




# Efficient feature selection for logical analysis of large-scale multi-class datasets

Kedong Yan<sup>1</sup> · Dongjing Miao<sup>2</sup> · Cui Guo<sup>3</sup> · Chanying Huang<sup>1</sup> 

Accepted: 31 March 2021 / Published online: 9 April 2021

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

## Abstract

Feature selection in logical analysis of data (LAD) can be cast into a set covering problem. In this paper, extending the results on feature selection for binary classification using LAD, we present a mathematical model that selects a minimum set of necessary features for multi-class datasets and develop a heuristic algorithm that is both memory and time efficient for this model correspondingly. The utility of the algorithm is illustrated on a small example and the superiority of our work is demonstrated through experiments on 6 real-life multi-class datasets from UCI repository.

**Keywords** Logical Analysis of Data · Supervised Learning · Feature Selection · Multi-classification · Set Covering

## 1 Introduction

LAD is a supervised learning method based on Boolean logic and combinatorial optimization. It was introduced by Hammer (1986) and consummated by Hammer and his co-workers Crama et al. (1988). LAD has wide applications across many research domains, for example, medical science (Alexe et al. (2006, 2004, 2003); Brannon et al.

---

✉ Chanying Huang  
hcy@njust.edu.cn

Kedong Yan  
yan@njust.edu.cn

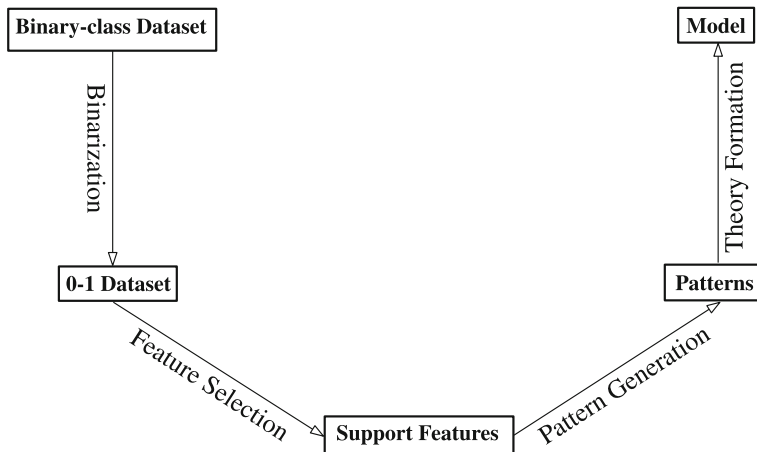
Dongjing Miao  
miaodongjing@hit.edu.cn

Cui Guo  
cguo@stu.edu.cn

<sup>1</sup> School of Computer Science and Engineering, Nanjing University of Science and Technology, 200 Xiaolingwei, Nanjing 210094, China

<sup>2</sup> Faculty of Computing, Harbin Institute of Technology, 92 Xidazhi, Harbin 150001, China

<sup>3</sup> Business School, Shantou University, 243 Daxue Road, Shantou 515063, China



**Fig. 1** Four sequential steps in LAD

(2010); Brauner et al. (2007); Gubskaya et al. (2011); Kim and Ryoo (2008); Kohli et al. (2006); Kronek and Reddy (2008); Yacout et al. (2013)), mechanical engineering (Jocelyn et al. (2017, 2018); Mortada et al. (2011, 2012, 2014); Ragab et al. (2016, 2018); Shaban et al. (2015, 2017a, b)), industrial control system in cyber-physical systems (Das et al. (2020)), to name a few. Generally speaking, for the successful application of LAD on real-life datasets, four stages need to be carried out in succession (Boros et al. (2000)), i.e., binarization, feature selection, pattern generation, and theory formation, as shown in Fig. 1.

The first step, binarization, transforms a dataset that consists of numerical and/or categorical features into a 0-1 dataset where features take only 0-1 values. In this process, each feature in the original dataset is replaced by several 0-1 features. Hence the transformed 0-1 dataset contains much more features after binarization. Like many other feature selection methods in machine learning literature, the second stage in LAD aims at reducing computational difficulty for subsequent processing and improving the performance of the predictor/classifier obtained in the fourth stage. Pattern generation, the key step of LAD, discovers a set of patterns in a separate-an-conquer manner. A pattern is a piece of knowledge formed by a conjunction of 0-1 features or their negations that distinguishes at least one example of one type of data from all of the other type. The final stage, theory formation, builds a model via pattern ensemble using weighted linear combination.

Feature selection is an important procedure in data analysis. It has wide application in bioinformatics (see, for example, Cai et al. (2006); Cai et al. (2007); Yang et al. (2006).) In LAD, feature selection is an NP-hard problem, like other data related problems (Miao et al. (2018); Cai et al. (2019); Miao et al. (2020b, a).) Feature selection in LAD is usually formulated into the well-known minimum set covering (SC) problem (see, e.g., Boros et al. (2000); Alexe et al. (2007).) Unlike its subsequent stage, pattern generation, which draws a lot of research efforts (Alexe and Hammer (2006a, b); Bonates et al. (2008); Ryoo and Jang (2009); Guo and Ryoo (2012); Yan and Ryoo

(2017a, b, 2019b, a)), the literature specifically addressing feature selection problem in LAD is relatively scarce (Bruni (2007); Ryoo and Jang (2007).) To be specific, Bruni (2007) reformulated feature selection problem from SC into a weighted SC problem, and Ryoo and Jang (2007) designed a memory and time efficient procedure called implicit SC-based feature selection for large-scale datasets. Both of these results are aimed at binary classification problems. With the increasing application of LAD on various domains, more and more real-life datasets at hand contain more than two classes of data.

For the problem of logical analysis of a multi-class dataset, the common practice is decomposing it into several binary classification problems (Avila-Herrera and Subasi (2015); Kim and Choi (2015); Bain et al. (2020).) Two most frequently used approaches that transform multi-classification problem into binary classification problems are ‘one versus one’ (OvO) and ‘one versus rest’ (OvR), respectively (see, e.g., Galar et al. (2011).) As the name indicates, OvO produces a binary classification problem for each pair of classes, and OvR generates a binary classification problem for one class and the rest of classes. Thus for a multi-class dataset with  $\ell$  ( $\ell \geq 3$ ) classes, OvO gives  $\frac{\ell(\ell-1)}{2}$  binary classification problems, whereas OvR yields  $\ell$  binary classification problems. LAD usually adopts OvR approach. Ryoo and Jang (2007) tested their implicit SC-based feature selection procedure on multi-class datasets using this method. There is a drawback in using OvR for feature selection in LAD. When the features selected from each binary classification problem are merged, there may exist many redundant features since each binary classification problem considers feature selection problem locally without interaction with each other. This can lead to a very complex model in the theory formation step. To overcome this drawback, we consider feature selection for multi-classification problem from a global point of view in this paper. As will be detailed later, we formulate a mathematical model for this problem and generalize the procedure developed by Ryoo and Jang (2007) to select features for multi-class dataset in one shot.

As for the organization of the paper, Section 2 gives a brief introduction of binarization and feature selection in LAD. The implicit SC-based feature selection procedure is also summarized in this section for self-containedness. Section 3 deals with multi-class datasets, it gives a mathematical formulation for the feature selection problem and devises an algorithm for the model. Section 4 illustrates the process of feature selection procedure in Section 3 step by step via a small multi-class dataset. Section 5 tests the efficacy and efficiency through experiments with 6 large-scale multi-class datasets from Lichman (2013). Finally, Section 6 concludes this paper.

## 2 Preliminaries and overview of implicit SC-based feature selection for binary classification

In this section, we briefly describe the first two steps of LAD and give an overview of implicit SC-based feature selection presented in Ryoo and Jang (2007) for binary classification problems to make this paper self-contained.

## 2.1 Binarization

A typical dataset  $S$  in LAD consists of two subsets. One subset is labeled as positive  $S^+$ , and the other negative  $S^-$ . For real-life datasets, most of them consist of nominal (e.g., color, shape, weather, etc.) and/or numerical features. In view of characteristics of LAD, Boros et al. (1997) handles such kind of dataset by transforming it into 0-1 equivalent such that the resulting dataset is contradiction free, i.e.,  $S^+ \cap S^- = \emptyset$ . That is to say, there should not exist a pair of + and – observations that have the same 0-1 representation. A binarization like this is called *proper*, which works like the following.

For a nominal feature  $\mathcal{F}$ , it is straightforward to convert it into  $K$  0-1 features  $a_1, \dots, a_K$  if it has  $K$  distinct values  $\mu_1, \dots, \mu_K$ . That is, for  $k = 1, \dots, K$ , set

$$a_k = \begin{cases} 1, & \text{if } \mathcal{F} = \mu_k, \\ 0, & \text{otherwise.} \end{cases}$$

With regard to a numerical feature  $\mathcal{F}$ , one needs to sort its values in increasing order as  $\mu^{(1)} < \mu^{(2)} < \dots < \mu^{(K)}$ , where  $K$  denotes the number of distinct values in feature  $\mathcal{F}$ . In this ascending sequence, if two adjacent numbers belong to different classes, a cutpoint, taking the middle value of the two numbers, is introduced. Such a cutpoint is called *essential*, and it is sufficient to use only essential cutpoints to binarize a dataset. For a numerical feature  $\mathcal{F}$ , suppose  $K'$  ( $1 \leq K' \leq K - 1$ ) essential cutpoints  $\alpha_1, \dots, \alpha_{K'}$  are needed, then  $\mathcal{F}$  is transformed into  $K'$  0-1 features  $a_1, \dots, a_{K'}$  with

$$a_k = \begin{cases} 1, & \text{if } \mathcal{F} \geq \alpha_k, \\ 0, & \text{otherwise.} \end{cases}$$

for  $k = 1, \dots, K'$ .

## 2.2 Feature selection

After the binarization step, a dataset  $S$  usually has a huge number of 0-1 features, which contains redundant information and hampers the efficient generation of patterns in next stage. A minimum and necessary set of features called *minimum support set* need to be identified, where ‘necessary’ means the resulting dataset is contradiction free after the selection process. To this end, suppose there are  $n$  0-1 features in  $S$ , denoted by  $a_j, j \in N := \{1, \dots, n\}$ . For an example  $A_i, i \in S$ , let  $A_{ij}$  denote the value of its  $j$ -th feature. To formulate a mathematical programming model for feature selection, the following indicator variables are naturally due for  $j \in N$ , that is,

$$x_j = \begin{cases} 1, & \text{if feature } a_j \text{ is selected,} \\ 0, & \text{otherwise.} \end{cases}$$

Feature selection requires each pair of examples from different classes stay different with respect to the selected features. For two 0-1 vectors  $\mathbf{b}_1$  and  $\mathbf{b}_2$ , their difference is usually measured by Hamming distance, i.e., the number of 1's in the fingerprint  $\mathbf{b}_1 \oplus \mathbf{b}_2$ , where ' $\oplus$ ' denotes exclusive-OR operator. Based on the discussion above, the minimum and necessary set of 0-1 features can be found by solving the following problem:

$$(FS)_b \left\{ \begin{array}{l} \min_{\mathbf{x} \in \{0,1\}^n} \sum_{j \in N} x_j \\ \text{s.t.} \quad \sum_{j \in N} (A_{ij} \oplus A_{kj}) x_j \geq 1, i \in S^+, k \in S^- \end{array} \right.$$

It is easy to see that  $(FS)_b$  is an SC problem. There exist a number of heuristic algorithms (Chvatal (1979); Balas and Carrera (1996); Ceria et al. (1998); Fisher and Kedia (1990); Caprara et al. (1999)) for the solution of such problem. Among these algorithms, the greedy heuristic by Chvatal (1979) is the most famous one and becomes the standard procedure for feature selection in LAD (Boros et al. (2000).) For application of the greedy heuristic, an SC matrix with  $|S^+| \times |S^-|$  rows needs to be explicitly built. Let  $M$  represent the row index set of the SC matrix, i.e.,  $M := S^+ \times S^-$ , and  $I_j \subseteq M$  denote the index set of rows that are covered by feature  $a_j$  for  $j \in N$ . In addition, denote  $N_t \subseteq N$  as the feature index set selected by the greedy heuristic and  $M_u \subseteq M$  as the uncovered row index by  $N_t$ . Then the unselected feature index set  $N_u = N \setminus N_t$ . With these notation, the greedy heuristic henceforth referred as standard FS is given in Algorithm 1.

---

**Algorithm 1** standard FS
 

---

input: contradiction and duplicate free 0-1 data  $A_i, i \in S = S^+ \cup S^-$ .

output: an index set of features  $N_t \subseteq N$ .

1: set  $M_u = M, N_t = \emptyset, N_u = N$ .

2: **while**  $M_u \neq \emptyset$  **do**

3: set  $f \leftarrow \operatorname{argmin} \left\{ \frac{1}{|M_u \cap I_j|} \mid j \in N_u \right\}$ . Break ties by selecting the feature with lowest index.

4: set  $N_t \leftarrow N_t \cup \{f\}, N_u \leftarrow N_u \setminus \{f\}$ .

5: set  $M_u \leftarrow M_u \setminus I_f$ .

6: **end while**

---

When  $S$  is large, the SC matrix can be very huge, which occupies so much memory that the dataset under consideration can not be analyzed. The equivalent feature selection procedure proposed in Ryoo and Jang (2007) requires much less memory and computational time than `standard FS` by circumventing the explicit construction of the SC matrix. For this purpose, a term is defined as  $t := \bigwedge_{j \in N_t} l_j$  for  $N_t \subseteq N$ , where  $l_j = a_j$  or its negation  $l_j = \bar{a}_j$ . In addition, for a term  $t$ , let

$$C^\bullet(t) := \left\{ i \in S^\bullet \mid t(A_i) = \prod_{\substack{l_j=a_j, \\ j \in N_t}} \prod_{\substack{l_j=\bar{a}_j, \\ j \in N_t}} \bar{A}_{ij} = 1 \right\}$$

for  $\bullet \in \{+, -\}$ . Then for  $j \in N$ ,

$$I_j = (C^+(a_j) \times C^-(\bar{a}_j)) \cup (C^+(\bar{a}_j) \times C^-(a_j))$$

Furthermore, let

$$T := \left\{ t = \bigwedge_{j \in N_t} l_j \mid C^+(t) \neq \emptyset, C^-(t) \neq \emptyset \right\} \quad (1)$$

Then

$$M_u = \bigcup_{t \in T} (C^+(t) \times C^-(t)),$$

and for  $j \in N_u$ ,

$$|M_u \cap I_j| = \sum_{t \in T} (|C^+(t \wedge a_j)| \times |C^-(t \wedge \bar{a}_j)| + |C^+(t \wedge \bar{a}_j)| \times |C^-(t \wedge a_j)|). \quad (2)$$

Using the results above, Ryoo and Jang (2007) devised a memory efficient heuristic procedure in Algorithm 2 henceforth dubbed as `implicit FS` for feature selection. When testing on real-life binary classification problems, `implicit FS` are much more efficient than `standard FS`.

### 3 Implicit SC-based feature selection for multi-classification

In this section, we are concerned with a multi-class 0-1 dataset  $S$  that contains  $\ell$  ( $\ell \geq 3$ ) types of examples. Let  $L := \{1, \dots, \ell\}$  and denote the set of examples of type  $p \in L$  by  $S^p$ . As the binary class case,  $S$  is a duplicate and contradiction free dataset with

$$S = \bigcup_{p \in L} S^p$$

**Algorithm 2** implicit FSB

input: contradiction and duplicate free 0-1 data  $A_i, i \in S = S^+ \cup S^-$ .

output: an index set of features  $N_t \subseteq N$ .

- 1: get  $C^\bullet(a_j), C^\bullet(\bar{a}_j)$  for  $\bullet \in \{+, -\}, j \in N$ .
- 2: set  $f \leftarrow \operatorname{argmax}\{|I_j| \mid j \in N\}$ , break ties by lowest index first rule.
- 3: set  $N_t \leftarrow \{f\}$  and obtain  $T$  via (1).
- 4:  $N_u = N \setminus N_t$ .
- 5: **while**  $T \neq \emptyset$  **do**
- 6: set  $f \leftarrow \operatorname{argmin}\left\{\frac{1}{|M_u \cap I_j|} \mid j \in N_u\right\}$ , where  $|M_u \cap I_j|$  is calculated via (2).  
Break ties by lowest index first rule.
- 7: set  $N_t \leftarrow N_t \cup \{f\}, N_u \leftarrow N_u \setminus \{f\}$ .
- 8: obtain  $T$  via (1).
- 9: **end while**

and  $S^p \cap S^q = \emptyset$  for  $p, q \in L, p \neq q$ .

Although  $S$  is a multi-class dataset, the rationale behind feature selection does not change. The selected minimum support set should separate examples in one type from those in other types. In other words, each example from one class should have at least Hamming distance 1 from every example in the remaining classes. By extending the formulation of  $(FS)_b$ , we obtain the following mathematical model that selects a minimum support set for logical analysis of multi-class dataset:

$$(FS)_m \left\{ \begin{array}{l} \min_{\mathbf{x} \in \{0,1\}^n} \sum_{j \in N} x_j \\ \text{s.t.} \sum_{j \in N} (A_{ij} \oplus A_{kj}) x_j \geq 1, \quad i \in S^p, k \in S^q, p, q \in L, p < q \end{array} \right.$$

The following result is immediate.

**Theorem 1** Let  $\mathbf{x}^*$  be an optimal feasible solution to  $(FS)_m$ , then  $F := \{j \in N \mid x_j^* = 1\}$  is a minimum support set.

**Proof** Suppose  $\mathbf{x}$  is a feasible solution to  $(FS)_m$ . For each pair of examples from different classes, for example,  $A_i, i \in S^p, A_k, k \in S^q, p \neq q$ , the constraint guarantees

that they are at least Hamming distance 1 apart from each other after feature selection. In addition, the objective function ensures the optimal solution  $\mathbf{x}^*$  gives the minimum support set.  $\square$

A few remarks are due here.

**Remark 1** Model  $(FS)_m$  reduces to  $(FS)_b$  when  $\ell = 2$ .

**Remark 2** In model  $(FS)_m$ , the restriction  $p < q$  for  $p, q \in L$  avoids the introduction of redundant rows in the constraint matrix.

**Remark 3** If `standard fs` is adopted for the solution of  $(FS)_m$ , one needs to explicitly construct an SC matrix with row index set

$$M = \bigcup_{\substack{p, q \in L \\ p < q}} (S^p \times S^q).$$

Thus the number of rows in the matrix is

$$|M| = \sum_{\substack{p, q \in L \\ p < q}} (|S^p| \times |S^q|).$$

For readers' smooth transition from material in Ryoo and Jang (2007), we inherit the notation therein and make changes if necessary.

As regards the term  $t$  defined in Section 2, let

$$C^p(t) := \left\{ i \in S^p \mid t(A_i) = \prod_{\substack{l_j = a_j, \\ j \in N_t}} A_{ij} \prod_{\substack{l_j = \bar{a}_j, \\ j \in N_t}} \bar{A}_{ij} = 1 \right\}$$

for  $p \in L$ .

**Proposition 1** For  $j \in N$ , the row index set  $I_j \subset M$  that feature  $a_j, j \in N$  covers is given by

$$I_j = \bigcup_{\substack{p, q \in L \\ p < q}} \{(C^p(a_j) \times C^q(\bar{a}_j)) \cup (C^p(\bar{a}_j) \times C^q(a_j))\}. \quad (3)$$

Given  $N_t \subseteq N$ , one needs to determine the rows with indices in  $M$  that are covered or not covered by the set of features with indices in  $N_t$ . Note that a row in  $M$  is not covered if the corresponding pair of examples  $A_i, i \in S^p, A_k, k \in S^q$  that produces it satisfy  $t(A_i) = 1, t(A_k) = 1$ , which means  $A_i$  and  $A_k$  take the same value with respect to the feature index set  $N_t$ .

**Proposition 2** For  $N_t \subseteq N$  and  $p, q \in L, p \neq q$ , the row index set  $M_u^{p,q}(t) \subset M$  not covered by features with indices in  $N_t$  is given by

$$M_u^{p,q}(t) = C^p(t) \times C^q(t), \quad (4)$$



where  $t = \bigwedge_{j \in N_t} l_j$ .

**Lemma 1** For  $N_t \subseteq N, N_t \neq \emptyset$ , let  $T$  the set of all possible terms that can be constructed from  $N_t$  with form  $t = \bigwedge_{j \in N_t} l_j$ . Then the row index set  $M_u \subset M$  that is not covered by the set of features in  $N_t$  is given by

$$M_u = \bigcup_{\substack{p, q \in L \\ p < q}} \bigcup_{t \in T} M_u^{p, q}(t).$$

Let

$$T^{p, q} := \left\{ t = \bigwedge_{j \in N_t} l_j \mid C^p(t) \neq \emptyset, C^q(t) \neq \emptyset \right\}$$

for  $p, q \in L, p < q$ , and redefine  $T$  as

$$T = \bigcup_{\substack{p, q \in L \\ p < q}} T^{p, q}. \tag{5}$$

Then  $M_u$  can be computed in the following way.

**Theorem 2** For  $N_t \subseteq N, N_t \neq \emptyset$ , define  $T$  via (5). Then the row index set  $M_u \subset M$  that is not covered by the set of features in  $N_t$  is given by

$$M_u = \bigcup_{\substack{p, q \in L \\ p < q}} \bigcup_{t \in T} (C^p(t) \times C^q(t)). \tag{6}$$

**Proof** Note that in (4), if either  $C^p(t) = \emptyset$  or  $C^q(t) = \emptyset$  for  $p, q \in L, p < q$ , then  $M_u^{p, q}(t) = \emptyset$ . For such a term  $t$ , it can be excluded in the calculation of  $M_u$ . In other words, we only need to consider terms in  $T$  defined via (5). This completes the proof. □

In each iteration of the feature selection, the candidate features need to be evaluated to select a feature index  $j$  which yields the largest  $|M_u \cap I_j|$  among all features with indices in  $N_u$ . With the notation introduced above, we first show how to compute  $M_u \cap I_j$  below.

**Theorem 3** For  $N_t \subset N, N_t \neq \emptyset$ , the row index set that is not covered by the set of features with indices in  $N_t$  but covered by features with indices in  $N_t \cup \{j\}, j \in N_u$  is given by

$$M_u \cap I_j = \bigcup_{\substack{p, q \in L \\ p < q}} \bigcup_{t \in T} \{ (C^p(t \wedge a_j) \times C^q(t \wedge \bar{a}_j)) \cup (C^p(t \wedge \bar{a}_j) \times C^q(t \wedge a_j)) \} \tag{7}$$

**Proof** For simplicity and convenience in proof, let

$$\begin{aligned} M_u^{p,q} &= \bigcup_{t \in T} (C^p(t) \times C^q(t)) \quad \text{and} \quad I_j^{p,q} \\ &= (C^p(a_j) \times C^q(\bar{a}_j)) \cup (C^p(\bar{a}_j) \times C^q(a_j)). \end{aligned}$$

Then

$$M_u \cap I_j = \left( \bigcup_{\substack{p,q \in L \\ p < q}} M_u^{p,q} \right) \cap \left( \bigcup_{\substack{p,q \in L \\ p < q}} I_j^{p,q} \right).$$

First, note that for a pair of classes, the row index set constructed from examples of them has empty intersection with row index set from a different pair of classes. In other words, for  $p, p', q, q' \in L$  with  $p < q$  and  $p' < q'$ , if  $p \neq p'$  or  $q \neq q'$ , then  $M_u^{p,q} \cap I_j^{p',q'} = \emptyset$ . Therefore,

$$\begin{aligned} M_u \cap I_j &= \bigcup_{\substack{p,q \in L \\ p < q}} (M_u^{p,q} \cap I_j^{p,q}) \\ &= \bigcup_{\substack{p,q \in L \\ p < q}} \left( \bigcup_{t \in T} (C^p(t) \times C^q(t)) \right) \cap I_j^{p,q} \\ &= \bigcup_{\substack{p,q \in L \\ p < q}} \bigcup_{t \in T} ((C^p(t) \times C^q(t)) \cap I_j^{p,q}). \end{aligned}$$

Now it is essential to deal with  $(C^p(t) \times C^q(t)) \cap I_j^{p,q}$ , which is detailed as follows.

$$\begin{aligned} &(C^p(t) \times C^q(t)) \cap I_j^{p,q} \\ &= (C^p(t) \times C^q(t)) \cap \{(C^p(a_j) \times C^q(\bar{a}_j)) \cup (C^p(\bar{a}_j) \times C^q(a_j))\} \\ &= \{(C^p(t) \times C^q(t)) \cap (C^p(a_j) \times C^q(\bar{a}_j))\} \cup \{(C^p(t) \times C^q(t)) \\ &\quad \cap (C^p(\bar{a}_j) \times C^q(a_j))\} \\ &= \{(C^p(t) \cap C^p(a_j)) \times (C^q(t) \cap C^q(\bar{a}_j))\} \\ &\quad \cup \{(C^p(t) \cap C^p(\bar{a}_j)) \times (C^q(t) \cap C^q(a_j))\} \\ &= (C^p(t \wedge a_j) \times C^q(t \wedge \bar{a}_j)) \cup (C^p(t \wedge \bar{a}_j) \times C^q(t \wedge a_j)) \end{aligned}$$

This completes the proof.  $\square$

Theorem 3 naturally leads to the following result.

**Corollary 1** For  $N_t \subset N$ ,  $N_t \neq \emptyset$ , the number of rows that are not covered by the set of features with indices in  $N_t$  but covered by features in  $N_t \cup \{j\}$ ,  $j \in N_u$  is given by

$$|M_u \cap I_j| = \sum_{\substack{p,q \in L \\ p < q}} \sum_{t \in T} (|C^p(t \wedge a_j)| \times |C^q(t \wedge \bar{a}_j)| + |C^p(t \wedge \bar{a}_j)| \times |C^q(t \wedge a_j)|) \tag{8}$$

With the material presented above, now we are ready to introduce the procedure intended to select support features for large-scale multi-class datasets.

---

**Algorithm 3** *implicit FSM*

---

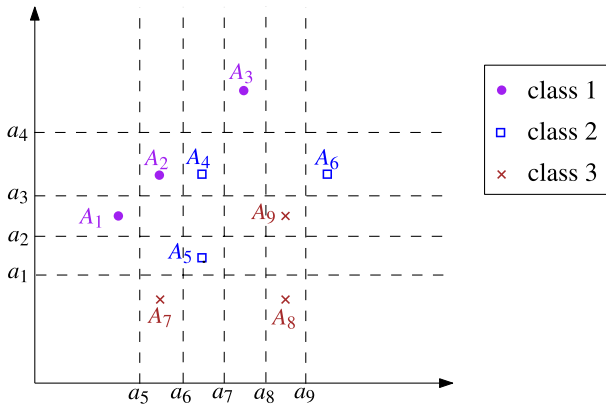
input: contradiction and duplicate free 0-1 data  $A_i, i \in S = \bigcup_{p \in L} S^p$ .

output: an index set of features  $N_t \subseteq N$ .

- 1: get  $C^p(a_j), C^p(\bar{a}_j)$  for  $p \in L, j \in N$ .
  - 2: set  $f \leftarrow \operatorname{argmax}\{|I_j| \mid j \in N\}$ , break ties by lowest index first rule.
  - 3: set  $N_t \leftarrow \{f\}$  and obtain  $T$  via (5).
  - 4:  $N_u = N \setminus N_t$ .
  - 5: **while**  $T \neq \emptyset$  **do**
  - 6:   set  $f \leftarrow \operatorname{argmin}\left\{\frac{1}{|M_u \cap I_j|} \mid j \in N_u\right\}$ , where  $|M_u \cap I_j|$  is calculated via (8).  
       Break ties by lowest index first rule.
  - 7:   set  $N_t \leftarrow N_t \cup \{f\}, N_u \leftarrow N_u \setminus \{f\}$ .
  - 8:   obtain  $T$  via (5).
  - 9: **end while**
- 

*implicit FSM* is equivalent to standard FS in that they both produce the same heuristic solution to  $(FS)_m$ , as stated in the theorem below.

**Theorem 4** *implicit FSM terminates and selects the same set of support features as standard FS.*



**Fig. 2** Illustrative dataset. (Each dashed line denotes a cutpoint, which corresponds to a binary feature. For a binary feature, an example takes value 0 if its value is less than the cutpoint associated to that feature)

**Table 1** 0-1 representation of illustrative dataset

Class	<i>i</i>	<i>a</i> <sub>1</sub>	<i>a</i> <sub>2</sub>	<i>a</i> <sub>3</sub>	<i>a</i> <sub>4</sub>	<i>a</i> <sub>5</sub>	<i>a</i> <sub>6</sub>	<i>a</i> <sub>7</sub>	<i>a</i> <sub>8</sub>	<i>a</i> <sub>9</sub>
1	1	1	1	0	0	0	0	0	0	0
	2	1	1	1	0	1	0	0	0	0
	3	1	1	1	1	1	1	1	0	0
2	4	1	1	1	0	1	1	0	0	0
	5	1	0	0	0	1	1	0	0	0
	6	1	1	1	0	1	1	1	1	1
3	7	0	0	0	0	1	0	0	0	0
	8	0	0	0	0	1	1	1	1	0
	9	1	1	0	0	1	1	1	1	0

**Proof** In each iteration, both of the algorithms examine the set of features  $a_j, j \in N_u$  in the same order and evaluate the same quantity  $\frac{1}{|M_u \cap I_j|}, j \in N_u$  for selecting next feature. The difference only lies in the way  $\frac{1}{|M_u \cap I_j|}, j \in N_u$  is calculated. For implicit FSM, it obtains the value via set operation, whereas standard FS counts the number of 1's among the SC matrix. Since the dataset under consideration is conflict free,  $(FS)_m$  is feasible and both algorithms terminate. This completes the proof. □

### 4 Illustrative example

In this section, we showcase `implicit` FSM via a small dataset. Suppose we have a dataset that consists of 3 classes of examples, and each class contains 3 examples, as shown in Fig. 2. For binarization of the dataset, 9 cutpoints (dashed lines in Fig. 2) need to be introduced, which correspond to 9 binary features  $a_1, \dots, a_9$ . The equivalent 0-1 representation of the dataset is given in Table 1.

For application of `standard` FS on this dataset, an SC matrix needs to be explicitly built as following:

$$\begin{aligned}
 (A_1 \oplus A_4) &\rightarrow (001011000) \\
 (A_1 \oplus A_5) &\rightarrow (010011000) \\
 (A_1 \oplus A_6) &\rightarrow (001011111) \\
 (A_2 \oplus A_4) &\rightarrow (000001000) \\
 (A_2 \oplus A_5) &\rightarrow (011001000) \\
 (A_2 \oplus A_6) &\rightarrow (000001111) \\
 (A_3 \oplus A_4) &\rightarrow (000100100) \\
 (A_3 \oplus A_5) &\rightarrow (011100100) \\
 (A_3 \oplus A_6) &\rightarrow (000100011) \\
 (A_1 \oplus A_7) &\rightarrow (110010000) \\
 (A_1 \oplus A_8) &\rightarrow (110011110) \\
 (A_1 \oplus A_9) &\rightarrow (000011110) \\
 (A_2 \oplus A_7) &\rightarrow (111000000) \\
 (A_2 \oplus A_8) &\rightarrow (111001110) \\
 (A_2 \oplus A_9) &\rightarrow (001001110) \\
 (A_3 \oplus A_7) &\rightarrow (111101100) \\
 (A_3 \oplus A_8) &\rightarrow (111100010) \\
 (A_3 \oplus A_9) &\rightarrow (001100010) \\
 (A_4 \oplus A_7) &\rightarrow (111001000) \\
 (A_4 \oplus A_8) &\rightarrow (111000110) \\
 (A_4 \oplus A_9) &\rightarrow (001000110) \\
 (A_5 \oplus A_7) &\rightarrow (100001000) \\
 (A_5 \oplus A_8) &\rightarrow (100000110) \\
 (A_5 \oplus A_9) &\rightarrow (010000110) \\
 (A_6 \oplus A_7) &\rightarrow (111001111) \\
 (A_6 \oplus A_8) &\rightarrow (111000001) \\
 (A_6 \oplus A_9) &\rightarrow (001000001)
 \end{aligned}$$

which contains  $3 \times (3 + 3) + 3 \times 3 = 27$  rows and 9 columns. For this problem, the reader can check that `standard` FS successively selects feature index in order  $3 \rightarrow 6 \rightarrow 7 \rightarrow 1 \rightarrow 4$  by breaking ties using lowest index first rule. Thus  $N_t = \{1, 3, 4, 6, 7\}$ .

**Initialization.** Following the steps in implicit FSM, we first get  $C^p(a_j), C^p(\bar{a}_j)$  for  $p \in \{1, 2, 3\}$  and  $j \in \{1, \dots, 9\}$  below:

$p$	1	2	3
$C^p(a_1)$	{1, 2, 3}	{4, 5, 6}	{9}
$C^p(\bar{a}_1)$	$\emptyset$	$\emptyset$	{7, 8}
$C^p(a_2)$	{1, 2, 3}	{4, 6}	{9}
$C^p(\bar{a}_2)$	$\emptyset$	{5}	{7, 8}
$C^p(a_3)$	{2, 3}	{4, 6}	$\emptyset$
$C^p(\bar{a}_3)$	{1}	{5}	{7, 8, 9}
$C^p(a_4)$	{3}	$\emptyset$	$\emptyset$
$C^p(\bar{a}_4)$	{1, 2}	{4, 5, 6}	{7, 8, 9}
$C^p(a_5)$	{2, 3}	{4, 5, 6}	{7, 8, 9}
$C^p(\bar{a}_5)$	{1}	$\emptyset$	$\emptyset$
$C^p(a_6)$	{3}	{4, 5, 6}	{8, 9}
$C^p(\bar{a}_6)$	{1, 2}	$\emptyset$	{7}
$C^p(a_7)$	{3}	{6}	{8, 9}
$C^p(\bar{a}_7)$	{1, 2}	{4, 5}	{7}
$C^p(a_8)$	$\emptyset$	{6}	{8, 9}
$C^p(\bar{a}_8)$	{1, 2, 3}	{4, 5}	{7}
$C^p(a_9)$	$\emptyset$	{6}	$\emptyset$
$C^p(\bar{a}_9)$	{1, 2, 3}	{4, 5}	{7, 8, 9}

We take  $j = 1$  as an example to show how to calculate  $|I_1|$ . Specifically,  $|I_1| = |C^1(a_1)| \times |C^2(\bar{a}_1)| + |C^1(\bar{a}_1)| \times |C^2(a_1)| + |C^1(a_1)| \times |C^3(\bar{a}_1)| + |C^1(\bar{a}_1)| \times |C^3(a_1)| + |C^2(a_1)| \times |C^3(\bar{a}_1)| + |C^2(\bar{a}_1)| \times |C^3(a_1)| = 3 \times 0 + 0 \times 3 + 3 \times 2 + 0 \times 1 + 3 \times 2 + 0 \times 1 = 12$ . The remaining  $|I_j|, j \in \{2, \dots, 9\}$  can be calculated in the same way. For readers' easy reference, we list them below:

$$\begin{aligned}
 |I_2| &= 3 \times 1 + 0 \times 2 + 3 \times 2 + 0 \times 1 + 2 \times 2 + 1 \times 1 = 14 \\
 |I_3| &= 2 \times 1 + 1 \times 2 + 2 \times 3 + 1 \times 0 + 2 \times 3 + 1 \times 0 = 16 \\
 |I_4| &= 1 \times 3 + 2 \times 0 + 1 \times 3 + 2 \times 0 + 0 \times 3 + 3 \times 0 = 6 \\
 |I_5| &= 2 \times 0 + 1 \times 3 + 2 \times 0 + 1 \times 3 + 3 \times 0 + 0 \times 3 = 6 \\
 |I_6| &= 1 \times 0 + 2 \times 3 + 1 \times 1 + 2 \times 2 + 3 \times 1 + 0 \times 2 = 14 \\
 |I_7| &= 1 \times 2 + 2 \times 1 + 1 \times 1 + 2 \times 2 + 1 \times 1 + 2 \times 2 = 14 \\
 |I_8| &= 0 \times 2 + 3 \times 1 + 0 \times 1 + 3 \times 2 + 1 \times 1 + 2 \times 2 = 14 \\
 |I_9| &= 0 \times 2 + 3 \times 1 + 0 \times 3 + 3 \times 0 + 1 \times 3 + 2 \times 0 = 6
 \end{aligned}$$

With  $|I_3|$  taking the largest value, feature index 3 is selected. Thus  $N_t = \{3\}, T = \{t_1 = a_3, t_2 = \bar{a}_3\}, N_u = \{1, 2, 4, 5, 6, 7, 8, 9\}$ .

**Iteration 1.** We gather  $C^p(t_1)$  and  $C^p(t_2)$  for  $p \in \{1, 2, 3\}$  below:

$p$	1	2	3
$C^p(t_1)$	{2, 3}	{4, 6}	$\emptyset$
$C^p(t_2)$	{1}	{5}	{7, 8, 9}

Based on the preceding information, the sets for calculating  $|M_u \cap I_j|$  are put in the following table for reference.

$p$	1	2	3
$C^p(t_1 \wedge a_1)$	{2, 3}	{4, 6}	$\emptyset$
$C^p(t_1 \wedge \bar{a}_1)$	$\emptyset$	$\emptyset$	$\emptyset$
$C^p(t_2 \wedge a_1)$	{1}	{5}	{9}
$C^p(t_2 \wedge \bar{a}_1)$	$\emptyset$	$\emptyset$	{7, 8}

Thus  $|M_u \cap I_1| = |C^1(t_1 \wedge a_1)| \times |C^2(t_1 \wedge \bar{a}_1)| + |C^1(t_1 \wedge a_1)| \times |C^3(t_1 \wedge \bar{a}_1)| + |C^2(t_1 \wedge a_1)| \times |C^3(t_1 \wedge \bar{a}_1)| + |C^1(t_1 \wedge \bar{a}_1)| \times |C^2(t_1 \wedge a_1)| + |C^1(t_1 \wedge \bar{a}_1)| \times |C^3(t_1 \wedge a_1)| + |C^2(t_1 \wedge \bar{a}_1)| \times |C^3(t_1 \wedge a_1)| + |C^1(t_2 \wedge a_1)| \times |C^2(t_2 \wedge \bar{a}_1)| + |C^1(t_2 \wedge a_1)| \times |C^3(t_2 \wedge \bar{a}_1)| + |C^2(t_2 \wedge a_1)| \times |C^3(t_2 \wedge \bar{a}_1)| + |C^1(t_2 \wedge \bar{a}_1)| \times |C^2(t_2 \wedge a_1)| + |C^1(t_2 \wedge \bar{a}_1)| \times |C^3(t_2 \wedge a_1)| + |C^2(t_2 \wedge \bar{a}_1)| \times |C^3(t_2 \wedge a_1)| = 2 \times 0 + 2 \times 0 + 2 \times 0 + 0 \times 2 + 0 \times 0 + 0 \times 0 + 1 \times 0 + 1 \times 2 + 1 \times 2 + 0 \times 1 + 0 \times 1 + 0 \times 1 = 4$ .  $|M_u \cap I_j|$  for  $j \in \{2, 4, 5, 6, 7, 8, 9\}$  can be obtained similarly. For reason of space, we omit the specifics and just give the numbers below:

$$\begin{aligned}
 |M_u \cap I_2| &= 4 \\
 |M_u \cap I_4| &= 2 \\
 |M_u \cap I_5| &= 4 \\
 |M_u \cap I_6| &= 6 \\
 |M_u \cap I_7| &= 6 \\
 |M_u \cap I_8| &= 6 \\
 |M_u \cap I_9| &= 2
 \end{aligned}$$

Feature index 6 is the first lowest index with largest value, so it is selected in this iteration. Therefore,  $N_t = \{3, 6\}$ ,  $T = \{t_1 = a_3a_6, t_2 = \bar{a}_3\bar{a}_6, t_3 = \bar{a}_3a_6\}$ ,  $N_u = \{1, 2, 4, 5, 7, 8, 9\}$ .

**Iteration 2.** Referring the sets  $C^p(t_1)$ ,  $C^p(t_2)$ ,  $C^p(t_3)$  for  $p \in \{1, 2, 3\}$ ,

$p$	1	2	3
$C^p(t_1)$	{3}	{4, 6}	$\emptyset$
$C^p(t_2)$	{1}	$\emptyset$	{7}
$C^p(t_3)$	$\emptyset$	{5}	{8, 9}

one obtains  $|M_u \cap I_j|$ ,  $j \in \{1, 2, 4, 5, 7, 8, 9\}$  below:

$$|M_u \cap I_1| = 2$$

$$\begin{aligned}
 |M_u \cap I_2| &= 2 \\
 |M_u \cap I_4| &= 2 \\
 |M_u \cap I_5| &= 1 \\
 |M_u \cap I_7| &= 3 \\
 |M_u \cap I_8| &= 3 \\
 |M_u \cap I_9| &= 1
 \end{aligned}$$

Thus feature index 7 is selected. We have  $N_t = \{3, 6, 7\}$ ,  $T = \{t_1 = a_3a_6a_7, t_2 = \bar{a}_3\bar{a}_6\bar{a}_7\}$ ,  $N_u = \{1, 2, 4, 5, 8, 9\}$ .

**Iteration 3.** On the basis of  $C^p(t_1)$ ,  $C^p(t_2)$  for  $p \in \{1, 2, 3\}$  below

$p$	1	2	3
$C^p(t_1)$	{3}	{6}	$\emptyset$
$C^p(t_2)$	{1}	$\emptyset$	{7}

$|M_u \cap I_j|$ ,  $j \in \{1, 2, 4, 5, 8, 9\}$  take value

$$\begin{aligned}
 |M_u \cap I_1| &= 1 \\
 |M_u \cap I_2| &= 1 \\
 |M_u \cap I_4| &= 1 \\
 |M_u \cap I_5| &= 1 \\
 |M_u \cap I_8| &= 1 \\
 |M_u \cap I_9| &= 1
 \end{aligned}$$

respectively. By lowest index first rule, feature index 1 is selected. As a result,  $N_t = \{1, 3, 6, 7\}$ ,  $T = \{t_1 = a_1a_3a_6a_7\}$ ,  $N_u = \{2, 4, 5, 8, 9\}$ .

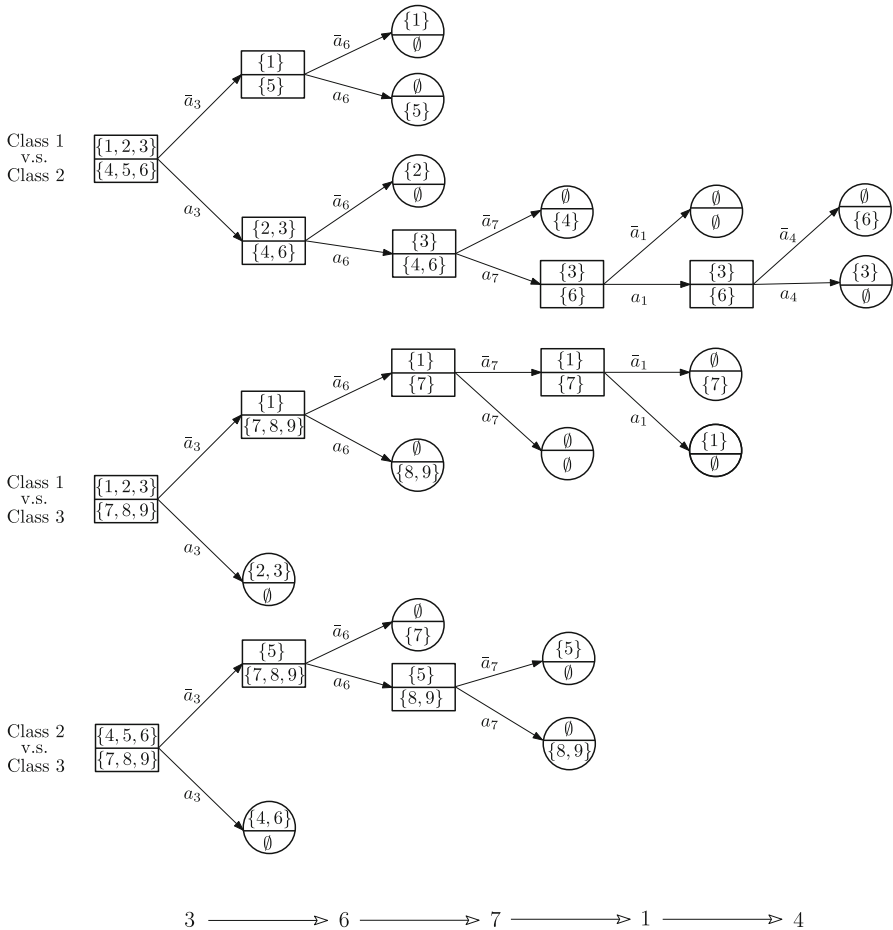
**Iteration 4.** With  $C^1(t_1) = \{3\}$ ,  $C^2(t_1) = \{6\}$   $C^3(t_1) = \emptyset$ , one can easily get

$$\begin{aligned}
 |M_u \cap I_2| &= 0 \\
 |M_u \cap I_4| &= 1 \\
 |M_u \cap I_5| &= 0 \\
 |M_u \cap I_8| &= 1 \\
 |M_u \cap I_9| &= 1
 \end{aligned}$$

Therefore, feature index 4 is selected. We have  $N_t = \{1, 3, 4, 6, 7\}$ ,  $N_u = \{2, 5, 8, 9\}$ . As  $T = \emptyset$  in this iteration, implicit FSM terminates and selects the same set of features in exactly the same order.

If we track the process of implicit FSM and record the information of term  $t$  and  $C^p(t)$ ,  $p \in \{1, 2, 3\}$ , we can draw a graph in Fig. 3. The graph comprises 3 binary trees, each of which tries to separate examples from a pair of different classes. For a non-root node, the path from root node to it corresponds to a term  $t$ . Note that each node of the tree is divided into two regions, the upper part and lower part, which





**Fig. 3** A forest of three binary trees built from the process of implicit FSM on dataset in Table 1. (Each tree separates a pair of classes, and each node in a tree is divided into upper and lower parts, which correspond to the two classes labeled next to the root node. Rectangles in the graph represent internal nodes, and circles denote leaf nodes. The sequence at the bottom indicates the order of features selected by implicit FSM)

contains  $C^p(t)$  and  $C^q(t)$  with  $p < q$ , respectively. For leaf nodes, it is easy to see that at least one of  $C^p(t)$  and  $C^q(t)$  is an empty set.

Recall implicit FSB for binary classification, an acute reader may note from this example that it works on one binary tree, whereas implicit FSM proposed in this paper deals with  $\frac{\ell(\ell - 1)}{2}$  binary trees simultaneously for a dataset containing  $\ell$  classes of data, with each tree trying to separate a pair of classes.

We close this section with the discussion on the memory requirement of standard FS and implicit FSM. With respect to the SC matrix constructed for standard FS, suppose each 0/1 is stored using 1 bit, then the number of bits occupied by the SC

**Table 2** Multi-class datasets from Lichman (2013)

Dataset (Abbreviation)	Number of		
	Examples	Original Features	Classes
Image Segmentation (img)	2310	19	7
Statlog Satellite Image (stim)	6435	36	6
Ann-thyroid (anth)	7200	21	3
Nursery (nurs)	12960	8	5
Letter Recognition (ltrc)	20000	16	26
Avila (avil)	20867	10	12

matrix are:

$$n \times |M| = n \sum_{\substack{p,q \in L \\ p < q}} (|S^p| \times |S^q|) \quad (9)$$

For implicit FSM, a smart implementation does not gather sets  $C^p(a_j)$ ,  $C^p(\bar{a}_j)$ ,  $\forall p \in L, j \in N$  explicitly to calculate  $|I_j|$ ,  $j \in N$ . One only needs to resort to the dataset. When selecting a feature with implicit FSM, only one layer of nodes are active during the selection process, all the previous layers of nodes can be dropped, so we do not need to store the whole forest (see Fig. 3.) Due to the intersection operation, the first layer (which consists of root nodes in the forest) requires most memory. Suppose it requires  $\kappa$  bits to hold the index of each example, then the amount of bits taken up by implicit FSM are:

$$\begin{aligned} & \kappa \times \sum_{\substack{p,q \in L \\ p < q}} (|S^p| + |S^q|) \\ &= \kappa(\ell - 1) \times \sum_{p \in L} |S^p| \\ &= \kappa(\ell - 1)|S| \end{aligned} \quad (10)$$

The number in (10) is orders of magnitude less than the one in (9). To see this, suppose we have a dataset with  $\ell = 3$ ,  $|S^1| = |S^2| = |S^3| = 10^4$ , and  $n = 10^3$ . In addition, the indices of examples are stored using `unsigned int` (which usually occupy 32 bits.) Then (9) yields  $3 \times 10^{11}$ , whereas (10) gives  $1.92 \times 10^6$ , which is 5 orders of magnitude less.

## 5 Numerical experiments

This section tests the efficiency of our proposed implicit FSM on large-scale real-life datasets. Toward this end, we run experiments on multi-class datasets obtained

**Table 3** Computational results

Dataset	Δ	Number of 0-1 Features		CPU Seconds (s)		Improvement <sup>◊</sup>
		†	‡	standard FS	implicit FSM	
imsg	2286900	9819	24	166.43	48.73	241%
stim	16777487	2264	51	251.91	76.53	229%
anth	3620732	737	17	14.15	0.96	1374%
nurs	57319460	27	17	18.80	0.26	7131%
ltrc	192300979	235	58	219.97	52.16	322%
avi1	167601215	32135	50	—*	13892.40	—

Δ: number of rows in SC matrix when using standard FS

†: number of 0-1 features before feature selection

‡: number of 0-1 features after feature selection

\*: dataset avi1 run out of memory and killed by operating system

◊: improvement calculated by  $\frac{\text{CPU seconds by standard FS} - \text{CPU seconds by implicit FSM}}{\text{CPU seconds by implicit FSM}} \times 100\%$

from UCI repository (Lichman (2013).) The information of these datasets are given in Table 2, where we list the number of examples, number of original features, and number of classes of each dataset, respectively. Note that for dataset that contains two parts, i.e. training and testing set, we merge them into one larger dataset.

The experiments were conducted on a machine with i9-9900KS CPU (Base Frequency 4.00GHz, 8 Cores 16 Threads) and 64GB RAM. The numerical results are summarized in Table 3. Numbers in the table are fairly self-explanatory. The two columns under ‘CPU Seconds’ suggest `implicit FSM` is exceptional and runs several times faster than `standard FS`.

It is worth mentioning that for dataset `avil`, `standard FS` failed to select a set of features due to its huge requirement of memory. Recall that each 0/1 is stored using one bit in our experiment, so the SC matrix constructed from dataset `avil` takes up  $167601215 \times 32135 / (1024^3 \times 8) \text{GB} \approx 627 \text{GB}$  of main memory, almost 10 times more than the RAM we have. The feature selection process was killed by operating system before any feature could be selected. On the other hand, `implicit FSM` only needs  $32 \times (12 - 1) \times 20867 / (1024 \times 8) \text{KB} \approx 894 \text{KB}$  of main memory, which is 5 orders of magnitude less than the memory needed by `standard FS`. So, in terms of memory occupation, `implicit FSM` is far superior to the counterpart `standard FS`.

## 6 Conclusion

In this paper, we addressed the feature selection problem in logical analysis of multi-class datasets. The mathematical formulation for this problem is given and a procedure that generalizes the one for solving binary classification problem in literature is developed accordingly. Numerical experiments on 6 real-life multi-class datasets echo our theoretical discussion and demonstrate the superiority of our algorithm over the standard one. With the aid of the proposed algorithm, more large-scale multi-class datasets can be analyzed using LAD.

**Funding** This research was supported by the National Natural Science Foundation of China (NSFC) under grant U19A2059, 61806095, 61802183 and 61972110, the Fundamental Research Funds for the Central Universities under grant 30920021130, the Jiangsu Planned Projects for Postdoctoral Research Funds under grant 1701146B, the Natural Science Foundation of Guangdong Province under grant 2017A030307026, National Foundation Raising Project of Shantou University under grant NFC16002, Scientific Research Start-up Funding Project of Shantou University under grant STF15003.

**Data availability and material** The data used in this paper can be found in UCI machine learning repository at <https://archive.ics.uci.edu/ml/index.php>

## Declaration

**Conflict of interest** The authors declare no conflict of interest.

**Code availability** The experiments were conducted using custom code written by the authors.

## References

- Alexe G, Hammer PL (2006a) Spanned patterns for the logical analysis of data. *Discret Appl Math* 154(7):1039–1049
- Alexe G, Alexe S, Liotta LA, Petricoin E, Reiss M, Hammer PL (2004) Ovarian cancer detection by logical analysis of proteomic data. *Proteomics* 4:766–783
- Alexe G, Alexe S, Axelrod DE, Bonates T, Lozina II, Reiss M, Hammer PL (2006) Breast cancer prognosis by combinatorial analysis of gene expression data. *Breast Cancer Research* 8R41
- Alexe G, Alexe S, Bonates TO, Kogan A (2007) Logical analysis of data - the vision of Peter L. Hammer. *Ann Math Artif Intell* 49:265–312
- Alexe S, Hammer PL (2006b) Accelerated algorithm for pattern detection in logical analysis of data. *Discret Appl Math* 154:1050–1063
- Alexe S, Blackstone E, Hammer PL, Ishwaran H, Lauer MS, Snader CEP (2003) Coronary risk prediction by logical analysis of data. *Ann Op Res* 119:15–42
- Avila-Herrera JF, Subasi MM (2015) Logical analysis of multi-class data. In: 2015 Latin American Computing Conference (CLEI), pp 1–10
- Bain TC, Avila-Herrera JF, Subasi E, Subasi MM (2020) Logical analysis of multiclass data with relaxed patterns. *Ann Op Res* 287:11–35
- Balas E, Carrera MC (1996) A dynamic subgradient-based branch-and-bound procedure for set covering problem. *Op Res* 44(6):875–890
- Bonates TO, Hammer PL, Kogan A (2008) Maximum patterns in datasets. *Discret Appl Math* 156(6):846–861
- Boros E, Hammer PL, Ibaraki T, Kogan A (1997) Logical analysis of numerical data. *Math Progr* 79:163–190
- Boros E, Hammer PL, Ibaraki T, Kogan A, Mayoraz E, Muchnik I (2000) An implementation of logical analysis of data. *IEEE Trans Knowl Data Eng* 12:292–306
- Brannon AR, Reddy A, Seiler M, Arreola A, Moore DT, Pruthi RS, Wallen EM, Nielsen M, Liu H, Nathanson KL, Ljungberg B, Zhao H, Brooks JD, Ganesan S, Bhanot G, Rathmell WK (2010) Molecular stratification of clear cell renal cell carcinoma by consensus clustering reveals distinct subtypes and survival patterns. *Genes Cancer* 1(2):152–163
- Brauner MW, Brauner N, Hammer PL, Lozina I, Valeyre D (2007) Logical analysis of computed tomography data to differentiate entities of idiopathic interstitial pneumonias. *Data Min Biomed* 7:193–208
- Bruni R (2007) Reformulation of the support set selection problem in the logical analysis of data. *Ann Op Res* 150:79–92
- Cai Z, Xu L, Shi Y, Salavatipour MR, Goebel R, Lin G (2006) Using gene clustering to identify discriminatory genes with higher classification accuracy. In: Sixth IEEE Symposium on BioInformatics and BioEngineering (BIBE'06), pp 235–242, 10.1109/BIBE.2006.253340
- Cai Z, Goebel R, Salavatipour MR, Lin G (2007) Selecting dissimilar genes for multi-class classification, an application in cancer subtyping. *BMC Bioinform* 8(206):1–15
- Cai Z, Miao D, Li Y (2019) Deletion propagation for multiple key preserving conjunctive queries: Approximations and complexity. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE), pp 506–517, 10.1109/ICDE.2019.00052
- Caprara A, Fischetti M, Toth P (1999) A heuristic method for the set covering problem. *Op Res* 47(5):730–743
- Ceria S, Nobili P, Sassano A (1998) A lagrangian-based heuristic for large-scale set covering problems. *Math Progr* 81(2):215–228
- Chvatal V (1979) A greedy heuristic for the set-covering problem. *Math Op Res* 4(3):233–235
- Crama Y, Hammer PL, Ibaraki T (1988) Cause-effect relationships and partially defined Boolean functions. *Ann Op Res* 16:299–326
- Das TK, Adepu S, Zhou J (2020) Anomaly detection in industrial control systems using logical analysis of data. *Comput Secur* 16:299–326
- Fisher ML, Kedia P (1990) Optimal solution of set covering/partitioning problems using dual heuristics. *Manag Sci* 36:674–688
- Galar M, Fernández A, Barrenechea E, Bustince H, Herrera F (2011) An overview of ensemble methods for binary classifiers in multi-class problems: experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognit* 44:1761–1776

- Gubskaya AV, Bonates TO, Kholodovych V, Hammer P, Welsh WJ, Langer R, Kohn J (2011) Logical analysis of data in structure-activity investigation of polymeric gene delivery. *Macromol Theory Simul* 20(4):275–285
- Guo C, Ryoo HS (2012) Compact MILP models for optimal and Pareto-optimal LAD patterns. *Discret Appl Math* 160:2339–2348
- Hammer PL (1986) Partially defined Boolean functions and cause-effect relationships
- Jocelyn S, Chinniah Y, Ouali M, Yacout S (2017) Application of logical analysis of data to machinery-related accident prevention based on scarce data. *Reliab Eng Syst Safety* 159:223–236
- Jocelyn S, Ouali MS, Chinniah Y (2018) Estimation of probability of harm in safety of machinery using an investigation systemic approach and logical analysis of data. *Safety Sci* 105:32–45
- Kim HH, Choi JY (2015) Pattern generation for multi-class LAD using iterative genetic algorithm with flexible chromosomes and multiple populations. *Expert Syst App* 42:833–843
- Kim K, Ryoo HS (2008) A LAD-based method for selecting short oligo probes for genotyping applications. *OR Spectr* 30:249–268
- Kohli R, Krishnamurti R, Jedidi K (2006) Subset-conjunctive rules for breast cancer diagnosis. *Discret Appl Math* 154:1100–1112
- Kronek LP, Reddy A (2008) Logical analysis of survival data: prognostic survival models by detecting high-degree interactions in right-censored data. *Bioinformatics* 24:i248–i253
- Lichman M (2013) UCI machine learning repository. <http://archive.ics.uci.edu/ml>
- Miao D, Cai Z, Li J (2018) On the complexity of bounded view propagation for conjunctive queries. *IEEE Trans Knowl Data Eng* 30(1):115–127
- Miao D, Cai Z, Li J, Gao X, Liu X (2020a) The computation of optimal subset repairs. *Proc VLDB Endow* 13(11):2061–2074
- Miao D, Cai Z, Liu X, Li J (2020b) Functional dependency restricted insertion propagation. *Theor Comput Sci* 819:1–8. <https://doi.org/10.1016/j.tcs.2017.03.043>
- Mortada M, Carroll T, Yacout S, Lakis A (2012) Rogue components: their effect and control using logical analysis of data. *J Intell Manuf* 23:289–302
- Mortada MA, Yacout S, Lakis A (2011) Diagnosis of rotor bearings using logical analysis of data. *J Qual Maint Eng* 17(4):371–397
- Mortada MA, Yacout S, Lakis A (2014) Fault diagnosis in power transformers using multi-class logical analysis of data. *J Intell Manuf* 25(6):1429–1439. <https://doi.org/10.1007/s10845-013-0750-1>
- Ragab A, Ouali M, Yacout S, Osman H (2016) Remaining useful life prediction using prognostic methodology based on logical analysis of data and kaplan-meier estimation. *J Intell Manuf* 27:943–958
- Ragab A, El-Koujok M, Poulin B, Amazouz M, Yacout S (2018) Fault diagnosis in industrial chemical process using interpretable patterns based on logical analysis of data. *Expert Syst Appl* 95:368–383
- Ryoo HS, Jang IY (2007) A heuristic method for selecting support features from large datasets. In: Kao M, Li X (eds) *Algorithmic Aspects in Information and Management, Third International Conference, AAIM 2007, Portland, OR, USA, June 6-8, 2007, Proceedings*, Springer, Lecture Notes in Computer Science, vol 4508, pp 411–423, 10.1007/978-3-540-72870-2\_39
- Ryoo HS, Jang IY (2009) MILP approach to pattern generation in logical analysis of data. *Discret Appl Math* 157:749–761
- Shaban Y, Yacout S, Balazinski M (2015) Tool wear monitoring and alarm system based on pattern recognition with logical analysis of data. *J Manuf Sci Eng* 137(3):1–14
- Shaban Y, Meshreki M, Yacout S, Balazinski M, Attia H (2017a) Process control based on pattern recognition for routing carbon fiber reinforced polymer. *J Intell Manuf* 28(1):165–179
- Shaban Y, Yacout S, Balazinski M, Jemielniak K (2017b) Cutting tool wear detection using multi-class logical analysis of data. *J Mach Sci Technol* 21(3):1–16
- Yacout S, Danish A, Saadany S, Kapongo J, Mani S, Gomes J (2013) Knowledge discovery from observational data of causal relationship between clinical procedure and alzheimer’s disease. *J Publ Health* 2:1–10
- Yan K, Ryoo HS (2017a) 0–1 multilinear programming as a unifying theory for LAD pattern generation. *Discret Appl Math* 218:21–39
- Yan K, Ryoo HS (2017b) Strong valid inequalities for Boolean logical pattern generation. *J Global Optim* 69(1):183–230
- Yan K, Ryoo HS (2019a) Cliques for multi-term linearization of 0-1 multilinear program for Boolean logical pattern generation. In: Thi HAL, Le HM, Dinh TP (eds) *Optimization of Complex Systems: Theory, Models, Algorithms and Applications, WCGO 2019, World Congress on Global Optimization*,

- Metz, France, 8–10 July, 2019, Springer, Advances in Intelligent Systems and Computing, vol 991, pp 376–386, 10.1007/978-3-030-21803-4\_38
- Yan K, Ryoo HS (2019b) A multi-term, polyhedral relaxation of a 0–1multilinear function for boolean logical pattern generation. *J Global Optim* 74(4):705–735
- Yang K, Cai Z, Li J, Lin G (2006) A stable gene selection in microarray data analysis. *BMC Bioinform* 7(228):1–16

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.