Neurocomputing 75 (2012) 199–209

Contents lists available at SciVerse ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Label-dependent node classification in the network[☆]

Przemyslaw Kazienko, Tomasz Kajdanowicz[*]

Wroclaw University of Technology, Wybrzeze Wyspianskiego 27, Wroclaw, Poland

ARTICLE INFO

Available online 17 September 2011

Keywords:
Classification
Node classification
Label-dependent classification
Label-dependent features
Collective classification
Classification in networks
LDBootstrapping
LDGibbs
Bootstrapping
Gibbs sampling

ABSTRACT

Relations between objects in various systems, such as hyperlinks connecting web pages, citations of scientific papers, conversations via email or social interactions in Web 2.0 portals are commonly modeled by networks. One of many interesting problems currently studied for such domains is node classification. Due to the nature of the networked data and the unavailability of collection of nodes' broad representation for training in majority of environments, only a very limited data may remain useful for classification. Therefore, there is a need for accurate and efficient algorithms that are able to perform good classification based only on scanty knowledge of network nodes.

A new approach of sampling algorithm—LDGibbs, used in the context of collective classification with application of label-dependent features, is proposed in the paper in order to provide more accurate generalization for sparse datasets. Additionally, a new LDBootstrapping algorithm based on label-dependent features has been developed. Both new algorithms include additional steps to extract new input features based on graph structures but limited only to the nodes of a given label. It means that a separate set of structural features is provided for each label. The comparison with the other approaches, in particular with standard Gibbs Sampling and bootstrapping provided satisfactory results and revealed LDGibbs's superiority.

1. Introduction

Relations between objects in many various systems, such as hyperlinks connecting web pages, papers citations, conversations via email and social interaction in social portals are commonly modeled by networks. Those models are further a base for different types of processing and analyzes, in particular for node classification (labeling of the nodes in the network), analysis of network dynamics (especially discovery of network evolution) as well as knowledge and information diffusion. One of the most important directions in research on networks is node classification, which has a deep theoretical background. However, due to new phenomenon appearing in artificial environments like social networks on the Internet, the problem of node classification is being recently re-invented and re-implemented.

Classification of nodes in the network may be performed either by means of known profiles of these nodes (regular concept of classification) or based on information derived from known classes (labels) of the other nodes in the network. This second approach may be easily extended by information about connections between nodes (structure of the network) and can be very useful in assigning labels to the nodes being classified. For example, it is very likely that a given web page x is related to sport (label *sport*), if x is linked by many other web pages about sport. In the case of social network, somebody is likely to be recognized as highly-educated, if one possesses close friends (strong, direct relationships in the social network), who are also highly-educated.

Hence, a form of collective classification should be provided, with simultaneous decision making on every node's label [21,28,35] rather than classifying each node separately. Such approach allows taking into account correlations between connected nodes, which deliver usually undervalued knowledge.

To utilize specific profile of the network in order to improve accuracy of classification, three distinct types of correlations could be considered while classifying an unknown node x [33]:

1. the correlations between the label (class) of a given node x and the known attributes of x, i.e. x's profile (regular approach of classification requiring, however, some knowledge about x's profile),
2. the correlations between the label of node x and all known attributes of nodes in x's neighborhood; it includes also the known neighbors' labels,
3. the correlations between the label of x and the nodes with the unknown labels in the neighborhood of x; it may also include the attributes (profiles) of these nodes.

In general, a term *collective classification* refers to the combined classification of interlinked nodes using all three above mentioned types of information [33]. The third kind of correlations

[☆]This work was supported by The Polish Ministry of Science and Higher Education, the research project, 2011–12, the research project, 2010–13, and Fellowship co-financed by European Union within European Social Fund.

[*] Corresponding author.

E-mail addresses: kazienko@pwr.wroc.pl (P. Kazienko), tomasz.kajdanowicz@pwr.wroc.pl (T. Kajdanowicz).

0925-2312/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2011.04.047

appears to be the least valuable since it is based on information about nodes with unknown labels, with no separation for each class. Note that the neighborhood can be considered with respect to the nearest or distant neighbors or even with consideration of the entire network starting from node *x*. The investigations on the usability of the first type of features (*x*'s profile) used for classification compared to the structural features derived for the *x*'s neighbors can be found in [15]. In this paper, only one kind of information is really used: structural features for *x*'s neighborhoods with known labels, i.e. the second type of correlations from the above list. Additionally, these neighborhoods are separated for each of the class type. This is classification based on label-dependent features.

In general, it happens in many applications that the labels of nodes in the neighborhood are unknown for majority of cases. It simultaneously enforces to infer the labels for all these unknown nodes in the network. In such cases, the classification of a single node may not be trivial and is highly affected by the class membership of those very few nodes which labels are known in the network.

To overcome this problem, it has been widely recognized that the class membership of the nodes should be inferred collectively instead of individually. Collective classification methods are based on the homophile hypothesis, in which linked nodes have a tendency to belong to the similar class.

Features used as an input for a classifier can be constructed in many different ways. In the context of networks, it can be done using all information associated with nodes as well as the complementary, network data, extracted from the structure of nodes' neighborhoods [36]. This last, structural information introduces a kind of generalization into classification process. After having network data processed and assigned to nodes as new attributes, any kind of classification technique may be applied.

Overall, two types of structural features may be constructed to perform node classification: label-independent and label-dependent. The former is calculated for each node using general information about network structure, e.g. degree centrality measure for node in its entire neighborhood (with respect to all neighbors, with known labels of all types). The latter, in turn, is also calculated based on the network structure but additionally they make use of the additional knowledge of neighboring nodes, namely, the class of the known labels. It means that in the label-dependent approach, the structural metrics are calculated for networks restricted only to nodes with a given label, separately for each label class. Obviously, label-dependent feature extraction is more computationally expensive but it results with higher classification accuracy compared to classification based on the regular label-independent features [14–16].

Many collective classification algorithms for networks have recently appeared in the literature. Overall, they may be based on either label-dependent or label-independent features. The most typical representative of collective classification methods—Gibbs Sampling algorithm described in Section 3.4 makes use only of label-independent features.

In this paper, a new general method for label-dependent structural features extraction together with a new version of Gibbs Sampling algorithm, which is based on these label-dependent features, for node classification is introduced and presented in this paper. This new algorithm called LDGibbs is compared with its basis, i.e. the standard Gibbs Sampling version (GS) and with another typical classification method—bootstrapping.

The new concept of collective classification is presented in Section 3. In particular, the method of structural both label-dependent and label-independent feature extraction in networks is depicted in Section 3.1. The following Sections 3.2 and 3.4 provide the description of two standard algorithms: bootstrapping and Gibbs Sampling, respectively. In Section 3.5, some disadvantages of Gibbs Sampling algorithm are pointed, whereas a new LDBootstrapping algorithm and a new LDGibbs algorithm are introduced in Sections 3.3 and 3.6, respectively. All methods are compared with each other in Sections 4 and 5. Moreover, the extended evaluation of the algorithms' accuracy for different contributions of the known labels in the entire network is performed using both binary and multiclass datasets. The experimental results are gathered in Section 5 and concluded in Section 6.

2. Related work

It is a common case that important information about some nodes and links may be missing or just unknown in the network. The task of recovering this information, i.e. node's and edge's labels, based on the available data is known as network or collective classification and can be utilized in many application domains like image processing, classification of documents, in direct marketing, etc.

There are a number of approaches for collective classification, which, in general, can be divided into two distinct types. The first one makes use of a collection of unnormalized local conditional classifiers successively applied to the unknown nodes whereas in the second one, collective classification problem is defined as optimization of one global objective function.

In general, network classification problems may be solved using two main approaches: within-network [4] and across-network inference. Within-network classification, for which training nodes are connected directly to other nodes, whose labels are to be classified, stays in contrast to across-network classification, where models learnt from one network are applied to another similar network [21]. Collective inference methods solving within-network classification problem may be divided in two groups: exact and approximate inference methods. Exact inference methods attempt to learn the joint probability distribution of class membership (labels). Among the best known methods of this group are those using Markov Random Fields (MRF). The extensions to MRF that take into account the observed attribute data are Conditional Random Fields and Relational Markov Networks proposed in [4]. Exact inference methods are computationally expensive. This is the reason why approximation methods, like the Gibbs Sampling, are much more often used for the within-network classification [23,25,34].

In the past, many researchers worked in the area of iterative classification for the networked data. Two most popular methods can be enumerated in this domain: Iterative Classification Algorithm (ICA) and Gibbs Sampling (GS), introduced by Geman & Geman in the image processing context. Both of them belong to so-called approximate inference algorithms basing on local conditional classifiers [33]. There are also some modified GS algorithms such as Parameterized Gibbs Sampling for Collective Classification (PGCC). It is a generalization of GS for collective classification that introduces an additional parameter to control how uncertain relational information is used. Loopy belief propagation [29] and relaxation labeling [2] are other examples of iterative classification algorithms based on Gibbs Sampling.

It is fairly simple to plug in any local classifier into ICA or GS algorithms and the following are the implementation examples: Naive Bayes [27]; logistic regression [21]; decision trees [12]; logistic regression, weighted-vote relational neighbor, class distribution relational neighbor [22], *k*-nearest neighbors [24], etc. There are some evidences to indicate that some local classifiers tend to provide higher accuracies than global ones, at least in the application domains, for which experiments have been conducted.

Lu & Getoor concluded that logistic regression outperforms naive Bayes, what was evaluated on the bibliography datasets and web page classification problems [21].

Another very important question is, what features to use to maximize the classification accuracy [13]. The choice of the proper aggregation operator is also relevant to this question. However, note that the precise solution strongly depends on the application domain. Within the statistical relational learning community, Perlich and Provost have studied aggregation extensively [30,31]. The previous research showed that new attribute values derived from the graph structure of the network, such as the betweenness centrality, may be beneficial to the accuracy of the classification task [7]. It was also confirmed by other research discussed in [15,16].

An important topic related to feature construction and iterative classification is ordering strategy that determines, in which order to visit the nodes iteratively to re-label them. Please note that the ordering of nodes influences the values of input features that are derived from the structure, i.e. connections to the nodes known so far, including just classified. A variety of sophisticated or very simple algorithms can be used for this purpose. Random ordering that is one of the simplest ordering strategies used with iterative classification algorithms can be quite robust [18–20].

Yet another problem is to ensure the convergence of the method. One possible way for the Gibbs sampler is to settle to a final state by using a simulated annealing approach [9]. Neville and Jensen used a simulated annealing approach in their iterative classification collective inference procedure, where a label for a given node was kept only if the classifier was confident enough, i.e. the confidence level exceeded a given threshold. If not, the label was set to null [27]. By slowly lowering this threshold, the system was eventually able to label all nodes. Such algorithms are called cautious since they seek to identify and exploit the more certain information. Otherwise, they are called aggressive. Gibbs sampling from cautious inference point of view is examined by McDowell et al. [24].

Neville and Jensen used GS in Relational Dependency Networks to extract a joint distribution of a dependency network and to show that this is possible regardless of the consistency of the model. In GS, all nodes have classes assigned at every iteration that is why the relational models will always see a fully labeled neighborhood, making prediction straightforward [22].

3. Label-dependent and label-independent node classification in the network

We can define a network as a graph $G=(V,E,X,L,Y,A)$, where V is a set of nodes; E is a set of directed edges (connections) e_{ij} from node v_i to node v_j, $E=\{e_{ij} : v_i,v_j \in V, i \neq j\}$; X is a set of attribute vectors x_i, a separate one for each node v_i, $X=\{x_i : v_i \in V \Leftrightarrow x_i \in X\}$; L is the set of distinct labels (classes) that can be assigned to nodes; Y is a list of actual label assignments to nodes, $Y=\{\langle v_i,y_i \rangle : v_i \in V \wedge y_i \in L\}$; A is a set of edge weights, $\forall a_{ij} \in A$ $a_{ij} \geq 0$ and a_{ij} indicates the strength of edge e_{ij}.

Classification in the network may be described as the process of inferring the values of the unknown labels y_i for the set of nodes $V^U \subset V$, based on the values of other known labels y_i from the set of nodes $V^K \subset V$ and the structural features derived from the connections between of unknown and known nodes. Obviously, $V^U \cup V^K = V$ and $V^U \cap V^K = \emptyset$.

The entire node classification process can be divided into two main tasks. The first one is a translation of the network data into a set of unified vectors x_i, one vector per each node v_i. In our case, vector x_i contains information derived from the network structure by feature extraction method, see Section 3.1. The second task is classical supervised classification that uses the previously obtained set of vectors x_i. During the entire process, vectors x_i may change for some iterative methods since the new labels provided by classification may be used in the following iterations.

3.1. Features extraction

Feature extraction from networks is a general term for the methods of constructing variables from the structure of the graph. It can provide additional information about position and importance of each node with respect to other nodes. The generated features may be either label-independent or label-dependent [6]. In this paper, we focus mainly on label-dependent features. Two example label-dependent features: betweenness and degree are presented in Section 3.1.2, whereas a general method for their extraction is proposed in Section 3.1.3.

3.1.1. Label-independent features

Label-independent (LI) features are calculated using the network structure, but not attributes or class labels of nodes. Their goal is to reflect a potential dependency between the class label of node x_i and x_i's label-dependent features. For example, there may be a following relation between centrality of the node and its class: nodes with the top centrality values belong to class '0', while the least central nodes to class '1'.

To calculate LI features only network structure is needed, and no other information, especially about node labels is necessary. However, LI features usually do not provide enough insight into dependencies existing between the network structure and the node labels. For that reason, there is a need of additional information, other type of features that can help in classification.

3.1.2. Label-dependent features

Label-dependent (LD) features incorporate both the structure of network and labels (or attributes) of the nodes. They are calculated as regular network measures for nodes but within a graph that consist of only nodes of the certain label (that is why we call them label-dependent). For that reason, LD features are calculated separately for each of the possible label. As a result, for a single structural measure, we have as many LD features as distinct labels exist: *card*(L).

Label-dependent features can be used as valuable sources of information about nodes' relations to train the classifier. There are plenty of structural metrics for nodes that can be used as label-dependent features. In this paper, only betweenness centrality and degree centrality are used as examples of LD features.

3.1.2.1. Label-dependent betweenness centrality. Label-dependent betweenness centrality of node v_i for label $l \in L$ pinpoints to what extent v_i is between other nodes with label l. Nodes with the high betweenness value are very important in the network as other nodes are connected with each other mainly through them. Label-dependent betweenness centrality $B(G_{li},v_i)$ of node v_i in graph G_{li} is the contribution of shortest paths between any pair of nodes in graph G_{li} passing via node v_i by the total number of shortest paths between any two nodes in this graph. Graph G_{li} is a subgraph of graph G that consists only of nodes having label l plus node v_i (set V_{li}) and all connections $e_{ij} \in E$ between nodes from V_{li}. Betweenness centrality $B(G_{li},v_i)$ is expressed as follows:

$$B(G_{li},v_i) = \sum_{v_j,v_k \in V_{li}; j \neq k \neq i} \frac{P(G_{li},v_j,v_k,v_i)}{P(G_{li},v_j,v_k)}, \qquad (1)$$

where $P(G_{li},v_j,v_k,v_i)$—a function that returns the number of shortest paths between v_j and v_k that pass through node v_i in graph G_{li}; $P(G_{li},v_j,v_k)$—a function returning the number of shortest paths between v_j and v_k in graph G_{li}.

Depending on the kind of connections label-dependent betweenness centrality can be computed either for directed paths or undirected paths. For the directed connections shortest paths from v_j to v_k may be different than from v_k to v_j. Moreover, in opposite to undirected graphs, it is more likely for directed graphs that there are no paths at all for some pairs of nodes.

3.1.2.2. Label-dependent degree centrality. Label-dependent degree centrality is defined as the number of nodes with the given label l connected to a given node v_i, i.e. number of edges incident with v_i. It is the simplest and most intuitive measure that can be used in the network analysis. Nodes with the high label-dependent degree centrality are recognized in the network as very important from the currently considered label's point of view. Degree centrality $D(G_{li},v_i)$ of node v_i in graph G_{li} can be computed with the following formula:

$$D(G_{li},v_i) = \frac{card(NN(G_{li},v_i))}{card(V)-1}, \tag{2}$$

where $NN(G_{li},v_i)$—a set of neighboring nodes of node v_i in graph G_{li}.

For the directed connections, we can considered separately in-degree centrality and out-degree centrality measures, which respect the number of edges incoming to or outgoing from the given node v_i, respectively.

3.1.3. General method for label-dependent features extraction

Network nodes can be characterized by a wide range of different structural measures, which were extensively studied in the literature. Most of them have been used in the label-independent context (regardless of labels of the nodes). However, the same measures can be utilized to extract label-dependent features. A general concept of creation of any label-dependent feature $M_l(G,l,v_i)$ for label l and node v_i in the network G applies label-independent feature (measure) M to the appropriate labeled sub-network G_{li}, as follows:

$$M_l(G,l,v_i) = M(G_{li},v_i), \tag{3}$$

where $M(G_{li},v_i)$—denotes any structural network measure for node v_i applied to sub-network G_{li}, e.g. degree, betweenness, clustering coefficient, prestige, closeness, density, node position [26], etc. G_{li} is a sub-graph obtained from graph G by removal of all nodes without label l except v_i together with connections of these nodes. In other words, G_{li} consists of set V_{li}, which contains nodes with label l and node v_i, and the set of all edges from G linking nodes from V_{li}. Obviously, $M_l(G,l,v_i)$ is computed separately for each label l using the appropriate sub-network G_{li}.

3.2. Node classification using bootstrapping

The bootstrapping method [33] is one of the most basic classification approaches utilized for networks. It is based on supervised classification without application of iterative approach. Each node is classified only once using information about node's known neighborhood (known labels). According to Algorithm 1 each network node v_i from V has a number of attributes computed within the whole graph G. Having features obtained for all nodes, classifier Φ is trained, i.e. its Θ parameters are optimized upon known attributes of nodes from the learning set V^K. Finally, each node with unknown label $v_i \in V^U$ is classified by means of classifier Φ.

Algorithm 1. Bootstrapping.

```
1: for each node $v_i \in V$ do
2:     compute $x_i$, i.e. $v_i$'s attributes using whole $G$
3: end for
4: train classifier $\Phi$ by $\Theta$ optimization using $V^K$
5: for each node $v_i \in V^U$ do
6:     $y_i \leftarrow \Phi(x_i,\Theta)$
7: end for
```

3.3. Node classification using LDBootstrapping

The label dependent bootstrapping method is an extension of standard bootstrapping classification approach utilized for networks. Similarly, it is based on supervised classification without application of iterative approach, however the classification is performed using richer information extracted from the network—label dependent attributes. As presented in Algorithm 2 each network node v_i from V has a number of LD attributes computed for the sub-graph G_{li} restricted to the nodes with label l and node v_i itself. These attributes are calculated separately for each label $l \in L$. Having features obtained for all nodes, classifier Φ is trained, i.e. its Θ parameters are optimized upon known attributes (LD features and labels) of nodes from the learning set V^K. Finally, each node with unknown label $v_i \in V^U$ is classified by means of classifier Φ.

Algorithm 2. LDBootstrapping.

```
1: for each node $v_i \in V$ do
2:     compute $x_i$, i.e. $v_i$'s LD attributes using $G_{li}$ separately for
   each label $l$
3: end for
4: train classifier $\Phi$ by $\Theta$ optimization using $V^K$
5: for each node $v_i \in V^U$ do
6:     $y_i \leftarrow \Phi(x_i,\Theta)$
7: end for
```

3.4. Gibbs sampling algorithm

Gibbs Sampling (GS) is utilized in the collective classification context with assumption that a local classifier is trained using initial network data and is accessible in iterative label settlement process. Having all the attribute values for nodes in the network, the classifier is used to estimate the conditional probability distribution of the labels for nodes. More about the need of introducing this assumption is related to further issues with traditional GS studied in [22,24]. The conditional probability distribution given by the local classifier therefore is an approximation of the true conditional probability distribution. More discussion and justification for this approach is provided by Neville and Jensen [28] where they used a similar form of GS for inference. It is worth mentioning that most of the inference algorithms available for practical tasks relating to collective classification are approximate [33]. Gibbs Sampling may be interpreted as a semi-supervised learning technique [11].

In Algorithm 3, there are four main steps performed: bootstrapping, burn-in, samples collection and final computation of labels. In the bootstrapping stage, the classifier Φ is trained by optimization of its Θ parameters. Burn-in step performs s iterations, in each one, classification of all nodes is executed. Iterations are performed according to generated ordering O. It is common case that random ordering is being chosen as O. Attributes of nodes with unknown labels V^U in each iteration are updated. Note that the recalculation of features (line 13) is made on the updated graph, i.e. on $G^{(n)}$.

The third part of the algorithm is a collection of classification results for each node. Labels for each $v_i \in O$ are sampled as well as counted, i.e. we obtain information about how many times label l was sampled for node v_i. After collecting a number of samples and fulfilling a stop criterion, labels y_i appearing mostly in sampling are assigned separately to each of the nodes $v_i \in V^U$. Maximum label count is a usual criterion of label assignment.

Algorithm 3. Gibbs sampling algorithm, the idea based on [33].

```
1: //bootstrapping
2: for each node $v_i \in V$ do
3:    compute $x_i$, i.e. $v_i$'s attributes using $G$
4: end for
5: train classifier $\Phi$ by $\Theta$ optimization using $V^K$
6: for each node $v_i \in V^U$ do
7:    $y_i \leftarrow \Phi(x_i,\Theta)$
8: end for
9: //burn-in
10: for $n=1$ to s do
11:    generate ordering $O$ over nodes in $V^U$
12:    for each node $v_i \in O$ do
13:       compute $x_i$ attributes using $G^{(n)}$
14:       $y_i \leftarrow \Phi(x_i,\Theta)$
15:    end for
16: end for
17: //initialize sample counts
18: for each node $v_i \in V$ do
19:    for label $l \in L$ do
20:       $c[i,l]=0$
21:    end for
22: end for
23: //collect samples
24: repeat
25:    generate ordering $O$ over nodes in $V^U$
26:    for each node $v_i \in O$ do
27:       compute $x_i$'s attributes using the updated $G$
28:       $y_i \leftarrow \Phi(x_i,\Theta)$
29:       $c[i,l] \leftarrow c[i,l]+1$
30:    end for
31: until stop condition
32: //compute final labels
33: for each node $v_i \in O$ do
34:    $y_i \leftarrow \mathrm{argmax}_{l \in L} c[i,l]$
35: end for
```

3.5. Gibbs sampling possible improvement areas

Considering possible improvement areas of Gibbs Sampling algorithm, four of them may be distinguished: aggregation, ordering, feature selection and classification. As the GS algorithm has the iterative nature and it requires the usage of network's structural features, some aggregation operators need to be defined in order to extract features' values from the structure of the network. In the past, variety aggregation operators were used, including count, mode, proportion, minimum, maximum and exists. Table 1 lists aggregation operators used in different researches.

Table 1
Aggregation operators appearing in the literature [33].

Reference	Operator
Friedman et al. [5]	Mode, count, SQL
Taskar et al. [35]	Mode, SQL
Richardson and Domingos [32]	FOL
Lu and Getoor [21]	Mode, count, exists
Macskassy and Provost [22]	Prop
Gupta et al. [8]	Mode, count
Gupta, Diwan, & Sarawagi [24]	Prop

Another important area that could be taken into account while improving GS accuracy is the ordering strategy, which determines the order in which nodes are classified in burn-in and samples collection step. Introducing new ordering strategies can be beneficial from classification accuracy point of view. However, there is some evidence that simple random ordering can give satisfactory results.

Also usage of a proper classifier may be crucial for highly accurate results but this choice strongly depends on profile of the data being examined. In many cases, the tradeoff between the higher accuracy and speed of algorithm processing has to be made. In Table 2 a list of classification algorithms already utilized and mentioned in the literature is presented.

Standard Gibbs Sampling approach is utilizing features derived from whole network's structure separately for each node (data instance). Other possible improvement is to provide fair feature extraction method, generating discriminative description of learning data, based on labeled sub-graphs—label dependent features. We address that problem in the proposal of LDGibbs algorithm.

3.6. LDGibbs algorithm

Algorithm 4. LDGibbs classification algorithm.

```
1: //bootstrapping
2: for each node $v_i \in V$ do
3:    compute $x_i$, i.e. $v_i$'s LD attributes using $G_{li}$ separately for each label $l$
4: end for
5: train classifier $\Phi$ by $\Theta$ optimization using $V^K$
6: for each node $v_i \in V^U$ do
7:    $y_i \leftarrow \Phi(x_i,\Theta)$
8: end for
9: //burn-in
10: for $n=1$ to $s$
11:    train classifier $\Phi$ by $\Theta$ optimization using $V^K$
12:    generate ordering $O$ over nodes in $V^U$
13:    for each node $v_i \in O$
14:       recompute $x_i$ (LD attributes) for the new $G_{li}^{(n)}$ for each label $l$
15:       $y_i \leftarrow \Phi(x_i,\Theta)$
16:    end for
17: end for
18: //initialize sample counts
19: for each node $v_i \in V$ do
20:    for label $l \in L$ do
21:       $c[i,l]=0$
22:    end for
23: end for
24: //collect samples
25: repeat
26:    train classifier $\Phi$ by $\Theta$ optimization using $V^K$
27:    generate ordering $O$ over nodes in $V^U$
28: for each node $v_i \in O$ do
29:       compute $x_i$'s, i.e. LD attributes using the updated $G_{li}$
30:       $y_i \leftarrow \Phi(x_i,\Theta)$
31:       $c[i,l] \leftarrow c[i,l]+1$
32:    end for
33: until stop condition
34: //compute final labels
35: for each node $v_i \in O$ do
36:    $y_i \leftarrow \mathrm{argmax}_{l \in L} c[i,l]$
37: end for
```

Table 2
Local classifiers used in collective classification reported in the literature [33].

Reference	Local classifier
Neville and Jensen [27]	Naïve Bayes
Lu and Getoor [21]	Logistic regression
Jensen et al. [12]	Naïve Bayes decision trees
Macskassy and Provost [22]	Naïve Bayes logistic regression, weighted-vote relational neighbor, class distribution relational neighbor
McDowell et al. [24]	Naïve Bayes, *k*-nearest neighbors

Table 3
An example of several iterations of the LDGibbs algorithm.

Node	Initial label	Label after iteration		
		1	...	*n*th
1	○	○	...	○
2	•	•	...	•
3	○	○	...	○
4	•	•	...	•
5	○	○	...	○
6	•	•	...	•
7	?	•	...	•
8	?	○	...	○
9	?	•	...	○
10	?	○	...	○
11	?	•	...	•
12	?	○	...	○
13	?	•	...	○
14	?	○	...	•

A new perspective on feature extraction allows to formulate LDGibss algorithm that depicts a classification approach which addresses the problem of label dependent feature extraction. As presented in Algorithm 4, the learning of classifier Φ is performed using features obtained from sub-graphs, composed of nodes labeled with the same labels l and edges between them. The number of distinct values of labels multiplied by the number of structural measures computed in each sub-graph states the number of extracted features for each node.

Label dependent features are calculated in the bootstrapping phase as well as in burn-in and samples collecting phases. Initial learning of the classifier takes into account only those nodes for which labels are known. Then, in the burn-in and samples collecting phases, nodes with initially unknown labels have some labels iteratively assigned. Afterward, the label-dependent features are recalculated for all the nodes in the network.

An example of several iterations of LDGibbs algorithm is presented in Table 3. It directly corresponds the case depicted in Figs. 1–3. Nodes 1–6 have their labels known (training set) and we want to assign proper labels to cases 7–14, Fig. 1, the second column in Table 3. Thus, the contribution of known nodes is 50%. After the first iteration of the burn-in phase (lines 10–17), the preliminary labels are assigned to the unknown nodes no. 7–14, Fig. 2, the third column in Table 3. Next, new label-dependent structural attributes are recomputed and as a result, labels of the initially unknown nodes no. 7–14 are reassigned according to the new knowledge gathered inside the model Φ. After the final iteration, the labels are settled, Fig. 3, the last column of Table 3. Afterward, the assigned labels may a bit change in the collect samples phase (line 30).

In comparison with standard GS algorithm, LDGibbs is extended by label dependent feature extraction providing additional information of each of nodes and supporting learning of base classifier in burn-in phase. Owing to this extension, the LDGibbs algorithm is better prepared especially for node classification with the very limited data for training.

Proposed enhancement and changes reveal significant accuracy improvement experimentally verified in the following sections.

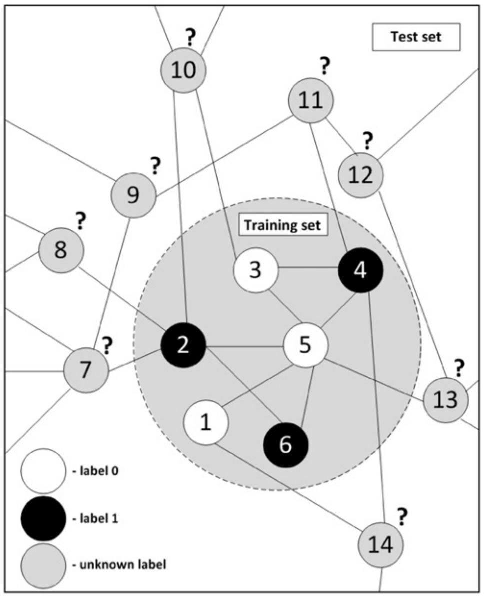

Fig. 1. Initial state of the dataset utilized in the bootstrapping phase of the LDGibbs algorithm.

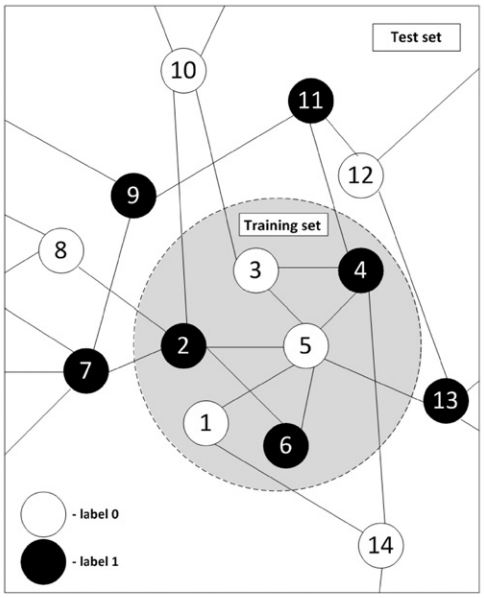

Fig. 2. The state of the dataset after the first iteration of the LDGibbs algorithm in the burn-in phase; all nodes with the unknown labels have been assigned with some preliminary labels.

4. Experimental setup

To create the test bed for proposed classification approach the experiment consisting of tests on six distinct datasets was designed in which the accuracy of LDGibbs, Standard Gibbs, LDBootstrapping and Bootstrapping methods was examined.

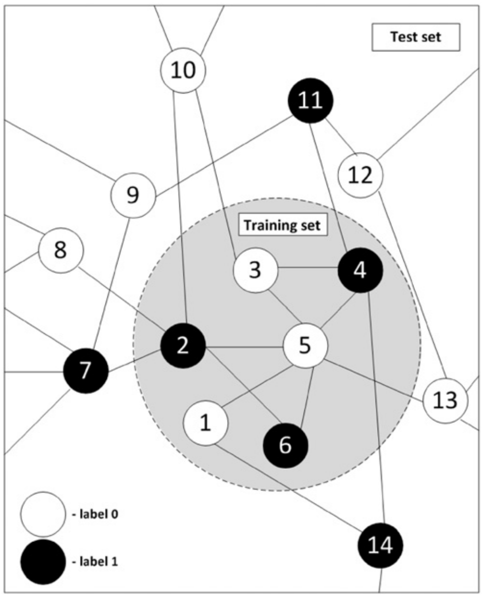

Fig. 3. The final state of the dataset after the termination of the burn-in phase of the LDGibbs algorithm (after the *n*th iteration); all labels of nodes with the initially unknown labels have been settled.

4.1. Datasets

Six distinct datasets derived from "Attendee Meta-Data" (AMD) were used in experiments. The source is available at UCI Network Data Repository (http://networkdata.ics.uci.edu/data.php?d=amdhope). The AMD dataset was a result of a project that took place during "The Last HOPE" Conference held in July 18–20, 2008, New York City, USA. At this conference RFID (Radio Frequency Identification) devices were distributed among participants and allowed to uniquely identify them and track in which sessions they attended. The dataset is build from information about descriptions of interests of participants, their interactions via instant messages, as well as their location over the course of the conference. Location tracking allowed to extract a list of attendances for each conference talk. In general, the most interesting for experiment information included in dataset are: information about conference participants, conference talks and presence on talks. Dataset consists of information about 767 different persons, 99 talks and 10,110 presences reported during talks.

In the experiment cleaning process was performed, excluding those participants who did not gave any information about their interests. After that 334 persons, 99 talks and 3141 presences have left. Afterward, a social network was built based on extracted data—ties in the network were constructed using the fact that attendees had participated in the same talks. The commitment function for ties between all participants v_i and v_j, $i \neq j$, was calculated as a fraction of number of v_i's and v_j's collective participations in talks to number of talks v_i has been to. As a result network had 10,738 connections.

Five of extracted datasets represented a problem with binary target interest of each participant, the sixth was a multiclass assignment from eight distinct classes. The number of extracted datasets was caused by necessity of reasonable low class imbalance in each of datasets. Following datasets were chosen for examination (AMD_1, AMD_2, AMD_3, AMD_4, AMD_5 and AMD_multiclass). The proportion between classes in binary datasets was 43% to 57%, 38% to 62%, 43% to 57%, 38% to 62% and 41% to 59%, respectively. The multiclass dataset had the following class distribution: 19%, 17%, 15%, 7%, 17%, 10%, 10%, 3%, for classes 1–8 respectively.

The basis for classification constituted only two general attributes computed as label-independent and label-dependent: betweenness and degree. Those attributes were chosen as well discriminating in dataset under research based on previous results from commitment function and procedure described in [14].

4.2. Evaluation procedure

The experiment consisted of a set of tests performed three times on each of datasets, each time testing the classification accuracy. Each test set assumed the examination of method on distinct training set. To perform the experiment each dataset was split randomly into 10 parts of equal (or approximately equal) size reflecting the class distribution of initial dataset. Methods were trained with training set built of 1, 2, 3, 4, 5, 6, 7, 8, 9 previously obtained parts and tested on the remaining ones using cross validation. For each fold, appropriate number of the instances (from 10% to 90%) were placed in the training set, and the rest were placed in the corresponding test. The same splits of the data were used for all examined methods.

What is worth mentioning the structure of network composed of datasets used in experiments does not imply the sampling procedure in which the training set is obtained. It reflects the nature of real-world classification problem performed in the network where the distribution of knowledge about nodes is random. The example of the random composition of training set is presented in Figs. 4–6 (drawn according to [17]). However, there may exist such a situation, where knowledge of nodes' labels is distributed in very specific way, e.g. collected labels are placed within clicks or particular groups of nodes. It has not been addressed in performed experiments.

Four methods for node classification were tested in the experiments: Bootstrapping, LDBootstrapping, Gibbs and LDGibbs. As seen in Algorithms 3 and 4 the burn-in phase may terminate after s iterations. In experiments s was set to 50. Moreover LDGibbs (Algorithm 4) contains other stopping criterion in collecting samples phase that may terminate execution. It was fulfilled when the average relative classification accuracy of the method did not exceed 5% between iteration for last five iterations, namely in nth iteration expression:

$$\frac{(accuracy(i) - accuracy(i-1))}{accuracy(i-1)} \leq 5\%$$

holds for $i \in \{n-4, n-3, n-2, n-1, n\}$, $n > 4$, $n \in \mathbb{N}$.

As Φ base classifier Neural Network classifier was arbitrary chosen. It was configured with parameters presented in Table 4.

The classification accuracy used as a base assessment measure in all the experiments was the proportion of true results (both true positives and true negatives) in the population. The accuracy of all four tested methods was examined for six datasets in binary

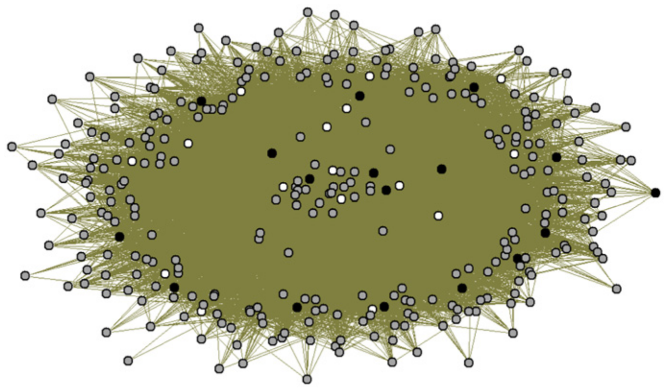

Fig. 4. An example of the social network with 10% of nodes with known labels (class 0—white circle, class 1—black circle, unknown class—gray circle).

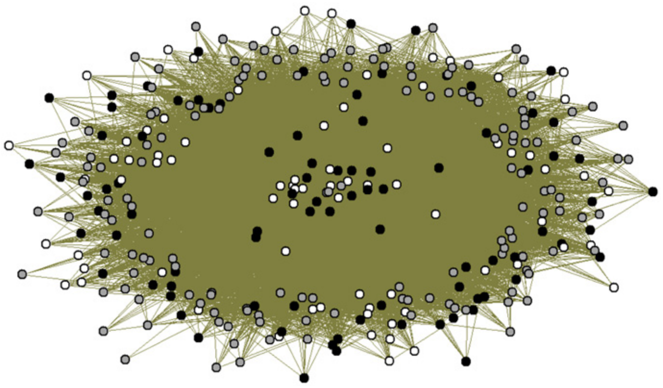
Fig. 5. An example of the social network with 50% of nodes with known labels (class 0—white circle, class 1—black circle, unknown class—gray circle).

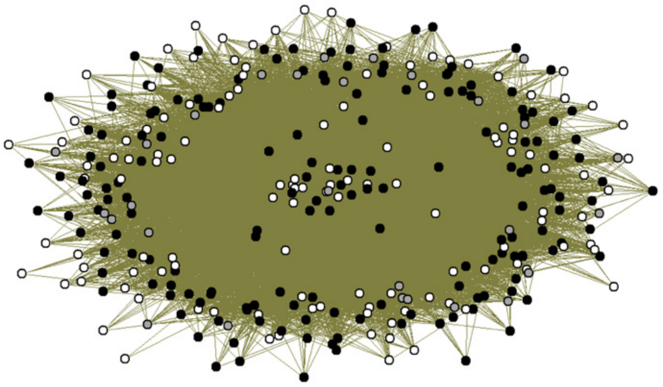
Fig. 6. An example of the social network with 90% of nodes with known labels (class 0—white circle, class 1—black circle, unknown class—gray circle).

Table 4
Neural network parameters used in the experiments.

Parameter	Value
Learning rate for the backpropagation algorithm	0.3
Momentum rate for the backpropagation algorithm	0.2
Number of epochs in training	500
Number of hidden layers	2

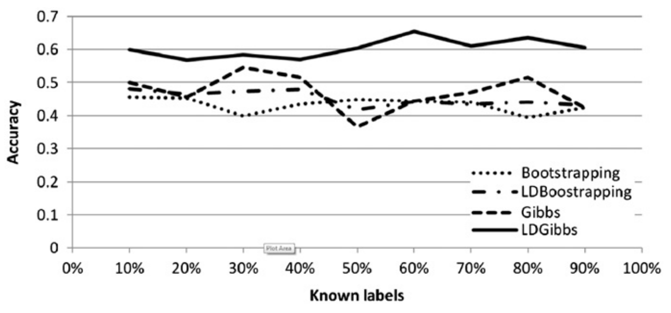

Fig. 7. Classification accuracy of Bootstrapping, LDBootstrapping, Gibbs and LDGibbs methods for distinct level of known labels on AMD_1 dataset.

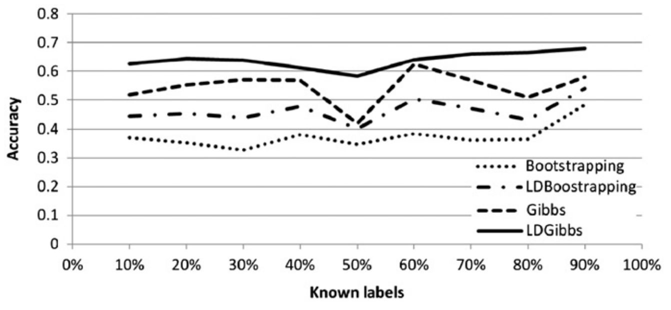

Fig. 8. Classification accuracy of Bootstrapping, LDBootstrapping, Gibbs and LDGibbs methods for distinct level of known labels on AMD_2 dataset.

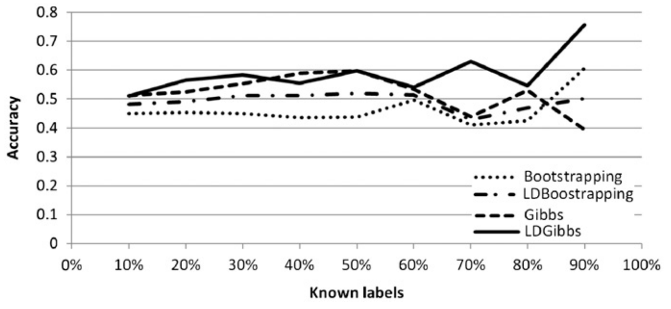

Fig. 9. Classification accuracy of Bootstrapping, LDBootstrapping, Gibbs and LDGibbs methods for distinct level of known labels on AMD_3 dataset.

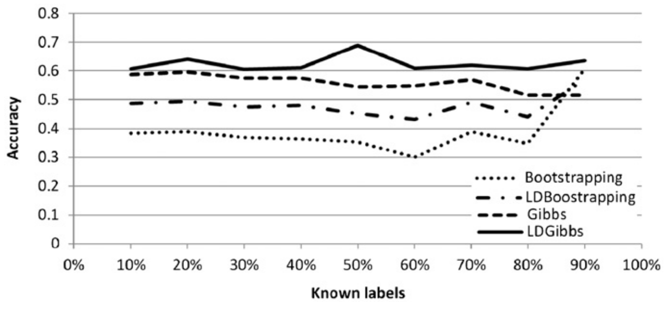

Fig. 10. Classification accuracy of Bootstrapping, LDBootstrapping, Gibbs and LDGibbs methods for distinct level of known labels on AMD_4 dataset.

and multiclass problem. According to method's performance the ranking was composed and summarized by Friedman's test.

5. Experimental results and discussion

The obtained results for binary classification have revealed that the classification accuracy using label-dependent features for the tested approaches depends on the level of known nodes, utilized in the training data (Figs. 7–11, Tables 5–9). It may be detected that usually the more known nodes the better accuracy. The best result ever (accuracy=75.8%) was obtained for dataset ADM_3, contribution of known nodes at the level of 90% and LDGibbs algorithm, see Table 7, whereas the worst accuracy (only 30%) was for the dataset AMD_4, 60% of known labels, and the bootstrapping algorithm. Similar regularities can be discovered for the multiclass problem, see Fig. 12 and Table 11.

However, the trend of better results for greater contribution of known labels is not always valid and strict. It comes from the nature of experiments: the known nodes were randomly selected from the entire node set and selection was repeated three times (the average is presented in figures). Hence, we can assume that the distribution of classes and label-dependent features for known and unknown nodes do not significantly differ. However, in real application, the known nodes are not equally distributed over the network and we could expect better accuracy for higher contribution of known nodes [10].

Based on the results for binary classification, we can state that label-dependent features provide better knowledge than typical label-independent ones, when used as the input for collective

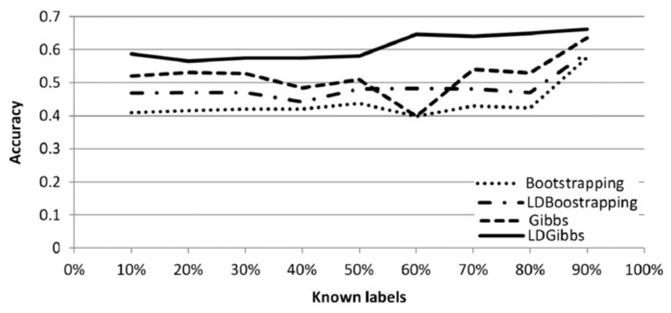

Fig. 11. Classification accuracy of Bootstrapping, LDBootstrapping, Gibbs and LDGibbs methods for distinct level of known labels on AMD_5 dataset.

Table 5
Classification accuracy obtained for Bootstrapping, LDBootstrapping, Gibbs Sampling and LDGibbs methods with distinct size of training set on AMD_1 dataset.

Known labels (%)	Bootstr.	LDBootstr.	Gibbs	LDGibbs
10	0.4567	0.481	0.5000	0.6020
20	0.4532	0.465	0.4569	0.5693
30	0.3991	0.473	0.5451	0.5837
40	0.4350	0.479	0.5150	0.5700
50	0.4491	0.417	0.3653	0.6048
60	0.4436	0.442	0.4436	0.6541
70	0.4400	0.435	0.4700	0.6100
80	0.3939	0.441	0.5152	0.6364
90	0.4242	0.432	0.4242	0.6061

Table 6
Classification accuracy obtained for Bootstrapping, LDBootstrapping, Gibbs Sampling and LDGibbs methods with distinct size of training set on AMD_2 dataset.

Known labels (%)	Bootstr.	LDBootstr.	Gibbs	LDGibbs
10	0.3700	0.445	0.5200	0.6267
20	0.3521	0.454	0.5543	0.6442
30	0.3262	0.439	0.5708	0.6395
40	0.3800	0.479	0.5700	0.6130
50	0.3473	0.401	0.4192	0.5840
60	0.3835	0.503	0.6241	0.6400
70	0.3600	0.471	0.5700	0.6600
80	0.3636	0.432	0.5100	0.6667
90	0.4848	0.541	0.5800	0.6800

Table 7
Classification accuracy obtained for Bootstrapping, LDBootstrapping, Gibbs Sampling and LDGibbs methods with distinct size of training set on AMD_3 dataset.

Known labels (%)	Bootstr.	LDBootstr.	Gibbs	LDGibbs
10	0.4500	0.4820	0.5133	0.5100
20	0.4532	0.4910	0.5243	0.5655
30	0.4506	0.5120	0.5536	0.5837
40	0.4350	0.5125	0.5900	0.5550
50	0.4371	0.5190	0.5988	0.5988
60	0.4962	0.5150	0.5338	0.5414
70	0.4100	0.4290	0.4400	0.6300
80	0.4242	0.4690	0.5303	0.5455
90	0.6061	0.5010	0.3939	0.7576

Table 8
Classification accuracy obtained for Bootstrapping, LDBootstrapping, Gibbs Sampling and LDGibbs methods with distinct size of training set on AMD_4 dataset.

Known labels (%)	Bootstr.	LDBootstr.	Gibbs	LDGibbs
10	0.3833	0.488	0.5867	0.6067
20	0.3895	0.494	0.5955	0.6404
30	0.3691	0.475	0.5751	0.6052
40	0.3650	0.481	0.5750	0.6100
50	0.3533	0.451	0.5449	0.6886
60	0.3008	0.432	0.5489	0.6090
70	0.3900	0.491	0.5700	0.6200
80	0.3485	0.442	0.5152	0.6061
90	0.6061	0.569	0.5152	0.6364

Table 9
Classification accuracy obtained for Bootstrapping, LDBootstrapping, Gibbs Sampling and LDGibbs methods with distinct size of training set on AMD_5 dataset.

Known labels (%)	Bootstr.	LDBootstr.	Gibbs	LDGibbs
10	0.4100	0.468	0.5200	0.5867
20	0.4157	0.471	0.5318	0.5655
30	0.4206	0.471	0.5279	0.5751
40	0.4200	0.442	0.4850	0.5750
50	0.4371	0.481	0.5090	0.5808
60	0.3985	0.482	0.3985	0.6466
70	0.4300	0.481	0.5400	0.6400
80	0.4242	0.471	0.5303	0.6500
90	0.5758	0.591	0.6364	0.6620

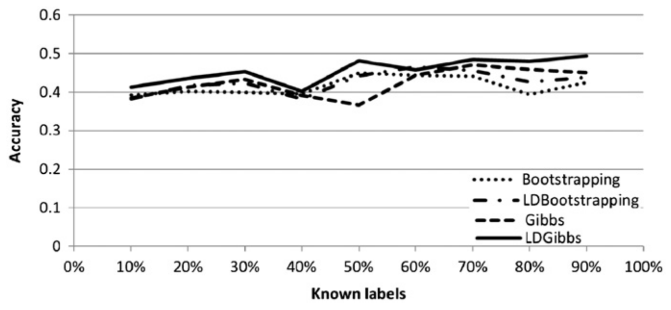

Fig. 12. Classification accuracy of Bootstrapping, LDBootstrapping, Gibbs and LDGibbs methods for distinct level of known labels on AMD_multiclass dataset (multiclass problem).

classification. As a result, almost always, LDBootstrapping is better than regular bootstrapping algorithm; it gains, in average: 4.5% for ADM_1, 23% for ADM_2, 6.5% for ADM_3, 23% for ADM_4, 10% for ADM_5, and 2% for ADM_6 compared to the independent approach. If we compared LDGibbs with Gibbs Sampling (Gibbs) in the similar way, we would observe that label-dependent method would outperform label-independent ones with 28%, 17%, 13%, 11%, 17%, and 7%. It means that usage of label-dependent structural features is almost always significantly better than the regular classification based on label-dependent measures.

The experimental result also revealed that bootstrapping is worse than Gibbs Sampling. It refers both their label-dependent and label-independent versions.

Analysis of results for the multiclass dataset confirms the general rules already discovered for binary classification, see Fig. 12 and Table 11. However, the differences between individual methods are smaller but the trends remain the same.

One of the most important conclusions drawn from the experimental studies is that the new LDGibbs algorithm is almost always the best collective classification algorithm among all considered (except only one case: AMD_3 and 40% of known labels). It usually surpasses Gibbs Sampling by a dozen or so percent and several dozen percent the bootstrapping.

In purpose of more precise statistical algorithm comparison, the average performance rank of each method was determined as follows: the method with the highest classification accuracy held the first position, while the method with the lowest accuracy held

Table 10
Method's ranking based on arithmetic test accuracy for binary classification problem.

Dataset	AMD_1	AMD_2	AMD_3
Bootstrapping	3.33 ± 0.82	4 ± 0	3.78 ± 0.63
LDBoostrapping	3.00 ± 0.67	3 ± 0.00	3 ± 0.00
Gibbs sampling	2.44 ± 0.68	2.00 ± 0.00	1.89 ± 0.87
LDGibbs	1.00 ± 0.00	1.00 ± 0.00	1.22 ± 0.42
Friedman	0.003	0	0.002

Dataset	AMD_4	AMD_5
Bootstrapping	3.78 ± 0.63	3.89 ± 0.31
LDBoostrapping	3 ± 0.00	2.89 ± 0.31
Gibbs sampling	2.22 ± 0.63	2.11 ± 0.31
LDGibbs	1.00 ± 0.00	1.00 ± 0.00
Friedman	0.000046	0.000015

Table 11
Classification accuracy obtained for Bootstrapping, LDBootstrapping, Gibbs and LDGibbs methods with distinct size of training set on AMD_multiclass dataset (multiclass problem).

Known labels (%)	Bootstr.	LDBootstr.	Gibbs	LDGibbs
10	0.3920	0.3810	0.3840	0.412
20	0.4020	0.4170	0.4120	0.435
30	0.3990	0.4231	0.4320	0.452
40	0.3950	0.3810	0.3910	0.402
50	0.4491	0.4412	0.3653	0.481
60	0.4436	0.4650	0.4436	0.456
70	0.4400	0.4550	0.4700	0.485
80	0.3939	0.4259	0.4580	0.479
90	0.4242	0.4380	0.4500	0.494

the last, fourth position in the ranking. Table 10 summarizes *p*-values returned by Friedman's procedure and the standard deviation of the rank obtained by each method.

In binary classification problem, the Friedman's test allows to reject the null hypothesis that all the methods performed the same on average at $\alpha = 0.05$ for all datasets AMD_1 to AMD_5, see Table 10. The result values of Friedman's test are actually very small and therefore, LDGibbs may be considered as a significantly different method from others and according to its results—the best among all methods evaluated.

6. Conclusions

Two new approaches to collective classification were proposed in the paper: a new sampling algorithm called LDGibbs and LDBootstrapping. Both were used in the context of relational dependency networks with application of label-dependent structural features. The general idea behind the new algorithms was to improve accuracy of node classification by introducing extended feature extraction technique, namely label-dependent features derived from the structure of the network limited to nodes of the individual labels.

Evaluation of the accuracy for the proposed LDGibbs and LDBootstrapping methods was performed with respect to different contribution of known nodes in the training data. Additionally, the results were compared to values obtained from the standard Gibbs Sampling algorithm as well as to the non-iterative classification. The outcome of the experimental studies was satisfactory and provided LDGibbs's superiority among all algorithms.

What is also important, all features in the algorithms based on the Gibbs scheme should be chosen carefully as their recalculation from iteratively changing label-dependent network's structure appears to be very time consuming. Additionally, the effectiveness and efficiency of final classification significantly depend on the selected base classifier, what also should be taken into consideration.

Collective classification has been a topic of active research for the past years but still a lot can be done in this area. In particular, there is a great need for robust and efficient classification algorithms for network structures, especially in hybrid artificial intelligence systems [1,3].

References

[1] A. Abraham, E. Corchado, J. Corchado, Hybrid learning machines, Neurocomputing 72 (2009) 2729–2730.
[2] S. Chakrabarti, B. Dom, P. Indyk, Enhanced hypertext categorization using hyperlinks, in: The Proceedings of the International Conference on Management of Data SIGMOD Conference, 1998, pp. 307–318.
[3] E. Corchado, A. Abraham, A.P.L.F. de Carvalho, Hybrid intelligent algorithms and applications, Inf. Sci. 180 (2010) 2633–2634.
[4] C. Desrosiers, G. Karypis, Within-network classification using local structure similarity, in: Lecture Notes in Computer Science, vol. 5781, 2009, pp. 260–275.
[5] N. Friedman, L. Getoor, D. Koller, A. Pfeffer, Learning probabilistic relational models, in: Proceedings of The International Joint Conference on Artificial Intelligence IJCAI-99, 1999, pp. 1300–1309.
[6] B. Gallagher, T. Eliassi-Rad, Leveraging label-independent features for classification in sparsely labeled networks: an empirical study, in: Proceedings of the Second ACM SIGKDD Workshop on Social Network Mining and Analysis, SNA-KDD'08, 2008.
[7] B. Gallagher, T. Eliassi-Rad, Leveraging Network Structure to Infer Missing Values in Relational Data, Lawrence Livermore National Laboratory Technical Report UCRL-TR-231993, 2007.
[8] R. Gupta, A.A. Diwan, S. Sarawagi, Efficient inference with cardinality-based clique potentials, in: Proceedings of the 24th International Conference on Machine Learning, 2007, pp. 329–336.
[9] S. Geman, D. Geman, Stochastic relaxation Gibbs distributions and the Bayesian restoration of images, IEEE Trans. Pattern Anal. Mach. Intell. 6 (1984) 721–741.
[10] L. Getoor, Link-based classification, in: Advanced Methods for Knowledge Discovery from Complex Data, Springer-Verlag, 2005, pp. 184–207.
[11] J. Jung, An evolutionary approach to query sampling for heterogeneous systems, Expert Syst. Appl. 37 (2010) 226–232.
[12] D. Jensen, J. Neville, B. Gallagher, Why collective inference improves relational classification, in: The Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004, pp. 593–598.
[13] D. Jensen, J. Neville, Linkage and autocorrelation cause feature selection bias in relational learning, in: Proceedings of the 19th International Conference on Machine Learning, 2002, pp. 259–266.
[14] T. Kajdanowicz, P. Kazienko, A method of label-dependent feature extraction in social networks, ICCCI'2010, The Second International Conference on Computational Collective Intelligence Technology and Applications, Lecture Notes in Artificial Intelligence, vol. 6422, 2010, pp. 11–21.
[15] T. Kajdanowicz, P. Kazienko, P. Doskocz, Label-dependent feature extraction in social networks for node classification, The Proceedings of Second International Conference on Social Informatics, Lecture Notes in Artificial Intelligence, vol. 6430, 2010, pp. 89–102.
[16] T. Kajdanowicz, P. Kazienko, P. Doskocz, K. Litwin, An assessment of node classification accuracy in social networks using label-dependent feature extraction, The Third World Summit on the Knowledge Society WSKS 2010, Communications in Computer and Information Science, vol. 111, 2010, pp. 125–130.
[17] T. Kamada, S. Kawai, An algorithm for drawing general undirected graphs, Inf. Process. Lett. 31 (1989) 7–15.
[18] A. Knobbe, M. deHaas, A. Siebes, Propositionalisation and aggregates, in: Proceedings of the Fifth European Conference on Principles of Data Mining and Knowledge Discovery, 2001, pp. 277–288.
[19] S. Kramer, N. Lavrac, P. Flach, Propositionalization approaches to relational data mining, in: S. Dezeroski (Ed.), Relational Data Mining, Springer-Verlag, 2001, pp. 262–286.
[20] M. Krogel, S. Rawles, F. Zeezny, P. Flach, N. Lavrac, S. Wrobel, Comparative evaluation of approaches to propositionalization, in: The Proceedings of 13th International Conference on Inductive Logic Programming, 2003, pp. 180–196.
[21] Q. Lu, L. Getoor, Link-based classification, in: Proceedings of the 20th International Conference on Machine Learning ICML, 2003, pp. 496–503.
[22] S. Macskassy, F. Provost, Classification in networked data: a toolkit and a univariate case study, J. Mach. Learn. Res. 8 (2007) 935–983.
[23] S. Macskassy, F. Provost, A simple relational classifier, in: The Proceedings of KDD-2003 Workshop on Multi-Relational Data Mining MRDM-2003, 2003, pp. 64–76.

[24] L.K. McDowell, K.M. Gupta, D.W. Aha, Cautious inference in collective classification, in: The Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, 2007, pp. 596–601.
[25] L. McDowell, K. Gupta, D. Aha, Case-based collective classification, in: Proceedings of the Twentieth International FLAIRS Conference, AAAI Press, 2007.
[26] K. Musial, P. Kazienko, P. Brodka, User position measures in social networks, in: The Third SNA-KDD Workshop on Social Network Mining and Analysis held in Conjunction with the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD, ACM Press, 2009 Article no. 6.
[27] J. Neville, D. Jensen, Iterative classification in relational data, in: Proceedings of the AAAI 2000 Workshop Learning Statistical Models from Relational Data, AAAI Press, 2000, pp. 42–49.
[28] J. Neville, D. Jensen, Collective classification with relational dependency networks, in: Proceedings of the Second International Workshop on Multi-Relational Data Mining, 2003, pp. 77–91.
[29] J. Pearl, Probabilistic Reasoning in Intelligent Systems, Morgan Kaufmann, 1988.
[30] C. Perlich, F. Provost, Aggregation-based feature invention and relational concept classes, in: ACM SIGKDD in the Proceedings of Ninth International Conference on Knowledge Discovery and Data Mining, 2003, pp. 167–176.
[31] C. Perlich, F. Provost, Distribution-based aggregation for relational learning with identifier attributes, Mach. Learn. 62 (1–2) (2006) 65–105.
[32] M. Richardson, P. Domingos, Markov logic networks, Mach. Learn. 62 (1–2) (2006) 107–136.
[33] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, T. Eliassi-Rad, Collective classification in network data, Artif. Intell. Mag. 29 (3) (2008) 93–106.
[34] P. Sen, L. Getoor, Empirical comparison of approximate inference algorithms for networked data, in: ICML Workshop on Open Problems in Statistical Relational Learning, 2006.
[35] B. Taskar, P. Abbeel, D. Koller, Discriminative probabilistic models for relational data, in: Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence, 2002, pp. 485–492.
[36] S. Wasserman, K. Faust, Social Network Analysis: Methods and Applications, Cambridge University Press, New York, 1994.

Przemysław Kazienko received his MSc and PhD degrees in computer science with honors, both from Wroclaw University of Technology, Poland, in 1991 and 2000, respectively. He obtained his habilitation degree from Silesian University of Technology, Poland, in 2009.

Recently, he serves as a professor of Wroclaw University of Technology at the Institute of Informatics, Poland. He was also a Research Fellow at Intelligent Systems Research Centre, British Telecom, UK in 2008. For several years, he held the position of the deputy director for development at Institute of Applied Informatics.

He was a co-chair of international workshops RAAWS’05, RAAWS’06, MMAML’10, MMAML’11 and a Guest Editor of New Generation Computing and International Journal of Computer Science & Applications. He regularly serves as a member of international programme committees and the reviewer for scientific conferences and prestige international journals. He is a member of Editorial Board of International Journal of Knowledge Society Research and International Journal of Human Capital and Information Technology Professionals. He has authored over 100 scholarly and research articles on a variety of areas related to multiple model classification, collective classification, social networks, social network analysis, knowledge management, collaborative systems, data mining, recommender systems, Information Retrieval, data security, and XML. He also initialized and led over 20 projects chiefly in cooperation with commercial companies, including large international corporations.

Tomasz Kajdanowicz was born in Poland in 1983. He received the MSc degree from Wroclaw University of Technology, Poland in 2008. He is currently a PhD student at Wroclaw University of Technology and simultaneously a consultant at Hewlett Packard, Poland. His research areas focus on social network analysis and hybrid information systems as well as their applications, especially in the industry. While participating in multiple research and development projects, he collaborates with leading financial enterprises in Poland. He authored about 20 scientific papers and articles.