CS CAPSTONE TECHNOLOGY REVIEW

NOVEMBER 9, 2019

REDUCING PATIENT DOSE FROM DIAGNOSTIC IMAGING USING MACHINE LEARNING

PREPARED FOR

OREGON STATE UNIVERSITY

STEVE REESE _____ _____
Signature *Date*

PREPARED BY

CS16: RADIOLOGICAL COUNTING

SEAN GILLEN _____ _____
Signature *Date*

Abstract

This document provides an overview of the technical choices made by Sean Gillen for the Oregon State University (OSU) Computer Science (CS) capstone team 16. The system created will apply an algorithm developed collaboratively at OSU with researchers at Georgetown University to demonstrate a net reduction in average patient radiation dose based upon radiation counting data collected by an analogous detector set up at the Oregon State University Training, Research, Isotopes, General Atomics (TRIGA) reactor. This project will use machine learning algorithms to create a proof-of-concept system to stop unnecessary irradiation when the desired outcome is reached and reduce the dose to patients. The choices made by Sean Gillen will be about the decision tree classifier implementation and data pre-processing tools used.

Because the group's work is under a Non-Disclosure Agreement (NDA), this document is being emailed instead of submitted to the Canvas grading system.

CONTENTS

1 Project Overview

1.1 Problem

Current radiological counting methods rely on long periods of radiation detection and analysis, often on the order of minutes. They do so in order to ensure enough data is available to yield an accurate result for the analysis being done, from spectroscopy to determine a compound's elemental makeup to x-ray medical imagery to identify structures within a patient's body. For some applications, the cost for an extra minute of data collection may just be wasted time for the technician. For other applications, the cost may be extra irradiation of a patient. Sometimes, an equal or sufficiently good image can be made with a shorter time of irradiation, but current tools do not always reach this optimal stopping time.

1.2 Solution

The group will apply several standard machine learning methods to analyze output from a spectrometer. The main steps involved will be:

1) Get the data from the spectrometer on a continuous basis
2) Process the data to ready it for analysis
3) Analyze the data with one of three machine learning models
4) Return a result after enough iterations of 1-3

2 My Responsibility

The group has divided the responsibility for the project's technical options by member. The system's components are going to use standard tools in order to allow the group to focus on creating a robust, simple system that demonstrates the potential for radiation counting systems. Many data processing libraries exist, so I will research ones fitting the project's other technologies and the group's prior experience. Since the client has requested three different machine learning models be used to perform analysis, each member has received a model to study and research implementation methods for. My model is classification using decision trees. Decision trees refer to a class of algorithms, so I will research their pros and cons.

My main three responsibilities are:

- Decision tree classifier subclass
- Decision tree classifier implementation library
- Data pre-processing library

3 Review of Technical Options

3.1 Decision Tree Subclasses

Decision trees have the common feature of being able to be represented by a tree graph where each node represents a decision. OSU PhD candidate Jessica Curtis, whose work is this project is continuing, discusses decision trees in her dissertation, noting that there are a "variety of computational approaches to be implemented." [1]

3.1.1 ID3

The ID3 algorithm was developed in the 1970s and is the basis for derivative decision tree algorithms. It is compared with several others in a 2014 comparative study by researchers at Sultan Moulay Slimane University. [2] ID3 is simpler than other decision tree algorithms, but has drawbacks including limited cleanup after tree creation to simplify it by removing unneeded nodes.

3.1.2 C4.5

As described in the 2014 Sultan Moulay Slimane University analysis, C4.5 improves upon several aspects of ID3. C4.5 was developed by J.R. Quinlan in the 1990s, and is currently a very common decision tree algorithm. [3] C4.5 allows for continuous data and provides other flexibility such as weighting attributes differently and allowing missing values.[2] After tree creation, C4.5 goes back and removes nodes that are unneeded. This improves the tree's performance because fewer nodes must be traversed to arrive at a leaf node.

3.1.3 C5.0

C5.0 is the later version of C4.5, developed by the same author J. R. Quinlan. The developer claims vastly better performance with C5.0, such as a reduction from 8 hours to 3 minutes to create a ruleset for their 'forest' data. [4] C5.0 is under a proprietary license. [5]

3.1.4 Decision

Decision trees produced by C4.5 will be sufficient for this project, but the improvements offered by C5.0 make it the most attractive overall. Because this project is meant to be real-time, performance in a key system component is important. If the group can use C5.0 with its more restrictive licensing, it will make the decision tree component simpler to create.

3.2 Decision Tree Implementation

3.2.1 Python Scikit-learn

Scikit-learn has the advantage of being within Python, which should make its use quite simple, should the group use Python as the main system language. It provides the class DecisionTreeClassifier, which can implement many algorithms. They include ID3, C4.5, C5.0, and CART (Classification And Regression Trees). This should make it simple to switch algorithm, should the group need to. [5]

3.2.2 WEKA C4.5

The Waikato Environment for Knowledge Analysis (WEKA) provides an implementation of the C4.5 algorithm. It also provides a rich Graphical User Interface (GUI) with tools for experimenting with the classifier parameters and output. It will be useful in experimenting with tuning the algorithm, but will probably be superseded by scikit-learn for ease of use and performance constraints to make the analysis real-time. WEKA uses Java, which could make it more difficult to achieve high performance due to the interaction needed between the Java runtime and our Python application.

3.2.3 Decision

The group plans to use Python's scikit-learn package in combination with WEKA, at least at the start. WEKA's Graphical User Interface (GUI) will make experimenting with and visualizing data simple. Python's scikit-learn package will offer

the ability to swap out different algorithms more easily. Using scikit-learn will simplify development: the data pre-processing can be done in Python and then sent to scikit-learn without having to convert the data again to another format needed by an external module.

3.3 Data pre-processing library

The system will read in data from a spectrometer by receiving data in a line-separated output file. The process of detecting new output should be straightforward and amount to continually checking the file's timestamp to see if it has changed since the last check. The file itself will be in list-mode format, where data is recorded for specific channels on each line. This format may be somewhat configurable but is not intended for direct input into a machine-learning model, so the group will probably need to do most of the transformation in our application. The group will arrive at a common internal format that is easily fed to our models.

To do so, we plan to use a tool that is easy to use and reasonably fast. Speed is important here because the group has now worked with some sample list-mode data that is on the order of hundreds of megabytes for a few minutes' output. The team is considering these options:

- Python's numpy library
- Python's pandas library, which uses numpy
- R language standard library

Below are the three options:

3.3.1 Python: numpy

Numpy is a python library intended for fast implementations of common mathematical routines. It achieves better performance on common data manipulation tasks than the Python standard library.

3.3.2 Python: pandas

Pandas, which uses Numpy, provides many features useful to manipulate data like the data we will receive from the spectrometer. It uses Numpy's fast routines while providing data processing functionality that the group will not have to spend time recreating. [6]

3.3.3 R language

The team already has some code created with the R language to convert CSV data to the format used by WEKA. The R language has a standard library with similar functionality to pandas, and R is an interpreted language like Python.

3.3.4 Decision

The group plans to primarily use a combination of numpy and pandas for data processing. Numpy provides improved performance from the Python standard library, and pandas leverage's Numpy to allow more complicated data transformations. The group will also use R in the initial stages at least because it already has relevant code, from the work done by Jessica Curtis at Oregon State Univesity.

REFERENCES

[1] J. R. Curtis, *Special Nuclear Material Classification and Mass Content Estimation Using Temporal Gamma-Ray Spectroscopy and Machine Learning Methods*. PhD thesis, Oregon State University, 2019.

[2] B. Hssina, A. Merbouha, H. Ezzikouri, and M. Erritali, "A comparative study of decision tree id3 and c4. 5," *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 2, pp. 0–0, 2014.

[3] J. R. Quinlan, "Improved use of continuous attributes in c4.5," *Journal of Artificial Intelligence Research 4*, vol. 4, pp. 77–90, 1996.

[4] "Is c5.0 better than c4.5?." https://www.rulequest.com/see5-comparison.html.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[6] W. Mckinney, "pandas: a foundational python library for data analysis and statistics," *Python High Performance Science Computer*, 01 2011.