Modeling Repetitive Patterns: A Bridge Between Pattern Theory and Data Mining

Ricardo Vilalta
Department of Computer Science
University of Houston
Houston, Texas, USA
Email: vilalta@cs.uh.edu

Luis Real
Department of Computer Science
Center for Research and Advanced Studies (CINVESTAV)
Guadalajara, Jalisco, México
Email: lreal@gdl.cinvestav.mx

***Abstract*—Traditional learning algorithms generate a predictive model by effectively partitioning the input or feature space in search for regions having a dominant single class. In this paper we point to the existence of problems where the relation among these regions corresponds to repetitive patterns that can be mapped to high-level models. We show how a formalism for the representation of patterns, also known as pattern theory, is instrumental to capture such relations. The idea is to verify patterns using mathematical constructs by combining primitive structures. We illustrate our ideas using parity problems, and show how bridging the gap between traditional supervised learning and pattern theory is a challenge that can bring large benefits to the data mining community.**

***Keywords*-pattern theory; repetitive patterns; classification; supervised learning.**

I. Introduction

A standard interpretation of classification or supervised learning is that of a partitioning of the input or variable space into a set of regions, each region having a dominant single class; the main idea is to train a predictive model that can serve to outline regions mapped to a unique output or class. A probabilistic representation of the data is appropriate when classes overlap, in which case regions are characterized by the class with the largest class-posterior probability [1], [2]. The model is assumed to achieve some degree of data compression [3], as the disjunction of regions normally embeds a shorter representation length when compared to an explicit enumeration of all training samples. The partitioning of the input-space follows a notoriously prevalent bias introduced by most learning algorithms; it assumes a clear degree of smoothness in the input-output distribution such that examples lying close to each other share the same output or class value. The design of learning algorithms revolves around this assumption, and holds valid for both local and global learners [4], connectionist and symbolic [5], or generative and discriminative [6]. During model selection, attention is frequently given to the type of decision boundaries needed to separate regions where there is a change in the dominant class; boundaries can vary from linear to highly non-linear and the mechanism to choose one type over another is based on attaining a balance between the bias and variance components of generalization error [2], [7], [8].

In this paper we point to the existence of problems that cannot be learned using traditional techniques in supervised learning. In particular, the presence of repetitive data patterns calls for additional levels of learning where partitioning the input space into regions is only the first step; additional levels are required to identify the possible occurrence of repetitiveness, and to construct predictive rules of a very different nature from those used in a traditional sense. As an example, Fig. 1 shows a two-dimensional input space where examples can take on two classes. There is an underlying repetitive pattern among examples where classes alternate between adjacent regions. Traditional learning attempts to identify regions (squares) without searching for any possible dependence among them. Under this approach, low-density regions could end up misclassified. One can observe that patterns of this nature can become quite complex; groups of regions can form super-regions with their own repetitive laws. Identifying these patterns forces us to move above the original outcome of current classifiers, in search for rules that define the set of allowable region configurations.

We stress the importance of broadening the learning bias outside traditional assumptions by arming our learning algorithms to cope with possible relations among class disjuncts. In this paper we describe one possible solution using notions of pattern theory. Pattern theory is a formalism where objects and processes in the real world can be modeled through mathematical constructs [9], [10]. The main idea is to model a process, phenomenon, or object of study, through the combination of primitive structures that follow a set of pre-defined combination –or coupling– rules to yield more complex structures. While pattern theory has been instrumental in fields such as computational anatomy [10], we claim its basic ideas can have far-reaching applications, including the construction of more robust classifiers.

This paper is organized as follows. Section II begins by providing basic concepts in classification and a brief introduction to pattern theory. We then explain in Section III how to bridge the gap between these two fields. Section IV illustrates a simple empirical example with results. Lastly, Section V gives a summary and conclusions.

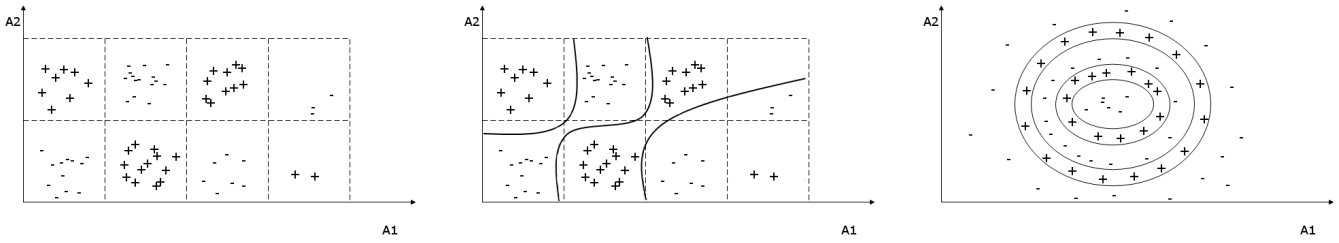

Fig. 1: A two-dimensional input space where examples take on two classes. Left: The target concept is characterized by an underlying repetitive pattern among regions (squares). Middle: Traditional classification techniques are designed to partition the input space into class-dominant regions, disregarding the presence of any repetitive pattern. Right: Repetitive patterns can show in many forms (e.g., concentric circles).

II. Preliminary Concepts

A. Basic Notation in Classification.

Let $(A_1, A_2, \cdots, A_n)$ be an n-component vector-valued random variable, where each A_i represents an attribute or feature; the space of all possible attribute vectors is called the input space $\mathcal{X}$. Let $\{y_1, y_2, \cdots, y_k\}$ be the possible classes, categories, or states of nature; the space of all possible classes is called the output space $\mathcal{Y}$. A classifier receives as input a set of training examples $T = \{(\mathbf{x}, y)\}$, $|T| = N$, where $\mathbf{x} = (a_1, a_2, \cdots, a_n)$ is a vector or point of the input space and y is a point of the output space. We assume T consists of independently and identically distributed (i.i.d.) examples obtained according to a fixed but unknown joint probability distribution in the input-output space $\mathcal{X} \times \mathcal{Y}$. The outcome of the classifier is a function f (or hypothesis) mapping the input space to the output space, $f : \mathcal{X} \to \mathcal{Y}$. Such mapping can be used to identify regions with a dominant single class, hereby referred to as DSC-regions.

B. Basic Concepts in Pattern Theory.

We now introduce basic notation in pattern theory as originally proposed by Grenander [9], [10]. The main idea is to provide a language to express regularities that are found in real processes or structures. The following discussion deviates from traditional classification; it is focused on the representation of patterns rather than class prediction. We bridge the gap between pattern theory and classification in subsequent sections. We begin by assuming a pattern is the composition of basic blocks or generators; a pattern can be fully described by detailing the nature of its generators and the rules of composition that determine how generators glue together. We narrow our discussion to situations where graph theory can be used as the representational language. Under this setting, generators can be represented as graph nodes where each link or bond coming out of a node can be connected to another bond if certain criteria is met. A set of connected generators forms a particular configuration; pattern theory will help us determine if a configuration follows a set of combination rules (i.e., will help us define the space of regular configurations).

More specifically, a basic building block or generator will be denoted as $g \in G$, where G is the generator space. A generator g will have attached to it bonds $\{b_i\}$ to combine with other generators; bonds will have corresponding values $\{\beta_i\}$. To know if two generators can be combined we make use of a bond value relation $\rho(\beta_i, \beta_j)$, that equals 1 or TRUE when such combination is valid, and equals 0 or FALSE otherwise. We can think of generators as pieces of a puzzle; each piece can only be joined with other pieces when bonds make a perfect fit.

To represent symmetries among patterns we can make use of similarity functions. A similarity $s \in S$ (where S is a similarity group) builds a mapping $s : G \to G$ that will help us know when two generators are effectively similar. For example, the generator space G will sometimes be easy to obtain from a smaller set called the initial generator space $G' \subset G$. We do that by employing a similarity group Δ such that any generator g can be obtained from another one $g' \in G'$ through a transformation $g = \delta g'$, where $\delta \in \Delta$.

As a practical illustration, let us define a two-dimensional board with alternating (e.g., white and black) tiles as in a checkerboard by defining an initial generator space $G' = \{g_0, g_1, g_2\}$ as shown in Fig. 2. Generator g_0 stands for one corner tile, g_1 stands for an inner tile, and g_2 stands for a border tile. As shown in Fig. 2, g_0 can be used as the bottom-left corner of the board; other corners can be easily represented by rotating g_0 through increments of 90^0 degrees or $\pi/2$ radians. We can define then a similarity group Δ made of rotation increments of $\pi/2$, $\Delta = \{\delta_{\frac{\pi}{2}}, \delta_{\pi}, \delta_{\frac{3}{2}\pi}\}$. The four corner tiles can now be made readily available through similarity transformations: $\delta_{\frac{\pi}{2}} g_0$, $\delta_{\pi} g_0$, and $\delta_{\frac{3}{2}\pi} g_0$. Equivalent transformations applied to generator g_2 enable us to produce the whole generator space G needed to build our checkerboard.

Fig. 3 shows how generators can be combined in this simple scenario; generators are merged with each other according to a binary bond value relation ρ. Specifically, let us assume each generator is attached a class label that can either take the value of white (w) or black (b); the bonds coming out of each generator will have the same value: β_w for white tiles and β_b for black tiles. It is now easy to see that forming a checkerboard is achieved by defining the bond value relation as follows: $\rho(\beta_w, \beta_b) = 1$ and $\rho(\beta_b, \beta_w) = 1$, while $\rho(\beta_w, \beta_w) = 0$ and $\rho(\beta_b, \beta_b) = 0$. This means two bonds can only be joined when they belong to tiles of different class (or color).

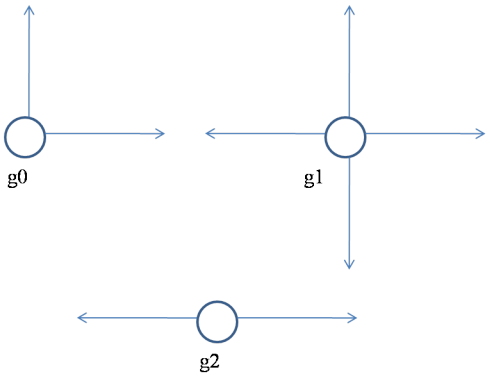

Fig. 2: Three basic generators from which other generators can be derived using rotational transformations.

The example given above is extremely simple. Real-world problems are expected to be of a different kind, where each generator can be assigned a class label that contains complex information (e.g., class vectors such as spatial coordinates); in addition, bonds will be expected to take on different values to define intricate structures.

In general, we define a configuration architecture of generators as $c = \sigma(g_0, \cdots, g_k)$, and represent it as a graph that shows how generators are combined in a specific form, as in Fig. 3. Those bonds connecting to other bonds will be called *internal*; those bonds that remain unconnected will be called *external*. For example, generator g_0 in Fig. 2 considered alone has two external bonds, while the graph shown in Fig. 3 is made of internal bonds only (because all bonds are connected). We say a configuration c is *locally regular* if every pair of internal connected bonds is allowed by the bond value relation. In other words, a configuration is locally regular if the following logical conjunction is true:

$$\bigwedge_{\beta_i,\beta_j} \rho(\beta_i,\beta_j) = \text{TRUE} \tag{1}$$

for all internal bonds β_i, β_j that connect together, and according to bond value relation ρ.

We can also talk of a configuration being *globally regular* when the graph represented by configuration c belongs to a family Σ of graphs such as linear graphs, those forming a lattice, or those forming a tree. When configuration c is both locally and globally regular we simply say it is regular; the space of all such configurations forms a space $\mathcal{C}(\mathcal{R})$ that we refer to as a *configuration space*, where $\mathcal{R}$ is referred to as a *regularity* and is defined through a 4-tuple:

$$\mathcal{R} = < G, S, \rho, \Sigma > \tag{2}$$

where G is the generator space, S is the similarity group, ρ is the bond value relation, and Σ is the graph family.

Lastly, it is important to mention that the bond value relation ρ can be made more informative by relaxing its binary domain to a real domain, where $\rho(\beta_i,\beta_j) \in \mathbb{R}$. This allows us to talk about the degree of regularity of a configuration. One can then define the probability of a configuration as follows:

$$p(c) = \frac{1}{Z} \prod_{i,j} \rho(\beta_i,\beta_j) \tag{3}$$

for all couplings (β_i,β_j), where Z is a normalization factor. Such extension enables us to deal with noisy domains where the presence of a regular pattern can only be partially observed.

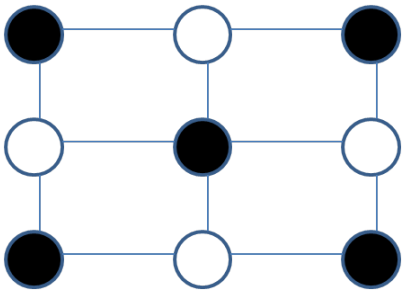

Fig. 3: A configuration of generators that make up a simple checkerboard pattern.

III. A Bridge Between Classification and Pattern Theory

Pattern theory can be employed for two main purposes: 1) synthesis and 2) analysis or inference. In synthesis, we verify how well the mathematical model capturing regularities resembles a real phenomenon; verification can be done through simulations, where any discrepancies with the real world can be used to refine such mathematical model. In analysis or inference, the goal is to design an algorithm that finds the pattern class to which the object under analysis belongs. This is the missing gap in the current understanding of classification of repetitive patterns, and which we address next.

A. Prediction Using Pattern Theory.

We refer to data patterns that repeat in a regular fashion by obeying certain (possibly stochastic) rules as *repetitive patterns*. In this section we discuss a general framework to learn repetitive patterns by exploiting the representational notation provided by pattern theory. Our general strategy comprises two steps. A first step identifies regions over the input-output space suggestive of the existence of a repetitive pattern; a second step formalizes the pattern identification step by assembling building blocks or generators over the data. We view both steps as complimentary; the second step is brought into play only when evidence is strong in favor of a repetitive pattern. In addition, the steps above assume the existence of few patterns that can be verified from data. The rationale is that it is unpractical to check all possible configurations among generators to find potential patterns. Instead we assume patterns a priori, and verify there existence on the training data.

Algorithm 1 shows a general framework to learn under the assumption of repetitive patterns. The idea is to first identify a set of DSC-regions $\{R_i\}$ that fall within a specified

Algorithm 1 A Learning Framework Using Pattern Theory

1: Let $\{R_i\}$ be the set of DSC regions according to family of configurations Σ
2: Assign generators $\{g_i\}$ to regions $\{R_i\}$
3: Let M be the corresponding pattern-theory model
4: Verify repetitive patterns according to Σ and ρ
5: **if** Pattern is present **then**
6: Use model M for prediction
7: **else**
8: Delegate prediction to a traditional learning algorithm
9: **end if**

family of configurations Σ. The next step is to verify if a repetitive pattern is present in the data. We do this by assigning generators to data regions. This is one of the major challenges we face to transition to pattern theory; DSC-regions need to be defined by assuming knowledge of potential repetitive patterns. Fig. 1, for example, shows a case where basic patterns (class-pure squares) bear no resemblance to regions found by a traditional learner (Fig. 1-middle). Clearly, the nature of the repetitive pattern can serve as knowledge that can be exploited to define our region set $\{R_i\}$.

Lastly, generating a predictive model that captures repetitive patterns carries an unavoidable degree of uncertainty, because our limited sample leaves gaps in the input-output space between generators. A mechanism is required to verify the connectivity of generators without invalidating the pattern under analysis. When a pattern is validated, the new model M outputs a prediction based on the class label that preserves the corresponding repetitive rule.

In short, verifying a repetitive patterns can be performed through a set of generators that form a configuration $c = \sigma(g_0, \cdots, g_k)$. Each generator g_i is defined having a set of bonds $B_s(g_i) = \{b_0^i, b_1^i, \cdots\}$, and bond values $B_v(g_i) = \{\beta_0^i, \beta_1^i, \cdots\}$. The goal is to find a configuration that stands as a good model of the distribution of the target concept over $\mathcal{X} \times \mathcal{Y}$. We begin by finding a *basic pattern*, a configuration c where there is no repetitive pattern within c (i.e., c contains a minimal pattern). Complex configurations can be defined in terms of simpler configurations $c = \sigma(c_0, c_1 \cdots)$ [9]. We illustrate these ideas through a practical scenario.

IV. The Parity Problem: A Case Study

Our study now turns to a concrete scenario that stands as a clear representative case where traditional classification techniques turn inadequate [11]. Parity is a concept that falls within the set of repetitive patterns; the parity concept can be simply stated as follows. For a binary-valued feature vector $(A_1, A_2, \cdots, A_n)$, where $A_i \in \{0,1\}$, the corresponding class takes value 0 when the number of features taking value 1 is even, and conversely the class takes value 1 when the number of features taking value 1 is odd. We can equivalently state the concept as follows:

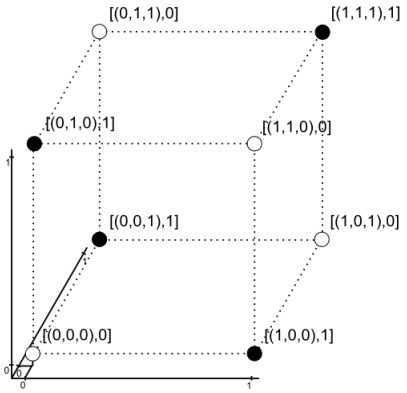

Fig. 4: 3-dimensional Parity Concept

$$f(\mathbf{x}) = \sum_{i=1}^{n} A_i \pmod 2 \tag{4}$$

In a 2-dimensional input space (with binary-valued features) the concept corresponds to logical operator XOR.

A. A Representation of 3-D Parity.

To better illustrate the notions of pattern theory, we begin by first defining a representational model for a 3-dimensional input space under the concept of parity, and then generalize it to the n-dimensional case. We assume each vector in the input space can in principle be represented with a generator, such that $|G| = |\mathcal{X}|$.

Fig. 4 shows a 3-dimensional parity mapping that is our object of representation. We aim at representing each class separately, where each class can have a representative generator from which all other vector points or generators can be derived. We refer to all generators of class 0 as G^0 and all generators of class 1 as G^1, and define an initial generator space G' by arbitrarily selecting one generator from each class. For example, we could choose g_1 corresponding to $x_1 = [(0,0,0),0]$ to represent class 0, and g_5 corresponding to $x_5 = [(0,0,1),1]$ to represent class 1 (Fig. 5). To obtain the other 3 generators for every class we have to adjust the position and direction of each representative generator through a set of transformations. We define a similarity group Δ to provide for such transformations, where each transformation $\delta_i \in \Delta$ is made of 3 rotational angles $\delta_i = (\theta_1, \theta_2, \theta_3)$, $\theta_j \in \{0, \pi\}$ (in radians), where $\pi = 180°$. We handle the following transformations: $\Delta = \{(\pi,0,0), (0,\pi,0), (0,0,\pi)\}$. Our representational notation in the realm of 3-dimensional parity can be summarized as follows:

$$\begin{aligned} &G = G^0 \cup G^1 \\ &G^0 = \{g_1, \cdots, g_4\}, \quad G^1 = \{g_5, \cdots, g_8\} \\ &G' = \{g_1, g_5\} \end{aligned} \tag{5}$$

$$\begin{aligned} &\Delta = \{\delta_i\}\ i = 1,2,3 \\ &\delta_i = \{(\theta_1^i, \theta_2^i, \theta_3^i) \mid \theta_j^i \in \{0, \pi\}\} \end{aligned}$$

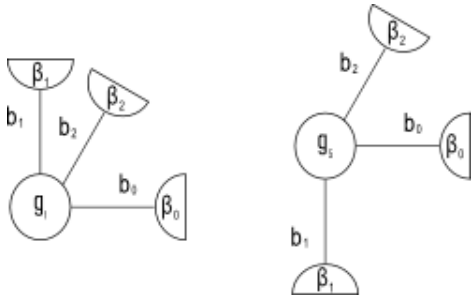

Fig. 5: Two representative generators for concept parity in a 3-dimensional space

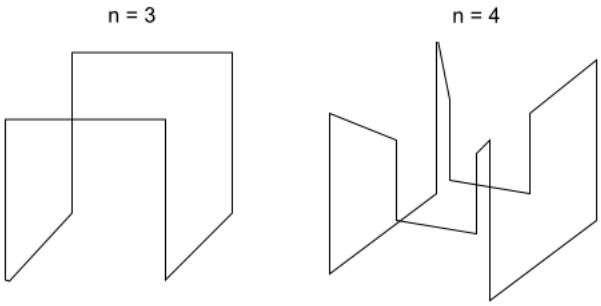

Fig. 6: Graphs for 3-dimensional and 4-dimensional domains

$$B_s(g_i) = (b_0^i, b_1^i, b_2^i)$$
$$B_v(g_i) = (\beta_0^i, \beta_1^i, \beta_2^i)$$

where $B_s(g_i)$ is the set of bonds for g_i, and $B_v(g_i)$ is the set of bond *values* for g_i. In our case bond values take on the class assigned to the corresponding vector in the input space (all bond values for a fixed generator are the same).

Lastly we define the bond-value relation $\rho(\beta_i, \beta_j)$. Since a class-0 generator always connects to a class-1 generator, we have the following relation:

$$\rho : B_v \times B_v \rightarrow \{\text{True}, \text{False}\} \tag{6}$$
$$\rho(\beta_i, \beta_j) = \begin{cases} \text{True} & if\ \beta_i \neq \beta_j \\ \text{False} & if\ \beta_i = \beta_j \end{cases}$$

B. Generalizing the Representation of Parity in n-Dimensions.

In the previous representational model, each generator has as many bonds as features or dimensions, $|B_s| = n$, which soon becomes a problem in high dimensional data sets. For an arbitrary n-dimensional space we can represent the concept of parity in a different manner. We note that in an n-dimensional hypercube, all vertices can be connected through a Hamiltonian path where every vertex connects to two neighbor vertices. It is then possible to define generators with 2 bonds, $|B_s| = 2$, and a set of transformation rules that can reach out to every vector in the instance space. Fig. 6 shows examples of Hamiltonian paths for $n = 3$ and $n = 4$ (the latter being a projection). Our representational notation for n-dimensional parity concepts is as follows:

Algorithm 2 Identifying parity concepts with pattern theory

```
Initialize no. of valid connections v = 0.
Find the k nearest neighbors around test example x
for each pair of nearest neighbors (x_i, x_j) do
    Find a path P_i of generators between x_i and x_j
    If P_i is incomplete, fill in gaps according to ρ
    If P_i is a valid connection then v = v + 1
end for
if v/k < τ (pattern is not present) then
    return Null (Refuse to output prediction for x)
else
    for each neighbor x_i of x do
        Find a path P_i of generators between x_i and x
        Let f_i be the class of the last generator in P_i assigned to x
    end for
    Let f* be the majority vote over {f_i}
    return f*
end if
```

$$G = G^0 \cup G^1 \tag{7}$$
$$G^0 = \{g_i\}, \quad G^1 = \{g_j\}, \quad G' = \{g_a, g_b\}$$

$$\Delta = \{\delta_i\}, \quad \delta_i = \{(\theta_1^i, \cdots, \theta_n^i) \mid \theta_j^i \in \{0, \pi, \pi/2\}\}$$

$$B_s(g_i) = (b_0^i, b_1^i)$$
$$B_v(g_i) = (\beta_0^i, \beta_1^i)$$

We just showed that even when the set of generators needed to produce the complete parity mapping is of size $|G| = 2^n$, we really only have to use two initial generators (with two bonds each), and obtain all others through a similarity group S. While the reduction of bonds saves processing time, the reduction of generators needed to work on certain region of the instance space saves on memory space.

C. A Learning Algorithm for Parity Concepts.

As mentioned before, it is convenient to begin under the assumption that a fixed type of regularity has been defined; an algorithm can then be devised to check for the presence of such regularity in local neighborhoods of the input-output space. Here, the regularity under study is that of the parity concept (as described above).

To show our ideas, we have implemented a simple algorithm (Algorithm 2) as follows. For a test example $\mathbf{x}$, we first check if the k neighbors around $\mathbf{x}$ provide evidence to assume the presence of a local regular configuration. The algorithm establishes a connection between every pair of neighbor points (excluding $\mathbf{x}$); a generator is attached to each neighbor point, and a path is created between the two points to check if the connections are valid according to a binary bond value relation ρ (here ρ corresponds to concept parity). Creating a connection is straightforward when the points are immediate

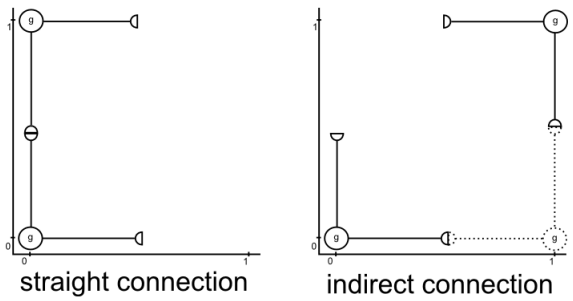

Fig. 7: Establishing a connection in a two-dimensional space. Left: Straight connections simply check if bonds can be connected according to bond value relation ρ. Right: Indirect connections need in addition finding a path between the two generators.

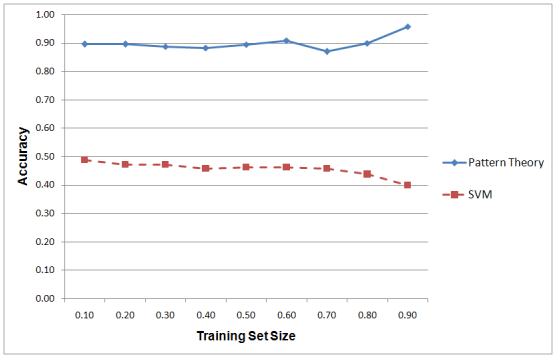

Fig. 8: Accuracy results on parity concept with 10% class noise. Our approach using notions of pattern theory is compared to SVMs.

neighbors (Fig. 7, left). But when this is not the case, we look for a path between the two generators to decide if the connection is valid (Fig. 7, right). Specifically, we adopt an optimistic approach to filling gaps, where we simply verify if a sequence of generators exists that obeys ρ. The algorithm favors the presence of a local regular configuration if the proportion of valid connections (for all pairs of neighbor points) is above a user-defined threshold τ. If a local regular configuration is favored, the algorithm generates a prediction for $\mathbf{x}$ according to the class that belongs to the last generator in the path that starts at each neighbor $\mathbf{x}_i$ and ends in $\mathbf{x}$ according to ρ. The final output for $\mathbf{x}$ is a majority voting over all k predictions.

Fig. 8 shows empirical results using artificial data comparing our approach (using $\tau = 0.7$) with support vector machines [12], [13], using a polynomial kernel. The learning task corresponds to parity with $n = 8$ attributes and 10% class noise. We use predictive accuracy (i.e., accuracy on a testing set) as the evaluation metric and vary the size of the training set by randomly drawing samples under a uniform distribution from the original set of size $N = 256$. The percentage of testing points rejected without prediction has a mean value of 4%. Confidence intervals assuming a normal distribution with a 95% confidence level have a maximum value of approx. ±2%. As shown in Fig. 8, our approach attains close to optimal performance (around Bayes' error). The traditional approach to learning –represented via SVMs– is unable to capture the relation among class-labels around every test example.

While previous work has shown alternative strategies to deal with parity concepts, our results serve as empirical evidence to support the feasibility of adapting pattern theory to concept learning. Generating predictive models based on the presence of repetitive patterns in data is a field mostly unexplored, and prone to enhance our learners beyond the traditional assumption of local class similarity.

V. Summary and Future Work

We point to the existence of learning problems where current classification algorithms are unequipped to deal with the existence of repetitive patterns in the input-output space. The key limitation consists of ignoring the potential interdependence among class-dominant regions. In this paper we propose a mechanism to look for repetitive patterns in data using notions of pattern theory; the idea is to represent basic regions in the input-output space as building blocks or generators that can be combined using (possible stochastic) rules. We view pattern theory not as an alternative solution to classification, but rather as an extension to our understanding of how to build accurate predictive models. Future work will look for real-world domains where the presence of repetitive rules can be discovered using our notions of pattern theory.

Acknowledgement

We are grateful to Roberto Valerio for his help in improving earlier versions of this manuscript. This material is based upon work supported by the National Science Foundation under Grant No. 0812372.

References

[1] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Wiley-Interscience, 2001.
[2] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. Springer, 2009.
[3] D. J. C. Mackay, *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.
[4] K. Huang, H. Yang, I. King, and M. Lyu, *Machine Learning: Modeling Data Locally and Globally*. Springer, 2008.
[5] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
[6] T. Jebara, *Machine Learning: Discriminative and Generative*. Kluwer Academic Publishers, 2004.
[7] E. Briscoe and J. Feldman, "Conceptual complexity and the bias/variance tradeoff," *Cognition*.
[8] K. P. Burnham and D. R. Anderson, *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*, 2nd ed. Springer, 2002.
[9] U. Grenander, *Elements of Pattern Theory*. The John Hopkins University Press, 1996.
[10] U. Grenander and M. Miller, *Pattern Theory: From Representation to Inference*. Oxford University Press, 2007.
[11] C. Thornton, "Parity: The problem that won't go away," in *Canadian Conference on Artificial Intelligence*. Springer, 1996, pp. 362–374.
[12] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.
[13] I. Steinwart and A. Christmann, *Support Vector Machines*. Springer, 2008.