High Level Pattern Classification via Tourist Walks in Networks

Thiago Christiano Silva and Liang Zhao, Senior Member, IEEE

Abstract—Complex networks refer to large-scale graphs with nontrivial connection patterns. The salient and interesting features that the complex network study offer in comparison to graph theory are the emphasis on the dynamical properties of the networks and the ability of inherently uncovering pattern formation of the vertices. In this paper, we present a hybrid data classification technique combining a low level and a high level classifier. The low level term can be equipped with any traditional classification techniques, which realize the classification task considering only physical features (e.g., geometrical or statistical features) of the input data. On the other hand, the high level term has the ability of detecting data patterns with semantic meanings. In this way, the classification is realized by means of the extraction of the underlying network’s features constructed from the input data. As a result, the high level classification process measures the compliance of the test instances with the pattern formation of the training data. Out of various high level perspectives that can be utilized to capture semantic meaning, we utilize the dynamical features that are generated from a tourist walker in a networked environment. Specifically, a weighted combination of transient and cycle lengths generated by the tourist walk is employed for that end. Furthermore, we show that the proposed technique is able to capture the organizational and complex features of the class component from a local to global fashion in a natural and intuitive way by altering the memory size of the tourist walk. Still in this work, we uncover the existence of a critical memory length, we say complex saturation, where any values larger than this critical point make no change in the transient and cycle lengths of the network component. Interestingly, our study shows that the proposed technique is able to further improve the already optimized performance of traditional classification techniques. Finally, we apply the proposed technique to the recognition of handwritten digit images and promising results have been obtained.

Index Terms—High level classification, tourist walks, supervised learning, complex networks.

I. INTRODUCTION

SUPERVISED data classification aims at generating a map from the input data to the corresponding desired output, for a given training set. The constructed map, called a classifier, is used to predict new input instances. Many supervised data classification techniques have been developed [1]–[3], such as k-nearest neighbors, Bayesian classifiers, neural networks, decision trees, committee machines, and so on. In essence, all these techniques train and, consequently, classify unlabeled data items according to the physical features (e.g., distance, similarity or distribution) of the input data. These techniques that predict class labels using only physical features are called low level classification techniques [4].

Usually, the data items are not isolated points in the attribute space, but instead tend to form certain patterns. For example, in Fig. 1 the two test instances represented by the triangle-shaped are most probably to be classified as pertaining to the square-shaped class if only physical features, such as distances among data instances, are considered. On the other hand, if we take into account the relationships among the data, we intuitively classify the triangle-shaped items as members of the circular-shaped class, since a clear pattern (lozenge) is formed. The human (animal) brain performs both low and high orders of learning and it has facility to identify patterns according to the semantic meaning of the input data. However, this kind of task, in general, is still hard to be performed by computers. Supervised data classification by considering not only physical attributes but also pattern formation is referred to as high level classification [4].

Fig. 1. A simple example of a supervised data classification task where two clear patterns are formed: the highly organized circular-shaped class and a rather dense square-shaped class. The goal is to classify the two triangle-shaped data items.

Traditional classification techniques often share the same vision: dividing the data space into sub-spaces, each of which representing a class. They are short in reproducing complex-formed or twisted classes. On the other hand, the salient feature of the proposed technique is that it provides two really distinct orthogonal visions: low level and high level visions for data classification. The former views the data’s classes from the perspective of their physical features, while the latter captures pattern formations of the data, which, in turn, permits the classifier to reproduce complex-formed and (or) twisted classes, i.e., a test instance can be put into a class if it conforms with the pattern formed by that class no matter how far it is from the center of that class. In this sense, strongly related techniques are co-training [5] and tri-training [6], which attempt to consider the cooperation of various classification techniques (ensemble), each focusing on a theoretically different “vision” of the data. However,
the involved classification techniques in co-training or tri-training are really only making statistically independent decisions, which still share, in essence, the same data vision. Another difficulty in these techniques resides in building these statistically independent visions. Since it is supposed that the data is generated from an unknown distribution, it is often hard to achieve such task. In our proposed work, the uncorrelated visions are naturally captured by the classifiers themselves, not by changing the content of the data items, but by “looking” at the data relationships in a different way (low and high levels).

Following the literature stream on such matter, there are several kinds of works related to high level classification, such as the Semantic Web [7]-[9], which uses ontologies to describe the semantics of the data, statistical relational learning, which realizes collective inference [10]-[14] or graph-based semisupervised learning [15], [16], and contextual classification techniques [17]-[23], which consider the spatial relationships between the individual pixels and the local and global configurations of neighboring pixels in an image for assigning classes.

From the viewpoint of high level classification, all the above mentioned approaches are quite restricted either to the types of semantic features to be extracted, such as Semantic Web, or to the types of data, such as the contextual classification, which is devoted to considering spatial relationship among pixels in image processing. To our knowledge, it is still lacking an explicit and general scheme to deal with high level classification in the literature, which is quite desirable for many applications, such as invariant pattern recognition. The current paper presents an endeavor to this direction.

As mentioned, low level classification usually presents difficulty in identifying the complex relationships among the data items. Consequently, these techniques are not suitable for uncovering semantically meaningful patterns formed by the data. This is because the data patterns are often not encountered with a fixed shape or distribution, instead, they are frequently determined by the local and/or global interactions among the data items. It is well known that the network representation can capture arbitrary levels of relationships or interactions of the input data. For this reason, we here show how the topological properties of the input data can help in identifying the pattern formation and, consequently, can be used for general high level classification. In this case, the topological properties are revealed by tourist walks. A tourist walk can be defined as follows: Given a set of cities, each time the tourist (walker) goes to the nearest city that has not been visited in the past, the tourist prevails to the nearest city that has not been visited in the past $\mu$ time steps [24]. It has been shown that tourist walk is useful for data clustering [25] and image processing [26]. However, all these kinds of works are realized in regular lattices. Here, we study tourist walk in networks and we show that it has the ability of capturing the topological properties of the underlying network in a local to global fashion. Moreover, the tourist walks approach applied to a networked environment is a relatively new approach taken here. Additionally, its utilization for discovering patterns in a network is a totally novel scheme in the literature.

In this paper, we propose a technique that combines the low level and the high level supervised data classifications. The idea of this paper is built upon the general framework proposed by [4]. The low level classification can be implemented by any traditional classification technique, while the high level classification exploits the complex topological properties of the underlying network constructed from the input data. In the original work introduced in [4], the high level classification problem is treated using three existing network measures in a combined way (assortativity, clustering coefficient, and average degree). Thus, a serious open problem is how one may choose other network measures in an intuitive way and also how to define the inference weight that is given for each of them. In this paper, a novel measure for high level classification is introduced by using tourist walks in networks. Since they are a deterministic dynamical process, the local and global information of the underlying network can be detected by the exclusive use of such measure. Moreover, the use of tourist walks presents some nontrivial advantages over the previous approach, such as:

- It is able to capture the organizational and complex features of the class component from a local to global fashion in a natural and intuitive way. For example, when the memory window of the tourist is low, it is able to extract local features of the graph component. As we increase the memory window, the dynamics of the walk compels the tourist to venture far away from its starting vertex. Hence, it is able to capture more global features of the graph component;
- It occurs that the tourist walk method presents a critical memory length, where any values larger than this critical point make no change in the transient and cycle lengths of the graph component. This is an interesting phenomenon, which is observed when the memory length reaches a sufficient high value. We say that, when this happen, the walks have reached the “complexity saturation” of the class component. In this occasion, the global topological and organizational features of the network are said to be completely characterized in the sense of tourist walks;
- In view of the intuitive dynamical properties displayed by a tourist walk, one can avoid the weight assignment among various network measures, which is a problem when static network measures are used, as occurs in [4]. This is because static measures provide partial or instant visions of the underlying network. Thus, one must artificially combine them to get a global vision.

Still in this paper, we show how the proposed technique can be used to solve general invariant pattern recognition problems [27]-[29], particularly when the pattern variances are nonlinear and there is not a closed form to describe the invariance.

The remainder of the paper is organized as follows. A detailed overview of the tourist walks is supplied in Section [11]. The proposed model is defined in Section [11]. Computer simulations are performed on synthetic and real-world data sets in Section [1V]. In Section [1V], we apply the proposed technique to manual digits recognition problem. Finally, Section [1V] concludes the paper.
II. Relevant Background: Tourist Walks

A tourist walk can be conceptualized as a walker (tourist) aiming at visiting sites (data items) in a \( d \)-dimensional map, representing the data set. At each discrete time step, the tourist follows a simple deterministic rule: it visits the nearest site which has not been visited in the previous \( \mu \) steps. In other words, the walker performs partially self-avoiding deterministic walks over the data set, where the self-avoiding factor is limited to the memory window \( \mu = 1 \). This quantity can be understood as a repulsive force emanating from the sites in this memory window, which prevents the walker from visiting them in this interval (refractory time). Therefore, it is prohibited that a trajectory to intersect itself inside this memory window. In spite of being a simple rule, it has been shown that this movement dynamic possesses complex behavior when \( \mu > 1 \) [24].

The tourist’s behavior heavily depends on the data set’s configuration and the starting site. In computational terms, the tourist’s movements are entirely realized by means of a neighborhood table. This table is constructed by ordering all the data items in relation to a specific site. This procedure is performed for every site of the data set.

Each tourist walk can be decomposed in two terms: (i) the initial transient part of length \( t \) and (ii) a cycle (attractor) with period \( c \). Figure 2 shows an illustration of a tourist walk with \( \mu = 1 \), i.e., the walker always goes to the nearest neighbor. In this case, one can see that the transient length is \( t = 3 \) and the cycle length \( c = 6 \).

Considering the attractor or cycle period as a walk section that begins and ends at the same site of the data set may lead one to think that, once the tourist visits a specific site, a new visit to it would configure an attractor. However, during a walk, a site may be re-visited without configuring an attractor. For instance, if we had chosen a \( \mu = 6 \) for the walk in Fig. 2 the re-visit that the tourist performs on the site 4 would have not configured an attractor, since the site 5 would still be forbidden to be visited again; hence, the tourist would be compelled to visit another site. This characteristic enables sophisticated trajectories over the data set, at cost of also increasing the difficulty of detecting an attractor.

A note that is worth pointing out is that, in the majority of the works related to these walks [23], [30], [31], the tourist may visit any other site other than the ones contained in its memory window. As \( \mu \) increases, there is a significant chance that the walker will begin performing large jumps in the data set, since the neighborhood is most likely to be already visited in its entirety within the time frame \( \mu \). As we will show in this work, in the context of classification, this is an undesirable characteristic that can be simply avoided by using a graph representation of the input data. In this way, the walker is only permitted to visit vertices, represented now by the sites, that are in its connected neighborhood (link). With this modified mechanism, it is most probable that, for large values of \( \mu \), depending on the network configuration, the walker will get trapped within a vertex, not being able to further visit other vertices of the neighborhood. In this scenario, we say that the walk only had a transient part and the cycle period is null \( (c = 0) \). Therefore, the tourist walks approach applied to a networked environment is a relatively new approach taken here. Additionally, its utilization for discovering patterns in a network is a totally novel scheme in the literature.

III. Model Description

In this section, a useful set of notations is presented, along with the premises of the underlying hybrid classification framework. Next, we give a quick overview on the particularities of the original hybrid classification framework [4]. Finally, the high level classification based on tourist walks is formally introduced, as well as an overview of its algorithm.

A. Notations and Premises

The hybrid classifier is designed to work in a supervised learning environment. In the following, some mathematical notations and premises are discussed.

Consider that \( \mathcal{X}_{\text{training}} = \{(x_1, y_1), \ldots, (x_l, y_l)\} \subset \mathcal{X} \times \mathcal{L} \) denotes the training set, which is composed of \( l \) labeled training instances. Each training instance, \( x_i \in \mathcal{X} \), is given a discrete label or target \( y_i \in \mathcal{L} \). Furthermore, each training instance is described by a \( d \)-dimensional vector, i.e., \( x_i = (f_1, \ldots, f_d) \), where each entry symbolizes a feature or descriptor of that item.

The goal here is to construct a hypothesis, in a way that the classifier maps \( x \mapsto y \). Commonly, the constructed classifier is checked with regard to its prediction power by submitting it to a test set \( \mathcal{X}_{\text{test}} = \{x_{i+1}, \ldots, x_{i+u}\} \), in which labels are not provided. In this case, each data item is called test instance. For an unbiased learning, the training and test sets must be disjoint, i.e., \( \mathcal{X}_{\text{training}} \cap \mathcal{X}_{\text{test}} = \emptyset \).

B. Overview of the Hybrid Classification Framework

In this section, we review the hybrid classification framework [4]. Specifically, in the following sections, the particularities of the training and classification phases, and the general definition of the classification scheme are discussed.

1) Training Phase: In this phase, the data in the training set are mapped into a graph \( \mathcal{G} \) using a network formation technique \( g : \mathcal{X}_{\text{training}} \mapsto \mathcal{G} = (\mathcal{V}, \mathcal{E}) \), where \( \mathcal{V} = \{1, \ldots, V\} \) is the set of vertices and \( \mathcal{E} \) is the set of edges. Each vertex in \( \mathcal{V} \) represents a training instance in \( \mathcal{X}_{\text{training}} \). As it will be described later, the pattern formation of the classes will be
extracted by using the complex topological features of this networked representation. Therefore, the network construction is vital for the prediction produced by the high level classifier.

In this stage, we first construct a network component for each class of the vector-based training set. The strategy to create edges depends on the type of region in which each vertex is. When the region is sparse, we utilize a $k$-nearest neighbor ($k$-NN) approach, whereas, when it is dense, we employ the $\epsilon$-radius technique. While the $k$-NN sets up an edge between the $k$ most similar vertices and the reference vertex, the $\epsilon$-radius method creates a link to whichever vertex that is within a predefined distance with radius $\epsilon$. The way that we classify a region as dense or sparse is by checking whether, within a circular region $\epsilon$ centered in the reference vertex, there are more than $k$ vertices of the same class as the reference vertex. If so, the region is classified as dense and the $\epsilon$-radius is selected; otherwise, the $k$-NN is elected. During the network formation, we are only permitted to create edges between vertices of the same class. The reason why we employ a $k$-NN in sparse region is to prevent the appearance of multiple graph components representing the same class. Therefore, we expect that this process will form isolated class components, in a way that each class is guaranteed to have a single and unique component representing it.

2) Classification Phase: In the classification phase, the unlabeled data items in the $X_{test}$ are presented to the classifier one by one. In contrast to the training phase, the class labels of the test instances are unknown. In view of that, we keep utilizing the edge formation strategy previously introduced but with slight changes: now, we do not consider the labels of the neighboring vertices. All the other technicalities remain the same. This prevents test instances from becoming singleton vertices, provided that $k > 0$.

With respect to the high order of learning, once the data item is inserted, each class analyzes, in isolation, the impact of the insertion of this data item on its respective class component by using a number of complex topological features. In the proposed high level model, each class retains an isolated graph component. Each of these components calculates the changes that occur in its pattern formation with the insertion of this test instance. If slight or no changes occur, then it is said that the test instance is in compliance with that class pattern. As a result, the high level classifier yields a great membership value for that test instance on that class. Conversely, if these changes dramatically modify the class pattern, then the high level classifier produces a small membership value on that class. These changes are quantified via network measures, each of which numerically translating the organization of the component from a local to global fashion. As we will see, we will extract information generated from the dynamical process of a tourist walker in a networked environment in an intuitive manner, such as to capture the pattern formation of the network in a local to global basis.

3) General Hybrid Classification Framework: The hybrid classification framework $F$ consists of a convex combination of two orthogonal terms, where each of which renders an uncorrelated vision about the data items, as follows:

i. A low level classifier, for instance, a decision tree, SVM, or a $k$-NN classifier. The vision that it stresses is the physical similarities among the data;

ii. A high level classifier, which is responsible for classifying a test instance according to its organizational or semantic meaning with the data. The vision that it values most is the pattern compliance of new test instances with the existing structure built up in the training process.

Mathematically, the membership of the test instance $x_i \in X_{test}$ with respect to the class $j \in \mathcal{L}$ yielded by the hybrid framework, here written as $F_i^{(j)}$, is given by:

$$F_i^{(j)} = (1 - \lambda)L_i^{(j)} + \lambda H_i^{(j)}.$$  

where $L_i^{(j)}, H_i^{(j)} \in [0, 1]$ represent the membership of the test instance $x_i$ towards class $j$ produced by the low and high level classifiers, respectively, and $\lambda \in [0, 1]$ is the compliance term, which plays the role of counterbalancing the classification decisions supplied by both low and high level classifiers. Note that, when $\rho = 0$, (1) reduces to a common low level classifier.

A test instance $x_i$ receives the label of the class $j \in \mathcal{L}$ that maximizes (1). Mathematically, the estimated label of $x_i$, $\hat{y}_x$, is decided according to the following expression:

$$\hat{y}_x = \arg \max_{j \in \mathcal{L}} F_i^{(j)}.$$  

Note that the predictions produced by both classifiers are combined via a linear combination to derive the prediction of the high level framework (meta-learning). Once the test instance $x_i$ gets classified, it is either discarded or incorporated to the training set with the corresponding predicted label. In the second case, only the edges created between the test instance and the class that it belongs to are maintained. Note that, in any of the two situations, each class is still represented by a single graph component.

C. Deriving the High Level Classification Technique Using Tourist Walks

Equation (1) supplies a general framework for the hybrid classification process, in the sense that various supervised data classification techniques can be brought into play. The first term of (1) is rather straightforward to implement, since it can be any traditional classification technique. The literature provides a myriad of supervised data classification techniques. Some of these include graph-based methods, decision trees, SVM and its variations, neural networks, Bayesian learning, among many others. However, to our knowledge, little has been done in the area of classifiers that take into account the patterns or organizational features inherently hidden into the relationships among the data items. Thus, we now proceed to a detailed analysis of the proposed high level classification term $H$.

Motivated by the intrinsic ability of describing topological structures among the data items, we propose a network-based (graph-based) technique for the high level classifier $H$. Specifically, the inference of pattern formation within the data is processed using the generated network. In order to do so,
the following structural constraints must be satisfied for any constructed network:

i. Each class is an isolated subgraph (graph component);
ii. Each class retains a representative and unique graph component.

As we have drawn attention to, the pattern formation of the data is quantified through a combination of network measures generated by a tourist walker in a networked environment. These measures are chosen in a way to cover relevant high level aspects of the class component. One can conceive these dynamical values as true network measures representing each class component. Having in mind the basic concepts revolving around tourist walks, the decision output of the high level classifier is given by:

$$H_i^{(j)} = \frac{\sum_{\mu=0}^{\mu_c} \left[ \alpha_t \left( 1 - T_i^{(j)}(\mu) \right) + \alpha_c \left( 1 - C_i^{(j)}(\mu) \right) \right]}{\sum_{\mu=0}^{\mu_c} \left[ \alpha_t \left( 1 - T_i^{(g)}(\mu) \right) + \alpha_c \left( 1 - C_i^{(g)}(\mu) \right) \right]}$$  \hspace{1cm} (3)

where \( \mu_c \) is a critical value that indicates the maximum memory length of the tourist walks, \( \alpha_t, \alpha_c \in [0, 1] \) are user-controllable coefficients that indicate the influence of each network measure in the process of classification, \( T_i^{(j)}(\mu) \) and \( C_i^{(j)}(\mu) \) are functions that depend on the transient and cycle lengths, respectively, of the tourist walk applied to the \( i \)th data item with regard to the class \( j \). These functions are responsible for providing an estimative whether or not the data item \( i \) under analysis possesses the same patterns of component \( j \). The denominator in (3) has been introduced solely for normalization matters. Indeed, in order to (3) to be a valid convex combination of network measures, \( \alpha_t \) and \( \alpha_c \) must be chosen such as to satisfy \( \alpha_t + \alpha_c = 1 \).

Regarding \( T_i^{(j)}(\mu) \) and \( C_i^{(j)}(\mu) \), they are given by the following expressions:

$$T_i^{(j)}(\mu) = \Delta t_i^{(j)}(\mu)p_i^{(j)}$$
$$C_i^{(j)}(\mu) = \Delta c_i^{(j)}(\mu)p_i^{(j)}$$  \hspace{1cm} (4)

where \( \Delta t_i^{(j)}(u), \Delta c_i^{(j)}(u) \in [0, 1] \) are the variations of the transient and cycle lengths that occur on the component representing class \( j \) if \( i \) joins it and \( p_i^{(j)} \in [0, 1] \) is the proportion of data items pertaining to class \( j \). Remembering that each class has a component representing itself, the strategy to check the pattern compliance of a test instance is to examine whether its insertion causes a great variation of the network measures representing the class component. In other words, if there is a small change in the network measures, the test instance is in compliance with all the other data items that comprise that class component, i.e., it follows the same pattern as the original members of that class. On the other hand, if its insertion is responsible for a significant variation of the component’s network measures, then probably the test instance may not belong to that class. This is exactly the behavior that (3) together with (4) propose, since a small variation of \( f(u) \) causes a large membership value output by \( H \); and vice versa.

In the following, we explain how to compute \( \Delta t_i^{(j)}(\mu) \) and \( \Delta c_i^{(j)}(\mu) \) that appear in (4). Firstly, we need to numerically quantify the transient and cycle lengths of a component. Since the tourist walks are strongly dependent on the starting vertices, for a fixed \( \mu \), we perform tourist walks initiating from each one of the vertices that are members of a class component. The transient and cycle lengths of the \( j \)th component, \( \langle t_i^{(j)}(\mu) \rangle \) and \( \langle c_i^{(j)}(\mu) \rangle \), are simply given by the average transient and cycle lengths of all its vertices, respectively. In order to estimate the variation of the component’s network measures, consider that \( x_i \in \mathbb{X}_{\text{test}} \) is a test instance. In relation to an arbitrary class \( j \), we virtually insert \( x_i \) into component \( j \) using the network formation technique that we have seen, and recalculate the new average transient and cycle lengths of this component. We denote these new values as \( \langle t_i^{(j)}(\mu) \rangle \) and \( \langle c_i^{(j)}(\mu) \rangle \), respectively. This procedure is performed for all classes \( j \in \mathbb{L} \). It may occur that some classes \( u \in \mathbb{L} \) will not share any connections with the test instance \( x_i \). Using this approach, \( \langle t_i^{(j)}(\mu) \rangle = \langle t_i^{(j)}(\mu) \rangle \) and \( \langle c_i^{(j)}(\mu) \rangle = \langle c_i^{(j)}(\mu) \rangle \), which is undesirable, since this configuration would state that \( x_i \) complies perfectly with class \( u \). In order to overcome this problem, a simple post-processing is necessary: For all components \( u \in \mathbb{L} \) that do not share at least 1 link with \( x_i \), we deliberately set \( \langle t_i^{(j)}(\mu) \rangle \) and \( \langle c_i^{(j)}(\mu) \rangle \) to a high value.

This high value must be greater than the largest variation that occurs in a component which shares a link with the data item under analysis. One may interpret this post-processing as a way to state that \( x_i \) does not share any pattern formation with class \( u \), since it is not even connected to it.

With all this information at hand, we are able to calculate \( \Delta t_i^{(j)}(\mu) \) and \( \Delta c_i^{(j)}(\mu) \), \( \forall j \in \mathbb{L} \), as follows:

$$\Delta t_i^{(j)}(\mu) = \frac{\left| \langle t_i^{(j)}(\mu) \rangle - \langle t_i^{(j)}(\mu) \rangle \right|}{\sum_{u \in \mathbb{L}} \left| \langle t_i^{(j)}(\mu) \rangle - \langle t_i^{(j)}(\mu) \rangle \right|}$$
$$\Delta c_i^{(j)}(\mu) = \frac{\left| \langle c_i^{(j)}(\mu) \rangle - \langle c_i^{(j)}(\mu) \rangle \right|}{\sum_{u \in \mathbb{L}} \left| \langle c_i^{(j)}(\mu) \rangle - \langle c_i^{(j)}(\mu) \rangle \right|}$$  \hspace{1cm} (5)

where the denominator is introduced only for normalization matters. According to (5), for insertions that result in a considerable variation of the component’s transient and cycle lengths, \( \Delta t_i^{(j)}(\mu) \) and \( \Delta c_i^{(j)}(\mu) \) will be high. In view of (4), \( T_i^{(j)}(\mu) \) and \( C_i^{(j)}(\mu) \) are expected to be also high, yielding a low membership value predicted by the high level classifier \( H_i^{(j)} \), as (3) reveals. On the other hand, for insertions that do not significantly interfere in the pattern formation of the data, \( \Delta t_i^{(j)}(\mu) \) and \( \Delta c_i^{(j)}(\mu) \) will be low, and, as a result, \( T_i^{(j)}(\mu) \) and \( C_i^{(j)}(\mu) \) are expected to be also low, producing a high membership value for the high level classifier \( H_i^{(j)} \), as (3) exposes.

The network-based high level classifier quantifies the variations of the transient and cycle lengths of tourist walks with limited memory \( \mu \) that occur in the class components when a test instance artificially joins each of them in isolation. According to (3), this procedure is performed for several values of the memory length \( \mu \), ranging from 0 (memoryless) to a critical value \( \mu_c \). This is done in order to capture complex
patterns of each of the representative class components in a local to global fashion. When \( \mu \) is small, the walks tend to possess a small transient and cycle parts, so that the walker does not wander far away from the starting vertex. In this way, the walking mechanism is responsible for capturing the local structures of the class component. On the other hand, when \( \mu \) increases, the walker is compelled to venture deep into the component, possibly very far away from its starting vertex. In this case, the walking process is responsible for capturing the global features of the component. In sum, the fundamental idea of the high level classifier is to make use of a mixture of local and global features of the class components by means of a combination of tourist walks with different values of \( \mu \).

Finally, we intuitively explain the role of \( p^{(j)} \in [0,1] \) in (4), i.e., the relative size of each component in the graph. In real-world databases, unbalanced classes are usually encountered. In general, a database frequently encompasses several classes of different sizes. A great portion of the network measures is very sensitive to the size of the components. In an attempt to soften this problem, we introduce in (4) the term \( p^{(j)} \), which is the proportion of vertices that class \( j \) has. Mathematically, it is given by:

\[
p^{(j)} = \frac{1}{V} \sum_{u=1}^{V} \mathbb{1}_{\{y_u = j\}},
\]

where \( V \) is the number of vertices and \( \mathbb{1}_{\{\cdot\}} \) is the indicator function that yields 1 if the argument is logically true, or 0, otherwise. In view of the introduction of this mechanism, we expect to obviate the effects of unbalanced classes in the classification process.

D. Algorithm

For didactic purposes, Algorithm 1 enumerates the sequence of steps needed to perform a high level classification of a single test instance, according to the rules that we have described in this section.

Algorithm 1: An overview of the high level classification procedure based on tourist walks for a single test instance.

1) Construct a set of network components \( \mathcal{G} = \{ \mathcal{G}_1, \ldots, \mathcal{G}_L \} \), for each class from the vector-based training set by using the combined \( k \)-NN and \( \epsilon \) rules from the training phase;
2) Calculate the average transient and cycle lengths for each network’s component, representing a class of training data;
3) Insert a test instance \( x_i \) into the formed graph by using the \( k \)-NN and \( \epsilon \) rules from the classification phase;
4) Calculate the new average transient and cycle lengths for each network’s component that received at least one link from the test instance;
5) Calculate the transient and cycle length variations by using Eq. (4);
6) Produce the high level decision value by using Eq. (3).

E. Linking the High Level Classifier based on Tourist Walks and the General Framework introduced in [4]

One may wonder how the high level classifier based on tourist walks given by (5) is plugged into the general framework for high level classification introduced in [4]. The following proposition links both approaches.

Proposition 1. The high level classifier based on tourist walks, whose decision equations are:

\[
H_i^{(j)} = \frac{\sum_{\mu_c=0}^{2\mu_c+2} v(\mu)(1 - V_i^{(j)}(\mu))}{\sum_{\mu_c=1}^{2\mu_c+2} v(\mu)(1 - V_i^{(j)}(\mu))},
\]

is a particular form of the generic high level framework given in [4] Eqs. (5) and (7):

\[
H_i^{(j)} = \frac{\sum_{\mu_c=0}^{M} \alpha(1 - T_i^{(j)}(\mu)) + \alpha_c(1 - C_i^{(j)}(\mu))}{\sum_{\mu_c=0}^{M} \alpha(1 - T_i^{(j)}(\mu)) + \alpha_c(1 - C_i^{(j)}(\mu))},
\]

where:

\[
v(\mu) = \begin{cases} 
\alpha_t & \text{if } \mu \leq \mu_c + 1 \\
\alpha_c & \text{if } \mu > \mu_c + 1
\end{cases},
\]

and

\[
V_i^{(j)}(\mu) = \begin{cases} 
T_i^{(j)}(\mu) & \text{if } \mu \leq \mu_c + 1 \\
C_i^{(j)}(\mu) & \text{if } \mu > \mu_c + 1
\end{cases},
\]

are piecewise functions defined over the domain \( \{1, \ldots, 2\mu_c + 2\} \). In view of [4, Eq. (6)], one must have that:

\[
\sum_{\mu_c=1}^{2\mu_c+2} v(\mu) = 1 \Rightarrow (\mu_c + 1)(\alpha_t + \alpha_c) = 1.
\]

Moreover, by equivalence, comparing the upper limits of the summations of (9) and (11), one has that:

\[
m = 2\mu_c + 2.
\]

Using [4, Eq. (6)] on (14), one has:
\[(\mu_c + 1)(\alpha_i + \alpha_c) = \sum_{\mu=1}^{2\mu_c+2} v(\mu) = 1 = \sum_{u=1}^{m} \alpha(u) = \sum_{u=1}^{2\mu_c+2} \alpha(u).\]  

(16)

Therefore, by polynomial equivalence, one can take:

\[
\alpha(u) = v(u) = \begin{cases} 
\alpha_i & \text{if } u \leq \mu_c + 1 \\
\alpha_c & \text{if } u > \mu_c + 1 .
\end{cases}
\]  

(17)

Using the same reasoning, we have that:

\[
f^{(j)}_i(u) = V^{(j)}_i(u) = \begin{cases} 
T^{(j)}_i(u) & \text{if } u \leq \mu_c + 1 \\
C^{(j)}_i(u) & \text{if } u > \mu_c + 1 .
\end{cases}
\]  

(18)

Finally, by looking at (8) and (10), we can infer that the tourist walk based classifier can be coupled into the hybrid framework by taking \(2\mu_c + 2\) network measures, such that the first \(\mu_c + 1\) are the transient lengths with increasing memory lengths, each of which weighted by \(\alpha_i\), and the remaining \(\mu_c + 1\) are cycle lengths with increasing memory lengths, each of which weighted by \(\alpha_c\). In view of this, the high level classifier based on tourist walks is, in fact, a particular implementation of the generic high level classifier.

IV. COMPUTER SIMULATIONS

In this section, we present computer simulation results in order to assess the effectiveness of the proposed hybrid classification model based on tourist walks. The error estimation method in these simulations is set to be the stratified 10-fold cross-validation.

A. Motivating and Illustrative Examples

In this section, we provide simple examples with the goal of showing the mechanics of the proposed hybrid classification technique. For this end, we simplify the parameters selection procedure, as follows: the weights given for the transient and cycle lengths are the same, i.e., \(\alpha_i = \alpha_c = 0.5\); and the critical tourist walk length \(\mu_c\) is set as being the size of the smallest class in the problem. Here, we design particular situations in which the use of the high order of learning is welcomed, since the reliance on mere physical measures would probably deceive a low level classifier.

As an introductory example, consider the synthetic data set supplied in Fig. 1 in the Introduction section. Our goal is to classify the two triangle-shaped data items (\(X^{(i)}_{\text{test}}\)). The items in \(X^{(i)}_{\text{test}}\) are inserted one by one using \(\epsilon = 0.05\) and \(k = 2\) to construct the network components before and after each insertion. For the purposes of this simulation, once a test instance is classified, it is discarded. With respect to the low level classifier, a fuzzy SVM classifier is utilized [32], which is equipped with optimization method criterion defined as the Karush-Kuhn-Tucker violation fixed at \(10^{-3}\) (the same condition suggested by [33]). The following four kernels are used: (i) Linear: \(u \cdot v\), (ii) RBF: \(\exp(-\gamma \| u - v \|^2)\), (iii) Sigmoid: \(\text{tanh}(\gamma u \cdot v + c)\), and (iv) Polynomial: \((c + u \cdot v)^d\). Additionally, the cost parameter \(C\) must be set in all simulations. For each low level classifier, we apply a fine-tuning parameter phase in order to choose a model with optimized classification results.

The networked data set exhibited in Fig. 1 has 2 classes, a red or circular-shaped class (16 vertices) and a blue or square-shaped class (58 vertices). Consequently, the proportions of vertices pertaining to each class are: \(p^{(\text{red})} = 21.64\%\) and \(p^{(\text{blue})} = 78.36\%\). Clearly, while the circular-shaped (red) class carries a perceivable pattern, which, geometrically speaking, is a lozenge-shaped lattice disposed in a 2-dimensional space, the blue (“square”) class does not indicate a strong pattern as the first class. In this particular situation, we will see that the results produced by the SVM by itself are not satisfactory. Even though the kernel is responsible for spawning the data items into a higher space via a nonlinear transformation, the test instances in question may only be distinguished by using semantic knowledge, i.e., the topological features of the classes. It is worth stressing that special kernels could be created in order to solve specific data distributions, but they would be totally peculiar for the problems at hand, making their usability constrained to a very few real situations. On the other side, the proposed high level technique captures pattern formation of the data in a general manner.

Table I reports the classification results of the two existing test instances for all the 4 kernels. For each kernel, the final prediction is given for three different values of the compliance term \(\lambda: \lambda = 0\) (pure SVM), \(\lambda = 0.5\) (equal weight for the decision of the SVM and the high level classifier), and \(\lambda = 0.9\) (a high weight for the high level classifier and a low weight for the SVM). We can clearly see that a pure SVM with well-known kernels is not able to correctly classify the data items shown in Fig. 1. However, if we make our final decision based on a mixture of the SVM and the high level classifier, one can verify that correct results can be obtained, i.e., the two triangle-shaped data items are classified to red or circular-shaped class.

![Fig. 3. Minimum value of the compliance term, \(\lambda_{\text{min}}\), that results in the correct classification of the missing test instances. Traditional techniques would definitely fail in correctly classifying the straight line that diametrically crosses the densely connected component pertaining to the blue or “square” class.](image-url)
shaped class. In the figure, one can see that there is a segment of line representing the red or circular-shaped class (9 vertices) and also a condensed rectangular class outlined by the blue or square-shaped class (1000 vertices). The network formation in the training and test phases uses $k = 1$ and $\epsilon = 0.07$ (this radius covers, for any vertex in the straight line, 2 adjacent vertices, except for the vertices in each end). The fuzzy SVM technique with RBF kernel ($C = 2^2$ and $\gamma = 2^{-1}$) is employed as the traditional low level classifier. The task is to classify the 14 test instances depicted by the big triangle-shape items from left to right. After a test instance is classified, it is incorporated into the training set with the corresponding predicted label. The graphic embedded in Fig. 3 shows the minimum required value of $\lambda_{\text{min}}$ for which the the triangle-shaped items are classified as members of the red or circular-shaped class. This graphic is constructed according to the triangle-shaped element that is exactly at the same position with respect to the $x$-axis in the scatter plot drawn above. For example, the first triangle-shaped data item can be correctly classified if one chooses $\lambda \geq \lambda_{\text{min}} \approx 0.37$. The second and third data items would require at least $\lambda_{\text{min}} = 0.81$ and $\lambda_{\text{min}} = 0.96$, respectively, and so on. Specifically, as the straight line crosses the condensed region of the blue or square-shaped class, the compliance term approaches $\lambda \to 1$, since one cannot establish its decision based on the low level classifier, because it would erroneously decide favorable to the blue or square-shaped class.

B. Parameter Sensitivity Analysis

In this section, we will perform several simulations with the goal of better understanding the influence of each parameter in the model.

1) Influence of the Compliance Term for Different High Level Classifiers: In this section, we study the influence of the compliance term $\lambda$ for three different types of high level classifiers, as follows: (i) one constructed solely using the component’s cycle length; (ii) one built exclusively using the component’s transient length; and (iii) the best weighted combination of these two measures. The optimization process is conducted by finding $\alpha_t \times \alpha_c \in \{0, 0.1, \ldots, 1\} \times \{0, 0.1, \ldots, 1\}$ (search space), subject to $\alpha_t + \alpha_c = 1$, which results in the highest accuracy rate of the model. Here, we show that, in general, the transient and cycles lengths are not correlated. The objective of this section is to make this point clear and demonstrate that, when they act together, they are able to produce better accuracy rates. In order to conduct these tests, classes with Gaussian distributions are used. The network in the classification phase is constructed with $k = 1$ and $\epsilon = 0.05$. The same $\epsilon$ is employed in the classification phase. The similarity measure is simply given by the reciprocal of the Euclidean measure. For the traditional classifier, the fuzzy SVM with RBF kernel is employed with $C = 2^4$ and $\gamma = 2^{-2}$. Finally, this process is repeated 100 times and the mean and the corresponding standard deviation for each value of $\lambda$ is reported.

Now, we advance to our first experiment in which we consider a rather simple classification scenario, where the classes are completely separated from each other, as Fig. 4a exhibits. Figure 4b depicts the accuracy rate of the proposed classifier vs. $\lambda$ for the three types of high level classifiers discussed before. The best weighted combination of the cycle and transient lengths is $\alpha_t = 0.6$ and $\alpha_c = 0.4$. One can see that the high level classifier constructed by the combination of the transient and cycle lengths is able to outperform the other high level classifiers constructed using only one of these measures. In addition, when $\lambda = 0$ (only the usage of the traditional classifier), almost no wrong labels are assigned. On the other hand, as $\lambda$ increases, the accuracy rate of the proposed technique starts to monotonically decrease. This is predictable, since the two classes are similar; hence, the network measures associated to each representative component are almost equivalent. Arguments such that there is no mixture between the classes and the spatial correlations of both classes reinforce this phenomenon. Therefore, it is natural that the high level classifier will become confused in classifying these data items under such conditions. This example shows that only the high level classifier is insufficient to get good classification results.

Next, let us examine the problem of classification illustrated in Fig. 4c, where the two classes slightly collide with each other. As a result of this phenomenon, a conflicting region is constituted. The data items which reside in this region are similar reinforce this phenomenon. Therefore, it is natural that the high level classifier will become confused in classifying these data items under such conditions. This example shows that only the high level classifier is insufficient to get good classification results.

| Kernel | Optimized Parameters | $\lambda = 0$ | $\lambda = 0.5$ | $\lambda = 0.9$ |
|--------|----------------------|--------------|---------------|---------------|
|        | $C$ | $\gamma$ | $\epsilon$ | Red Class | Blue Class | Red Class | Blue Class | Red Class | Blue Class |
| Left-most | Linear | $2^1$ | N/A | N/A | 0.47 | 0.60 | 0.54 | 0.46 | 0.64 | 0.36 |
|         | RBF | $2^0$ | $2^{-6}$ | N/A | N/A | 0.47 | 0.53 | 0.60 | 0.40 | 0.67 | 0.33 |
|         | Sigmoid | $2^0$ | $2^{-1}$ | -4.5 | N/A | 0.46 | 0.54 | 0.59 | 0.41 | 0.67 | 0.33 |
|         | Polynomial | $2^1$ | 3.0 | 4 | 0.44 | 0.56 | 0.57 | 0.44 | 0.65 | 0.35 |
| Right-most | Linear | $2^{-1}$ | N/A | N/A | 0.24 | 0.76 | 0.40 | 0.60 | 0.51 | 0.49 |
|          | RBF | $2^{-1}$ | $2^{-5}$ | N/A | N/A | 0.29 | 0.71 | 0.45 | 0.55 | 0.57 | 0.43 |
|          | Sigmoid | $2^{-1}$ | $2^{-3}$ | -6.5 | N/A | 0.31 | 0.69 | 0.47 | 0.53 | 0.58 | 0.42 |
|          | Polynomial | $2^{-1}$ | $2^{-5}$ | 5.0 | 2 | 0.28 | 0.72 | 0.44 | 0.56 | 0.57 | 0.43 |
the model. Specifically, the highest accuracy rate occurs when $\lambda \in \{0.15, 0.2\}$. The region of coalescence of both classes influences in the network construction, in such a way that the representative components of each class become slightly different. These arguments explain that a little relevance to the high level classifier decision can cause the accuracy rate to increase, since the network measures describing each component are slightly distinct.

We now proceed to inspect the classification problem proposed in Fig. 4c. In this particular situation, the two classes heavily collide with each other. Due to that, the two classes become almost indistinguishable. In consequence of the network formation technique, two very distinct and representative components will arise from these two classes. In this special scenario, the two components that comprise the network are expected to be possess different network properties, i.e., some unique structural pattern for each class is hoped for emerging. Figure 4c depicts the accuracy rate reached by the classifier against various values of $\lambda$. The best weighted combination here is $\alpha_t = 0.2$ and $\alpha_c = 0.8$. One can see that the accuracy rate keeps on increasing until a high value of the compliance term, namely $\lambda = 0.65$, where it achieves 65%, against 54% with $\lambda = 0$ for the high level classifier combining the transition and cycle lengths. This phenomenon is precisely explained by the complex structural patterns that each component exhibits, in which merely traditional classifiers would not be able to capture; therefore, a heavy weight on the high level classifier decision was responsible for this significant increase in the accuracy rate.

2) Influence of the Critical Memory Length: As we have seen, the high level classifier makes its prediction by using combinations of several tourist walks with memory lengths $\{0, 1, \ldots, \mu_c\}$. A natural question that arises is: is it really necessary to perform the computation of tourist walks with $\mu$ ranging from 0 to a maximum feasible number, i.e., the number of vertices in a component? In this section, we will empirically show that, usually, it is not necessary to perform all these computations. For this end, we will reinforce this argument by verifying this behavior in a synthetic data set and two real-world data sets.

We start out by revisiting the synthetic data set given in Fig. 1 shown in the Introduction section. The behavior of the transient and cycle lengths are displayed in Figs. 5a and 5b respectively. One can note that these two dynamical measures vary as $\mu$ increases up to a point where they reach a steady region in which no more oscillating is verified. When this happens, we say that the dynamics of the tourist walk have reached a saturation point, in the sense that further computations of tourist walks with larger memory lengths are redundant. Moreover, one can see that this saturation point is reached very quickly in relation to the graph components.

Continuing our exploration of this interesting phenomenon, we now turn our attention to two well-known data sets from the UCI Machine Learning Repository [34]: Iris (balanced classes) and Wine (unbalanced classes). Consider Figs. 6a and 6b where it is depicted the transient and cycle lengths for the classes of the Iris data set, and Figs. 7a and 7b in which it is displayed the same information for the Wine data set. With respect to the transient length behavior, we can see that, for both data sets, as $\mu$ increases, the transient length also increases. However, when $\mu$ is sufficiently large, the components’ transient lengths settle down in a flat region (like the previous case). On the other hand, for both data sets, the cycle length behavior is rather interesting, which can be
roughly divided in three different regions: (i) for a small \( \mu \), it is directly proportional to \( \mu \); (ii) for intermediate values of \( \mu \), it is inversely proportional to \( \mu \); and (iii) for sufficiently large values of \( \mu \), it also settles down in a steady region. One can interpret these results as follows:

- When \( \mu \) is small, it is very likely that the transient and cycle parts will also be small, because the memory of the tourist is very limited. We can conceive this as a walk with almost no restrictions;
- When \( \mu \) assumes an intermediate value, the transient length keeps increasing but the cycle length reaches a peak and starts to decrease afterwards. This peak characterizes the topological complexity of the component and varies from one to another. Hence, this is the most important region for capturing pattern formation of the class component by using the topological structure of the network.
- When \( \mu \) is large, the tourist has a greater chance of getting trapped in a vertex of the graph, once all the neighborhood of the visited vertex is contained in the memory window \( \mu \). In this scenario, the transient length is expected to be very high and the cycle length, null. This phenomenon explains the steady regions in Figs. 6a and 6b. In this region, we can say that the tourist walks have already covered all the global aspects of the class component, and increasing the memory length \( \mu \) will not capture any new topological features or pattern formation of the class. In this scenario, it is said that the tourism walks have completely described the topological complexity of the class component (saturation). In view of this, the calculation of tourist walks by further increasing \( \mu \) is redundant.

This analysis suggests that the accuracy of the high level classifier may not change given that we choose a \( \mu_c \) residing near these steady regions. This means that higher values for \( \mu_c \) will only cause redundant computations and the accuracy will not be enhanced. In order to check this, Figs. 8a and 8b reveal the behavior of the hybrid classification framework for different values of \( \mu_c \). We have used three distinct compliance terms, namely \( \lambda \in \{0, 0.05, 0.6\} \) for the Iris data set and \( \lambda \in \{0, 0.06, 0.7\} \) for the Wine data set. When \( \lambda = 0 \), only the low level classifier is used, in such a way that the value of \( \mu_c \) is irrelevant, since the high level term is disabled. With respect to the other cases, when \( \mu_c \geq 20 \), the model provides the same accuracy rates for both data sets, confirming our prediction that, once it reaches the steady region where the transient and cycle lengths settle down, subsequent increases of \( \mu_c \) do not change the accuracy of the model. In practical terms, according to our simulations, fixing the critical length \( \mu_c \) as \( 10 - 30\% \) of the component’s size is enough to get satisfactory results.

C. Simulations on Real-World Data Sets

In this section, we will apply the proposed framework to several well-known UCI data sets. The most relevant metadata of each data set is given in Table II. For a detailed description, refer to [34]. Concerning the numerical attributes, the reciprocal of the Euclidean distance is employed. For categorical examples, the overlap similarity measure [35] is utilized. All data sets are submitted to a standardization pre-processing step.

Here, the high level classifier is composed of the best weighted combination of transient and cycle lengths. The optimization process is done by encountering \( \alpha_c \times \alpha_c \in \{0, 0.1, \ldots, 1\} \times \{0, 0.1, \ldots, 1\} \) (search space), subjected to \( \alpha_c + \alpha_c = 1 \), which result in the highest accuracy rate of the model. The critical memory length is fixed to \( \mu_c = 0.3n_{\max} \), where \( n_{\max} \) indicates the size of the largest component. The parameter optimization results are given in last two columns of Table II.
Here, we will deal with two kinds of high level classifiers: (i) one in which the tourist walks are performed in a network constructed from the vector-based data set and (ii) one in which the tourist walks are realized in a lattice, i.e., the tourist is free to visit any other data site apart from the ones in the memory window \( \mu \). In the latter case, it is expected that the walker will perform long jumps in the data set once the memory length \( \mu \) assumes large values. It will be verified that such mechanism is not very welcomed in classification tasks. Furthermore, this serves as a strong argument for the employment and introduction of network-based high level classifiers.

Table III reports the results obtained by the proposed technique on the data sets listed in Table II. For comparison purposes, we evaluate the performance of the framework against different low level classifiers: Bayesian networks [36], Weighted \( k \)-nearest neighbors [37], Multi-layer perceptron (MLP) [38], Multi-class SVM (M-SVM) [32], [39]. The outcome of each algorithm is estimated by the average value over hundred runs of a stratified 10-fold cross-validation process. Also, for each result, three different types of results are indicated as follows:

- **“Pure”** row: only the low level classifier is utilized (\( \lambda = 0 \)). In this case, inside the parentheses are indicated the best parameters obtained from the optimization process for each technique;
- **“Networkless”** row: a mixture of low and high level classifiers is employed. The value inside the parentheses indicates the best compliance term \( \lambda \). The high level classifier is constructed using the best weighted combination of transient and cycle lengths with the weights respecting Table II. Moreover, the tourist walks are performed in a networkless environment, i.e., the tourist can visit any other site (data item) apart from those contained in the memory window \( \mu \);
- **“Network”** row: the same setup as before, but the tourist walks are conducted on a networked environment. In this case, the tourist can only visit vertices (items) that are in the neighborhood and not in the memory window \( \mu \). Here, the network in the training phase is built using \( k = 1 \) and \( \epsilon = 0.03 \). The values inside the parentheses exhibit: the \( \epsilon \) used in the classification phase and the best compliance term \( \lambda \), respectively.

For the sake of clarity, take the first entry of Table III. The pure low level classifier Bayesian Networks achieved an accuracy rate of \( 57.8 \pm 2.6 \) (\( \lambda = 0 \)). However, if we use the proposed technique in a networkless environment, the accuracy rate is refined, achieving \( 58.6 \pm 2.3 \) when \( \lambda = 0.04 \). Now, when the proposed technique is used in a network environment, the accuracy rate reaches \( 66.3 \pm 2.6 \) when \( \lambda = 0.28 \). In general, the proposed technique is able to boost the accuracy rates of the data sets under analysis. Furthermore, we can see that the networked high level classifier can outperform the networkless version.

**V. APPLICATION: HANDWRITTEN DIGITS RECOGNITION**

In this section, we provide an application of the proposed technique to handwritten digits recognition. Specifically, in Subsect. V we detail the experimental results obtained from the modified NIST set, a data sets composed of tens of thousands of real handwritten digits. Still in this section, we present two simple examples to show how the proposed technique can be used for invariant pattern recognition.

While recognizing individual digits is only one of a myriad of problems that involves specific designing of practical recognition system, it still is, undoubtedly, an excellent benchmark for comparing shape recognition methods. The data set in which we will conduct our studies hereon is named Modified NIST set [40]. This data set provides a training set with 60 000 samples and a test set of 10 000 samples. Each image has \( 28 \times 28 \) pixels. With respect to the high level classifier, the network in the training phase is constructed using \( k = 3 \) and \( \epsilon = 0.01 \). In the classification phase, the same \( \epsilon \) is used.

For comparison matters, we use the proposed high level classifier with 3 distinct low level classifiers. The high level classifier will remain constant as we have been using, i.e., with an underlying network. The techniques that will be exploited are listed below:

- A linear classifier implemented as an 1-layer neural network. We use the same experimental setup given in [40];
- A \( k \)-nearest neighbor classifier with an Euclidean distance measure between input images (\( k = 3 \));
- A \( k \)-nearest neighbor classifier with the similarity function given by a set of weighted eigenvalue measures [4]. Specifically, we use \( \phi = 4 \) eigenvalues and adopt the following \( \beta \) function: \( \beta(x) = 16 \exp(\frac{x}{4}) \).

Figure 9 shows the performance of these techniques acting together with the high level classifier in a networked environment. Our main goal here is to reveal that a mixture of traditional and high level classifier is able to increase the accuracy rate. For example, the linear neural network reached 88% of accuracy rate when only a traditional classifier is applied. A small increase in the compliance term is responsible for increasing the accuracy rate to 91% (\( \lambda = 0.2 \)). Regarding the \( k \)-nearest neighbor algorithm, for a pure traditional classifier, we have obtained 95% of accuracy rate, against 97.6% (\( \lambda = 0.25 \)). For the proposed weighted eigenvalue measure, we have obtained 98% of accuracy rate when \( \lambda = 0 \), against 99.1% when \( \lambda = 0.2 \). It is worth noting that the enhancement is significant. Even in the third case, the improvement is quite welcomed, because it is a hard task to increase an already very high accuracy rate. Moreover, one can see that maximum compliance term is intrinsic to the data set, since, for

| Data Set | # Samples | # Dimensions | # Classes | \( \alpha_k \) | \( \alpha_c \) |
|----------|-----------|--------------|-----------|--------------|--------------|
| Yeast    | 1484      | 8            | 10        | 0.40         | 0.60         |
| Teaching | 151       | 5            | 3         | 0.50         | 0.50         |
| Zoo      | 101       | 16           | 7         | 0.30         | 0.70         |
| Wine     | 178       | 13           | 3         | 0.40         | 0.60         |
| Iris     | 150       | 4            | 3         | 0.40         | 0.60         |
| Glass    | 214       | 9            | 6         | 0.50         | 0.50         |
| Vehicle  | 846       | 18           | 4         | 0.60         | 0.70         |
| Letter   | 20000     | 16           | 26        | 0.40         | 0.60         |

**TABLE II**

BRIEF INFORMATION OF THE DATA SETS.
TABLE III

ACCURACY RATE ACHIEVED BY SEVERAL LOW LEVEL CLASSIFICATION TECHNIQUES AND THE HIGH LEVEL CLASSIFIER WITH AND WITHOUT NETWORKS. IN THE ROW NAMED “PURE”, THE OPTIMIZED PARAMETERS OF EACH LOW LEVEL TECHNIQUE ARE REPORTED AS FOLLOWS: WEIGHTED k-NN (κ), MLP (NUMBER OF LAYERS, LEARNING RATE, MOMENTUM), AND FUZZY M-SVM (C, γ). IN THE ROW DENOMINATED “NETWORKLESS”, THE BEST COMPLIANCE TERM IS REPORTED IN THE PAIR OF PARENTHESSES. IN THE ROW CALLED “NETWORK”, THE ϵ (CLASSIFICATION PHASE) AND THE BEST COMPLIANCE TERM ARE EXHIBITED IN THE PAIR OF PARENTHESSES.

| Data Set | Type      | Bayesian Networks | Weighted k-NN | MLP | Fuzzy M-SVM |
|----------|-----------|-------------------|---------------|-----|-------------|
| Yeast    | Pure      | 57.8 ± 2.6        | 60.9 ± 3.6 (16) | 56.2 ± 3.9 (4.3, 0.2) | 58.0 ± 4.8 (21, 21) |
|          | Networkless | 58.6 ± 2.3 (0.04) | 62.0 ± 3.2 (0.07) | 56.9 ± 2.5 (0.05) | 60.2 ± 4.6 (0.14) |
|          | Network   | 66.3 ± 2.6 (0.05, 0.28) | 65.7 ± 4.0 (0.03, 0.19) | 63.3 ± 2.9 (0.05, 0.23) | 69.8 ± 3.8 (0.05, 0.28) |
| Teaching | Pure      | 61.3 ± 8.8        | 63.0 ± 12.3 (9) | 60.9 ± 9.4 (7, 0.2, 0.4) | 52.5 ± 7.9 (26, 26) |
|          | Networkless | 63.5 ± 9.3 (0.18) | 63.8 ± 10.6 (0.12) | 62.0 ± 7.7 (0.13) | 55.3 ± 8.6 (0.18) |
|          | Network   | 67.2 ± 6.6 (0.03, 0.24) | 68.5 ± 7.4 (0.04, 0.19) | 67.8 ± 6.1 (0.04, 0.26) | 62.7 ± 0.9 (0.04, 0.33) |
| Zoo      | Pure      | 95.9 ± 4.3        | 96.2 ± 5.8 (1) | 96.1 ± 6.9 (3, 0.4, 0.5) | 96.3 ± 6.4 (21, 21) |
|          | Networkless | 96.0 ± 3.6 (0.02) | 96.5 ± 5.2 (0.04) | 96.4 ± 6.6 (0.04) | 96.5 ± 4.5 (0.05) |
|          | Network   | 97.3 ± 4.3 (0.02, 0.06) | 97.5 ± 4.4 (0.02, 0.09) | 97.5 ± 4.2 (0.02, 0.09) | 97.5 ± 2.3 (0.02, 0.08) |
| Wine     | Pure      | 98.8 ± 0.7        | 94.6 ± 1.4 (1) | 97.8 ± 0.5 (3, 0.6, 0.4) | 98.9 ± 0.2 (21, 21) |
|          | Networkless | 98.8 ± 0.7 (0.00) | 94.6 ± 2.1 (0.00) | 97.9 ± 0.3 (0.03) | 98.9 ± 0.2 (0.00) |
|          | Network   | 98.8 ± 0.7 (0.00, 0.00) | 96.3 ± 1.0 (0.06, 0.06) | 98.6 ± 0.3 (0.02, 0.09) | 98.9 ± 0.2 (21, 21) |
| Iris     | Pure      | 92.7 ± 1.2        | 97.9 ± 3.3 (19) | 94.0 ± 2.9 (1, 0.3, 0.2) | 97.0 ± 4.6 (22, 22) |
|          | Networkless | 93.2 ± 1.9 (0.07) | 97.9 ± 3.3 (0.00) | 94.8 ± 2.8 (0.10) | 97.2 ± 3.7 (0.00) |
|          | Network   | 94.9 ± 0.4 (0.01, 0.15) | 98.3 ± 0.6 (0.01, 0.05) | 96.5 ± 1.1 (0.02, 0.21) | 98.1 ± 1.0 (0.02, 0.05) |
| Glass    | Pure      | 70.6 ± 7.7        | 71.8 ± 9.0 (1) | 67.3 ± 5.0 (7, 0.1, 0.3) | 72.4 ± 5.6 (20, 20) |
|          | Networkless | 71.5 ± 5.7 (0.14) | 72.7 ± 7.1 (0.16) | 68.8 ± 3.2 (0.22) | 73.3 ± 3.9 (0.12) |
|          | Network   | 79.2 ± 5.3 (0.03, 0.32) | 79.7 ± 5.0 (0.35) | 77.4 ± 5.5 (0.02, 0.30) | 80.1 ± 4.3 (0.03, 0.31) |
| Vehicle  | Pure      | 68.1 ± 3.8        | 67.6 ± 4.1 (5) | 69.0 ± 4.4 (5, 0.3, 0.2) | 84.4 ± 3.4 (20, 20) |
|          | Networkless | 70.0 ± 2.6 (0.19) | 69.4 ± 2.5 (0.18) | 70.3 ± 3.8 (0.13) | 84.4 ± 3.4 (0.00) |
|          | Network   | 74.1 ± 2.9 (0.05, 0.26) | 73.6 ± 3.0 (0.05, 0.24) | 74.7 ± 3.6 (0.07, 0.29) | 84.9 ± 2.7 (0.04, 0.07) |
| Letter   | Pure      | 74.4 ± 5.6        | 96.0 ± 7.6 (1) | 88.9 ± 9.9 (3, 0.2, 0.4) | 94.8 ± 1.7 (20, 20) |
|          | Networkless | 75.5 ± 4.6 (0.14) | 96.0 ± 7.6 (0.00) | 89.3 ± 7.4 (0.09) | 94.8 ± 1.7 (0.00) |
|          | Network   | 77.8 ± 3.6 (0.04, 0.17) | 96.0 ± 7.6 (-, 0.00) | 92.1 ± 4.1 (0.04, 0.19) | 94.8 ± 1.7 (-, 0.00) |

three completely distinct low level classifiers, the maximum accuracy rate is reached in the surroundings of λ = 0.225.

Fig. 9. A detailed analysis of the impact of the compliance term λ on different traditional low level techniques applied to the MNIST database. One can see that a mixture of the proposed traditional and high level techniques does give a boost in the accuracy rate in this real-world data set.

Figures 10a and 10b illustrate how the digit classification is carried out by using simple networks containing a small number of digits ‘5’ and ‘6’, randomly drawn from the MNIST data set. Firstly, let us consider the Fig. 10a where the digits ‘5’ with brown boxes and the digits ‘6’ with blue boxes represent the training set. The task now is to classify the test instance represented by the digit with a red box. If only low level classification is applied, the test digit is probably to be classified as a digit ‘6’, because it has more neighbors of digit ‘6’ than that of ‘5’. On the other hand, if we also consider high level classification, the test digit can be correctly classified as a digit ‘5’, because it complies more to the pattern formed by training digits ‘5’ than to the one formed by digits ‘6’. More specifically, if the test digit is put into the class ‘5’, it just extends the already formed “line” pattern. As a consequence, the inclusion of the test digit to the class ‘5’ generates small variations of the component measures. However, if the test digit is inserted into the class ‘6’, larger variations of the component measures occur, since cycles are formed in the component (before insertion of the test digit, there is no cycle in the component). Figures 11a and 11b show the transient and cycle lengths as well as the corresponding variations as a function of μ, when the test digit is inserted into the component of digit ‘5’. We see that the variations are very small, indicating the strong compliance of the test digit with the pattern formed by training digits ‘5’. On the other hand, Figs. 11c and 11d show the same information, when the test digit is put into the component of digit ‘6’. Here, we see that larger variations occur, which means that the test digit does not conform to the pattern formed by the component of digits ‘6’. As a result, the test instance is correctly classified as a digit ‘5’. The same reasoning can be applied to the digit network shown by Fig. 10b. In this case, the transient and cycle lengths as well as the corresponding variations are shown in Figs. 11c - 11d, when the test digit is inserted into the component of digit ‘5’ or ‘6’, respectively. In this situation, the test instance
Fig. 10. Illustration of the pattern formation impact in a subset of samples extracted from the MNIST data set. The training instances are displayed by the brown (digit 5) and blue (digit 6) colors. The test instances are indicated by a red color (bigger sizes). Insertion of a test instance whose real class is: (a) the digit 5 and (b) the digit 6.

Fig. 11. Transient and cycle lengths and the corresponding variations due to the insertion of a test instance. With regard to Fig. 10a: (a) and (b) transient and cycle lengths of the brown class (digit 5) and their variations by virtue of the insertion of the red instance (test instance); (c) and (d) same information for the blue class (digit 6). With regard to Fig. 10b: (e) and (f) transient and cycle lengths of the brown class (digit 5) and their corresponding variations due to the insertion of the red instance (test instance) with respect to the brown class; (g) and (h) same information for the blue class (digit 6).

is classified as a digit ‘6’.

VI. CONCLUSIONS

In this work, we have proposed an alternative and novel technique for data classification, which combines both low and high level characteristics of the data. The former classifies data instances by their physical features and the latter measures the compliance of the test instance with the pattern formation of the input data. To this end, tourist walks have been employed to capture the complex topological properties of the network constructed from the input data. A quite interesting feature of the proposed technique is that the high level term’s influence has to be increased in order to get correct classification as the complexity of the class configuration increased. This means that the high level term is specially useful in complex situations of classification. Also, it is worth observing that the application of the tourist walks dynamics in the context of high level classifier is also a novel approach in the literature. We have shown that, even though such walk is constructed under very simple rules, it is still able to capture topological features of the underlying network in a local to global basis. In addition, we uncover a critical memory length, above which no change occurs in the transient and cycle lengths of the network component.

We hope our work can provide an alternative way to the understanding of high level semantic machine learning. As a future work, mechanisms for taking representative samples, rather than recalculating all the tourist walks for all the vertices
in the network, are going to be considered.

ACKNOWLEDGMENT

This work is supported by the São Paulo State Research Foundation (FAPESP) and by the Brazilian National Research Council (CNPq).

REFERENCES

[1] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley-Interscience, 2000.
[2] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York, NY: John Wiley & Sons, Inc., 2001.
[3] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
[4] T. C. Silva and L. Zhao, “Network-based high level data classification,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 6, pp. 954–970, 2012.
[5] A. Blum and T. Mitchell, “Combining labeled and unlabeled data with co-training,” in *COLT. Proceedings of the Workshop on Computational Learning Theory*, 1998, pp. 92–100.
[6] 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.
[7] T. Berners-Lee, J. Hendler, and O. Lassila, “The semantic web,” *Scientific American*, vol. 284, no. 5, pp. 34–43, 2001.
[8] N. Shadbolt, W. Hall, and T. Berners-Lee, “The semantic web revisited,” *IEEE Intelligent Systems*, vol. 6, pp. 96–101, 2006.
[9] L. Feigenbaum, I. Herman, T. Hongsermeier, E. Neumann, and S. Stephens, “The semantic web in action,” *Science*, vol. 297, no. 6, pp. 90–97, 2007.
[10] T. Zhang, A. Popescul, and B. Dom, “Linear prediction models with graph regularization for web-page categorization.” in *KDD*, ACM, 2006, pp. 821–826.
[11] S. A. Macskassy and F. Provost, “Classification in networked data: A toolkit and a univariate case study,” *J. Mach. Learn. Res.*, vol. 8, pp. 935–983, 2007.
[12] S. Zhu, K. Yu, Y. Chi, and Y. Gong, “Combining content and link for classification using matrix factorization.” in *SIGIR*, ACM, 2007, pp. 487–494.
[13] B. Gallagher, H. Tong, T. Eliassi-rad, and C. Faloutsos, “Using ghost edges for classification in sparsely labeled networks,” in *Knowledge Discovery and Data Mining*, 2008, pp. 256–264.
[14] D. Zhang and R. Mao, “Classifying networked entities with modularity kernels,” in *International Conference on Information and Knowledge Management*, 2008, pp. 113–122.
[15] X. Zhu, “Semi-supervised learning literature survey,” *Computer Science*, University of Wisconsin-Madison, Tech. Rep. 1530, 2005.
[16] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*, ser. Adaptive Computation and Machine Learning. Cambridge, MA: The MIT Press, 2006.
[17] R. W. Donaldson and G. T. Toussaint, “Use of contextual constraints in recognition of contour-traced handprinted characters,” *IEEE Trans. Computers*, pp. 1096–1099, 1970.
[18] E. Binaghi, I. Gallo, and M. Pepe, “A cognitive pyramid for contextual classification of remote sensing images,” *IEEE Trans. Geoscience and Remote Sensing*, vol. 41, no. 12, pp. 2906–2922, 2003.
[19] D. Tuia, G. Camps-Valls, G. Matasci, and M. Kanevski, “Learning relevant image features with multiple-kernel classification,” *IEEE Trans. Geoscience and Remote Sensing*, vol. 48, no. 10, pp. 3780–3791, 2010.
[20] Y. Tian, Q. Yang, T. Huang, C. X. Ling, and W. Gao, “Learning contextual dependency network models for link-based classification,” *IEEE Trans. Data and Knowledge Engineering*, vol. 18, no. 11, pp. 1482–1496, 2006.
[21] D. Lu and Q. Weng, “survey of image classification methods and techniques for improving classification performance,” *International Journal of Remote Sensing*, vol. 28, no. 5, pp. 823–870, 2007.
[22] D. Williams, X. Liao, Y. Xue, L. Carin, and B. Krishnapuram, “On classification with incomplete data,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, no. 3, pp. 427–436, 2007.
[23] A. Micheli, “Neural network for graphs: A contextual constructive approach,” *IEEE Trans. Neural Networks*, vol. 20, no. 3, pp. 498–511, 2009.
[24] G. F. Lima, A. S. Martinez, and O. Kinouchi, “Deterministic walks in random media,” *Phy. Rev. Lett.*, vol. 87, p. 010603, 2001.
[25] M. G. Campitelli, P. D. Batista, O. Kinouchi, and A. S. Martinez, “Deterministic walks as an algorithm of pattern recognition,” *Physical Review E*, vol. 74, p. 026703, 2006.
[26] A. R. Backes, W. N. Gonçalves, A. S. Martinez, and O. M. Bruno, “Texture analysis and classification using deterministic touristic walk,” *Pattern Recognition*, vol. 43, pp. 685–694, 2010.
[27] O. C. Hamsici and A. M. Martinez, “Rotation invariant kernels and their application to shape analysis,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, no. 5, pp. 947–954, 2010.
[28] U. Park, Y. Tong, and A. K. Jain, “Age-invariant face recognition,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, no. 5, pp. 947–954, 2010.
[29] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, “Robust object recognition with cortex-like mechanisms,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, no. 3, pp. 411–426, 2007.
[30] H. E. Stanley and S. V. Buldyrev, “Statistical physics - the salesman and the tourist,” *Nature*, vol. 413, pp. 373–374, 2001.
[31] O. Kinouchi, A. S. Martinez, G. F. Lima, G. M. Lourencó, and S. Risau-Gusman, “Deterministic walks in random networks: an application to thesaurus graphs,” *Physica A*, vol. 315, pp. 665–676, 2002.
[32] C.-F. Lin and S.-D. Wang, “Fuzzy support vector machines,” *IEEE Transactions on Neural Networks*, vol. 13, pp. 464–471, 2002.
[33] C.-W. Hsu and C.-J. Lin, “A comparison of methods for multi-class support vector machines,” *IEEE Transactions on Neural Networks*, vol. 13, pp. 415–425, 2002.
[34] A. Frank and A. Asuncion, “UCI machine learning repository,” 2010.
[35] S. Borjia, V. Chandola, and V. Kumar, “Similarity measures for categorical data: A comparative evaluation,” in *SIAM Data Mining Conference*, 2008, pp. 234–254.
[36] R. E. Neapolitan, *Learning Bayesian Networks*, 1st ed., 2003.
[37] R. E. Neapolitan, *Learning Bayesian Networks*, 1st ed., 2003.
[38] R. E. Neapolitan, *Learning Bayesian Networks*, 1st ed., 2003.
[39] S. Abe and T. Inoue, “Fuzzy support vector machines for multiclass problems,” in *in European Symposium on Artificial Neural Networks, 2002*, pp. 113–118.
[40] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.