ALC 2019/2020

1^{st} Project – Decision Tree Construction with SAT/MaxSAT/PB

16 October 2019, version 1.1

Overview

The 1^{st} ALC project is to develop a software tool for constructing decision trees [2, 1]. In order to solve this problem, students must use solvers for Satisfiability (SAT), Maximum Satisfiability (MaxSAT), Pseudo-Boolean Satisfiability (PBS) or Pseudo-Boolean Optimization (PBO).

Problem Specification

Given a set of samples and an integer N, the task is to decide whether there exists a decision tree with N nodes correctly classifying the samples. All features have only values 0 and 1. Further, samples always fall into one of the classes positive (T) and negative (F).

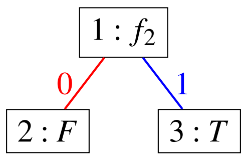


Figure 1: Simple decision tree with the nodes $\{1,2,3\}$ deciding only on feature f_2.

Figure 1 shows a simple decision tree with 3 nodes, where the root node decides on the feature f_2 and responds T if the feature is 1 and responds F if the feature is 0. Such tree correctly classifies for instance the samples positive $= \{(0,1)\}$, negative $= \{(0,0)\}$. Observe that the tree can ignore f_1 because its value is always 0 in the given samples.

Features may repeat on a single branch. Effectively, this means that the program should find a decision tree for any $N \geq N_0$ if there exists one for N_0.

Project Goals

You are to implement a tool, or optionally a set of tools, invoked with command `proj1`. This set of tools should use a SAT/MaxSAT/PB solver to decide whether a decision tree of size N

exists for a given N and set of samples.

Your tool does not take any command-line arguments. The problem instance is to be read from the standard input.

Consider an instance file named `samples.jfp`. The tool is expected to be executed as follows:

```
proj1 < samples.smp > solution.txt
```

The tool must write the solution to the standard output, which can then be redirected to a file (e.g., `solution.txt`).

The programming languages to be used are only C/C++, Java or Python. The formats of the files used by the tool are described below.

File Formats

You can assume that all input files follow the description provided in this document. There is no need to check if the input file is correct. Additionally, all lines (input or output) must terminate with the end-of-line character.

Input Format

The input file representing a problem instance is a text file that follows the following format:

- One line with two integers K and N defining the number of features and nodes, respectively.
- A sequence of lines where each line consists of $K+1$ characters 0/1, separated by a single space.
- The first K characters determine the values of features in the sample, and the last character determines its class.
- It is guaranteed that there are no conflicting samples, i.e., there are never two lines with the same values of features but different class.

You may assume that $N > 1$ and that it is odd.

Output Format

The output of the program representing an optimal solution to the problem instance must comply with the following format:

- If no tree exists, the program produces a single line with the word “UNSAT”.

Otherwise, the program produces a tree in the following format consisting of 4 types of lines.

- "l i j" representing that j is a **l**eft (0) child of i
- "r i j" representing that j is a **r**ight (1) child of i
- "c i v" leaf i responds with the **c**lass v
- "a r i" the feature r is **a**ssigned to internal node i

The order of these lines is not important and any line that does not begin with either of the characters l, c, r, a, will be ignored. It is always assumed that node 1 is the root. The tree must be binary, i.e., a node either has no children or it has two (left, right). The tree must be consistent with the given set of samples. In particular, for any sample s, there must not be a path in the tree consistent with the sample s, where the leaf at the end of the path responds with a different class than the sample s.

Important: The final version to be submitted for evaluation must comply with the described output. Project submissions that do not comply will be severely penalized, since each incorrect output will be considered as a wrong answer. An application that verifies if the output complies with the description is available on the course's website.

Example

The file describing the problem discussed above is described by the following input.

```
2 3
0 0 0
0 1 1
```

A solution corresponding to Figure 1 would be:

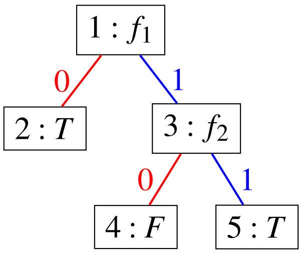
```
l 1 2
r 1 3
a 2 1
c 2 0
c 3 1
```

Another example with still 2 features but requiring 5 nodes is the following.

```
2 5
0 0 1
0 1 1
1 1 1
1 0 0
```

```
l 1 2
r 1 3
l 3 4
r 3 5
c 2 1
c 4 0
c 5 1
a 1 1
a 2 3
```

This tree can be visualized as in the following figure.



The following set of samples has no solution with 5 nodes.

```
2 5
0 0 1
0 1 0
1 1 1
1 0 0
```

Therefore, the program should simply print

```
UNSAT
```

Additional Information

The project is to be implemented in groups of one or two students.

The project is to be submitted through the course website. Jointly with your source code, you should submit a short text file describing the main characteristics of your project.

The evaluation will be made taking into account correctness given a reasonable amount of CPU time (80%) and efficiency (20%).

The input and output formats described in this document must be strictly followed.

Project Dates

- Project published: 11/10/2019.
- Project due: 01/11/2019 at 23:00.

Omissions & Errors

Any detected omissions or errors will be added to future versions of this document. Any required clarifications will be made available through the course's official website.

Versions

11/10/2019,	version 1.0:	Original version.
16/10/2019,	version 1.1:	Fixed small issues with the visualized trees.

References

[1] N. Narodytska, A. Ignatiev, F. Pereira, and J. Marques-Silva. Learning optimal decision trees with SAT. In J. Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 1362–1368. ijcai.org, 2018.

[2] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.