph-based methods, as we will survey below. Here we propose a more general probabilistic, non-parametric semi-supervised classification schema and demonstrate its benefits by implementing it with a probabilistic k nearest neighbor classifier that is conceptually simpler and yet compares favorably against state-of-the-art methods for semi-supervised classification on a large collection of datasets being considerably more efficient.

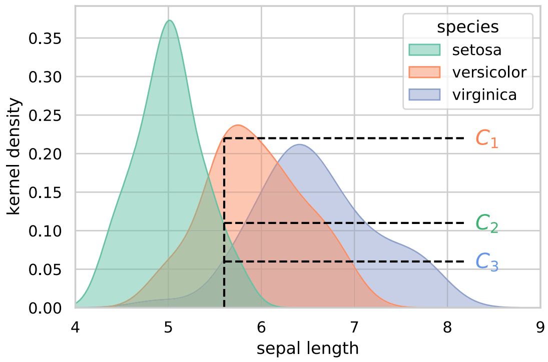


Fig. 1. Some classifier's model (estimated class-conditional probability density distributions). Deciding at any point according to the maximum class (posterior) probability and using only that label later renders further propagation or later induction oblivious of the evidence for other classes. Example data from the Iris dataset.

In the remainder, we give an overview of related work on semi-supervised classification (Sect. 2), introduce the general concept and a concrete implementation of our method (Sect. 3), study its performance on a large collection of datasets and compare against several state-of-the-art methods in terms of effectiveness and efficiency (Sect. 4), and conclude with a short discussion of some properties of the compared methods (Sect. 5).

2 Related Work

There are many variations of semi-supervised learning such as *self-training* [16] and *co-training* [2]. Transductive learning [20] is a part of the foundation of semi-supervised learning and relates to an approach that uses both labelled and unlabelled data as training data, $TR = L \cup U$, to predict labels for U. Some of the most popular methods in this field have been surveyed in books broadly discussing the area [5,27]. Well known methods such as graph classification [12, 23] and support vector machines have been adapted to the transductive setting [9], and have been used in combination with Laplacian regularization as for Gaussian Field Harmonic Function (GFHF) [25,26]. In the following we discuss some methods that are more closely related to our approach.

The basic idea of GFHF [25,26] is to model a transition probability in the graph representing the dataset, typically using the RBF kernel. All nodes are associated with a class label distribution which is updated following the transition probabilities until convergence. GFHF also was seminal for Laplacian support vector machines and Laplacian regularized least squares [1]. GFHF propagates the transition probabilities estimated by the RBF kernel on the complete graph. In each iteration the propagation of transition probabilities is normalized to maintain a probability interpretation, and the method iterates until convergence.

Learning with Local and Global Consistency (LGC) [23] is inspired by GFHF and mainly differs in the propagation process, including a parameter α that determines how the information from the previous and the current iteration are weighted. LGC also uses the RBF kernel to generate a weight matrix for the complete graph and normalizes the weights to maintain label probability distributions for propagation in each iteration.

Szummer and Jaakola [18] proposed one of the first semi-supervised classification algorithms, using Markov random walks from a number of random points in the dataset, and the RBF kernel (similarly as Zhu et al. [26]) for determining the edge weights to estimate the class conditional probability. They use kNNs for finding the local manifold structure. Substantial differences between the method presented in this paper and Szummer and Jaakola's is the choice of the kernel used to define the label distributions, their use of a symmetrized kNN graph, and their probability estimation procedure. Furthermore their method requires several additional parameters such as the time parameter t for the number of steps in the Markov process to govern smoothness and the σ parameter of the RBF kernel for edge weights, and the number of random starting points.

Liu and Chang [12] introduced the RMGT algorithm, a graph based algorithm that utilizes the combinatorial Laplacian matrix to describe the local manifold structure. They introduce a new underlying graph topology referred to as the symmetry favored kNN graph, which adds weights to bidirectional edges in the directed kNN graph. De Sousa and Batista [17] extended this algorithm further (RMGTHOR) by modifying the regularization framework to use a normalized Laplacian, or a Laplacian with a degree higher than 1, instead of the combinatorial Laplacian used in the earlier methods.

3 Label Probability Distribution Propagation

3.1 Motivation

In semi-supervised classification, a core assumption is that the labelled subset L of the training set is much smaller both in relative and absolute terms than the unlabelled subset U, i.e., $|L| \ll |U|$. Therefore a learner should be extra careful when propagating labels because of the high probability of propagating errors when decisions are based on insufficient information. With further propagation of potentially erroneous labels such errors can spread and have a severe impact on the quality of the transduction.

This is the core motivation for our proposal to not propagate class *labels* but instead to propagate the *distribution* of class labels (or class label *probabilities*) from instances in L to instances in U. Such class probabilities can be determined in principle using any probabilistic classifier.

3.2 General Schema

In Fig. 1 we see three probability density functions for the three classes over the sepal length attribute in the Iris dataset. We propose to propagate label probability distributions to unlabelled instances in a semi-supervised manner using the (estimated) probability density for each class starting from a straightforward Bayesian schema:

With the prior class probability $\Pr(c)$ for each class $c \in C$ and Bayes theorem, the probability for instance x to belong to a class c can be estimated by:

$$\Pr(c|x) \propto \frac{\hat{f}(x|c)\Pr(c)}{\sum_{c_i \in C} \hat{f}(x|c_i)\Pr(c_i)} \tag{1}$$

where $\hat{f}$ is some estimate of the probability density (which could be a direct probability estimate, if some classifier delivers that).

Such estimated label probability distributions are assigned to all instances $x \in U$, such that the label y_i of instance x_i is in fact a label distribution:

$$y_i = (\Pr(c_1|x_i), \Pr(c_2|x_i), \cdots, \Pr(c_n|x_i))^\top \tag{2}$$

Consider an unlabelled instance x in Fig. 1. The probability density for each class is given by c_1, c_2, and c_3, and the probability for belonging to each class can then be calculated by Eq. (1). The standard maximum likelihood prediction would predict class c_3 and immediately lose the information that the other two classes, albeit less likely, still carry a non-negligible probability that could be helpful to decide close cases downstream, when using x transductively (or even for induction later on). This is because, after the assignment of a label probability distribution to instance x, x is moved from the set of unlabelled data U to the set of labelled data L and is used for the transductive labelling process that continues until $U = \emptyset$ and all training instances are labelled.

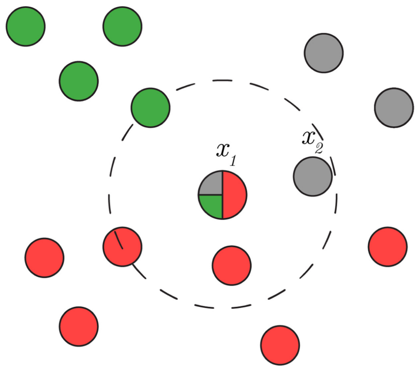
3.3 *k*NN-Label Distribution

To test this concept in semi-supervised classification we employ a probabilistic *k*NN classifier to estimate label probability distributions in a non-parametric way and describe an algorithm to propagate label probability density distributions using *k*NN, resulting in an algorithm *k*NN *Label Distribution Propagation* (*k*NN-LDP). Propagating label probability distributions allows data instances to have a soft labelling and the transduction to account for this label distribution when calculating the label probability distributions of unlabelled neighbors.

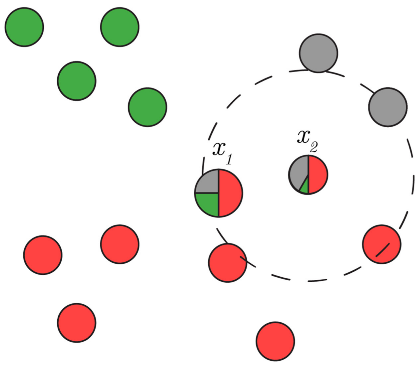
While we keep all information of the label probability distributions as long as possible, we can at any point derive a crisp labelling of a query object if needed, taking the maximum of the assigned class probabilities. Also note that, although we focus on the transduction here, we could also apply the same algorithm for induction beyond the training data to predict the class (or the label probability distribution) for any unseen query object.

In the supervised scenario of using the *k*NN classifier, the label probability distribution for some instance x is given by the class-conditional density estimates based on the k nearest neighbors of x taken over the labeled training data L [7,22]:

$$\hat{f}(x|c_j) = \frac{|\{x_\ell \in k\mathrm{NN}(x) \cap c_j\}|}{|\{x_\ell \in L \cap c_j\}| \cdot Vol_{k\mathrm{NN}(x)}} \tag{3}$$



(a) An unlabelled instance in the training set receives a label distribution from labelled neighbors (note that we have a tie in the neighborhood and therefore effectively use $k+1$ neighbors).



(b) An unlabelled instance in the training set receives the label distributions from neighbors, including examples labeled by label distribution propagation (no tie here).

Fig. 2. kNN label distribution propagation, using $k = 3$, without and with partially labelled examples.

where $|\cdot|$ denotes the cardinality of a set and $Vol_{k\text{NN}(x)}$ denotes the volume needed to cover k nearest neighbors of x, centered at x. The shape of this volume will depend on the employed distance function. Note, however, that the volume cancels nicely out when putting this into Eq. (1).

In the semi-supervised scenario tackled here, an instance among the nearest neighbors might not have a crisp label but a label probability distribution itself, or no label for instances $\in U$. For getting a well-defined probability distribution we can treat the "unknown" case as a special class. Accounting for partial labels in Eq. 3 thus yields

$$\hat{f}(x|c_j) = \frac{\sum_{x_\ell \in k\text{NN}(x)} \Pr(c_j|x_\ell)}{\sum_{x_\ell \in L} \Pr(c_j|x_\ell) \cdot Vol_{k\text{NN}(x)}} \tag{4}$$

Using this in Eq. (1), the probability for each class c in the label distribution, depending on the label probability distributions of the k nearest neighbors, is therefore given by

$$\Pr(c|x) = \frac{\sum_{x_\ell \in k\text{NN}(x)} \Pr(c|x_\ell)}{|\, k\text{NN}(x)|} \tag{5}$$

We illustrate the method in Fig. 2. The example $x_1 \in U$ receives label information from its k nearest neighbors. One of the nearest neighbors, $x_2 \in U$, is unlabelled. As a result, x_1 would now carry partial label information which we can interpret as a label probability distribution (Fig. 2(a)). Next, example x_2 is processed and receives label information from its neighbors, including x_1, thus not just counting labels of classes but considering the label probability distribution over the k nearest neighbors (Fig. 2(b)).

3.4 Abstention in Case of Insufficient Information

We have to account for a potential complication in this process of assigning label probability distributions. There might be insufficient information to assign some label probability distribution to some given unlabelled instance. This is a complication that is not unlikely in the semi-supervised scenario, where we assume much less labeled than unlabeled training data.

If we encounter such an instance that cannot get assigned any class probabilities, we assign a special label signaling the fact that the label distribution is unknown, thus employing the concept of abstaining classifiers [15], although we do not propose here to optimize the classifier w.r.t. abstention.

3.5 Propagation Algorithm

To use as much information as possible for the label assignment in just two passes over the data we start with the instances where most information is available, that is where the sum of the class probabilities over the neighbors (except for the class "unknown") is maximal, and continue to process instances with decreasing order w.r.t. this available information (which might change over time). This requires checking all neighborhoods in advance. For the sake of efficiency, the forward and reverse k nearest neighbors should be indexed in this first pass.

The information that can be used for label distribution assignments can be captured in weights:

$$w(x) = \sum_{c \in C \setminus \{\text{“unknown”}\}} \Pr(c|x) \tag{6}$$

These weights w are used to keep the instances sorted in decreasing order w.r.t. the available information in some priority queue. This way, as much information as possible is used in one sweep over the data, updating the label probability distributions and the weights of the reverse k nearest neighbors (RkNN) of updated instances (i.e., those that are affected by an update of the current instance). This might make instances climb up in the priority queue if their weight changed because their neighbors got label distribution information assigned. In the example of Fig. 2, this would be the case for $x_2 \in \mathrm{R}k\mathrm{NN}(x_1)$: after having assigned a label probability distribution to x_1, $w(x_2)$ will increase. Then we can assign an estimate of the label distribution for each instance, as defined in Eqs. (2) and (5). Note that the definitions of probability distributions include the class "unknown", such that probabilities sum up to one. A sketch of the procedure is provided in Algorithm 1.

Algorithm 1. kNN-LDP

```
1: for all $x \in U$ do
2:    index forward and reverse $k$ nearest neighbors ( $k$NN, R$k$NN )
3:    $x.w \leftarrow \sum_{c \in C \setminus \{\text{unknown}\}} \Pr(c|x)$ {Eq. (6)}
4: end for
5: $PQ_U \leftarrow$ priority-queue($U$) {decreasing order w.r.t. $x.w$}
6: while $PQ_U$.size $> 0$ do
7:    $x \leftarrow PQ_U$.getMax()
8:    if $x.w > 0$ then
9:       $x.y \leftarrow (\Pr(c|x))_{c \in C}$ {Eqs. (2) and (5)}
10:      for all $p \in$ R$k$NN($x$) do
11:         $p.w \leftarrow \sum_{c \in C \setminus \{\text{unknown}\}} \Pr(c|p)$ {Eq. (6)}
12:         $PQ_U$.update($p$)
13:      end for
14:      $L \leftarrow L \cup \{x\}$
15:   else
16:      $x.y \leftarrow$ unknown
17:      for all $p \in PQ_U$ do
18:         $p.y \leftarrow$ unknown
19:         $L \leftarrow L \cup \{p\}$
20:      end for
21:      $PQ_U \leftarrow \emptyset$
22:   end if
23: end while
```

3.6 Advantages and Disadvantages of kNN-LDP

The label distribution propagation algorithm proposed here also has an underlying graph interpretation when implemented with a k nearest neighbor classifier. It has some advantages over other graph-based methods. The asymptotic runtime of the kNN-based label distribution propagation algorithm is identical to that of finding the kNN, i.e., the operation of identifying the nearest neighbors is the computational bottleneck as in many applications and could naturally benefit from employing efficient neighborhood search methods [10,11]. Yet, due to the heuristic order of processing instances, our method, as opposed to many competitors, does not require any iterations.

The runtime for graph-based algorithms depends on the topology of the graph but is typically higher than the runtime of kNN-LDP which only makes neighborhood queries for the unlabelled data. A mutual-kNN graph requires the computation of the nearest neighbors of all labelled and unlabelled instances which takes $\mathcal{O}(n^2)$ for a dataset of size n. If Ozaki's graph connection method [13] is used, computing the complete similarity graph takes $\mathcal{O}(n^2)$, which cannot be improved. Finding the minimum spanning tree (or rather maximum spanning tree, as it is based on similarities, not distances) takes $\mathcal{O}((V+E)\log V)$, where V is the number of vertices, and E is the number of edges in the graph. For the complete graph this takes $n + \frac{n(n-1)}{2} \log \frac{n(n-1)}{2}$ using Prim's algorithm.

Table 1. Competitors and their implementations

Method	Impl. Name	Impl. Source
GFHF [26]	Label Propagation	Scikit-Learn [14]
LGC [23]	Label Spreading	
LapRLS [1]	LapRls	https://github.com/HugoooPerrin/semi-supervised-learning
LapSVM [1]	LapSvm	
RMGT [12]	RMGT	de Sousa & Batista [17]
RMGTHOR [17]	RMGTHOR	
This paper	kNN LDP	https://github.com/Goettcke/kNN_LDP

Although the kNN Label Distribution Propagation is different from the most common graph-based algorithms it also comes with some of the same disadvantages. When constructing an adjacency matrix in nearest-neighbor graph-based semi-supervised learning algorithms, the number of components plays an essential role in the success of the label propagation. A similar problem is present in kNN-LDP, if an unlabelled instance cannot be reached by the propagation through neighborhoods, i.e., if an unlabelled instance resides in a graph component without any labelled instances. This problem tends to occur more with a smaller ratio of $\frac{L}{U}$ and a smaller value of k, and tends to affect the label propagation late in the process.

There are different strategies to tackle this problem. One could be to increase k until at least one neighbor carries label information. However, sparse graphs are observed empirically to perform better than dense or complete graphs [28], as they have a higher sensitivity to detecting the local manifold which the data points lie on. Another solution, as seen in related work, is to assign the majority class if no other information is available.

The solution we have chosen for kNN-LDP is to make the classifier abstaining from making a decision where it does not have the information and giving it the class label "unknown". It should be noted that this is a clear disadvantage in the comparison, as such a label will always count as an error. We will see in the evaluation that trading these errors in the evaluation metric for not propagating potential errors to further decisions seems to pay off.

4 Experimental Evaluation

4.1 Competitors

As competitors we selected the more closely related methods GFHF [26], LGC [23], RMGT [12], and the more recent RMGTHOR [17] method. The Laplacian regression method, Laplacian Regularized Least Squares (LapRLS) and Laplacian Support Vector Machines (LapSVM) [1] were also evaluated. We used publicly available implementations, an overview is provided in Table 1.

Table 2. Datasets used for comparative evaluation.

Dataset	Classes	Attributes	Instances	Dataset	Classes	Attributes	Instances
australian	2	14	690	page-blocks	5	10	5472
banana	2	2	5300	phoneme	2	5	5404
breast	2	30	569	segment	7	19	2310
bupa	2	6	345	spambase	2	57	4597
cleveland	5	13	297	spectfheart	2	44	267
contraceptive	3	9	1473	tae	3	5	151
dermatology	6	34	358	vowel	11	13	990
glass	6	9	214	wine	3	13	178
hayes roth	3	4	160	wine-red	6	11	1599
heart	2	13	270	wine-white	7	11	4898
iris	3	4	150	wisconsin	2	9	683
led7digit	10	7	500	COIL	2	50	1500
mammo. mass	2	5	830	digit-1	2	50	1500
monk-2	2	6	432	G-241C	2	50	1500
movement libras	15	90	360	G-241N	2	50	1500
new thyroid	3	5	215	USPS	2	50	1500

4.2 Parameters

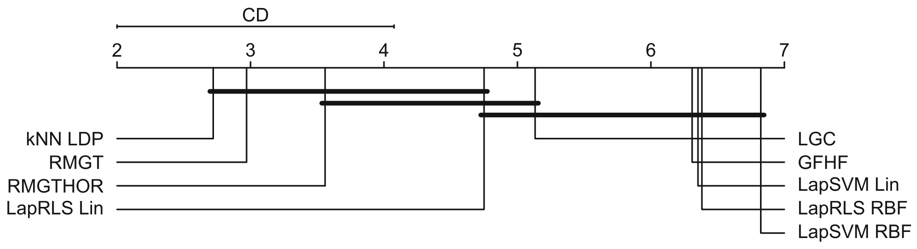
In the experiments all distances are Euclidean. LapSVM and LapRLS were tested in two settings, using the Linear kernel and the RBF kernel. GFHF and LGC use the RBF kernel. For LGC we used $\alpha = 0.2$ as it is the default value in the Scikit-Learn implementation. RMGT uses combinatorial Laplacian regularization. RMGTHOR uses the normalized Laplacian. Both methods use local linear embedding for building the weight matrices. RMGTHOR uses a Laplacian degree of 1. For all algorithms based on k nearest neighbors we invariably set $k = 10$, following findings that some small value is typically a good choice for the local density estimation and for determining the local manifold [28].

For the methods taking an RBF kernel, $\sigma_M \in \{0.1, 0.5, 1\}$ was tested. For the parameters γ_L and γ_M for LapSVM and LapRLS a grid search was performed for all combinations of the values $\gamma_L \in \{0.1, 0.5, 1\}$ and $\gamma_M \in \{0.1, 0.5, 1\}$, that is using the same range as the original publication [1]. The grid search tries all combinations of the parameters in the parameter sets, and for each dataset the best result achieved is extracted and used for comparison.

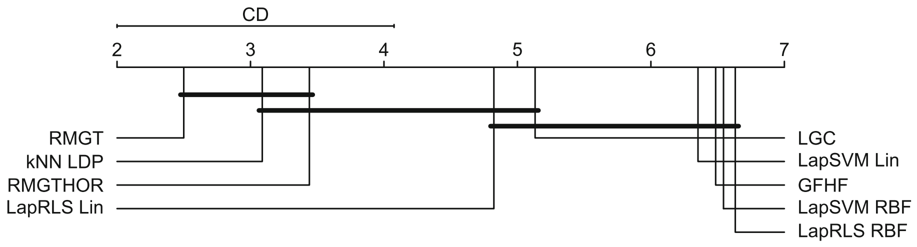
Parameter optimization is necessary for these methods to avoid poor performance. It should be noted, though, that selecting the best results for these methods gives them an advantage in the comparison.

4.3 Datasets

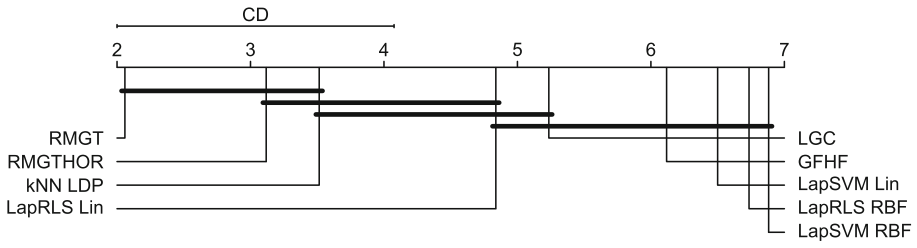
We have evaluated our method and state-of-the-art competitors on the 5 datasets commonly used in semi-supervised classification benchmarks [3] as well as on 27 datasets used by Triguero et al. [19]. The datasets have been selected such that



(a) 5% labelled and 95% unlabelled data



(b) 10% labelled and 90 % unlabelled data



(c) 20% labelled and 80% unlabelled data

Fig. 3. Critical difference plots showing the relation between the tested algorithms transductive performance on different proportions of available label information.

all algorithms compared to in this study could be applied without adjustments. An overview on the datasets is provided in Table 2.

4.4 Evaluation Setup

Each dataset has been split into labelled training data and unlabelled training data, in three different split proportions (labelled, unlabelled): $[0.05, 0.95]$, $[0.1, 0.9]$, and $[0.2, 0.8]$. In each setting, we average the test results over 64 random samples. We measure accuracy on crisp decisions derived from the label probability distributions post hoc in case of *k*NN-LDP, and the built-in predict functions for other methods. We count as error when *k*NN-LDP is abstaining from classifying an instance.

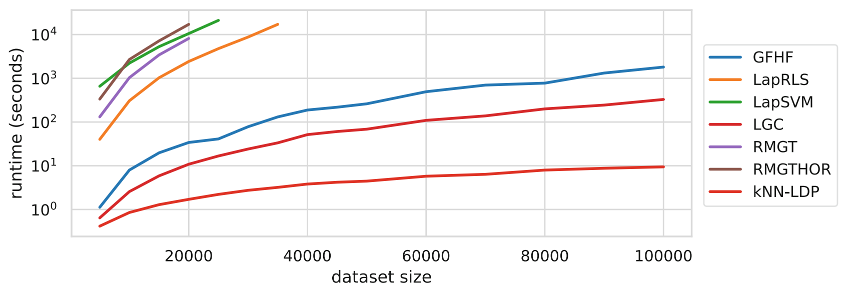


Fig. 4. Runtime in seconds scaling with dataset size.

4.5 Results

For assessing the performance of the different methods, we show "critical difference plots" following the methodology described by Demšar [6] for assessing the statistical significance in the ranking of the compared methods in terms of accuracy in Fig. 3. These plots visualize the mean rank over all datasets and the critical difference given the number of datasets and algorithms used in the statistic. If the mean rank for a method is connected to the mean rank of another method by a horizontal bar, the two methods are within the critical difference and their performance is not significantly different.

kNN-LDP often achieves the highest accuracy score and is significantly better than several competitors, and not different from the other methods with statistical significance. The advantage of kNN-LDP tends to be more prominent with a smaller proportion of label information. It performs best for the smallest fraction of labelled data, which is the most important scenario for semi-supervised learning. In all scenarios, kNN-LDP forms a top group together with RMGT and RMGTHOR where the differences are not statistically significant. We will see next, however, that kNN-LDP is much more efficient than all the other methods, in particular kNN-LDP beats RMGT and RMGTHOR by a large margin in terms of scalability.

4.6 Scalability

For testing the scalability of the algorithms we generated 2-dimensional datasets with two-class problems using the Scikit-Learn *make_classification* function, increasing the dataset size. The classification problems all have uniform class distributions, and the algorithms were given 10% labelled training data. The test hardware used consists of an AMD EPYC 7501 32-Core Processor, and 256 GB available memory. All algorithms were run with default parameters in these tests. We depict the results of the scalability experiment in Fig. 4.

The rapid increase in runtime for LapRLS, LapSVM, RMGT, and RMGTHOR and their also rapidly increasing demands on system memory prevented tests on larger dataset sizes. However, the disadvantage of these methods in terms of runtime and scalability behavior is already quite clear at this point (note the logarithmic scale of the runtime axis). GFHF and LGC remain more competitive to our method, although they are considerably slower. Given the logarithmic scale, the scalability performance of kNN-LDP is clearly superior even if we would disregard its advantage in absolute runtime accounting for possible implementation advantages [11].

5 Conclusion

We studied an elegant non-parametric method with a clear interpretation in terms of density estimation and Bayesian reasoning here that performs as good as or better than state-of-the-art methods on a large collection of datasets even though it was put on a disadvantage compared to other methods in two aspects:

First, it is a fundamental requirement in graph-based algorithms that each instance (i.e., a vertex in some kNN graph) must belong to a component in which at least one other vertex is labelled. While other methods use undirected, symmetrized, or even complete graphs to adhere to this assumption, in the case of kNN-LDP the assumption is more likely to be violated because the kNN graph is inherently directed. As a consequence each unlabelled vertex should not only be in a component with at least one other labelled vertex but also be directly connected to it. While other methods use various heuristics to solve cases where this assumption is violated and can be sometimes correct with that, we simply abstain from a decision which as such will always count as an error. However, this abstention fits and contributes to the fundamental motivation and strategy of our method: to avoid the propagation of errors, which will be of utmost importance in the induction on test or new data.

Second, we performed grid search optimization of several parameters for the competing Laplacian methods and the methods using an RBF kernel. Without such parameter tuning, these methods would not be able to achieve reasonable performance. No such parameter optimization was done for the k-value used in the kNN-LDP method, and the other methods using nearest neighbor information, where some small value is typically a good choice for the local density estimation and for determining the local manifold [28].

In terms of efficiency and scalability, our method is clearly outperforming the competitors.

References

1. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. J. Mach. Learn. Res. **7**, 2399–2434 (2006)
2. Blum, A., Mitchell, T.M.: Combining labeled and unlabeled data with co-training. In: COLT, pp. 92–100 (1998)
3. Chapelle, O., Schölkopf, B., Zien, A.: Analysis of benchmarks. In: Semi-Supervised Learning [5], pp. 376–393

4. Chapelle, O., Schölkopf, B., Zien, A.: A discussion of semi-supervised learning and transduction. In: Semi-Supervised Learning [5], pp. 473–478
5. Chapelle, O., Schölkopf, B., Zien, A. (eds.): Semi-Supervised Learning. The MIT Press, Cambridge (2006)
6. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
7. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley, Hoboken (2001)
8. Castro Gertrudes, J., Zimek, A., Sander, J., Campello, R.J.G.B.: A unified view of density-based methods for semi-supervised clustering and classification. Data Min. Knowl. Discov. **33**(6), 1894–1952 (2019). https://doi.org/10.1007/s10618-019-00651-1
9. Joachims, T.: Transductive inference for text classification using support vector machines. In: ICML, pp. 200–209 (1999)
10. Kirner, E., Schubert, E., Zimek, A.: Good and bad neighborhood approximations for outlier detection ensembles. In: SISAP, pp. 173–187 (2017). Springer, Cham. https://doi.org/10.1007/978-3-319-68474-1_12
11. Kriegel, H.-P., Schubert, E., Zimek, A.: The (black) art of runtime evaluation: are we comparing algorithms or implementations? Knowl. Inf. Syst. **52**(2), 341–378 (2016). https://doi.org/10.1007/s10115-016-1004-2
12. Liu, W., Chang, S.: Robust multi-class transductive learning with graphs. In: CVPR, pp. 381–388 (2009)
13. Ozaki, K., Shimbo, M., Komachi, M., Matsumoto, Y.: Using the mutual k-nearest neighbor graphs for semi-supervised classification on natural language data. In: CoNLL, pp. 154–162. ACL (2011)
14. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
15. Pietraszek, T.: On the use of ROC analysis for the optimization of abstaining classifiers. Mach. Learn. **68**(2), 137–169 (2007)
16. Scudder, H.J., III.: Probability of error of some adaptive pattern-recognition machines. IEEE Trans. Inf. Theory **11**(3), 363–371 (1965)
17. de Sousa, A.R., Batista, G.E.A.P.A.: Robust multi-class graph transduction with higher order regularization. In: IJCNN, pp. 1–8 (2015)
18. Szummer, M., Jaakkola, T.S.: Partially labeled classification with Markov random walks. In: NIPS, pp. 945–952 (2001)
19. Triguero, I., Sáez, J.A., Luengo, J., García, S., Herrera, F.: On the characterization of noise filters for self-training semi-supervised in nearest neighbor classification. Neurocomputing **132**, 30–41 (2014)
20. Vapnik, V.: Statistical Learning Theory. Wiley, Hoboken (1998)
21. Vapnik, V.: Transductive inference and semi-supervised learning. In: Chapelle et al. [5], pp. 452–472
22. Zaki, M.J., Meira, W., Jr.: Data Mining and Analysis: Fundamental Concepts and Algorithms. Cambridge University Press, Cambridge (2014)
23. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: NIPS, pp. 321–328 (2003)
24. Zhou, D., Schölkopf, B.: Discrete regularization. In: Chapelle et al. [5], pp. 236–249
25. Zhu, X., Ghahramani, Z.: Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107. School of Computer Science, Carnegie Mellon University (2002)
26. Zhu, X., Ghahramani, Z., Lafferty, J.D.: Semi-supervised learning using Gaussian fields and harmonic functions. In: ICML, pp. 912–919 (2003)

27. Zhu, X., Goldberg, A.B.: Introduction to Semi-Supervised Learning. Morgan & Claypool Publishers, San Rafael (2009)
28. Zhu, X.J.: Semi-supervised learning literature survey. Technical Report. University of Wisconsin-Madison Department of Computer Sciences (2005)