CSC410 Project 8 PAC Learning Automata using Examples

Mehmet Atmaca, Cem Kutay Yildirim, Yunhao Mao, Derin Ozerman

December 7th 2017

1 Introduction

In this project, we used PAC learning strategy to developed a Python program that learns and builds an finite deterministic automata from by providing examples.

We implemented an learner class that uses RPNI algorithm to construct and draw DFAs from given example sets. We also implemented a teacher class that can take an existing DFAs can generate positive and negative examples, there is also a subclass characteristics sets teacher sub class that can generate expanded tree sets for given DFA.

2 Description

The project consists of 2 parts, learner and teacher, they can work independently or combine together to use to train a learner.

We used a simple Python DFA library called python-automata[1] that provides the data-structure for easy creation, manipulation and examine a DFA. The library was modified by us for our needs.

The other library we used is pygraphviz[2], this is used for ease of drawing DFAs to PNG files. In order for the drawing to work, Graphviz[3] program needed to be installed since what the library is providing a set Python APIs.

There is also a helper module dfa_parser, it can be input a Python dictionary and return an instance of DFA, or convert a DFA object into dictionary representation.

Learner is the class that implemented RPNI algorithm. RPNI algorithm was implemented based on the book [4] and paper [6]. It works by creating a prefix

acceptor tree which is the initial DFA accepting all of the given positive examples. RPNI algorithm then tries to merge child states into the parent while changing the transition of states accordingly. After a candidate state merge is done, the DFA is checked against given examples and expected outcomes. If any of the labelled examples are incorrectly recognized by the candidate merged DFA, the candidate DFA is discarded and another candidate is created by choosing the next state to merge. The RPNI algorithm repeats itself until all of the positive examples are recognized and negative examples are rejected.

Teacher is the class that parse DFA and generate examples positive or negative strings. The strings are generated randomly. The teacher can be requested for a positive or negative example and used by the learner. The Teacher also has a subclass called CharacteristicSetTeacher, this class implemented the functions of getting a characteristic set of a given DFA. We implemented the algorithm based on the paper *On Characteristic Sets and Degrees of Finite Automata*[5]. As a matter of fact, the characteristic sets we generated by expanding the DFA in to a tree T(A) and get all the paths from the tree and form set $|T(A)|$.

3 Result

While using the teacher that was created from section 1 of the project scope, the non characteristic sets were created from random binary strings and then labelled with the DFA given to the teacher. Compared to the characteristic set teacher, the learner was not always able to correctly guess the DFA. It was observed that increasing the number of examples given to the learner helps with accuracy of the guess although since the RPNI algorithm's complexity is polynomial time it becomes unfeasible to guess DFA due to increased time considerations.

We tested on providing a set of examples from characteristic set teacher. The learner will run the RNPI algorithm based on that set and the result is below:

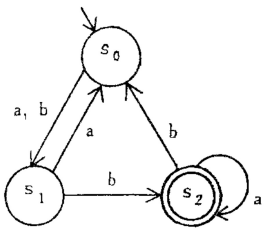

Figure 1: Original

Figure 1 is the original DFA, and figure 2 is the DFA generated by RPNI after fed in the characteristic set. We discovered that the DFAs aren't exactly

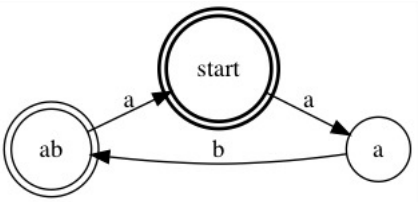

Figure 2: Generated

the same, this is due to the nature of RPNI algorithm bases on accept/reject to construct the DFA instead of specific states. We discovered the resulting DFA satisfies all the input examples from characteristic set teacher.

Performance wise, we measured that:

4 Questions

1. How many examples would you need?

   The number of examples depends on the number of accepted states. RPNI algorithm requires a certain set of positive examples to generate a prefix tree automaton(PTA). This set of postive examples must be structurally complete, which means covers each transition of DFA (except the transitions associated with the dead state) and uses every element of the set of final states of DFA as an accepting state[6]. The counter examples we need is at least 2N - 1 where N is the state of the DFA.

References

[1] https://github.com/reverie/python-automata

[2] https://github.com/pygraphviz/pygraphviz

[3] "Graphviz - Graph Visualization Software." www.graphviz.org/

[4] Higuera, David de la. "12 - Informed Learners." Grammatical Inference: Learning Automata and Grammars, Cambridge University Press, 2010.

[5] Tanatsugu, Keisuke, and Setsuo Arikawa. "On Characteristic Sets and Degrees of Finite Automata." SpringerLink, Kluwer Academic Publishers-Plenum Publishers, link.springer.com/article/10.1007/BF00991485.

[6] Honavar, Vasant G. "Learning DFA From Simple Examples." Machine Learning