Abstract— Binary classification, used for Boolean data, is a particular case of supervised machine learning. The Logical Analysis of Data (LAD) has first emerged as a supervised binary classification method, later improved to include non-Boolean observations. Using Java™ language, we implement the LAD method and build the induction process (called theory) from scratch. The underlying set covering optimization problem arising in pattern selection is resolved using a solver from GAMS. Our improved implementation, through the avoidance of top-down stage, ensures that pattern generation process stops when all the observations are covered. The main findings are that the LAD method supported by the improved selection procedure gives better results, making the processing faster with less memory demand.

Although LAD has been used for about two decades now, there is no general program that can be used to solve problems. Researchers have to re-write the same functions for different problems. In order to address this issue, we present a general implementation of the most basic functions required by LAD as a starting point towards a complete environment like WEKA¹, one of the related tools used in machine learning and data mining.

II. LAD METHODOLOGY

A. Background

In 1986, Peter L. Hammer described the basic idea of LAD for the first time in a lecture given at the International Conference on Multi-attribute Decision Making via OR-based Expert Systems [2]. This seminal idea was later developed in [3]. Those first contributions were later followed by several research studies, many of which can be found in the list of references given in [4]. As for many other research areas, in early LAD contributions, the concentration was put on theoretical improvements and on basic computational implementation. At a later stage, attention was focused on practical applications spanning areas from medicine to credit risk ratings. The objective of the present paper is to use the theoretical foundations of the LAD method in order to end up with a tangible software product to be used in diversified applications.

B. Motivation

We can summarize the main points that have motivated our work on LAD as follows:
- It is a branch of applied machine learning / data mining with many academic contributions and diversified application areas spanning major vital and real-life sectors.
- It produced many new algorithms; some of these are still under improvement.
- It gives high accuracy than other similar methods making the LAD method an important subject area.
- Up to date, there is no general program or toolbox for LAD.

¹ http://www.cs.waikato.ac.nz/ml/weka/
C. Main Applications

The LAD has been applied to the following: [4],
- Diagnosed acute ischemic stroke.
- Revealed, for the first time, a correlation between some chemical structures and their efficiency in transfecting DNA.
- LAD was used to identify Chinese labor productivity patterns. It explained changes in productivity and led to a decision support system (DSS) aimed at increasing productivity of labor in China’s provinces.
- Contributed to breast cancer diagnosis.

More recent contributions include the application of LAD to forecast show rate at an airliner. LAD offers better predictions than the current tool [5]. LAD has been applied to a very small database of belt-conveyor-related accidents. The method is used as innovative approach to prevent machinery-related accidents [6]. Despite rare data, LAD produces patterns with adequate classification accuracy.

D. Methodology

1) Patterns

LAD is based on pattern generation. This step generates a set of Boolean rules called patterns that characterize the observations membership to one class only. Patterns are the main building block of the LAD method, because an observation can be classified into the corresponding class based on them. A pattern is composed of a conjunction of literals; a literal is a Boolean variable or its negation.

Strictly speaking, a pattern \( p \) of degree \( a \) is the conjunction of \( a \) literals such that it is true for at least one observation of one class, and false for all observations of all other classes. If a pattern is true for a certain observation, then it is said to cover that particular observation. Less strictly, a pattern \( p \) allows for a larger percentage of coverage for one class and a smaller coverage for other classes [1].

2) Positive prime patterns and negative prime patterns

Positive (resp. negative) prime patterns of the partially defined Boolean function \( f \) are the prime implicants of \( f \) covering at least one point in the set of positive (resp. negative) observations. Positive prime patterns (PPPs) and negative prime patterns (NPPs) are used in the pattern generation procedure.

3) Pattern generation

There are basically three different approaches in the generation of PPPs (resp. NPPs) [1], [7].

a) Top-down approach

The method starts by coupling to every positive observation its characteristic term, which is obviously a pattern. Even after the removal of some literals, this term might remain a pattern. The top-down procedure eliminates literals one by one until reaching a prime pattern.

b) Bottom-up approach

At the opposite, the bottom-up approach begins with terms of degree one covering some positive observations. This term is confirmed a positive pattern if it covers none of the negative observations. In the case where the term is not a pattern, literals are added one by one to the term until obtaining a pattern. This is a term enumeration technique which can be computationally very costly, and practically unfeasible.

For example, if we consider the case with \( n=3 \) variables and degree \( d=2 \). What are the possible patterns? We have to choose all combinations of words of length 2. In the case of unary variables, the combinations are simply given by:

\[
\binom{3}{2} = \frac{3!}{2!(3-2)!} = 3. \quad (1)
\]

The result is the set of combinations \( \{ x_1 x_2 , x_2 x_3 , x_1 x_3 \} \). For the case of Boolean variables, this result is multiplied by \( 2^n \), with the exponent representing a prescribed degree. For the Boolean case, the total combinations of terms are:

\[
\binom{3}{2} \times 2^3 = 3 \times 8 = 12 \text{ terms} \quad (2)
\]

It is easy to check by induction that, for the general case of \( n \) Boolean variables and degree \( d \), the number of terms of length \( d \) is given by:

\[
\text{#terms} = \binom{n}{d} \times 2^d = \frac{n!}{d!(n-d)!} \times 2^d \quad (3)
\]

As a result, even for \( n \) having a reasonable practical value (say, \( n=30 \)), the number of terms is a very rapidly growing function of \( d \); hence the extreme selectivity process of the terms.

c) Hybrid approach and algorithm

This approach is achieved by a combination of bottom-up and top-down approaches, while favoring the bottom-up strategy. It starts by the bottom-up approach to produce all the patterns of small degrees, (4 or 5, say). Then it uses a top-down approach to cover those positive observations (if any) that remained uncovered after the bottom-up step. The pattern generation process is directed by two rational objectives, namely simplicity and comprehensiveness. Simplicity gives preference to the production of short patterns and comprehensiveness attempts to cover every positive observation by at least one pattern.

The basic overall steps of the Hybrid Pattern Generation Algorithm are described in Figure 1 below.

| Step 1 | First apply Bottom-up Pattern Generation Algorithm Output P(D) // set of PPPs up to a degree D; |
| Step 2 | For any term \( a \) in positive observations (S+) not covered by any pattern in \( P(D) \) obtained in step 1; Do \( Pa := \text{minterm}(a) \) Apply Top-Down Pattern Generation Algorithm Endfor |
| Step 3 | Eliminate any redundant patterns; |
| Step 4 | Output P. |

Algorithm 1 - Hybrid Pattern Generation Algorithm for NNPs/PPPs Generation
4) Theory formation
   a) Characteristics of a theory

   It is expected that an efficiently chosen group of patterns can be used for building a general classification rule. This rule is no more than an extension of a partially defined Boolean function and is referred to as a theory. A good theory should capture all the significant aspects of the problem. The disjunction of positive patterns is called a positive theory and vice versa. An extension $\varphi$ of $f$ such that $\varphi$ is a positive theory and $\bar{\varphi}$ is a negative theory is called a bi-theory [4].

   b) Discriminant

   The weighted sum of positive and negative prime patterns is called a discriminant and is represented as follows:

   $\Delta = \sum_{i=1}^{r} w_i P_i + \sum_{i=0}^{s} w_i N_i \quad (4)$

   A large positive (resp. negative) value of $\Delta$ suggests a positive (resp. negative) character of the new observation.

III. LAD IMPLEMENTATION

   In our context, the main goal behind implementation is the production of a software capable of solving classification problems using LAD. We start by modeling the solution using Unified Modeling Language (UML®) [8] and then obtain the corresponding Java™ code [9]. Because LAD is based on optimizing the number of generated patterns via selection, we therefore need a ready-made linear programming software. For this task, we use GAMS [10] as a high-level optimization language, principally its MIP solver. Some authors have used mixed 0-1 integer and linear programming (MILP) approach to produce patterns [11]. The interactions of UML®, and Java™, on the one hand, and GAMS on the other hand, are described below.

A. System Architecture
   1) Overview of the solution

   Figure 1 describes the overall architecture of the proposed solution. The Java™ implementation addresses patterns generation (PG) and GAMS solves the patterns selection (PS) via a linear programming solver.

   ![Figure 1 – General Architecture](image1)

B. System Architecture
   1) Results in IDE JCreator™

   The classes depicted in Figure 2 above are now expanded using the Java™ IDE JCreator™. The description of the three classes (Archive, Observations, Term), their data, and their methods are shown in Figure 3, 4 & 5, respectively. These figures are self-explanatory. We start by the most general class to the most specific one.

   a) Class “Archive”

   Figure 3 shows the Class “Archive”

   ![Figure 2 – LAD UML® Class Diagram](image2)

   ![Figure 3 The Class “Archive”](image3)

---

2 Java™ is a trademark of Oracle™, Inc.
3 http://www.gams.com
4 http://jcreator.com/
b) The Class “Observation”

The class “Observation” is described in Figure 4.

![Figure 4 - The Class “Observation”](image)

2) Specialized Software - GAMS

a) GAMS models

General-purpose languages, such as Java™, lack some advanced features such as ready-made optimization linear programming (LP) built-in functions. To address this issue, we chose General Algebraic Modeling System (GAMS) and in particular its MIP solver; the choice being motivated by free online availability. GAMS incorporates 12 optimization models, namely LP, MIP, NLP, MCP, MPEC, CNS, DNLP, MINLP, QCP, MIQCP, Stochastic, Global. In addition, GAMS handles 34 solvers. Among these solvers, we need MIP to solve the LP set covering problem arising in LAD.

5 http://archive.ics.uci.edu/ml/datasets/SPECT+Heart

IV. RESULTS AND CONTRIBUTION

A. Total coverage as stopping criterion

The original hybrid algorithm would keep on generating new patterns as long as it is possible even if all of the observations were covered. Total coverage is a mechanism we implemented in Java™ to make sure pattern generation will stop the moment all observations are covered. This allowed total coverage with less patterns.

B. Hybrid algorithm testing (no pattern selection)

In this section, our goal is to test the hybrid algorithm with no use of pattern selection. This means that only Java™ is used. We begin now with the training dataset and generate the prime patterns (both positive and negative). The dataset consists of 19 distinct positive observations and 10 distinct negatives. The original dataset was 20 observations for both but including redundancies.

1) PPPs generation and positive part of discriminant

Figure 6 shows the training dataset used, namely SPECT (Single Proton Emission Computed Tomography) images dataset from University of California at Irvine (UCI), for heart disease classification.

![Figure 6 - Training dataset](image)

Figure 7 shows PPPs and positive part of the discriminant.
C. Effect of using pattern selection

The LAD method suggests that the use of pattern selection leads to an increase of the classification accuracy. Out of the patterns generated through Java™ program, we proceed to select the minimum number of patterns in order to cover all observations irrespective of being positive or negative. This is tackled by solving a set covering problem using GAMS as tested below.

1) Pattern selection

In this step, Java™ exports its output to GAMS as input. Using GAMS, a set covering problem is solved in order to get the minimum number of patterns representing all observations. Figure 9 shows a result of the process for selecting PPPs. The columns (j1 to j19) indicate the observations and the rows (k1 to k45) indicate the patterns.

2) Testing and classification accuracy

For the positive observations (POs), the accuracy before selection is 75% (15/20) and 55% (11/20) after selection. The accuracy deteriorates. The negative

The negative part of the discriminant is obtained in the same manner as for the positive part. In order for a positive (resp. negative) observation to be classified as positive (resp. negative), the determinant must be strictly positive (resp. negative).

3) Testing and classification accuracy

Testing gave correct classification of 75% of positive observations and 5% of negative ones. This gives us an overall average accuracy of 40%.

The negative part of the discriminant is obtained in the same manner as for the positive part. In order for a positive (resp. negative) observation to be classified as positive (resp. negative), the determinant must be strictly positive (resp. negative).
observations (NOs) accuracy, however, changed to the better. It improved from 5% (1/20) to 55% (11/20). The improvement is important. Figure 11 shows POs classification before and after GAMS operation.

![Figure 11 – POS CLASSIFICATION BEFORE AND AFTER GAMS](image)

Figure 12 shows NOs classification before and after GAMS operation.

![Figure 12 – NOS CLASSIFICATION BEFORE AND AFTER GAMS](image)

3) Overall accuracy

TABLE 1 summarizes the accuracy comparison between the two cases; with and without selection.

<table>
<thead>
<tr>
<th></th>
<th>Without Selection</th>
<th>With Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>POs classification accuracy (%)</td>
<td>75</td>
<td>55</td>
</tr>
<tr>
<td>NOs classification accuracy (%)</td>
<td>5</td>
<td>55</td>
</tr>
<tr>
<td>Overall average (%)</td>
<td>40</td>
<td>55</td>
</tr>
<tr>
<td>Total improvement (%)</td>
<td>37.5</td>
<td></td>
</tr>
</tbody>
</table>

The overall accuracy is positively affected by the selection procedure. The overall accuracy increased by 37.5% improving from 40% to 55%.

V. CONCLUSION

It appears from the literature that LAD method is among the most powerful classification and pattern recognition tools with applications in many diversified fields. LAD method can be used to automate the tasks of finding patterns hidden in the data behavior. It’s a good way to convert data into information, and then into knowledge as a prelude to good decision making.

A LAD tool has been described using state-of-the-art software engineering practices and procedures. The LAD method has been implemented and tested on small examples and on a real-life dataset, namely SPECT for heart disease classification. The implementation of prime pattern generation, theory formation and prime patterns selection are the main phases of our work. The basic algorithms for LAD are BOTTOM-UP ALGORITHM, TOP-DOWN ALGORITHM and HYBRID ALGORITHM. These have been implemented in Java™ along with the SELECTION ALGORITHM arising in the set covering problem, implemented in GAMS-MIP. The program encompassed all the LAD phases from data input to the obtainment of the discriminant for at least one real-life example.

The overall accuracy improvement (37.5%) for degree-3-patterns based on SPECT usage suggests that pattern selection methods have to play an important role in LAD-based classification improvement. The improvement of LAD as a computational system will induce important impacts on the future of abstraction, reconstruction and understanding of large real-life datasets.

ACKNOWLEDGMENT

This research was partially supported by King Fahd University of Petroleum and Minerals, KFUPM, Dhahran, Saudi Arabia, as part of the Bachelor of Science Degree in Industrial and Systems Engineering. I am highly thankful to my supervisor Hany Osman who provided valuable expertise that greatly assisted the research, although he may not approve all the interpretations provided in this paper.

REFERENCES


