2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

A Consensus-Based Semi-Supervised Growing Neural Gas

Vinícius R. Máximo, Marcos G. Quiles, and Mariá C. V. Nascimento

***Abstract*— In this paper, we propose a new semi-supervised growing neural gas (GNG) model, named *Consensus-Based Semi-Supervised* GNG, or CSSGNG, in which both labeled and unlabeled data are used to train the network. In contrast to former adaptations of the GNG to semi-supervised classification, such as the SSGNG and OSSGNG models, the CSSGNG does not assign a single scalar label value to each neuron. Instead of the scalar, a vector containing the representativeness level of every class is associated with each neuron. Moreover, to propagate the labels among the neurons the CSSGNG employs a consensus approach. Computer experiments show that our model on average can deliver better classification results in comparison to the SSGNG and OSSGNG models.**

I. Introduction

SEMI-SUPERVISED learning has become an important research topic in machine learning. It lies between two orthodox machine learning paradigms: unsupervised learning and supervised learning. In contrast to the supervised learning, which requires a completely labeled dataset as input, or unsupervised learning in which no knowledge of labeled data is considered, semi-supervised learning (SSL) techniques can learn from both labeled and unlabeled data.

Usually, unlabeled data is less costly to obtain than labeled data; thus, semi-supervised learning becomes an interesting and cheaper approach in situations where only a small fraction of the data can be labeled. Moreover, published results have shown that, in practice, unlabeled can improve the performance of supervised classification and also improve clustering when the labeled data is used as constraints to guide the search for optimal partition in the data [1]–[3].

There exists a variety of SSL methods built under fairly distinct approaches, which includes: generative models [4], [5]; multi-view methods [6]–[10]; cluster-than-label methods [11], [12]; low-density separation models, such as Transductive Support Vector Machines (TSVM) [13]; and graph-based methods [14]–[20]. For more information regarding the SSL methods, we suggest the reading of [1], [21], [22].

Machine learning approaches based on topological maps, such as the Kohonen Self-Organizing Maps (SOM) [23] and the Growing Neural Gas (GNG) [24] have been widely applied to unsupervised problems for representing high-dimensional feature space into a low dimensional map. In special, the GNG is a self-organizing incremental network that can learn the intrinsic topological relation of a dataset in an incremental fashion. During the learning phase, the GNG can adjust its topology by automatically adding and removing neurons and connections [24], [25], which is not allowed in the SOM. Thus, the GNG-based algorithms are suitable for non-stationary learning settings.

In its original form, GNG is an unsupervised technique albeit it can be easily applied to perform data classification in both supervised and semi-supervised paradigms. In performing supervised data classification, Heinke and Hamker [26] have shown that the GNG can outperform some traditional classifiers, such as the MLP network. Moreover, their study has also suggested that the GNG has not a high sensitivity on parameters change, which is a quite important finding.

Concerning the semi-supervised scenario, one can find a few studies as in [27], [28]. Zaki and Yin [27] developed an algorithm named SSGNG (Semi-Supervised Growing Neural Gas) using a self-training approach to solve a semi-supervised classification problem. For the same purpose, Beyer and Cimiano [28] presented the OSSGNG algorithm (On-line Semi-Supervised Growing Neural Gas). Unlike in [27], in their model, labeled and unlabeled samples are processed in an on-line fashion without the need to store the labeled samples as considered in [27]. Both SSGNG and OSSGNG are able to provide good classification results even though some improvements can still be pursued. For example, for a number of applications, these strategies may require a high computational time due to their label propagation characteristics. More information about these models and their limitations are depicted in the next section.

In this paper, we propose a novel GNG-based method for SSL, named *Consensus-Based Semi-Supervised* GNG, or CSSGNG. In contrast to the former approaches, through a consensus process, our model can deliver better classification results and also more information with regard to the classification confidence of each sample. This last information can be used, for example, to query the specialist about new labels, i.e. active learning, and also to solve multi-class classification problems.

This paper is organized as follows. Section 2 revisits the original GNG algorithm and its former adaptation to SSL. The semi-supervised classification method proposed in this paper is presented in Section 3. Computer

Vinícius R. Máximo, Marcos G. Quiles, and Mariá C. V. Nascimento are with the Institute of Science and Technology (ICT), Federal University of São Paulo (Unifesp), São José dos Campos, SP, Brazil, (phone: +55 12 3309-9582; email: vinymax10@gmail.com, {quiles, mcv.nascimento}@unifesp.br).

This research was supported by São Paulo Research Foundation (FAPESP) and by the Brazilian National Research Council (CNPq).

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

experiments are presented in Section 4. Finally, Section 5 draws some concluding remarks and points up some future work.

II. Background

This section revisits the original GNG model [24] and its two main extensions to semi-supervised classification: the SSGNG and the OSSGNG algorithms.

A. The Original GNG

Growing Neural Gas (GNG) is a competitive self-organizing and incremental neural network. It was proposed by Fritzke in 1994 [24] and can been seen as a model that combines the Competitive Hebbian Learning model (CHL) [29] with the Growing Cell Structures (GCS) [30]. The CHL model, proposed by Martinetz [29], introduced the concept of creating edges between two winner neurons for a given input data. On the other hand, the GCS model, proposed by Fritzke [30], introduced a mechanism to dynamically create neurons.

Roughly speaking, the GNG algorithm can be described as follows: the network starts with two neurons randomly disposed in the feature space. In each iteration, a sample x_i is selected from the input space and presented to the network. Following a competitive learning mechanism, a pair of neurons, s_1 and s_2 which are, respectively, the first and the second winners, are selected.

Algorithm 1: Present Sample.

Input: Sample x_i

Find the first, s_1, and the second, s_2, winner neurons.

$s_1 = \arg\min_{j \in V} (\mathbf{x}_i - \mathbf{w}_j)^2$

$s_2 = \arg\min_{j \in V \setminus \{s_1\}} (\mathbf{x}_i - \mathbf{w}_j)^2$

if *there is a link between* s_1 *and* s_2 **then**

  reset the age of the link to 0.

end

else create the link and set its age to 0.

Update the local error E_{s_1} by:

$\Delta E_{s_1} = (\mathbf{x}_i - \mathbf{w}_{s_1})^2$

Update the weights of the winner neuron s_1.

$\Delta \mathbf{w}_{s_1} = \varepsilon_b (\mathbf{x}_i - \mathbf{w}_{s_1})$

Update the weights of the neighborhood of the winner neuron s_1.

$\Delta \mathbf{w}_j = \varepsilon_n (\mathbf{x}_i - \mathbf{w}_j) \quad \forall\, j \in \delta(s_1).$

Increment the ages of all links adjacent to s_1.

Remove all links with age larger than a_{max}.

Remove neurons with no links.

Decrease the accumulated error of all neurons by multiplying them with a constant β.

$E_i = \beta E_i \quad \forall\, i \in V.$

Between this pair of neurons, a connection is (re)established. Afterwards, the weights of the winner neuron and its neighborhood, according to the network, are updated leading to the movement of those neurons towards the position of the input signal in the feature space. These updates are with regard to the neuron attributes, as summarized in Algorithm 1.

After a predefined number of iterations λ, new neurons are included in the network, as shown in Algorithm 2. These neurons are inserted in order to reduce the accumulated error in the network. The accumulated error of a neuron i is denoted by E_i. The insertion process works by, firstly, selecting a pair of neurons in the GNG network which will be the new neuron's neighbors. This pair is composed by the neuron q with the highest accumulated error, ($E_q \geq E_j \ \forall j$), and the neuron f within the neighborhood of neuron q (here, the neighborhood of a neuron q is denoted by $\delta(q)$) with the highest error ($E_f \geq E_j \ \forall j \in \delta(q)$). An overall description of the original GNG algorithm is depicted in Algorithm 3.

Algorithm 2: Insert new neuron

Input: Input data

Insert a new neuron r according to the following procedure:

Find the neuron q with the larger accumulated error.

$q = \arg\max_{i \in R} (E_i)$

Among the neighbors of q, find f with the largest accumulated error.

$f = \arg\max_{viz \in \delta(q)} (E_{viz})$

Insert a new neuron, r, between q and f.

$\mathbf{w}_r = \frac{(\mathbf{w}_q + \mathbf{w}_f)}{2}$

Create links $\{q, r\}$ and $\{r, f\}$ and delete the original link $\{q, f\}$.

Decrease the error of q and f by means of the constant α and set the error of r according to the mean of the errors of q and f.

$E_q = \alpha E_q \qquad E_f = \alpha E_f \qquad E_r = \frac{(E_q + E_f)}{2}$

Owning to its structure, the GNG network can be seen as a graph $G = \{V, M\}$, in which V is the set of neurons (or vertices) and M the set of edges between neurons.

Algorithm 3: The original GNG algorithm for unsupervised learning

Input: Set of input data

while *The stopping criterion is not reached* **do**

  Present Sample($\mathbf{x}_i$)

  if *the current iteration is a multiple of* λ **then**

    Insert new neuron()

  end

end

Although several parameters must be empirically set for the suitable functioning of the GNG network, most

of them have well-known default values. Actually, as reported in [26], the GNG is not sensitive to parameters' changes. For instance, good convergence on the classification results has been reported by using the following setup:

- Transient to add a new neuron: $\lambda = (300, 1000)$
- Link's maximum age: $a_{max} = (80, 120)$
- Winner learning coefficient: $\varepsilon_b \approx 10^{-2}$
- Neighborhood learning coefficient: $\varepsilon_n \approx 10^{-3}$
- Neighboring error reduction: $\alpha = 0.5$
- Error reduction rate: $\beta = 0.995$

B. Former GNG Adaptations to SSL

There are two main adaptations of the GNG neural network to semi-supervised classification, the Semi-Supervised Growing Neural Gas (SSGNG) [27] and the Online Semi-Supervised Growing Neural Gas (OSSGNG) algorithm [28], both already mentioned before. The former, proposed by Zaki and Yin [27], follows a self-training approach: first the model is trained with the labeled set; then, the network classifies the unlabeled samples which are gradually added to the labeled set. Iteratively, the network is retrained using both the labeled and classified unlabeled sets until the labels assigned to the unlabeled samples become stable.

In the same scenario, the OSSGNG algorithm processes labeled and unlabeled samples in an online fashion without the need to store the labeled samples as considered in [27]. Moreover, an additional step is used to assign a label or even to recalculate the former label of the winner neuron after the presentation of a labeled input data. This algorithm also makes use of a prediction function for labeling unlabeled data. Finally, it is worth noting that the outcomes provided by the OSSGNG are quite similar to those generated by the SSGNG.

In the SSL scenario, SSGNG and OSSGNG can provide good classification results albeit some improvements can still be pursued. Nevertheless, they are characterized by the global propagation of the labels among the dataset. This trait may lead to a high computational processing, which may not be desirable for medium to large-scale applications. Moreover, in both models the label is associated to a single scalar value, thus, the only information that can be retrieved from a neuron is whether a sample belongs or to not to that class. It can be a problem when the sample lies in between two classes.

Taking this and other relevant issues into account, we propose, through a consensus approach, an attempt to overcome the possible drawbacks found in the GNG adaptations to SSL, mainly those limitations with respect to memory storage and label information. Hence, we propose a new GNG-based online method for SSL, aiming at improving classification results, primarily, with respect to the classification confidence of each sample.

III. Model Description

In this section, we introduce our Consensus-Based GNG model for semi-supervised classification (CS-SGNG). For describing this model, first, consider a set of n samples from which r of them are labeled. These samples, represented by the set $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_r, \mathbf{x}_{r+1}, ..., \mathbf{x}_n\} \subset \mathbb{R}^m$, compose the input dataset in a m dimensional space. Each sample $x_i \in X$ with $i \leq r$ has an associated label $y_i \in L$, where $L = \{1, 2, ..., t\}$ defines the label set, being t the number of classes.

Our model consists of a GNG network trained with both labeled and unlabeled samples from X. In addition to the weight vector that represents the position of the neuron into the feature space, each neuron also has a *pertinence vector* which defines its classification ability. Instead of using a single scalar to label each neuron, such as considered in SSGNG and OSSGNG models, here we adopt a vector that represents the neuron's pertinence to each class of the problem. Thus, if necessary more information about the class assigned to each neuron can be provided. Further, the model can be straightly applied to the multi-class classification problem.

Formally, let $\mathbf{c}_j = \{c_{j,1}, c_{j,2}, ..., c_{j,t}\}$ be the pertinence vector of neuron j, in which each scalar $c_{j,l}$ defines the representativeness level of a class l assigned to neuron j. If we normalize the vector $\mathbf{c}_j$ in order to make $\sum_{l \in L} c_{j,l} = 1$, for each sample $\mathbf{x}_i$ represented by neuron j, $c_{j,l} = p(\mathbf{x}_i|l)$ for all class $l \in L$. For assigning a label to a sample x_i, consider its winner neuron j, i.e., $y_i = \arg\max_{l \in L}(c_{j,l})$. Nevertheless, instead of using a single winner neuron, as we will show in Section 3, better results can be achieved by considering a set of the κ-nearest neurons.

The learning phase follows the standard unsupervised GNG algorithm to evolve the network. In order to perform the classification, we must set correct values for the pertinence vectors $\mathbf{c}_j$, for all neurons j. To solve this problem, we developed two major modifications into the unsupervised GNG algorithm. First, we show a mechanism to absorb the available label information. Afterwards, we introduce a consensus mechanism for propagating the labels among the neurons from the network.

A. Label Absorption

The label absorption mechanism is responsible for setting the label information into the pertinence vector of the winner neuron s_1. It means, during the learning phase, whenever a labeled sample is presented to the network, the label information is used to set the vector $\mathbf{c}_{s_1}$ of the winner neuron s_1. Algorithm 4 shows the absorption label method proposed in this paper.

The pertinence vector $\mathbf{c}_{s_1}$ of the neuron s_1 assumes a value equal to 1 only in $l = y_i$, the other positions are set to zero (Line 2 of Alg. 4). This approach guarantees that the winner neuron s_1 gets a higher probability of

Algorithm 4: Label Absorption.

Input: The label y_i of sample x_i and the winner neuron s_1.

1 **for each** $l \in L$ **do**
2 $\quad c_{s_1,l} = \begin{cases} 1, & \text{if } l = y_i \\ 0, & \text{otherwise.} \end{cases}$
3 **end**
4 **for each** $j \in \delta(s_1)$ **do**
5 $\quad c_{j,y_i} = 1$
6 **end**
7 **for each** $j \in \delta(s_1)$ **do**
8 $\quad$ Run the consensus method for neuron j III-B.
9 **end**

being labeled with the same class y_i of the input sample x_i. In addition, the label information is also propagated to the neighbor neurons of s_1 albeit, in contrast to the changes in the $\mathbf{c}_{s_1}$, only $c_{j,l}$ is set to 1. The positions of this pertinence vector associated with the other classes do not change.

After the local changes in s_1 and in its neighborhood $\delta(s_1)$, the label information is spread through the network in order to reach a label equilibrium, or agreement, among the neighbor neurons. It means the pertinence vector associated with each neuron is not set only by the label information provided by samples that activate this neuron. It is the result of the combination of the local information provided by the labeled samples with the neighborhood information, thus, a consensus by the whole of neighbor neurons. This consensus approach is thoroughly described in the next section.

B. Consensus

Consensus problems can be found in several study fields (see [31]). Here, the consensus consists in reaching to an agreement with respect to the pertinence values among neighbor neurons. For instance, the consensus for a neuron j with regard to a class l is defined as the average of $c_{i,l}$ among every $i \in \delta(j)$ weighted by the inverse of their distance to j ($d_{j,i}$) plus its own pertinence value for the label l, $c_{j,l}$:

$$c_{j,l} = \gamma \left(\hat{d}_j + \sum_{i \in \delta(j)} d_{j,i} \right) \left(\frac{c_{j,l}}{\hat{d}_j} + \sum_{i \in \delta(j)} \frac{c_{i,l}}{d_{j,i}} \right) \forall l \in L \tag{1}$$

where γ is an attenuation factor and $\hat{d}_j = \min_{i \in \delta(j)} d_{j,i}$. Following Eq. (1), a neuron j represents the class l with a probability defined by its own pertinence value combined with pertinence values of its neighborhood.

The attenuation factor γ is responsible for reducing the pertinence value of a given label according to the distance of neuron j to a labeled neuron. Thus, the further away a neuron j is from a labeled neuron i, the lower its pertinence value is for the label assigned to i.

C. The CSSGNG algorithm

As aforementioned, the CSSGNG can be seen as an extension of the unsupervised GNG algorithm with two main adaptations: the *label absorption mechanism* (Sec. III-A), that transforms the unsupervised GNG into a semi-supervised model; and the *consensus* (Sec. III-B), which defines the pertinence of each class over the neurons. Algorithm 5 depicts an overall description of the proposed algorithm.

The label absorption occurs in an on-line fashion, it means the labels are absorbed on the fly during the learning phase. Consequently, in each iteration, the algorithm must take two situations into account: whether the input sample x_i is a labeled or an unlabeled sample. If x_i is an unlabeled sample, the algorithm follows the same standard steps of the original GNG. Otherwise, if x_i is labeled, the label y_i is absorbed by the winner neuron and by its neighborhood (See lines 3-5 of Alg. 5).

One can see that the consensus method acts on the neighborhood of s_1 just after the end of the label absorption (lines 7-9 of Alg. 4). As a consequence, neurons representing regions of the feature space with no labeled samples will not be part of the consensus process. In order to solve this limitation, the consensus is run for the whole network every time a new neuron is inserted, or at every λ steps.

The insertion of new neurons is very similar to the unsupervised GNG even though, here, this insertion considers the setting of the pertinence vector. The process is analogous to the strategy used to define the weight vector w of the new neuron j:

$$\mathbf{c}_j = \frac{\mathbf{c}_q + \mathbf{c}_f}{2} \tag{2}$$

which is the halfway between neurons q and f.

After the learning phase, responsible for generating the network and labeling all neurons, the model is ready to assign labels to the unlabeled data. For performing this task, there are two different ways. The first consists in labeling each unlabeled sample x_i presented to the network by straightly using the pertinence vector of the winner neuron s_1. In this case, y_i is computed by finding the largest pertinence value, i.e., $y_i = \arg\max(\mathbf{c}_{s_1})$.

The second approach is also a consensus-based strategy for labeling unlabeled data. For such, the pertinence vector of each unlabeled data x_i presented to the network is proportional to the average vector among the pertinence vectors of its κ-nearest neurons. The pertinence vector of x_i also considers the sum of the distances between every neuron $j \in \delta(\kappa)$ and the samples x_i, where $\delta(\kappa)$ is the set of the κ nearest neurons of x_i. Therefore, one may calculate $\hat{\mathbf{c}}$ by using the following equation:

$$\hat{\mathbf{c}} = \left(\sum_{j \in \delta(\kappa)} d_{j,x_i} \right) \left(\sum_{j \in \delta(\kappa)} \frac{c_{j,l}}{d_{j,x_i}} \right) \quad \forall l \in L \tag{3}$$

Finally, in this second approach, x_i is labeled according to $y_i = \arg\max(\hat{\mathbf{c}})$. Algorithm 5 presents a pseudocode summarizing all steps of the proposed consensus strategy, the CSSGNG algorithm.

Algorithm 5: CSSGNG - Consensus-Based Semi-Supervised GNG

Input: Set of input data
while *The stopping criterion is not reached* **do**
  Present Sample($\mathbf{x}_i$)
  if x_i *is labeled* **then**
    Neuron s_1 **absorbs the label** y_i **following to Alg. 4**
  end
  if *the current iteration is a multiple of* λ **then**
    Insert new neuron()
    for each *neuron* $j \in V$ **do**
      Run the Consensus method accordingly to Equation 1
    end
  end
end

Next section presents some computer experiments carried out using our model.

IV. Computer Experiments

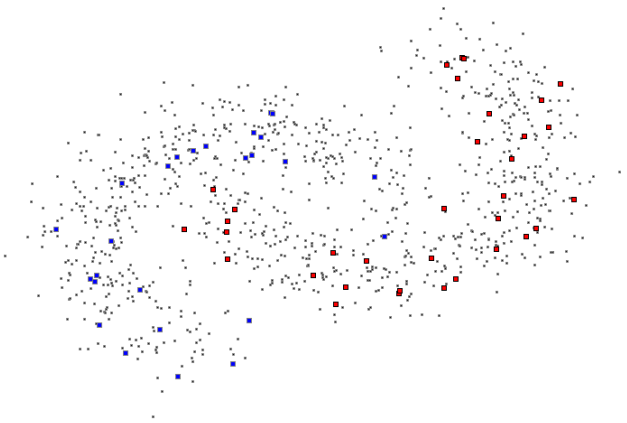
In this section, we present a comprehensive evaluation of the results achieved by the model proposed in this paper. The simulations are divided into three parts. First, we display an illustrative simulation using a synthetic dataset. The main purpose behind this simulation is for analyzing the performance of the model in a well-known scenario. Second, we evaluate the classification accuracy by varying κ-nearest neurons to compute $\hat{\mathbf{c}}$ (Eq. 3). Finally, for the sake of comparison, we evaluate our model by using the same methodology and datasets considered in [1].

The values for the parameters of the CSSGNG network, depicted in Table I, were empirically decided and held constant in all simulations. Actually, as mentioned in Section II-A, the GNG parameters have well-known values for which good results are achieved, thus only the parameters γ and κ, introduced in our model, went through a fine-tuning by a trial-and-error approach in order to improve further the classification accuracy of the model. Additionally, we employed a maximum number of 200 neurons as the stop criterion of CSSGNG. It is worth noting that the number of neurons is not a core parameter. For any number between 150 and 250, we barely obtain the same results for all datasets.

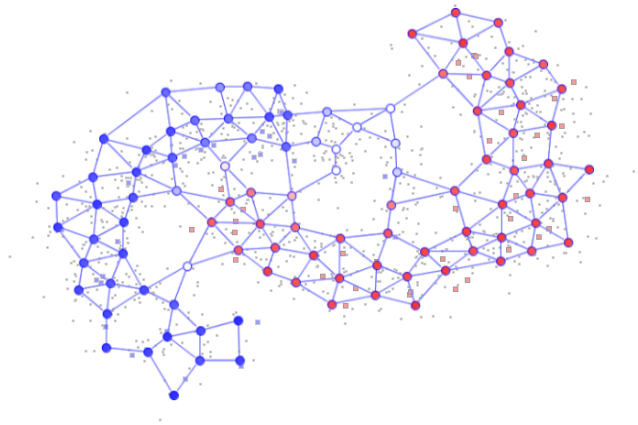
The first simulation we carried out was with the dataset showed in Fig. 1(a), where just 6% of samples are labeled (the red and blue squares). This dataset is composed of 1000 samples equally divided into two classes. The accuracy achieved in a set of 100 realizations was superior to 98%. In order to facilitate the visualization

TABLE I
Parameters setting of the CSSGNG network.

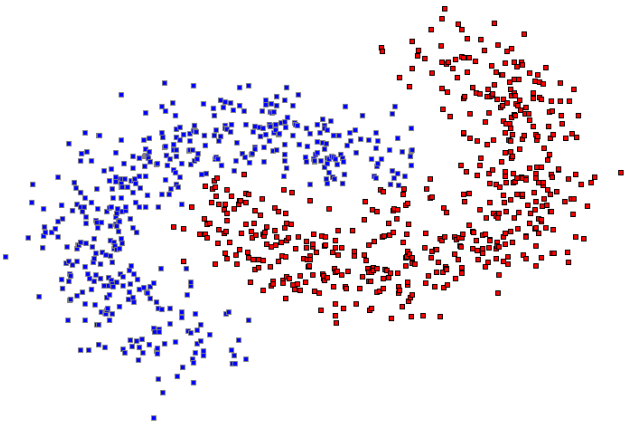
ε_b	ε_n	λ	α	β	γ	a_{max}	κ
0.04	0.007	1000	0.5	0.995	0.95	120	4



(a) Input dataset



(b) Network



(c) Classification Result

Fig. 1. Illustration of the proposed model using a synthetic dataset.

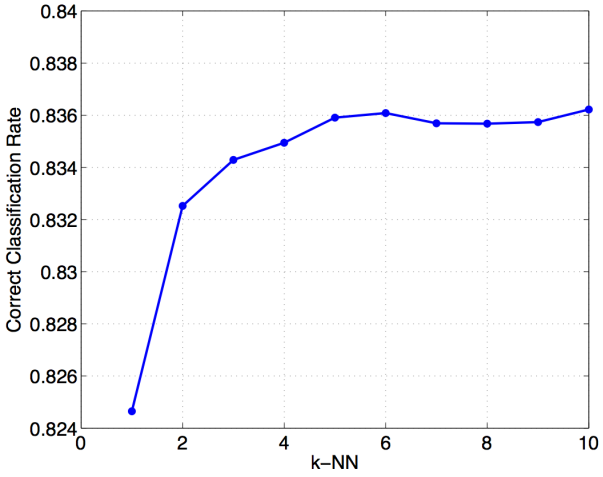


Fig. 2. Correct classification rate versus κ-nearest neurons. Each point in the trace represent the average of 100 realization in each of the datasets depicted in Table II.

of the network, we limited the number of neurons to 100. Figure 1(b) depicts a network with 100 neurons trained with the dataset presented in (a). Neurons are labeled with either red or blue colors according to the class they represent. The color saturation indicates the label confidence, i.e., if the color of a neuron has a low saturation (close to the white color), the probability $p(x|y)$ assumes similar values $\forall y$. The final classification result is shown in Fig. 1(c)

Figure 2 shows the classification accuracy by varying κ-nearest neurons. The results is an average of several simulations using all datasets considered in [1]. It can be observed that the accuracy is slightly improved when a set of neighbor neurons are taken into account. Actually, in order to not compromise the computational cost of the classification procedure and to obtain some accuracy gain, we set $\kappa = 4$. This setup is used in the following simulation.

For the sake of comparison, the third part of the simulations were conducted with benchmark datasets studied in [1][1]. Some information about these datasets are presented in Table II. It is worth noting that all datasets are balanced, except the USPS, in which the classes are imbalanced with relative sizes of $1:4$.

The same methodology used in [1] was used in this work, *i.e.*, the experiments were divided into two scenarios: one with 10 samples of each dataset were labeled and the other with 100 samples of each dataset were labeled. For each dataset, the same 12 *folds* provided by [1] were taken into acount, enabling to a straight comparison with all methods analyzed in [1]. Specifically, we compare our results with the best and worst results achieved by the methods analyzed in [1] for each dataset and also

[1]Available online at `http://www.kyb.tuebingen.mpg.de/ssl-book`

TABLE II
DATASETS FROM [1]

Dataset	Class	Dimension	Samples	Type
g241c	2	241	1500	synthetic
g241n	2	241	1500	synthetic
Digit1	2	241	1500	synthetic
USPS	2	241	1500	real
COIL	6	241	1500	real
BCI	2	117	400	real

with the arithmetic average of all methods regarding to each dataset (worst, best, and average in Tables III-IV). Moreover, we also compare our results with those presented in [27], [28], and [19].

It is worth noting that, for the 10 labeled samples scenario, the results for the SSGNG and the OSSGNG models are missing. This scenario was not considered by the authors in [27], [28].

Results with the 10−labeled and 100−labeled samples configurations are presented in Tables III and IV, respectively.

Regarding to the 10−labeled samples scenario, depicted in Table III, our model achieved competitive results providing accuracy superior to the average, and, in most cases, close to the best results reported in [1]. Actually, in the USPS dataset, our model reached the best accuracy. In comparison to the model proposed in [19], our results were superior for all datasets taken into account.

In Table IV, we depict our results in comparison to those results reported in [1] and also the results achieved by the SSGNG,the OSSGNG, and [19] models. Again, it is possible to observe that our model achieved results superior or equivalent to the SSGNG and the OSSGNG models in all datasets, except to the BCI dataset. It should be stated that by analyzing the outcome of both models, we were not able to identify the reasons why our model had a lower performance in this particular dataset.

Similarly to the previous scenario, our model also provided superior results in contrast to those achieved in [19]. Moreover, in comparison to the models analyzed in [1], we achieved results superior to the average and, for some datasets, accuracy close to the best results. Besides, by analyzing the average results in all datasets, one can observe that ours are higher than the average achieved by the SSGNG, OSSGNG, and [19], which demonstrates the accuracy and applicability of our model.

Finally, a negative aspect highlighted in Tables lies in the standard deviation, especially when only a few labeled samples are considered. This higher standard deviation in comparison to the model [19] can be interpreted as a higher sensitivity of our model regarding to selection of the labeled samples. However, as it will be stated in the conclusions, the consensus approach proposed in this paper can be used to guide the selection of samples in an active learning approach. Thus, diminishing or eliminating this sensitivity problem.

TABLE III

ACCURACY RATE FOR THE CHAPELLE BENCHMARK DATASETS: G241C, G24D, DIGIT1, USPS, COIL, BCI, 10 LABELED SAMPLES

Datasets	Proposed Model	Model [19]	Average	Worst	Best
g241c	74.04±11.39%	55.95±3.30%	59.54%	50.41%	77.24%
g241d	70.11±14.97%	56.78±2.89%	55.84%	49.37%	81.27%
Digit1	87.00±6.98%	76.42±5.79%	85.05%	69.40%	94.56%
USPS	**85.58±3.95%**	80.51±3.55%	80.88%	74.64%	83.93%
COIL	38.95±6.25%	35.50±4.41%	35.83%	32.50%	45.46%
BCI	51.77±2.35%	51.26±2.83%	50.69%	49.64%	52.05%
Average	**67.91%**	59.40%			

TABLE IV

ACCURACY RATE FOR THE CHAPELLE BENCHMARK DATASETS: G241C, G24D, DIGIT1, USPS, COIL, BCI, 100 LABELED SAMPLES

Datasets	Proposed Model	Model [19]	SSGNG [27]	OSSGNG [28]	Average	Worst	Best
g241c	79.42±6.44%	59.72±2.25%	49.64%	52.98%	72.91%	55.95%	86.51%
g241d	90.26±2.43%	62.51±1.61%	44.03%	62.66%	72.32%	56.79%	95.05%
Digit1	96.14±1.02%	94.16±1.38%	96.20%	96.77%	96.21%	93.85%	97.56%
USPS	93.76±1.38%	91.37±0.97%	92.58%	93.07%	93.13%	90.23%	95.32%
COIL	81.02±2.16%	81.02±1.83%	75.49%	81.45%	80.05%	71.29%	90.39%
BCI	60.29±1.94%	55.17±1.77%	**80.51%**	79.47%	58.36%	52.11%	66.75%
Average	**83.48%**	73.99%	73.08%	77.73%			

V. CONCLUSIONS

We have proposed a new model based on the Growing Neural Gas network for semi-supervised classification. In contrast to previous GNG-Based SSL models in which a single scalar were used to assign a label to each neuron, here a pertinence vector has been considered to store the representativeness level of each class of the problem. Thus, more information about the classes can be retrieved from each neuron. Moreover, the label propagation through the network was performed by a consensus approach proposed in this work. Finally, the experiments have shown that the proposed approach was able to achieve superior results in comparison to the SSGNG and the OSSGNG models and competitive results in comparison to the best ones reported in the literature. As a future work we intend to explore the consensus information of the pertinence vector in order to query the specialist about labels (active learning). For instance, by analyzing the pertinence vector, one can find the most confident neurons and also those who have no confidence about which class it belongs to. Thus, that outcome might be used to indicate the regions of the feature space that need further inspection of the specialist. In addition, by using the intrinsic characteristics of the pertinence vector, we intend to used our model to solve multi-class classification problems.

REFERENCES

[1] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*. Cambridge, MA: The MIT Press, 2006.

[2] B. Kulis, S. Basu, I. Dhillon, and R. Mooney, "Semi-supervised graph-clustering: a kernel approach," in *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, Bonn, Germany, 2005, pp. 19–26.

[3] W. Chen and G. Feng, "Spectral clustering: A semi-supervised approach," *Neurocomputing*, vol. 77, no. 1, pp. 229–242, 2012.

[4] K. Nigam, A. K. McCallum, S. Thrun, and T. M. Mitchell, "Text classification from labeled and unlabeled documents using em," *Machine Learning*, vol. 39, pp. 103–134, 2000.

[5] A. Fujino, N. Ueda, and K. Saito, "A hybrid generative/discriminative approach to semi-supervised classifier design," in *AAAI-05, Proceedings of the Twentieth National Conference on Artificial Intelligence*, 2005, pp. 764–769.

[6] A. Blum and T. M. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the Eleventh Annual Conference on Computational Learning Theory - COLT-98*. ACM Press, 1998, pp. 92–100.

[7] T. M. Mitchell, "The role of unlabeled data in supervised learning," in *Proceedings of the Sixth International Colloquium on Cognitive Science*, 1999.

[8] Z.-H. Zhou and M. Li, "Tri-training: exploiting unlabeled data using three classifiers," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 11, pp. 1529–1541, 2005.

[9] ——, "Semisupervised regression with cotraining-style algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 11, pp. 1479–1493, 2007.

[10] G. Jin, R. Raich, and D. J. Miller, "A generative semi-supervised model for multi-view learning when some views are label-free," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2013)*. Vancouver, Canada: IEEE Press, 2013, pp. 3302–3306.

[11] A. Demiriz, K. P. Bennett, and M. J. Embrechts, "Semi-supervised clustering using genetic algorithms," in *Proceedings of Artificial Neural Networks in Engineering (ANNIE-99*. ASME Press, 1999, pp. 809–814.

[12] R. Dara, S. Kremer, and D. Stacey, "Clustering unlabeled data with soms improves classification of labeled real-world data," in *Proceedings of the World Congress on Computational Intelligence (WCCI)*, 2002, pp. 2237–2242.

[13] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley-Interscience, September 2008.

[14] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," Carnegie Mellon University, Pittsburgh, Tech. Rep. CMU-CALD-02-107, 2002. [Online]. Available: http://citeseer.ist.psu.edu/581346.html

[15] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the Twentieth International Conference on Machine Learning*, 2003, pp. 912–919.

[16] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Advances in Neural Information Processing Systems*, vol. 16. MIT Press, 2004, pp. 321–328. [Online]. Available: http://www.kyb.tuebingen.mpg.de/bs/people/weston/localglobal.pdf

[17] F. Wang and C. Zhang, "Label propagation through linear neighborhoods," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 1, pp. 55–67, Jan. 2008.

[18] M. G. Quiles, L. Zhao, F. A. Breve, and A. Rocha, "Label propagation through neuronal synchrony," in *The 2010 International Joint Conference on Neural Networks (IJCNN'2010)*, Barcelona, Espanha, 2010, pp. 2517–2524.

[19] M. G. Quiles, M. P. Basgalupp, and R. Barros, "An oscillatory correlation model for semi-supervised classification," *Learning and Nonlinear Models*, vol. 11, pp. 3–10, 2013.

[20] M. Karasuyama and H. Mamitsuka, "Multiple graph label propagation by sparse integration," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 12, pp. 1999–2012, 2013.

[21] X. Zhu and A. B. Goldberg, *Introduction to semi-supervised learning*, ser. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2009.

[22] Z. H. Zhou and M. Li, "Semi-supervised learning by disagreement," *Knowledge and Information Systems*, vol. 24, no. 3, pp. 415–439, 2010.

[23] T. Kohonen, *Self-Organizing Maps*, 3rd ed., ser. Springer Series in Information Sciences. Springer, 2000.

[24] B. Fritzke, "A growing neural gas network learns topologies," in *Advances in Neural Information Processing Systems 7*. MIT Press, 1995, pp. 625–632.

[25] J. Graham and J. Starzyk, "A hybrid self-organizing neural gas based network," in *IEEE International Joint Conference on Neural Networks (IJCNN 2008)*. Hong Kong: IEEE Press, 2008, pp. 3806–3813.

[26] D. Heinke and F. H. Hamker, "Comparing neural networks: A benchmark on growing neural gas, growing cell structures, and fuzzy artmap," *IEEE Transactions on Neural Networks*, vol. 9, no. 6, pp. 1279–1291, 1998.

[27] S. Mohd Zaki and H. Yin, "Semi-supervised growing neural gas for face recognition," in *Proceedings of the 9th International Conference on Intelligent Data Engineering and Automated Learning*, ser. IDEAL '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 525–532. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-88906-9_66

[28] O. Beyer and P. Cimiano, "Online semi-supervised growing neural gas," *International Journal of Neural Systems*, vol. 22, no. 05, p. 1250023, 2012, pMID: 22992211. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S0129065712500232

[29] T. Martinetz, "Competitive hebbian learning rule forms perfectly topology preserving maps," in *Proceedings of the International Conference on Artificial Neural Networks (ICANN-93)*, S. Gielen and B. Kappen, Eds. Amsterdam, The Netherlands: Springer, 1993, pp. 427–434.

[30] B. Fritzke, "Growing cell structures - a self-organizing network for unsupervised and supervised learning," *Neural Networks*, vol. 7, pp. 1441–1460, 1993.

[31] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Procedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.