Complexity-Based Tools for Automated Learning

Timothy Law Snyder
Department of Computer Science
Georgetown University
Washington, DC 20057

J. Michael Steele
Department of Statistics
The Wharton School
University of Pennsylvania
Philadelphia, PA 19104

EXTENDED ABSTRACT (4800 words, August 14, 1991)

A theoretical basis for automated learning is provided that is motivated by the need for tools for training and testing image recognition devices. The foundation rests extensively on ideas from theoretical computer science, and the model easily extends to the general problem of automated learning.

1. Introduction

The objective of this paper is to develop a complexity-based paradigm for the training of image recognition devices. The models and examples used offer a bridge between the problems of automated learning and some classical results from theoretical computer science. Though we use the image generation/recognition problem as a prototypical example, our paradigm is general enough to apply to virtually any form of automated learning.

Most of the technologies currently applied to image recognition require training the recognizer. In this training, the recognizer is provided a large number of incidences of feasible input images and is given feedback on the accuracy of its responses. For the training to be effective, the training data must be appropriate and often needs to be extensive. In the well-studied context of optical character recognition, one can easily generate relevant data in the quantities required, but in other important contexts, the practical problems associated with the physical generation of the training data are nearly insurmountable. Even in less complex circumstances, if one requires a high level of recognition correctness, say on the order of one error per million trials, then one immediately finds a compelling need for training data sets that are much larger than can be achieved by physical generation.

The traditional problems that arise when trying to generate data suitable for training a learning device are not related to complexity theory. In image generation, for example, one quickly finds critical problems in statistical modeling, image modeling, and image perturbation. Still, for the problems of generation of instances for training and testing the quality of learning, there is a foundational need for the generation problem to be *easy* and the for recognition problem to be *hard*. Though these intuitive descriptions do not conform to off-the-shelf tools of computational complexity, the insights of computational complexity still can be made relevant to the problems one faces in practice.

When one considers the rigorous complexity theory of automated training, a useful paradigm that arises is that of an *adversary-based* generation/recognition. We present such a paradigm in Section 2, along with a key example. The example deals with a problem from the theory of graphs rather than true imagery, but its simplicity sorts out many issues that have been considered problematic for some time.

Besides simply bridging the gap between traditional automated learning and classical complexity theory, our paradigm offers other advantages. For example, the generator is afforded virtually complete control over the difficulty of the problem instance presented to the recognizer. Section 3 develops several scenarios that the generator has at its disposal; these include *complete information* versus *partial information*, and *subversive partial information*. The distinctions seem to be new—yet essential—to the rigorous discussion of image generation driven by theoretical computer science.

Section 4 contains additional examples that illustrate the range of new possibilities the paradigm offers. Included are the specification of the data to be transmitted to the recognizer, the impact that variations in this data have on the complexity of recognizing the image, and algorithmic issues involved with generating the images. Section 5 illustrates strategies that the recognizer can use for the scenarios developed in Section 3.

2. A Generation/Recognition Paradigm

Our goal is to generate a set of high-complexity images that are in certain circumstances amenable to low-complexity recognition. We begin by noting that there is a massive body of literature that deals with the complexity of problem solving, especially from the computational standpoint (see, *e.g.*, Aho, Hopcroft, and Ullman (1974), Papadimitriou and Steiglitz (1982), and Garey and Johnson (1979)).

In the analyses, discussions, and examples that follow, we uniformly use the worst-case measure of theoretical computer science as our measure of complexity, and we assume the reader is familiar with the complexity classes P and NP-complete. We define NP-complete problems to be of *high-complexity* in terms of recognition; problems in P, however, we consider to be *low-complexity* recognition problems. This simple distinction results in a wide variety of image-generation examples, as we will show. For us, of course, a *problem instance* is an image and some supplementary information, which we define as follows.

Let $\mathcal{X}$ be a set of images. Of these, only a subset will be feasible; let the feasible set of images be $\mathcal{F} \subset \mathcal{X}$. The images in $\mathcal{F}$ belong to two categories. The first category is the set of images in $\mathcal{F}$ that possess some predetermined criterion or criteria; these images comprise the set of "yes" instances. The second category is the subset of all images of $\mathcal{F}$ that fail to possess at least one of the features of merit that distinguish the yes instances from the no (non-yes) instances. For any image $x \in \mathcal{X}$, let

$$y(x) = \begin{cases} 1, & \text{if } x \text{ is a yes instance;} \\ 0, & \text{if } x \text{ is not a yes instance.} \end{cases} \tag{2.1}$$

So, the function $y : \mathcal{X} \to \{0, 1\}$ partitions the set of images into "yes" and "no" instances.

Let the set of yes instances be $\mathcal{Y} \subset \mathcal{F} \subset \mathcal{X}$, *i.e.*, $\mathcal{Y} = \{x : y(x) = 1,\ x \in \mathcal{F}\}$. The recognizer needs to determine, for a given problem instance, whether $x \in \mathcal{Y}$. Note that the definition of $\mathcal{Y}$ requires that a yes instance must be feasible.

We can completely define the set $\mathcal{Y}$ of yes instances by an *attribute set* $A = \{A_1, A_2, \ldots, A_k\}$, where each A_i is an *attribute* of a yes instance. Formally, for all $1 \leq i \leq k$, let $A_i : \mathcal{X} \rightarrow \{0, 1\}$, where $A_i(x) = 1$ if and only if the image x possesses the attribute associated with A_i. For an image to be considered a yes instance, it must possess all the A_i. Furthermore, we require that if the recognizer if in possession of the entire set A, then, for all $x \in \mathcal{X}$, there exists a polynomial-time algorithm to determine whether $x \in \mathcal{Y}$.

The notion of attributes allows us to model the notion of *partial information*, which we define as a proper subset A' of A. The *complete information* set, naturally, is A, and the *no information* corresponds to $A = \emptyset$.

Along with the attribute set A, the generator creates (or is given) a *feasibility set* $B = \{B_1, B_2, \ldots, B_j\}$. The B_i are also yes/no functions, with $B_i : \mathcal{X} \rightarrow \{0, 1\}$, for $1 \leq i \leq j$, and $B_i(x) = 1$ if and only if x satisfies the feasibility condition associated with B_i. An image $x \in \mathcal{F}$ if and only if $B_i(x) = 1$ for all $1 \leq i \leq j$. We restrict the feasibility conditions to be such that, for all $x \in \mathcal{X}$, the computation of $B(x)$ can be accomplished in polynomial time.

Combining the conditions required by the attribution and feasibility sets, we can now completely characterize $\mathcal{Y}$:

$$x \in \mathcal{Y} \iff A_i(x) = 1 \text{ for all } 1 \leq i \leq k \text{ and } B_i(x) = 1 \text{ for all } 1 \leq i \leq j. \tag{2.2}$$

For some problems and applications feasibility is not an issue. In these cases, we set $B = \emptyset$ and $\mathcal{F} = \mathcal{X}$, thereby making all images of $\mathcal{X}$ feasible, focusing only on the attribute set of the images.

We now have a model that allows us to present our generation/recognition paradigm. In words, the paradigm is as follows. First, the full attribute and feasibility sets are established, *i.e.*, A and B (hence, $\mathcal{Y}$ and $\mathcal{F}$) are determined. The following process is then repeated. The *image generator* creates, using some algorithm or system of rules, an image $x \in \mathcal{X}$ along with the value of $y(x)$, and, sometimes, some extra information in the form of a subset of A. The subset can be empty, a proper subset of A, or all of A. The generator then passes x and the selected attributional information to the *image recognizer*, which must determine $y(x)$. Based on previous experience, the recognizer proceeds by making the best selection it is able to make for the value of $y(x)$, and it passes this "best guess" to the generator. The generator then makes the recognizer privy to the true value of $y(x)$, informing the recognizer of the correctness of its guess. The process is repeated, with the generator controlling the difficulty of the recognizer's task by its selection of the attributes that are passed along with each images.

More formally, let G be our image generator, and let R be the image recognizer. Though we could formally model G and R using functions, we choose this notation to keep the description simple. The following eight-step process characterizes the image generation and recognition paradigm:

1. G establishes (or is given) $\mathcal{X}$, A, and B;
2. From A and B, G forms the problem instance $(x, A', B, y(x))$, where $x \in \mathcal{X}$, $A' \subseteq A$, and

$y : \mathcal{X} \to \{0,1\}$, with $y(x) = 1 \iff x \in \mathcal{Y}$;

3. G passes (x, A', B) (but not $y(x)$) to R;
4. R first reconciles whether $x \in \mathcal{F}$ by computing $B_i(x)$ for all $1 \le i \le j$ (by definition, this can be done in polynomial time);
5. If $B_i(x) = 1$ for all $1 \le i \le j$ (*i.e.*, if $x \in \mathcal{F}$), then R attempts to reconcile whether $x \in \mathcal{Y}$ by computing, if possible, $A_i(x)$, for all $1 \le i \le k$;
6. From the results of Steps 4 and 5, R then guesses whether $x \in \mathcal{Y}$ by computing $r(x)$, where $r : \mathcal{X} \to \{0,1\}$ is R's best guess for $y(x)$;
7. R passes $r(x)$ to G; and
8. G computes and passes to R the value $\mathbf{1}(r(x))$, where $\mathbf{1}(r(x)) = 1$ if and only if $r(x) = y(x)$ (*i.e.*, if and only if R's guess $r(x)$ is correct).

Of course, Steps 2 through 8, which comprise a single iteration of the paradigm, are repeated as the recognizer is trained. We note that the entire feasibility set B is passed to the recognizer. This means that the recognizer can determine the feasibility of x in low-complexity time. Hence, the potentially laborious part of the recognizer's task is in dealing with the A_i's, which are selectively revealed at the discretion of the generator.

The generation/recognition paradigm gives us a framework we will use in future examples and augmented models. We first present an example that will guide us through the concepts associated with these models. Though the example is perhaps too simple, we will see that it contains a surprising richness in terms of the generation and recognition issues it invokes.

Let $\mathcal{X} = \{\, G : G \text{ is a graph } \}$, so that the image generated is a graph $G = (V, E) \in \mathcal{X}$, and let $d(v)$, for a vertex $v \in V$, be the degree of v. The following complete attribution set is known to the image generator:

$$A = \{A_1, A_2\},$$

where

$$\begin{aligned} A_1(G) = 1 &\iff G \text{ has a Hamiltonian cycle, and} \\ A_2(G) = 1 &\iff \text{For all } v \in V,\ d(v) = 2. \end{aligned} \tag{2.3}$$

Furthermore, let $B = \emptyset$, so that $\mathcal{F} = \mathcal{X}$.

Clearly, the complete problem instance (G, A) is in P because, if the recognizer determines that A_2 is true for G, then the normally NP-complete attribute A_1 can be easily reconciled just by ascertaining whether G is connected. This gives us a linear-time algorithm for determining $y(G)$ in the presence of the complete information set A, provided the recognizer is smart enough to resolve A_2 before attempting to resolve A_1. (The problem of the order of resolution of the attributes by the recognizer will be formally addressed in Section 5.)

But, the generator has the option of passing any subset of A to the recognizer. In the next section, we survey the strategies the generator can use to vary the difficulty of the recognizer's task.

3. Options for the Generator: Complete, Partial, and No Information

By its definition of the attribute set A, complete knowledge of A leads to low-complexity (polynomial-time) recognition. In the generation/recognition paradigm, however, the generator may pass only partial information $A' \subset A$ or no information at all to the recognizer. This leads to three general scenarios.

3.1. The No-Information Scenario

If the problem instance is void of supplementary information and consists only of an image, then the recognizer must perform its task in the presence of *no information*. The perils of this scenario are discussed in the full version of this paper.

3.2. The Partial-Information Scenario: Subversive vs. Non-Subversive Information

In most circumstances in which a recognizer is likely to be used, the recognizer has clues as to what it is attempting to recognize in the form of reference standards, pre-programmed procedures, or A_i that arrive with the current or previous images. This corresponds to the partial information scenario $A' \subset A$, with $A' \neq A$.

We can subdivide the partial-information scenario into two categories: *subversive* and *non-subversive* partial information. The key feature that separates the two categories is the nature of the information that is withheld by the generator, *i.e.*, the attributes belonging to $A - A'$. If the problem instance formed by an image and the partial information set A' belongs to P, then we say the partial information is *non-subversive*, for the information is complete enough to guarantee the *existence* of a polynomial-time algorithm for the problem instance, making it of either current or eventual low-complexity.

Note that non-subversive partial information does not, however, guarantee that the recognizer will discover all the key ingredients, A, that allow it to determine without error the value $y(x)$ for all $x \in \mathcal{X}$. So, even in the presence of non-subversive partial information, the recognizer's task remains non-trivial.

In some cases, key or critical attributes of A will be withheld from the recognizer. If these ingredients are such that the problem instance resulting from an image and A' is an instance of a problem that is NP-complete, then we say that the partial information is *subversive*, since the withheld information in $A - A'$ is necessary if the image is to be recognized with low complexity.

To illustrate these concepts, we return to the example of Section 2, where $\mathcal{X} = \{ G : G \text{ is a graph } \}$ and $A = \{A_1, A_2\}$ and A_1 and A_2 correspond to G having a Hamiltonian cycle and being of regular-degree two, as specified by (2.3).

Consider the recognizer's task in the presence of subversive partial information. If the recognizer is given the image G along with

$$A' = \{A_1\}, \tag{3.1}$$

without the augmenting information in A_2 that makes the problem tractable, it must resolve whether G has a Hamiltonian cycle. In general, this is an NP-complete problem.

Furthermore, consider now the situation in which the recognizer is given A' as defined in Equation (3.1), and G contains a Hamiltonian cycle but does not satisfy attribute A_2 in (2.3). In this case, even if the recognizer is lucky enough to find a Hamiltonian cycle in G and therefore guess that $y(G) = 1$ (by transmitting $r(G) = 1$ to the generator), its efforts would be frustrated when the generator responds with $y(G) = 0$ (actually, $\mathbf{1}(r(G)) = 0$) since A_2 is not satisfied in G. This is why we call partial information such as A' in (3.1) subversive: without the information in $A - A'$, the problem is NP-complete, and, even if the NP-complete problem instance is "solved," the recognizer is still faced with a situation that is barely better than having no information at all.

Since NP-complete problems are intractable, it is of paramount priority that the recognizer be able to identify subversive partial information when it is received. Since so many NP-complete problems are well-cataloged (Garey and Johnson (1979)), we deem this to be a feasible task. In Section 5, we consider general options available to the recognizer when it is faced with subversive partial information.

Before doing so, we first present, in the next section, more examples that serve as reasonable systems for the generation of recognizable high-complexity images.

4. Examples of High-Complexity Generation with Low-Complexity Recognition

This section contains examples of problems that suit our goal of high-complexity images that are recognizable in low-complexity time for a system (recognizer) with complete information. To make the recognizer's task more difficult, the generator will usually choose an attribute set that results in the transmission of subversive partial information.

Our examples come in three flavors: (1) graph-theoretical, (2) string-based, and (3) image-like. For each example, we will discuss the problem definition, the proposed partial information to be transmitted to the recognizer, the issues involved with recognizing the image, and the issues, including feasibility and available algorithms, involved with generating the images.

We begin with the graph-theoretical examples, the first of which is the prototypical example of the last section. The reader is reminded that the case $B = \emptyset$ corresponds to $\mathcal{F} = \mathcal{X}$, *i.e.*, the case in which feasibility is not an issue.

4.1. Graph-Theoretical Examples

1. Hamiltonian Cycles in 2-Regular Graphs

1.A. *Problem Definition*:

Let $\mathcal{X} = \{\, G : G \text{ is a graph } \}$ and $A = \{A_1, A_2\}$, where, for $G = (V, E) \in \mathcal{X}$, $A_1(G) = 1$ if and only if G has a Hamiltonian cycle and $A_2(G) = 1$ if and only if $d(v) = 2$ for all $v \in V$. Also, let $B = \emptyset$.

1.B. *Partial Information*:

$A' = \{A_1\}$; subversive.

1.C. *Recognition Issues:*

For the complete information case, the recognizer needs only to realize that $x \in \mathcal{Y}$ if and only if G is connected. Once this is accomplished, the recognizer can determine membership in $\mathcal{Y}$ in linear time using any search method, say, depth-first search (Tarjan (1972)). The partial information case is subversive.

1.D. *Generation Issues:*

Generating the images is trivial: To generate a $G \in \mathcal{X}$ such that $y(G) = 1$, the generator simply chooses a non-negative integer n and forms a cycle on n vertices. An image G that fails to be Hamiltonian but satisfies A_2 is simply a set of disjoint cycles, and a G that is Hamiltonian but fails A_2 can be generated by beginning with a simple cycle and augmenting the cycle graph with additional edges. If the generator wishes to generate graphs that satisfy neither condition, the key issue is the Hamiltonian path. It is not trivial to generate a highly-complex graph that is guaranteed to not be Hamiltonian. Perhaps the easiest way is to generate a graph that is disconnected. A second and more reasonable example guaranteed to be non-Hamiltonian is a bipartite graph whose total number of vertices is odd. Many canonical graphs, such as the Peterson, Horton, and Herschel graphs, are also known to be non-Hamiltonian (Bondy and Murty (1976)).

2. Traveling Salesman Tours of Short Length; Christofides' Heuristic

2.A. *Problem Definition:*

The *traveling salesman problem* is to find, in a graph $G = (V, E)$ with weight function $w : E \rightarrow \mathbb{R}$, a Hamiltonian cycle of smallest weight, where the weight of the cycle C is $\sum_{e \in C} w(e)$. The traveling salesman problem is *NP*-complete. One heuristic method that finds a reasonably short tour is *Christofides' heuristic* (Christofides (1976)), which is guaranteed to find a tour T_C such that weight(T_C)/weight$(T_\star) \leq 3/2$, where $T_\star$ is an optimal traveling salesman tour. Let $\mathcal{X}$ be the set of all weighted graphs. For $G \in \mathcal{X}$, let $A = \{A_1, A_2\}$, where $A_1(G) = 1$ if and only if G contains a tour of weight less than α, where α is given. Let $A_2(G) = 1$ if and only if G contains a tour of weight less than α that can be found by Christofides' heuristic. Also, let $B = \emptyset$.

2.B. *Partial Information:*

$A' = \{A_1\}$; subversive.

2.C. *Recognition Issues:*

Recognition is easy in the presence of complete information, for Christofides' heuristic runs in polynomial time. If the image G fails to have a tour of weight less than α that can be found by Christofides' heuristic, then $G \notin \mathcal{Y}$. On the other hand, if $A_2(G) = 1$, then $A_1(G) = 1$, as well, for the optimal tour has weight no more than that of a tour found by Christofides' heuristic.

2.D. *Generation Issues:*

To generate graphs that have traveling salesman tours of weight less than α, simply start with a cycle, say, of length n, assign a weight of $\alpha/n - \epsilon$, where $\epsilon > 0$, to each edge in the cycle, then augment the cycle with any number of edges, assigning weights to the new edges arbitrarily (or randomly). To build a graph with no traveling salesman tour of length less than α, again begin with a cycle on n vertices, then augment the graph with as many edges as desired. But, this time, assign a weight of α/n or more to each edge. Since the traveling salesman tour must contain exactly n edges, all tours are guaranteed to be of weight at least α, which is too much for the graph to belong to $\mathcal{Y}$.

3. Euclidean Minimum Steiner Trees with One Steiner Point

3.A. *Problem Definition*:

The *Euclidean Steiner tree problem* is to find a minimum-length graph that spans a given set of points $S \subset \mathbb{R}^d$, where the interpoint distances are determined by the Euclidean (L_2) metric and the allowable edges come from the complete graph on S. The length of the optimal graph, which which is called a *Euclidean minimum Steiner tree*, is sum of its edge-lengths. The Euclidean Steiner tree problem is *NP*-complete (Garey, Graham, and Johnson (1976)). When forming a minimum Steiner tree of S, the tree can be augmented with points that do not belong to S; such points are called *Steiner points*. Let $\mathcal{X}$ be the collection of all point sets S of cardinality n. For $S \in \mathcal{X}$, let $A = \{A_1, A_2\}$, where $A_1(S) = 1$ if and only if S has a minimum Steiner tree of length less than α, where α is given. Let $A_2(S) = 1$ if and only if S has a Steiner tree of length less than α, when the problem is constrained so that a maximum of one Steiner point is permitted. Also, let $B = \emptyset$.

3.B. *Partial Information*:

$A' = \{A_1\}$; subversive

3.C. *Recognition Issues*:

Recognition is surprisingly easy in the presence of complete information. Though the problem associated with A_1 is *NP*-complete, the problem associated with A_2, the 1-*Steiner problem*, has an efficient quadratic solution (Georgakopoulos and Papadimitriou (1987)). Since the 1-Steiner problem associated with attribute A_2 is solvable, and since the existence of a Steiner tree of length less than α using only one Steiner point implies the existence of a minimum-length Steiner tree of length less than α (using any number of Steiner points), the recognition problem is solvable in the presence of complete information.

3.D. *Generation Issues*:

Given α, the generation issues for this problem are tough, for it is difficult to find a point set that guarantees a minimum Steiner tree of a given length. The best way to generate point sets that do or do not satisfy attribute A_2 for the 1-Steiner problem is to choose α *after* constructing the point set. In other words, choose any set S, then compute the length of a minimum 1-Steiner tree using Georgakopoulos and Papadimitriou's algorithm. Once the length of a minimum 1-Steiner tree is known, α can be chosen to exceed or not exceed the length. Another strategy, for point sets in the unit cube, chooses an α that exceeds the maximum possible length that the Steiner tree can assume (see Snyder (1990)).

4.2. String-Based Examples