High Level Classification for Pattern Recognition

Thiago C. Silva, Thiago H. Cupertino, and Liang Zhao
Department of Computer Sciences, Institute of Mathematics and Computer Science (ICMC)
University of São Paulo (USP)
Av. Trabalhador São-carlense, 400, 13560-970, São Carlos, SP, Brazil
e-mail: {thiagoch, thiagohc, zhao}@icmc.usp.br

***Abstract*—Traditional data classification techniques consider only physical features of input data in order to construct their hypotheses. On the other hand, the human (animal) brain performs both low and high order learning and it has facility to identify patterns according to the semantic meaning of input data. In this paper, we propose a data classification technique by combining the low level and the high level learning. The low level term can be implemented by any classification technique, while the high level classification is realized by the extraction of features of the underlying network constructed from the input data. Thus, the former classifies data instances by their physical features, while the latter measures the compliance to the pattern formation of the data. Our study shows that the proposed technique can not only realize classification according to the pattern formation, but it is also able to improve the performance of traditional classification techniques. An application on handwritten digits recognition is performed, revealing that higher classification rates can be obtained when we have a proper mixture of low and high level classifiers.**

***Keywords*-High level classification; complex networks.**

I. Introduction and Related Work

Supervised Data classification aims at creating a map from the input data to a corresponding known output in the training phase. The constructed map, called classifier, is then used to predict new input instances. Many classification techniques have been developed [1], [2], such as linear discriminant functions, K-nearest neighbors, Bayesian decision theory, neural networks, committee machine, Support Vector Machine (SVM) and so on. However, all those techniques train and classify data considering only the physical features (e.g., distance or similarity). Here, we call them *low level classification* techniques. However, data items are not isolated points in the attribute space but tend to form certain patterns. For instance, in Fig. 1, the data points represented by the "triangles" (black) are most probably to be classified as pertaining to the "square" (blue) class if only physical features, such as distances between data, are considered. On the other hand, if we take into account the global relationship among the data, we intuitively classify the "triangles" items as members of the "circle" (red) class, since a clear visual pattern of "eight" is formed. The human brain performs both low and high order learning and has facility to identify patterns accordingly to the semantic meaning of the input data. Nevertheless, this kind of task is still hard to be performed by computers. Data classification considering not only physical attributes but also pattern formation is here referred to *high level classification*.

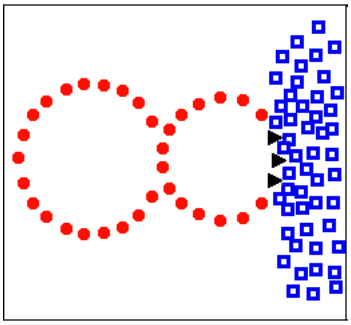

Fig. 1. A simple example of a data classification task where there exists a class with a clear visual pattern, in this case, the red ("eight") class. The goal is to classify the black "triangles" data items. Traditional (low level) classifiers would have trouble to classify such items, since they only derive their decision based on merely physical measures.

There are several kinds of works related to high level classification. One of them is the co-training technique [3], [4], a semi-supervised learning process [5]. It requires two views of the data and assumes that each example is described using two different feature sets that provide different and complementary information about the instances. The co-training technique firstly learns a separate classifier for each view using any labeled examples. Then, the most confident predictions of each classifier on the unlabeled data are then used to iteratively construct additional labeled training data. By considering different views of the same data set, some kinds of data relationships determined by the predefined views may be uncovered. Another related technique is the committee machine [6], which consists of an ensemble of classifiers. In this case, each classifier makes a decision by itself and all these decisions are combined into a single response. The combined response of the committee machine is supposed to be superior to those of its constituent experts. Since each classifier has a particular view to the input data, the combination of them may reveal relationship among input data. The Semantic Web [7], [8] is another strongly related work. The idea of Semantic Web is "an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation" [9]. Semantic Web uses ontologies to describe the semantics of the data. Is a promising idea but it also has many difficult challenges. A key challenge in creating Semantic Web is the semantic mapping among the ontologies. Maybe the most related works

to high level classification are the contextual classification techniques [10], [11], which consider the spatial relationships between individual pixels and the local and global configurations of neighboring pixels to assign classes. The underlying assumption is that the neighboring pixels are strongly related, i.e., the structure of classes are likely to be found in the presence of others. However, from the viewpoint of general high level classification, all the above mentioned approaches are quite restricted either to the types of semantic features to be extracted, such as co-training and ensemble method, or to the types of data, such as the contextual classification. 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 [12], [13]. This work presents an effort on this direction.

Another research area which is fundamental to this work is the *Complex Networks*. A complex network refers to a large scale graph with nontrivial connection patterns [14], [15]. Such networks have emerged as a unified representation of complex systems in various branches of science [16]. Complex networks are ubiquitous in nature and everyday life. Examples include the Internet, the World Wide Web, biological neural networks, social networks between individuals and among companies and organizations, food webs, metabolic networks and distribution as the bloodstream, postal delivery routes and distribution of electric energy etc. Many network measures have been developed to characterize nodes, subnetworks and the whole network [17], [18]. Such measures can also be used to distinguish one type of network from another. One of the salient features of complex networks, as data representation tools, is the ability to describe the topological structure of the data. Such representation not only emphasizes the physical distances among nodes, but also captures from local to global relationships among data instances. Consequently, it is a suitable tool to uncover pattern formation. For example, it has been shown that the topological structure is quite useful to detect various cluster forms using a data clustering algorithm with a unique distance measure [19].

In this paper, we propose a novel technique by combining the low level and the high level data classifications using a networked data representation. The low level classification can be implemented by any classification technique, while the high level classification is realized by the extraction of features of the underlying network constructed from the input data. Thus, the whole classification technique consists of two terms: the first considers classifying data items solely by their physical features and the second measures the compliance to the pattern formation from the input data items. Numerical studies show that the proposed technique can not only execute classification accordingly to the pattern formation, but also can improve the performance of traditional classification techniques. Interestingly, we find that in general good classification results can be obtained by the combination of the two terms.

The remainder of the paper is organized as follows. The definition of the proposed model is described in Section II. In Section III, we provide computer simulation results to show the high level classification performance and its particular characteristics. In Section IV, we apply the proposed technique to manual digits recognition problem. We show that the high level classification can really improve the recognition precision. Finally, Section V concludes the paper.

II. Model Description

A training set is denoted here as $\mathbf{X}_{\text{training}} = \{(f_1, y_1), \ldots, (f_l, y_l)\}$, where the first component of the ith tuple $f_i = \{x_1, \ldots, x_d\}$ denotes the attributes of the d-dimensional ith data item. The second component $y_i \in \mathcal{L} = \{L_1, \ldots, L_n\}$ characterizes the class label or target associated to that data item. The goal here is to learn a mapping from x to y. Usually, the constructed classifier is checked by using a test set $\mathbf{X}_{\text{test}} = \{f_1, \ldots, f_u\}$, in which labels are not provided. For an unbiased learning, we have $\mathbf{X}_{\text{training}} \cap \mathbf{X}_{\text{test}} = \emptyset$.

With these definitions at hand, we now advance to the network formation. The network will be constructed solely based on the data presented in the training set $\mathbf{X}_{\text{training}}$. In this case, every data item (x_i, y_i) is represented as a vertex. A connection between two vertices will take place accordingly to the similarity and the classes of each data item. Consider that $\rho(i, j)$ denotes the distance between vertices i and j, then a connection will be formed between the vertices i and j if the following conditions are satisfied (ϵNN technique): (i) $\rho(i, j) \leq \epsilon$ and (ii) $y_i = y_j$, where ϵ is a fixed positive constant, i.e., vertices i and j are within a predefined similarity (it is used the Euclidean distance in this work) and are instances of the same class. If it happens that no data item of the same label is within the circular region with radius ϵ, then we deliberately make a random link between the data item under analysis and any other data item with the same label. This enforces a single component for each class. The constructed network will then be used to extract pattern information from the original input data.

A. The High Level Classification Technique

The classification technique proposed in this work can be comprehended as a high level (semantic) classifier. Therefore, it is capable of processing the test set in a more intelligent form by using a mixture of local features, provided by the physical distance among the data instances, and global semantic features, which are induced by the unveiled class patterns formed by the data.

With the purpose of being able to represent such mixture, we propose a classifier M that consists of a convex combination of a low level classifier and a high level classifier. Mathematically, the probability that a data item $i \in \mathbf{X}_{\text{test}}$ be classified within the class $j \in \mathcal{L}$, is given by:

$$M_i^{(j)} = (1 - \lambda) T_i^{(j)} + \lambda C_i^{(j)} \tag{1}$$

where $T_i^{(j)} \in [0, 1]$ denotes the probability (decision) produced by an arbitrary traditional (low level) classifier; $C_i^{(j)} \in$

$[0, 1]$ indicates the probability predicted by a high level classifier; $\lambda \in [0, 1]$ is the *compliance term* which plays the role of counterbalancing the classification decision predicted by both classifiers. Whenever $T_i^{(j)} = 1$ and $C_i^{(j)} = 1$, we may deduce that the i carries all the characteristics of class j. On the other hand, whenever $T_i^{(j)} = 0$ and $C_i^{(j)} = 0$, we may infer that the i does not present any similarities with the items of class j. Values in between these two extremes lead to natural uncertainness in the classification process and are found in the majority of executions of a classification task. The same reasoning can be applied to $M_i^{(j)}$. From Eq. (1), one can minimize the classification error by just choosing the class j characterized by the highest $M_i^{(j)}$. Therefore, Eq. (1) provides a fuzzy classification method. Moreover, it is valuable to salient that, when $\lambda = 0$, Eq. (1) reduces to a common low level classifier.

Equation (1) supplies a general framework for the classification process, in the sense that any technique could be easily brought into play. The first term of Eq. (1) is rather straightforward to implement, since it can be any traditional classification technique.

The network-based (graph-based) technique underlying the high level classifier has the characteristic that each class corresponds to an isolated subgraph (component) within the network. Thus, a class component must always be connected, so that every class is represented by and unique component. With that in mind, we make use of a combination of well-known network measures developed in the complex network literature to compose the high level classifier C. These measures are chosen in a way to cover relevant high level aspects of the class component. The number of measures to be plugged into the high level classifier is user-controllable; thus, let us suppose that K measures are selected to comprise the classifier. Mathematically, the high level classification probability that a data item $i \in \mathbf{X}_{\text{test}}$ belongs to the class $j \in \mathcal{L}$ is given by:

$$C_i^{(j)} = \frac{\sum_{u=1}^{K} \alpha(u) \left[1 - f_i^{(j)}(u)\right]}{\sum_{g=1}^{|\mathcal{L}|} \sum_{u=1}^{K} \alpha(u) \left[1 - f_i^{(g)}(u)\right]} \tag{2}$$

where $\alpha(u) \in [0, 1], \forall u \in \{1, \ldots, K\}$, is a user-controllable coefficient that indicates the influence of each network measure in the process of classification; $|\mathcal{L}|$ denotes the number of distinct classes in the problem, and $f_i^{(j)}(u)$ is a function that depends on the uth network measure applied to the ith data item regarding the class j. This function is responsible for providing an answer whether the data item i in analysis possesses the same patterns of the component j or not. Indeed, in order to Eq. (2) to be a valid convex combination of network measures, the following constraint on the coefficients must always be satisfied:

$$\sum_{u=1}^{K} \alpha(u) = 1 \tag{3}$$

Regarding the $f_i^{(j)}(u)$, it possesses a general closed form given by:

$$f_i^{(j)}(u) = \Delta G_i^{(j)}(u) p^{(j)} \tag{4}$$

where $\Delta G_i^{(j)}(u) \in [0, 1]$ is the variation on the uth network measure that occurs on the component representing class j if i joins such component and $p^{(j)} \in [0, 1]$ is the proportion of data items pertaining to the class j. Remembering that each class has a component representing itself, it is naturally to check the pattern compliance of a new data item to be classified by examining whether its insertion to a component results in a great variation of all the network measures. In other words, if there is a small change in the network measures, the new data item is in compliance with all the other data items that comprise that 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 on the network measures of that component, then probably the data item in question may not belong to that class. This is exactly the behavior that Eq. (2) together with Eq. (4) propose. Consider that a data item i is to be classified in an arbitrary classification problem and consider also that there are 2 possible classes, namely A and B, $K = 1$, i.e., we are using only one network measure, and classes A and B are equally sized. Hypothetically, say that $\Delta G_i^{(\mathrm{A})}(1) = 0.7$ and $\Delta G_i^{(\mathrm{B})}(1) = 0.3$. Clearly, this data item has a bigger chance to belong to the class B, since if it joins class B, it would produce less variation on the network measure. Indeed, that is what Eq. (2) precisely produces.

We proceed to explain the role of the $p^{(j)} \in [0, 1]$ in Eq. (4). In real world databases, we usually encounter unbalanced classes. Roughly speaking, a database frequently encompasses several classes of different sizes. In general, some network measures are very sensitive to the size of the components. In an attempt to soften this problem, we introduce in Eq. (4) the term $p^{(j)}$, which is the proportion of vertices of the component representing class j. On account of that, we expect to obviate the size effects in the classification process. For instance, consider an arbitrary data item i and two classes, A and B, such that the A's size is 10 times bigger than B's. In this case, $p^{(A)} = {}^{10}/_{11}$ and $p^{(B)} = {}^{1}/_{11}$. Without the term $p^{(j)}$, it is expected that variations on the network measure due to the insertion of i in A to be considerably smaller than B, no matter how data item i complies to class B. By considering the term $p^{(j)}$, the bigger value of $p^{(\mathrm{A})}$ over $p^{(\mathrm{B})}$ diminishes the above mentioned undesirable effect.

B. Composition of the High Level Classifier

In this subsection we introduce the selected set of network measures that compose the high level classifier, namely: assortativity, clustering coefficient, and average degree. In spite of having chosen these measures, it is worth to emphasize that other network measures can be also plugged into the high level classifier through Eq. (2).

1) Assortativity: The assortativity measure numerically translates the preference for vertices of a network to attach to others that are similar or different regarding the vertex's degree in a structural sense [20]. In general, r lies between -1 and 1. When $r = 1$, the network is said to have perfect assortative mixing patterns, while at $r = -1$ the network is completely disassortative.

Remembering that each class owns a representative and unique component, we can calculate the assortativity with regards to each of the existing classes. Let $\mathbb{U}_j = \{u : y_{i_u} = j \wedge y_{k_u} = j\}$, where u represents an edge, $i_u, k_u \in \mathbf{X}_{\text{training}}$ indicate the vertices at each end of the edge u. In another terms, $\mathbb{U}_j$ encompasses all the edges within the class j. With these considerations, the assortativity of class j is given by:

$$r^{(j)} = \frac{M^{-1}\sum_{u\in\mathbb{U}_j} i_u k_u - \left[M^{-1}\sum_{u\in\mathbb{U}_j}\frac{1}{2}(i_u + k_u)\right]^2}{M^{-1}\sum_{u\in\mathbb{U}_j}\frac{1}{2}(i_u^2 + k_u^2) - \left[M^{-1}\sum_{u\in\mathbb{U}_j}\frac{1}{2}(i_u + k_u)\right]} \tag{5}$$

Let us now derive $\Delta G_i^{(j)}(1)$ using the assortativity measure. Consider that an arbitrary data item i is to be classified. In relation to an arbitrary class j, we first calculate the actual assortativity measure of the component representing class j. Let it be denoted by $r^{(j)}$. Then, we virtually insert the data item i into component j using the ϵNN technique, as exposed in the previous section, and recalculate the new assortativity measure. Let the new value be denominated $r'^{(j)}$. We perform this procedure to all classes $j \in \mathcal{L}$. However, if there are no connections from that data item i to a specific component, say pertaining to class $k \in \mathcal{L}$, by this approach, we would have $r^{(k)} = r'^{(k)}$, which is undesirable, since this configuration would state that the data item i is as similar as possible to the component of class k. In order to overcome this problem, a simple post-processing is necessary as follows. For all components $k \in \mathcal{L}$ that do not share at least 1 link with the data item i, we deliberately set $r^{(k)} = -1$ and $r'^{(k)} = 1$, i.e., the maximum possible difference , since r ranges from $[-1, 1]$.

With all this information at hands, we are able to calculate $\Delta G_i^{(j)}(1)$ for all $j \in \mathcal{L}$ as follows:

$$\Delta G_i^{(j)}(1) = \frac{|r'^{(j)} - r^{(j)}|}{\sum_{u=1}^{|\mathcal{L}|}|r'^{(u)} - r^{(u)}|} \tag{6}$$

where the denominator is introduced only for normalization matters. According to Eq. (6), for insertions that result in a considerable variation regarding the assortativity quantity, $\Delta G_i^{(j)}(1)$ will be high, and, consequently, by Eq. (4), $f_i^{(j)}(1)$ is expected to be also high, yielding a low value for the high level classifier C, as Eq. (2) reveals. On the other hand, for insertions that do not cause a considerable variation regarding the assortativity quantity, $\Delta G_i^{(j)}(1)$ will be low, and, as a result, by Eq. (4), $f_i^{(j)}(1)$ is expected to be also low, producing a high value for the high level classifier C, as Eq. (2) exposes.

2) Clustering Coefficient: The clustering coefficient measure quantifies the degree to which local nodes in a network tend to cluster together. Evidence suggests that in many real-world networks, and in particular social networks, nodes tend to create tightly knit groups characterized by a relatively high density of ties [21]. Here, we use the measure originally proposed by Watts and Strogatz [21]. The local clustering coefficient of a vertex in a graph quantifies how close its neighbors are to being a clique (complete graph). Mathematically speaking, the local clustering coefficient for a given vertex i pertaining to the representative component of class j is given by:

$$C_i^{(j)} = \frac{|e_{uk}|}{k_i^{(j)}\left(k_i^{(j)} - 1\right)} \tag{7}$$

where $|e_{uk}|$ the number of links shared by the direct neighbors of vertex i (number of triangles formed by vertex i and any of its two neighbors) and $k_i^{(j)}$ is the degree of vertex i of component j. By Eq. (7), we see that $C_i^{(j)} \in [0, 1]$. Having calculated the local clustering coefficient of all vertices that belong to the class j, we are able to define the component's average clustering coefficient by:

$$C^{(j)} = \frac{1}{n_j}\sum_{i=1}^{n_j} C_i^{(j)} \tag{8}$$

where n_j symbolizes the number of vertices in component j and $C^{(j)} \in [0, 1]$. Roughly speaking, the clustering coefficient tells how well connected the neighborhood of the node is. If the neighborhood is fully connected, the clustering coefficient is 1 and a value close to 0 means that there are hardly any connections in the neighborhood. With that in mind, we motivate the use of such measure by using the following facts: components with large clustering coefficient are found to have a modular structure with high density of local connections, while components with small average clustering values tend to posses many long range connections, destroying local structures.

The derivation of $\Delta G_i^{(j)}(2)$ is rather analogous to the previous case, except for a simple detail: In this case, for all components $k \in \mathcal{L}$ that do not share at least 1 link with the data item i to be classified, we intentionally fix $C^{(k)} = 0$ and $C'^{(k)} = 1$, i.e., the maximum possible difference, since C ranges from $[0, 1]$, where $C^{(k)}$ and $C'^{(k)}$ represent the clustering coefficients of the component k before and after the data item to be classified is inserted, respectively. In this way, we are able to define $\Delta G_i^{(j)}(2)$ as:

$$\Delta G_i^{(j)}(2) = \frac{|C'^{(j)} - C^{(j)}|}{\sum_{u=1}^{|\mathcal{L}|}|C'^{(u)} - C^{(u)}|} \tag{9}$$

3) Average Degree: The average degree measure is relatively a simple measure and statistically quantifies the average degree of the vertices of a component. This measure by itself

is weak in terms of finding patterns in the network, since the mean value may not exactly quantify the degrees of the majority of vertices in a component. However, if it is jointly used with another measures, such as assortativity and clustering coefficient, its recognition power significantly increases. Here, its usage is motivated by the fact that components with degrees alike are valid candidates to pertain to the same class. In this case, the average degree of an arbitrary component representing class j is given by:

$$\langle k^{(j)} \rangle = \frac{1}{n_j} \sum_{i=1}^{n_j} k_i^{(j)} \tag{10}$$

where $k_i^{(j)}$ indicates the degree of vertex i from the component j. The derivation of $\Delta G_i^{(j)}(3)$ is rather analogously to the previous case, except for a simple detail: In this case, for all components $u \in \mathcal{L}$ that do not share at least 1 link with the data item i to be classified, we purposefully assign $\langle k'^{(u)} \rangle = \max\left(\langle k^{(u)} \rangle - \min_i \left(k_i^{(u)}\right), \max_i \left(k_i^{(u)}\right) - \langle k^{(u)} \rangle\right)$, i.e., the maximum possible difference from the mean of the component, i.e., $\langle k^{(j)} \rangle$. In this way, we are able to define $\Delta G_i^{(j)}(3)$ as:

$$\Delta G_i^{(j)}(3) = \frac{|\langle k'^{(u)} \rangle - \langle k^{(u)} \rangle|}{\sum_{u=1}^{|\mathcal{L}|} |\langle k'^{(u)} \rangle - \langle k^{(u)} \rangle|} \tag{11}$$

Again, $\langle k'^{(u)} \rangle$ and $\langle k^{(u)} \rangle$ represent average degree of the component u before and after the data item to be classified is inserted, respectively.

III. Computer Simulations

In this section it is presented the results from a simulation using artificial data in order to show the effectiveness of the model. For this matter, we will give an equal weight for each of the network measures previously introduced, i.e., we will set $\alpha(1) = \alpha(2) = \alpha(3) = 1/3$, according to Eq. (2). Note that the constraint imposed by Eq. (3) is satisfied. Usually, these parameters are not sensible to the final result of the algorithm. The only sensible parameter to the model is λ, which counterbalances the influence of the low and high level classifiers.

In this simulation, we are going to empirically calculate the minimum compliance term λ for which all the data items from the test set are correctly classified. Before going any further, let us clarify the concept of orientation in the learning. The orientation in a learning process is given by the preference of predicting some data item to the detriment of the others, i.e., the orientation provides a natural ordering for the classification task. A good orientation can supply results much more relevant than to those orientations that do not base their arguments in a good heuristic or measure. Consider the classification problem arranged in Fig. 2, where there is a segment of line represented by the red or "circle" class (7 vertices) and a condensed rectangular class outlined by the blue or "square" class (100 vertices). Hence, we have that $p^{\text{red}} = 7/107$ and $p^{\text{blue}} = 100/107$. Regarding the network formation, we have used the ϵNN technique with $\epsilon = 0.07$ (this radius covers, for any vertex in the straight line, 2 adjacent vertices, except for the vertices in each end). For the low level classifier, we will employ the ϵNN classifier with $\epsilon = 0.07$, i.e., we predict the class of the data item plainly by the proportion of vertices of each class that are contained in ϵ-radius taking as the center the vertex to be classified. The task is to classify the 10 test data items described by the big "triangles". Clearly, the orientation will play a main role in the classification problem here. Let us fix the preference for classifying vertices from left to right. The graphic embedded in Fig. 2 shows the minimum value allowed for the λ in order to the data items to be correctly classified. For the first data item to be classified, $\lambda = 0$ suffices, i.e., any efficient traditional classifier could correctly predict this data item. However, from the second to the seventh, higher minimum compliance term values are required. Specifically, as the straight line diametrically crosses the condensed region pertaining to the blue class, the minimum feasible compliance term approaches $\lambda \to 1$, since we cannot establish our decision from the low level classifier, because it would erroneously decide favorable to the blue or "square" class. The data items to be classified in the border of the blue or "square" class, namely the second and seventh, require a lower value in relation to those situated in the middle of the blue's territory. From the eighth data item forward, a $\lambda = 0$ will suffice.

In order to give a clear idea of how the network measures vary, in Table I it is reported all the calculated relevant measures of the contextual classification regarding the first 5 left-most big "triangle" data items in Fig. 2 and the corresponding classification decision for $\lambda = 0$ and $\lambda = 0.95$. A careful look at Table 2 corroborates the importance that the proportion of vertices plays in the process, as it is responsible for counterbalancing the nominal variations of each network measure against the size of the component. In this case, the small variations that occur due to the insertion of a new data item in the blue class is amplified by the high value of $p^{(\text{blue})}$. Conversely, the sizable variations that take place in the red class by virtue of the addition of a new data item is softened by the low value of $p^{(\text{red})}$. Specifically, we can observe that the first big "triangle" data item (number 1) is correctly classified as a member of the red ("circle") class with 100% certainty, no matter which classifier we use, i.e., $\lambda = 0$ (solely a low level classifier) or $\lambda = 0.95$ (mixture of low and high level classifiers). This is expectable, since there is no blue example in the vicinity of the first "triangle" data item. However, as we cross the blue component, the low level classifier ($\lambda = 0$) always misclassifies the examples, due to the high density of examples pertained to the blue class. The highly organized data that composes the red class is sufficient to the high level classifier ($\lambda = 0.95$) to decide favorable for the red class, in detriment to the blue class, as one can see from the data items from 2 to 5.

TABLE I

SNAPSHOT OF THE MEASURES CAPTURED IN THE CONTEXTUAL CLASSIFICATION TASK OF THE BIG "TRIANGLE" DATA ITEMS IN FIG. 2. THE ITEMS ARE ENUMERATED FROM LEFT TO RIGHT (ORIENTATION). THE LEFT-MOST "TRIANGLE" DATA ITEM IS THE ITEM 1 AND SO FORTH, UP TO THE FIFTH ITEM (NUMBER 5). BOLDED VALUES IN THE TWO RIGHT-MOST COLUMNS EMPHASIZE THE FINAL CLASSIFICATION OF THE ALGORITHM FOR $\lambda = 0$ AND $\lambda = 0.95$.

Item	**Class**	**Calculated Network Measures**									**Final Classification Result**	
		Assortativity			**Clustering Coeff.**			**Mean Degree**			$\lambda = 0$	$\lambda = 0.95$
		$r^{(.)}$	$r'^{(.)}$	$\Delta G_i^{(.)}(1)$	$C^{(.)}$	$C'^{(.)}$	$\Delta G_i^{(.)}(2)$	$\langle k^{(.)} \rangle$	$\langle k'^{(.)} \rangle$	$\Delta G_i^{(.)}(3)$		
1	**Red**	0.9519	0.9597	0.0000[1]	0.0000	0.0000	0.0000[1]	1.7143	1.7500	0.0000[1]	**1.0000**	**1.0000**
	Blue	0.5626	0.5626	1.0000[1]	0.6745	0.6745	1.0000[1]	9.5489	9.5489	1.0000[1]	0.0000	0.0000
2	**Red**	0.9597	0.9653	0.4095	0.0000	0.0000	0.0000	1.7500	1.7778	0.3377	0.1973	**0.7518**
	Blue	0.5626	0.5707	0.5905	0.6745	0.6651	1.0000	9.5489	9.4944	0.6623	**0.8027**	0.2482
3	**Red**	0.9653	0.9695	0.3936	0.0000	0.0000	0.0000	1.7778	1.8000	0.3735	0.0036	**0.7406**
	Blue	0.5626	0.5691	0.6064	0.6745	0.6696	1.0000	9.5489	9.5116	0.6265	**0.9964**	0.2594
4	**Red**	0.9695	0.9728	0.3070	0.0000	0.0000	0.0000	1.8000	1.8182	0.3897	0.0289	**0.7544**
	Blue	0.5626	0.5701	0.6930	0.6745	0.6705	1.0000	9.5489	9.5204	0.6103	**0.9711**	0.2456
5	**Red**	0.9728	0.9755	0.4794	0.0000	0.0000	0.0000	1.8182	1.8333	0.8718	0.0004	**0.6480**
	Blue	0.5626	0.5655	0.5206	0.6745	0.6719	1.0000	9.5489	9.5511	0.1282	**0.9996**	0.3520

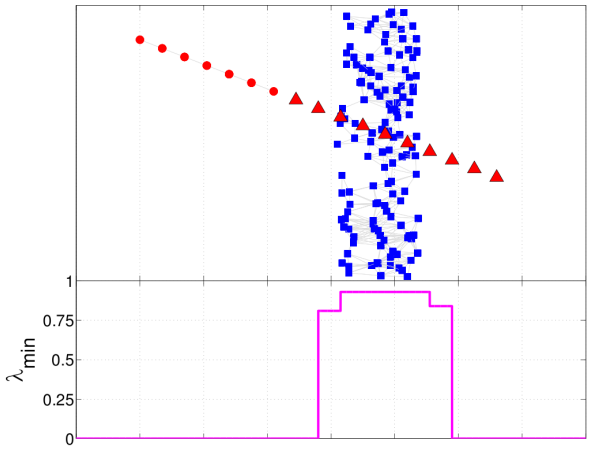

Fig. 2. A detailed analysis of the minimum value of the compliance term λ that would result in the correct classification of the missing data items of the straight line (red or "circle" class) in a two-class problem. If a lesser value of λ would be chosen, then the missing data items would be classified pertaining to the blue class. The orientation was set from left to right. Whenever a data item is classified, it is incorporated to the corresponding most similar class. Traditional techniques would definitely fail to correctly classify the straight line that diametrally crosses the densely connected component pertaining to the blue or "square" class.

IV. APPLICATION: HANDWRITTEN DIGIT RECOGNITION

In this section we provide an application to our contextual high level classifier. In order to do so, a network formation technique is explained, as well as the simulations results obtained from the MNIST database [22].

[1]Since no edge is shared from the data item to the blue class, we set the network measure variations regarding the insertion of the data item 1 to their maximum value (as explained in the previous section).

A. The Network Formation Technique

In a graph-based data representation, the images (data items) are represented by the vertices, whereas the relationships between them are given by the links. A link connecting two vertices (images) holds a weight that numerically translates the similarity between them. Each image can be represented by a "square" matrix $\eta \times \eta$. For rectangle images, a pre-processing is required to transform it into a square image. Also, we conventionally set the pixels' values range to lie within the interval $[0, 1]$ by normalization. Thus, an arbitrary data item (image) x_i can be seen as a matrix with dimensions $\eta \times \eta$, where each pixel $x_i{}^{(u,j)} \in [0, 1], \forall (u, j) \in \{1, \ldots, \eta\} \times \{1, \ldots, \eta\}$.

In order to construct the network, we are required to define a similarity measure. The traditional pixel-per-pixel distance is rather insufficient in terms of reliably when representing data, since such measure is very sensitive to rotation and scale modifications. Hence, we have considered in this work a measure based on eigenvalues. The eigenvalues carry essential information of the images they are derived from [23]: the greater its value, more information about the image it conveys.

Therefore, the similarity measure between two images are calculated as follows. First of all, we remove the mean associated to each image, for having a common basis for comparison, then we calculate the $\phi < \eta$ largest eigenvalues for each image discarding the smaller ones. Also, in order to give more emphasis to the largest eigenvalues, a weight is associated to each one so that the larger an eigenvalue is the larger will be its associated weight.

Summarizing, in order to compute the similarity between two images, e.g., x_i and x_j, firstly we sort their ϕ associated eigenvalues: $|\lambda_i^{(1)}| \geq |\lambda_i^{(2)}| \geq \ldots \geq |\lambda_i^{(\phi)}|$ and $|\lambda_j^{(1)}| \geq |\lambda_j^{(2)}| \geq \ldots \geq |\lambda_j^{(\phi)}|$, with $|\lambda_i^{(k)}|$ being the kth eigenvalue of the ith data item. Finally, the dissimilarity measure ρ between images x_i and x_j is then:

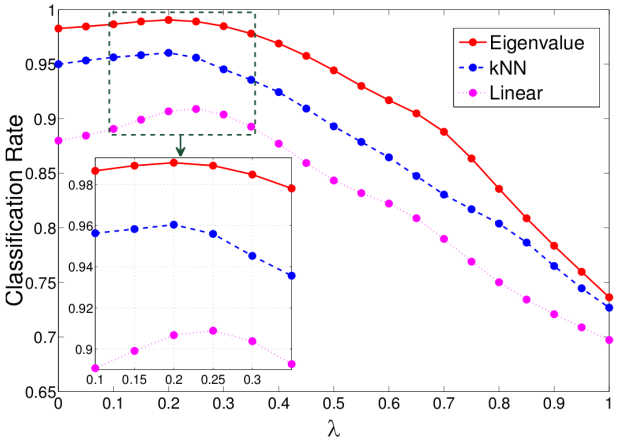

Fig. 3. A detailed analysis of the impact of the compliance term λ on different traditional low level techniques applied to the MNIST database. The high level classifier was fixed to a convex combination of assortativity, clustering coefficient and average degree. One can see that a mixture of the proposed traditional and high level techniques does give a boost in the classification rate in this real world data set. The red line shows the results for a network constructed according to our proposed network formation technique with only the 4 highest eigenvalues. The low level classifier used here was the ϵ-neighbors, with $\epsilon = 0.014$. The blue line indicates the outcome using a simple kNN classifier with $k = 3$ as the traditional classifier. The magenta line depicts the results reached for a linear neural network with 10 neurons in the output layer used as the traditional classifier.

$$\rho(i,j) = \frac{1}{\rho_{max}} \sum_{k=1}^{\phi} \beta(k) \left[|\lambda_i^{(k)}| - |\lambda_j^{(k)}| \right]^2 \qquad (12)$$

where $\rho \in [0,1]$, $\rho_{max} > 0$ is a normalization constant, $\beta : \mathbb{N}^* \rightarrow (0,\infty)$ indicates a monotonically decreasing weight function that can be arbitrarily chosen by the user.

B. Experimental Results

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 database in which we will conduct our studies hereon is entitled Modified NIST set [22]. This database provides a training set with 60.000 samples and a test set of 10.000 samples. Furthermore, in order to make the computation more efficient, we have conducted a pre-processing step. Specifically, we will normalize all the gray-level pixels from the image and reduce its size to fit in a 20×20 pixel box, while preserving their aspect ratio.

In this case, we will make use of the dissimilarity measure based on the first 4 eigenvalues of each image out of 20 eigenvalues, since the image has dimensions 20×20. By virtue of that, we will employ as the function β in the Eq. (12) an exponential decreasing function with a time constant fixed at $\tau = 3$ and a scaling factor given by 16, i.e., $\beta(x) = 16\exp(\frac{x}{3})$. Since, this function is mapped into the interval $(0,\infty)$ and is a monotonically decreasing function, it follows that this β function meets all the aforementioned requirements. Specifically in this situation, we have that the weights associated to each eigenvalue are: $\beta(1) = 11.46$, $\beta(2) = 8.21$, $\beta(3) = 5.89$, and $\beta(4) = 4.22$.

For the sake of comparison, we will test the high level contextual 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 through an 1-layer neural network. No pre-processing was performed. For more details about the neural network parameters, see [22];
- A k-nearest neighbor classifier with an Euclidean distance measure between input images with $k = 3$. No pre-processing was performed. For more details, one can refer to [22];
- The ϵNN classifier with an ϵNN network formation technique with the similarity measure given by a weighted um of the $\phi = 4$ greatest eigenvalues. On the contrary of the previously two techniques, this technique requires a network. In all simulations, $\epsilon = 0.014$ for the both steps (classification and network formation).

We have intentionally provided three techniques with distinct learning paradigms to show how easily we can plug in any kind of classifier in the model. Figure 3 shows the comparison results reached by each algorithm. Our main goal here is to reveal that a mixture of traditional and high level classifier is able to increase the classification rate.

For example, the linear neural network reached 88% of classification rate when we have used only a traditional classifier ($\lambda = 0$). A little increase of the compliance term, $\lambda = 0.25$, is responsible for a boost in the classification rate. In this case, it achieves almost 91%. Regarding the k-nearest neighbor algorithm, for a pure traditional classifier, we have obtained 95% of classification rate, against 96% when $\lambda = 0.2$. In the last algorithm, namely the ϵNN classifier with our proposed measure, we have obtained 98.49% of correctly classified data items for $\lambda = 0$, against 99.06% when $\lambda = 0.2$. It is worth noting that the enhancement is significant. Even in the case of the ϵNN classifier with an eigenvalue-based distance, the improvement is quite welcomed, because it is a hard task to increase an already very high classification rate. Moreover, one can see that maximum compliance term is intrinsic to the data set, since, for three completely distinct low level classifiers, the maximum classification reached is achieved in the surroundings of $\lambda = 0.225$.

For the sake of completeness, in Fig. 4 we provide five samples that have been misclassified using the ϵNN classifier for $\lambda = 0$, but can be successfully classified when $\lambda = 0.2$ (the compliance term with the highest classification rate). For all these samples, a little aid of the high level classifier in this process permitted that all these samples to be correctly classified. In this situation, the traditional classifier cannot correctly predict those data items since it only uses some sort of physical measure between the data. When we raise the influence of the high level classifier decision, i.e., mathematically setting $\lambda = 0.2$, it uses the information inherently

(a) (b) (c) (d) (e)

Fig. 4. Illustration of 5 samples pertaining to the MNIST database that were misclassified using $\lambda = 0$ (only the traditional technique), but have been correctly classified when using $\lambda = 0.2$, i.e., a mixture of traditional and high level classifiers. (a) A handwritten "zero" sample image that was misclassified to a "eight" pattern when $\lambda = 0$ (test sample number 8326). (b) A handwritten "two" sample image that was misclassified to a "seven" pattern when $\lambda = 0$ (test sample number 8705). (c) A handwritten "four" sample image that was misclassified to a "nine" pattern when $\lambda = 0$ (test sample number 9793). (d) A handwritten "five" sample image that was misclassified to a "six" pattern when $\lambda = 0$ (test sample number 3322). (e) A handwritten "nine" sample image that was misclassified to a "four" pattern when $\lambda = 0$ (test sample number 4301).

embedded within the representative component of each class to derive high level decisions, which, jointly with the low level classifier predictions, enabled the contextual classifier M to correctly classify the samples.

V. Conclusions

In this work, we have proposed a novel combination of a low and a high level classifiers, which can be considered as a general framework, to perform supervised data classification. Thus, the whole classification technique consists of two terms, where the first term classifies data instances by their physical features and the second term measures the compliance to the pattern formation from the input data. Additionally, this paper also provides a novel network-based high level classifier that bases its prediction on the organizational changes that the new item to be classified produces in the data. The high level classifier is customizable and can be easily extended or modified by the addition of new network measures, such as degree entropy, component spectrum, average edge reciprocity, matching indices, among many others (see [18] for details about these measures).

We have provided a simple synthetic example in order to demonstrate how the network measures work before and after the insertion of the data item, giving a clear view of the variation of the data organization. A quite interesting feature of the proposed technique is that the high level term influence had 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. Additionally, an application on handwritten digits recognition has been presented and higher accuracy rates have been obtained using the combination of the two terms in relation to classifiers solely based on physical distances. More importantly, our study shows that the proposed technique not only presents innovation for machine learning theory, but also can achieve good classification results in real applications.

Acknowledgment

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

References

[1] T. Mitchell, *Machine Learning*. McGraw Hill, 1997.
[2] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley-Interscience, 2008.
[3] 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.
[4] T. M. Mitchell, "The role of unlabeled data in supervised learning," in *Proceedings of the Sixth International Colloquium on Cognitive Science*, 1999.
[5] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*, ser. Adaptive Computation and Machine Learning. Cambridge, MA: The MIT Press, 2006.
[6] S. Haykin, *Neural Networks, a comprehensive foundation*. Prentice Hall, 1999.
[7] N. Shadbolt, W. Hall, and T. Berners-Lee, "The semantic web revisited," *IEEE Intelligent Systems*, vol. 6, pp. 96–101, 2006.
[8] L. Feigenbaum, I. Herman, T. Hongsermeier, E. Neumann, and S. Stephens, "The semantic web in action," *Scientific American*, vol. 297, no. 6, pp. 90–97, 2007.
[9] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American*, vol. 284, no. 5, pp. 34–43, 2001.
[10] 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.
[11] A. Micheli, "Neural network for graphs: A contextual constructive approach," *IEEE Trans. Neural Networks*, vol. 20, no. 3, pp. 498–511, 2009.
[12] J. Wood, "Invariant pattern recognition: A review," *Pattern Recognition*, vol. 29, pp. 1–17, 1996.
[13] G. Chen, T. Bui, and A. Krzyzak, "Invariant pattern recognition using radom, dual-tree complex wavelet and fourier transforms," *Pattern Recognition*, vol. 42, no. 9, pp. 2013–2019, 2009.
[14] M. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, no. 2, pp. 167–256, 2003.
[15] A. Barrat, M. Barthlemy, and A. Vespignani, *Dynamical Processes on Complex Networks*. Cambridge University Press, 2008.
[16] S. Bornholdt and H. G. Schuster, *Handbook of graphs and networks: from the genome to the internet*. Wiley, 2003.
[17] M. E. J. Newman, *Networks: An Introduction*. Oxford University Press, 2010.
[18] L. F. Costa, F. A. Rodrigues, G. Travieso, and P. R. V. Boas, "Characterization of complex networks: A survey of measurements," *Advances in Physics*, vol. 56, no. 1, pp. 167–242, 2007.
[19] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, pp. 75–174, 2010.
[20] M. E. J. Newman, "Mixing patterns in networks," *Physical Review E*, vol. 67, no. 2, p. 026126, 2003.
[21] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.
[22] 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.
[23] I. T. Jolliffe, *Principal Component Analysis*, 1st ed. Springer Series in Statistics, 2002.