Chapter 4

Some Fast Heuristics for Inferring a Boolean Function from Examples

4.1 Some Background Information

The previous two chapters discussed the development and key mathematical properties of some branch-and-bound (B&B) approaches for inferring a Boolean function in the form of a compact (i.e., with as few clauses as possible) CNF or DNF expression from two collections of disjoint examples. As was described in Chapters 2 and 3, the B&B approaches may take a long time to run (actually, they are of exponential time complexity).

This chapter presents a simple heuristic approach which may take much less time (of polynomial time complexity) than the B&B approaches. This heuristic attempts to return a small (but not necessarily minimum or near minimum) number of clauses of the CNF or DNF expressions inferred from two disjoint collections of examples. Some variants of this heuristic are also discussed. This chapter is based on the developments first reported in [Deshpande and Triantaphyllou, 1998].

In this chapter we consider two closely related problems. The first one is how to infer a Boolean function fast from two disjoint collections of positive and negative examples. All the examples are again assumed to be binary vectors and we also assume that we know them precisely. The second problem considers cases in which we have partial knowledge of some of the examples.

In order to help fix ideas, consider the following two disjoint sets of positive and negative examples (which were also used in the previous two chapters):

\[
E^+ = \begin{bmatrix}
0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 \\
1 & 0 & 0 & 1
\end{bmatrix}
\quad \text{and} \quad
E^- = \begin{bmatrix}
1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0
\end{bmatrix}.
\]

Given these data we want to derive a single Boolean function (in CNF or DNF form) that satisfies the requirements implied in the previous examples. For the CNF...
case, we would like each clause (i.e., each disjunction of the CNF expression) to be accepted by each of the positive examples, while each negative example to be rejected by at least one of the clauses. For instance, the following CNF expression satisfies these requirements:

\[(A_2 \lor A_4) \land (\bar{A}_2 \lor \bar{A}_3) \land (A_1 \lor A_3 \lor \bar{A}_4).\]

This, in essence, is Problem 1, that is, how to construct a set (of hopefully small size) of clauses which would correctly classify all the available positive and negative examples and hopefully classify new examples with high accuracy.

Next, in order to illustrate Problem 2, we consider the following hypothetical sample of input data:

\[E^+ = \begin{bmatrix} 0 & * & 0 & 0 \\ 1 & 0 & * & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \quad E^- = \begin{bmatrix} 0 & 0 & * & 1 \\ * & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \text{and} \]

\[E^U = \begin{bmatrix} 0 & 1 & 1 & * \\ 1 & 1 & * & * \end{bmatrix},\]

In this chapter the \(E^U\) set will denote the set with the undecidable (not to be confused with unclassified) examples. The symbol “*” in the three data sets represents attributes whose binary values are missing.

In the previous \(E^+\) and \(E^-\) data sets it is assumed that the missing data values (indicated by “*”) did not prohibit the “oracle” (i.e., the hidden function) from classifying the corresponding examples as positive or negative. For instance, the first positive example in \(E^+\) (i.e., \((0, *, 0, 0)\)) implies that this example should be positive regardless of the actual nature of the “*” element. This observation indicates that the following two examples (note that \(2 = 2^1\), where 1 is the number of the missing elements in that example): \((0, 0, 0, 0)\) and \((0, 1, 0, 0)\) are also positive examples. That is, for positive and negative examples the missing values can be treated as do not care cases (i.e., they can be either 1 or 0 without changing the classification of that example). The notion of the do not care concept was first introduced by Kamath, et al., in [1993] in order to condense the information representation in this type of learning problems.

An obvious restriction for the data in the two sets to be valid is that every possible pair of a positive and a negative example should have at least one of their common fixed attributes with a different value. For instance, when \(n = 8\), the examples \((1, 1, *, *, 0, 0, *, *)\) and \((1, 1, *, 0, 0, 0, *, 1)\) cannot belong to different classes (i.e., one to be positive and the other to be negative). This is true because the example \((1, 1, 0, 0, 0, 0, 1)\) is implied by either of the previous two examples which have missing (i.e., do not care) elements.

To further illustrate the concept of the undecidable examples consider the following Boolean function defined on five attributes (i.e., \(n = 5\)):

\[(A_1 \lor A_4) \land (A_2 \lor \bar{A}_3 \lor A_5).\]