CS CAPSTONE PROJECT HAND-OFF

MAY 30, 2020

RADIATION SPECTRUM ANALYSIS USING MACHINE LEARNING

PREPARED FOR

OREGON STATE UNIVERSITY

DR. STEVE REESE

PREPARED BY

GROUP 16

RADIOLOGICAL COUNTING

IAN BROWN

SEAN GILLEN

YIHONG LIU

Abstract

The Capstone group created a machine learning system to perform real-time classification of radiation spectrum data from a spectrometer in the OSU Radiation Center. This report collects the group's documents and discusses the project and next steps.

CONTENTS

FOREWORD: RELEASE NOTES 1.0

Note: for overall project documentation, see Section 7: Project Documentation.

Depending on the direction that the clients want to go, there are several items left to work on:

Manager: Tie the different models together for real-time simultaneous use.

Model/training data persistence: Create system for saving training data generated from the input .dat files. The models trained from this data can also be saved, but the parsing/creation of the training data itself from the ListPRO spectrometer output files is most of the runtime of using the models. The team started on this using the python package pickle, which could be the simplest way, but did not get to a finished component.

Models: Refactor shared code among three models. They are mostly the same in terms of training data setup.

Performance: Decide optimal frequency to rerun the classifier's predictions.

Testing: The team created a script to simulate ListPRO, but more testing could be done using the real ListPRO. In theory, it should not make a difference, but it's possible that the simulator script is different in a way that affects operation. This requires being logged onto the computer that has the spectrometer attached (we were using d102-03, in the Radiation Center). It works over remote desktop, since the detector is usually left on all the time for data collection.

Interface: The client has not expressed an interest in having more of an interface, but a graphical interface could be created that would allow adjusting different parameters, such as the model(s) and training sets used.

1 Introduction to Project

The purpose of the project was to develop a proof-of-concept application that implements Machine Learning (ML) algorithms for radiation spectrum analysis. The project was requested by Dr. Steven Reese, director of the Oregon State University Radiation Center. Development of the project was intended to demonstrate the feasibility of a patent belonging to Dr. Reese. If done properly, the project has the potential to make improvements to the field of radiation spectrum analysis. Specifically, it was confirmed that the application can identify certain radioisotopes very quickly using ML. Ortec GammaVision, which can also perform identification, served as a baseline comparison.

The clients for this project were Dr. Reese, Dr. Ophir Frieder, an ML specialist at Georgetown University, and Jessica Curtis, a former PhD student at OSU. Dr. Reese was in charge of delegating work for the team and was our main contact. We met with him frequently and provided updates on our progress. Dr. Frieder provided insight to some of the ML models we used and suggestions for training. Jessica supplied us with some starter code used in her thesis and consulted with Dr. Reese on the science behind the project.

The members of the team were Ian Brown, Sean Gillen, and Yihong Liu. Ian was responsible for the naive Bayes model and some of the work in the real-time parsing component of the application. Sean developed the decision tree and decision forest models and completed the work necessary for reading the Ortec spectra files, simulating detector output, and implementing a watchdog for responding to data changes in real time. Yihong was in charge of the neural network model and tuned performance using many different parameters for this classifier.

Fortunately, the transition to remote work in the spring term did not severely impact our project. By this time, our application was functionally complete and we were able to run tests locally. Then, we sent these results to Dr. Reese who was able to compare them with his own tests of GammaVision beforehand at the lab. The work remaining for the

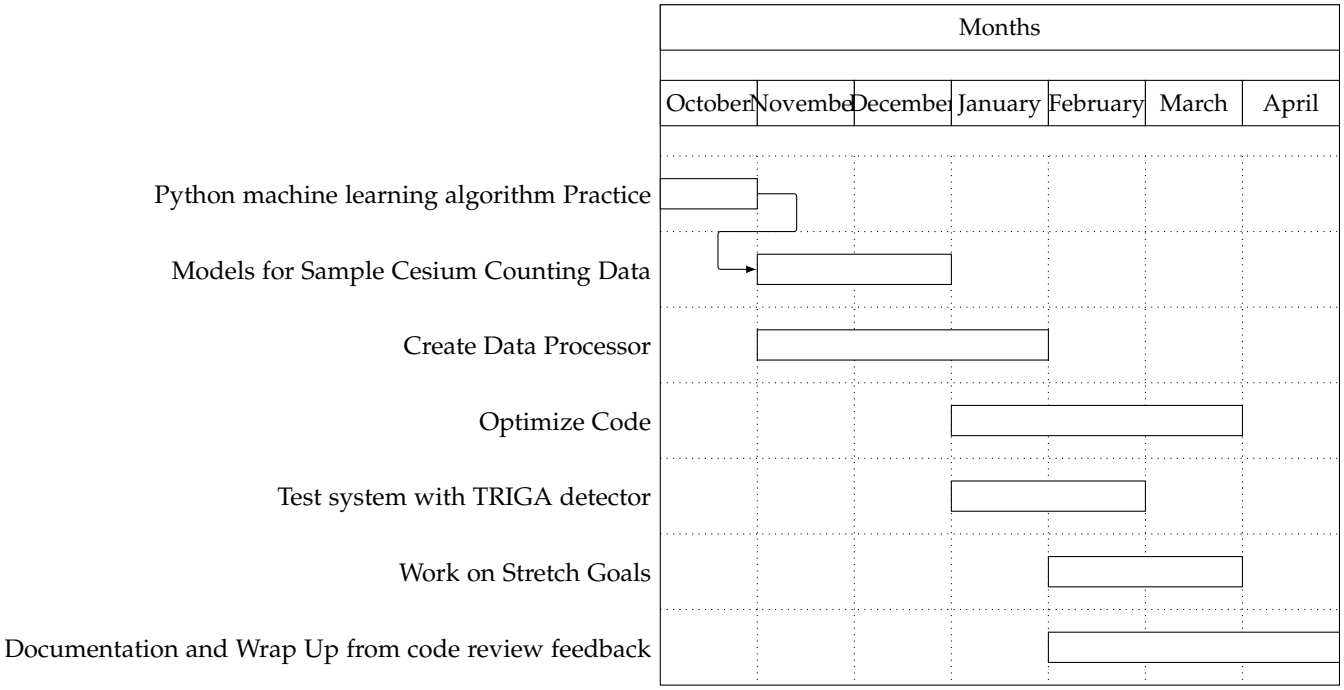
project is to link the file watcher component with the machine learning component so that the classifiers are ran in real time while new data is written to a file. For more information, see the Foreword of this document.

2 Requirements Document

Revisions

Section	Original	New
1.1 - Purpose	• determine when an optimal dosage of radiation has been reached	• classify the appearance of different isotopes in the radiation source.
1.2 - Scope	• Using Weka to software to do the machine model development.	• Switched to use python with scikit-learn library of machine learning.

Updated Gantt Chart



CS CAPSTONE REQUIREMENTS DOCUMENT

MAY 30, 2020

REDUCING PATIENT DOSE FROM DIAGNOSTIC IMAGING USING MACHINE LEARNING

PREPARED FOR

OREGON STATE UNIVERSITY

DR. STEVE REESE _____ _____
Signature *Date*

PREPARED BY

RADIOLOGICAL COUNTING

IAN BROWN _____ _____
Signature *Date*

SEAN GILLEN _____ _____
Signature *Date*

YIHONG LIU _____ _____
Signature *Date*

Abstract

The OSU capstone team will demonstrate a proof-of-concept reduction in spectroscopy classification time through the use of machine learning, following an algorithm developed collaboratively at OSU and Georgetown University. To do so, the team will use radiation counting data collected by an analogous detector set up at the Oregon State University Training, Research, Isotopes, General Atomics (TRIGA) reactor. Three different machine learning models will be trained and tested with this data to utilize their respective strengths. This document is covered under a Non-Disclosure Agreement (NDA) limiting access to its signers, the project's stakeholders, and OSU employees (including Teaching Assistants).

CONTENTS

1 INTRODUCTION

1.1 Purpose

The software will apply one or more machine learning algorithms to incoming radiation counting data during emission of radiation. The algorithm will process this data and attempt to determine when an optimal dosage of radiation has been reached. Once this occurs, or if a predetermined maximum threshold dosage is reached, the emitter will be signalled to stop or reduce radiation levels.

1.2 Scope

The first task required for the project will be to write software that manipulates existing data into a specific format. This software may leverage previous Python code done by OSU postgraduate students to perform the conversions. Specifically, the output data will be formatted for use by the Waikato Environment for Knowledge Analysis (Weka) software package. The goal of this task will be the visualization of data and machine learning algorithm performance.

The aforementioned data will contain radiation counting information to be used by machine learning algorithms. Usually, a spectrometer displays this information in real time using a graphical layout. Human operators are then able to visually identify which wave frequencies have a high occurrence of counts. This will appear as a peak in activity. Since a program will be doing this reading instead, the data is outputted in *list mode*. Instead of being displayed as a graph, data will be written to a file that contains an entry for every count occurrence on a wave frequency. A timestamp will also be included. Converting this data to the format used by Weka will allow us to utilize the software's visualization and statistical tools for implementing and evaluating our machine learning algorithms.

1.3 Product Overview

1.3.1 Product Perspective

The machine learning algorithm and relevant data will exist as part of a larger data analyzer unit. This will in turn constitute a single piece of a larger system. The system may vary depending on use cases, but can be categorized as either *active* or *passive*. Active instances will contain a radiation emitter while passive instances will not. The analyzer will receive radiation counting data in real time and will control the emitter if one is present. A display will also be connected to the data analyzer. This will provide feedback to operators during execution.

1.3.2 Product Functions

The product will provide an interface for the system operator to add parameters. When configured, the system must be able to start receiving data and return with real-time feedback on the imaging process using trained machine-learning models. When it determines it has reached the stopping point, it will signal the imagery machine to cease radiation emission and notify the operator of completion.

1.3.3 User Characteristics

Users of this system will be primarily spectroscopy experts who are evaluating the new technique. They may or may not have software development experience.

1.3.4 Limitations

The data available for machine learning is limited by the fidelity of the radiation detector. Because of the setup and safety precautions needed to test the system, the majority of the development and evaluation process will be virtual. Further work will be needed to verify findings and hone the solution for medical use.

Also, because machine learning algorithms are inherently based on probability, the program will not be 100% accurate. The project will begin with a required minimum certainty value of 95%. This may be lowered or raised based on experimental results. Ideally, a final product will make this configurable to fit the requirements of varying use cases.

Finally, the project will be limited by the simplistic methods of determining the presence or identification of radioactive elements. Certain materials may only be identified using highly sophisticated approaches. Due to the team's lack of knowledge in this field, these approaches will be outside the scope of the project.

2 SPECIFIC REQUIREMENTS

The system will process incoming radiation data using three different machine learning methods: naive Bayes, decision tree, and neural network. These methods have different performance characteristics and thus will be used as complements to arrive at a result. Also, implementing the different machine learning methods will help to figure out the pros and cons of each one, to aid arriving at a single best algorithm.

2.1 Usability Requirements

The system will be a proof of concept, so it does not need to cater as much to non-experts. It will need to be understood by persons familiar with spectroscopy. However, setting the system up should be made as simple as possible.

2.2 Code Quality

The underlying code that will be running this application must be maintainable for teams in the future to improve upon. This means that the code must be homogeneous and well documented. These qualities will allow any portion of code to be understandable just from looking at its source and to have the reason for its existence documented.

2.3 Flexibility

The system should account for future changes to the algorithms used. The use of a version control system should allow for flexibility in implementing different features, with the ability to roll back easily to previous methods if needed.

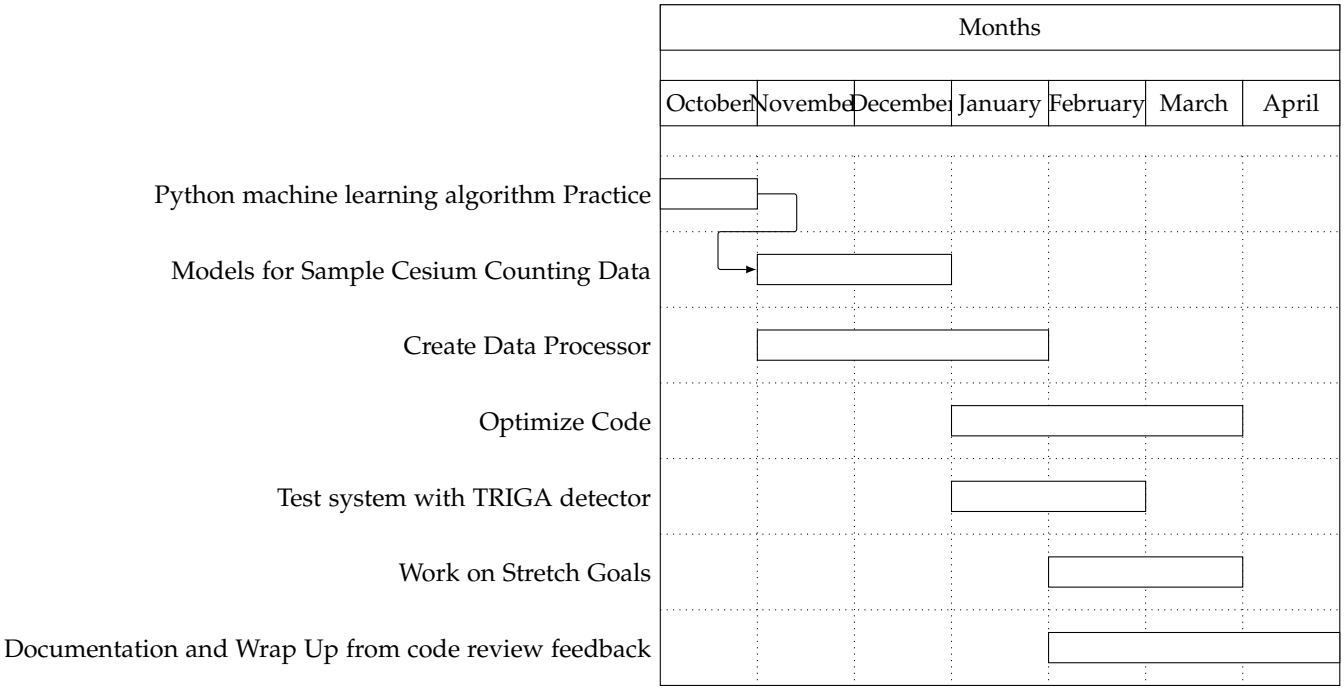
2.4 Safety

Because the system developed by this team will be a proof of concept, the main safety concern is ensuring use of standard procedures when collecting radiation data, as determined by experts at the OSU Radiation Center. The users will be trained, and need to understand how to interact with the radiation sources used, safely. In the long run, derivative/continued works will need to match or exceed the safety of current diagnostic imagery systems. In order to maintain this baseline, the system will have constraints on its analysis. If the analysis fails entirely, the system reverts back to a module implementing current radiological counting methods. If the analysis suggests continuing past a point suggested by the module implementing current counting methods, the system will require user input and only continue to a pre-configured maximum dosage.

2.5 Performance Requirements

In the long run, the system will need to support one user: the radiologist controlling the imagery process. Once the model is trained, the processing needs to be real-time in order to provide continuous feedback on the progress of the imagery. As the spectrometer produces usable data on the millisecond scale, processing time should be as close to that as possible.

3 GANTT CHART



3 DESIGN DOCUMENT

CS CAPSTONE DESIGN DOCUMENT

DECEMBER 4, 2019

REDUCING PATIENT DOSE FROM DIAGNOSTIC IMAGING USING MACHINE LEARNING

PREPARED FOR

OREGON STATE UNIVERSITY

DR. STEVE REESE _____ _____

Signature *Date*

PREPARED BY

RADIOLOGICAL COUNTING

IAN BROWN _____ _____

Signature *Date*

SEAN GILLEN _____ _____

Signature *Date*

YIHONG LIU _____ _____

Signature *Date*

Abstract

The capstone team will create a system to arrive at a spectroscopy classification result faster than conventional spectrometers through the use of real-time processing of radiation data. The system will be controlled by a manager module that receives data from the spectrometer and sends it to three trained machine learning models, which prompt for more data until a confidence value is reached. The main focus will be to develop the accuracy and speed of the system enough for it to be a compelling demonstration of the algorithm described in a 2019 patent by OSU and Georgetown University researchers. This document is covered under a Non-Disclosure Agreement (NDA) limiting access to its signers, the project's stakeholders, and OSU employees (including Teaching Assistants).

CONTENTS

1 Introduction

1.1 Purpose

This design document details the technologies to be used during the development of the machine learning algorithms as well as the rationale behind each decision. After reading this document, a developer should know what components are needed for the project as well as the tools that can be used to implement the solution.

1.2 Scope

The scope of this document includes the design decisions and rationale for each aspect of the machine learning project. For each aspect, the desired functionality will be summarized and the methods, techniques, and/or tools that the team has decided to use to achieve said functionality will be discussed.

1.3 Intended Audience

This document is intended for the project client and the senior capstone advising team. Both the client and capstone advisors will use this document to examine and analyze the project team's design choices as well as monitor their progress during development for the whole year.

2 Glossary

- ADC: Analog-to-Digital Converter. An ADC is a piece of hardware that converts analog signals such as sound waves into digital signals. This allows data to be used by software. The ADC used in this project converts a spectrum of radiation counts into a discrete number of counts for every energy channel.
- API: Application Programming Interface. This describes the functions a developer can use with a third-party tool.
- C4.5, C5.0: Decision tree algorithms.
- Pandas: data processing library for the Python programming language.
- Python: interpreted programming language developed in the mid-1990s. It is known for enabling fast prototyping, and today has a large number of libraries available for data processing and machine learning.
- Tensorflow: math library developed by Google which can be used to apply machine learning models, such as a neural network.
- TRIGA: Training, Research, Isotopes, General Atomics. A class of nuclear research reactor designed and manufactured by General Atomics. One such reactor is present in OSU's radiation center, and equipment used in the reactor bay will be used by this project.
- Weka: Waikato Environment for Knowledge Analysis. An open-source graphical tool enabling quick application of different machine learning algorithms.

3 Body

3.1 Stakeholders

The main stakeholders for this project are the authors of the patent it falls under. They are:

- Dr. Steve Reese, Oregon State University

- Dr. Ophir Frieder, Georgetown University
- Jessica Curtis, Oregon State University

3.2 Design Viewpoints

3.2.1 Context viewpoint

The system will take radiation counting data from an ADC and return a classification to the user's display. The system's user will be one of the project's stakeholders.

System black box components:

- ADC and equipment it interfaces with on its own
- Computer receiving ADC data
- Computer display/application

The system requires other components to operate, such as the radiation detector itself, and radiation source. These will be ignored for the most part in this design, except for how they affect the ADC's operation and output. This is because the system is only meant to change the classification stage, which happens entirely from ADC data.

The user will start the system by opening the application on a computer in the OSU TRIGA reactor bay. The computer will be connected to the ADC, allowing it to receive data. The user will activate the ADC and the external systems to which the ADC connects in order to start the collection of radiation counts. The application developed will detect the start of output from the ADC, and begin the process of classification.

During the classification process, the system will display real-time outputs from the machine learning models with respective confidence values. Once the confidence values reach a user-set value, the application will send a message to the user. As this version of the system will not control the ADC or radiation source, counting will continue until the user switches off the ADC module or this application, or the ADC reaches its preset time threshold.

3.2.2 Composition viewpoint

The system consists of an ADC, serializer, manager, classifier, and some sort of display. The ADC takes signals from a spectrometer and outputs a file containing data on the counts it performed. The serializer then opens this file and interprets its contents. Then, it constructs a list of data structures that represent this information. The manager will invoke one of three previously-constructed classifiers and feed the list of data from the serializer into it. The classifier itself will take in a data set and attempt to classify it as a specific radioactive material. If the accuracy of the classification is high enough, the manager will stop execution and output the result to one or more displays and/or files. If the accuracy is not high enough, the manager will request more data from the serializer.

While the interval between requests for more data and the threshold for classifier accuracy should be configurable, this project will use 10ms and 95% respectively. Additionally, any reasonable machine learning classifier could work in this system. However, this project will limit these to decision tree, naive Bayes, and neural network.

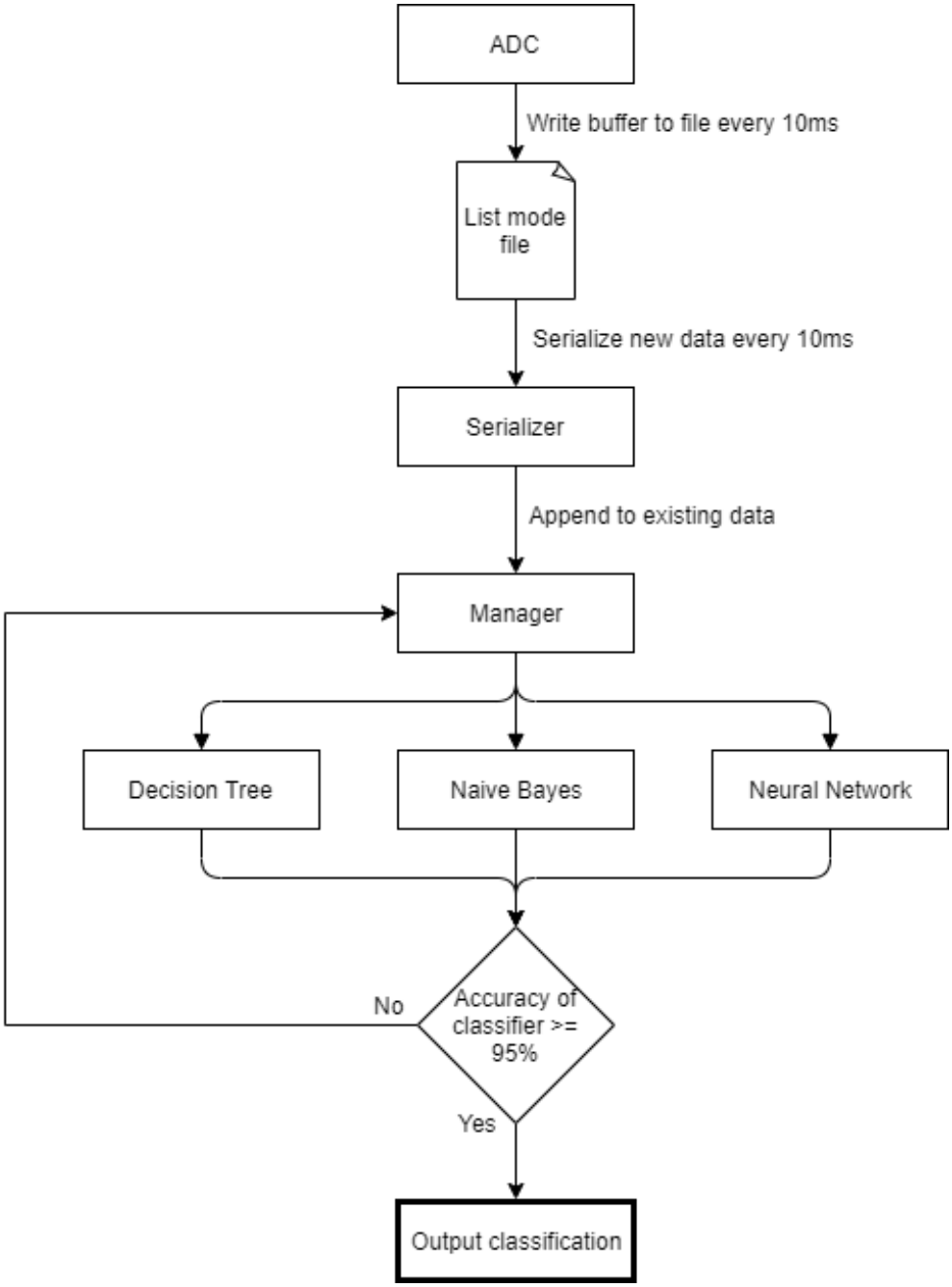


Fig. 1. System diagram

3.2.3 Interaction viewpoint

- `Serializer`
  - `get_data`: Opens the list mode file created by the ADC, serializes the contents to a list of objects, and returns this list. This method will take a byte offset parameter and also return a byte offset. This decision was made so that serialization can be performed incrementally. Every time the method is called, it will start at a location in the file given by the offset parameter, read data for a specified file length, and then

return the serialized data and byte offset at the location of the file in which it stopped. The file name can be parameterized or hardcoded for a slight performance increase if possible.

- `Manager`
  - `main`: The entrypoint of the system. Execution flow will remain here until a classification has been reached with sufficient accuracy. Command line arguments can be passed for configuration changes.
  - `append_data`: This method will perform the call to `Serializer.get_data`. It will also handle updating the working data set with the new data. The byte offset returned by the serializer will also need to be stored in a variable. Then, this can be passed in to `Serializer.get_data` to continue where it left off from the previous call. `Manager.main` will call this method with high frequency, so performance will be important here.
  - `check_classification`: Calls `Classifier.classify` and handles the result. After this call, the accuracy will be compared to the configured threshold. If the accuracy is less than the threshold, control will be passed back to `main` and the loop will continue. If the accuracy is greater than or equal to the threshold, the classification will be passed to one or more external systems, such as a display. A file may also be generated with some results about the overall execution of the system.
- `Classifier`
  - `classify`: Runs classifier on a provided data set. The result of this function should be a classification and a probability representing the estimated accuracy of the classification.

3.3 Design Views

3.3.1 Context view

This system sits between the ADC and the user, so the black box components are only those for the classification part of the process.

3.3.2 Composition view

Since our project will be implemented in Python, the composition was based on the concept of distinct modules. In this context, modules are analogous to classes in the object-oriented programming paradigm. Also, the ADC component and choice of classifiers in the classifier component were previously decided by the patent that the project is based on.

3.3.3 Interaction view

The classifier module will exist as a third-party Python package. Because of this, it is represented as having one main method. The serializer module also has one method, although this will be developed by the group. This is due to the serializer being a fairly simple module. Finally, the idea of the manager is that it will function as a coordinator for the other two modules. It will not export any methods, but it will be the main entrypoint of the system. This module will also be where most of the data being generated resides during runtime.

3.4 Design Rationale

3.4.1 System Interface

Because the main purpose of the project is a proof-of-concept for an algorithm, the interface will not likely be used much by those outside of the development/stakeholder group.

3.4.2 Main Development Language

The group chose to develop the project using the Python programming language. The main benefit to this choice is the simplicity and readability of Python code over other popular languages. Also, since the main goal of the project is to function as a proof of concept, other factors such as performance play less of a part, as long as the code can be understood by the stakeholders. Python also supports several packages such as scikit-learn, TensorFlow, and PyTorch which perform well for applications of machine learning.

4 Tech Review

4.1 Tech Review - Ian Brown

Oregon State University
College of Engineering

CS Capstone Technology Review

May 28, 2020

Reducing Patient Dose from Diagnostic Imaging Using Machine Learning

Prepared for

Oregon State University

Steve Reese

Prepared by

Group 16

Radiological Counting

Ian Brown

Abstract

Several factors are necessary to consider when making decisions on the technologies to use for each part of the project. While all Naive Bayes classifiers function by using Bayes' Theorem, there are variants that behave slightly different depending on the type of data being used. The main characteristic of each of these variants is how data with continuous values are handled. When considering the programming language that the application will be implemented in, metrics such as performance, simplicity, portability, and package selection all need to be considered. In this case, some metrics matter more than others. A choice can be made by gauging the most valuable ones and how each of the languages under consideration perform in them.

CONTENTS

1 NAIVE BAYES CLASSIFIER VARIANTS

One of the machine learning classifier types required to be implemented in this project is Naive Bayes. This is a probabilistic classifier, meaning it is relatively simple to implement and fast to execute. Several variants of Naive Bayes classifiers exist. For this project, we will consider Gaussian, Multinomial, and Complement Naive Bayes classifiers.

Bayes' Theorem, from which the family of Naive Bayes classifiers are built on, is fairly straightforward to apply when the values of each feature belong to one or more discrete classes (e.g. "red", "green", "yellow"). Since there is just one method of applying the rule to these types of features, each variant will behave roughly in the same manner if all data appears like this. However, it is quite common for models to have features that are continuous, or "real". An example of this could be a *temperature* feature whose value is a number representing degrees Farenheit. Due to the infinitely possible values for this feature, it must be accounted for in some other way in order to apply Bayes' Theorem. The main difference between each of the Naive Bayes variants is the method in which they handle (or do not handle) real-valued features.

1.1 Gaussian Naive Bayes Classifier

The Gaussian variant is usually the most common choice when values of features are continuous [1]. It works by assuming that each feature has a normal distribution. In other words, the method assumes that the feature's values appear to fit a symmetric bell curve.

1.2 Multinomial Naive Bayes Classifier

While the Gaussian classifier explicitly handles continuous values, the multinomial variant does not. Instead, values are assumed to be discrete, meaning there must be a finite number of classes in which any value of a feature belongs to. This assumption makes the multinomial variant especially well-suited for use with classifying sequences of words [2]. For example, by taking each word in a document such as an email to be a feature, the algorithm can classify the document as spam or not spam. Despite its inability to handle continuous features, the variant can still be applied using a discretization method described later.

1.3 Complement Naive Bayes Classifier

A less popular Naive Bayes variant is known as Complement Naive Bayes (CNB). Derived from the multinomial variant, CNB is especially useful when the values of features in training data sets appear to be imbalanced [2]. Further, this particular case of better performance has been proved empirically by the creators of CNB [2]. It has been noted that the CNB also often outperforms the multinomial variant in general for applications like text classification [3].

1.4 Comparison of Options

As previously stated, Naive Bayes variants differ from each other by their capacity to account for real-valued features. Therefore, the choice of variant hinges on how each of the features will be represented in the model.

1.4.1 All Discrete Features

If all of our features are discrete, any variant may be directly applied to our data set. However, the main benefit of the Gaussian variant in simplifying the distribution of continuous values will no longer be applicable, so it should

not be considered for this case. In choosing between the multinomial and complement variants, the aforementioned evidence supporting the complement's tendency to outperform the multinomial seems promising. Because of this, the Complement Naive Bayes classifier will be used if features are to be represented as discreate classes.

1.4.2 At Least One Continuous Feature

If one or more features must be represented as continuous values, we could either use the Gaussian classifier directly or first discretize our features and apply either the multinomial or complement variant. This discretization would be accomplished by partitioning the data set into a number of equal-size subsets. For example, continuing from the *temperature* feature example, subsets could include $<60°$, 60-70°, 70-80°and $>80°$. While this discretization process would work, it would be detrimental to our application for a few reasons.

First, adding a discretization step will result in extra processing time needed to carry out calculations. Instead of loading sample data directly as training data, we would need an extra step of splitting up this data into each subset. The second issue is the choice of how many subsets should be used. If the value is too low, the classifier will not be accurate enough. If too few subsets are used, the final classification might be correct, but may also be unhelpful due to a large range. Alternatively, if the value is too high, the performance impact could be too large due to maintaining many data structures to classify each data entry. Because of these reasons, it would make more sense to use the Gaussian variant if features are continuous

It is the group's understanding that the spectrometer used to generate radiation counting data uses a discrete number of channels. For this reason, it can be assumed that the feature set will be discrete and therefore, the Complement Naive Bayes classifier will be chosen. If the implementation requires at least one continuous-value feature, the Gaussian variant will be chosen instead.

2 Programming Language for Implementation

One of the more fundamental decisions for this project is the choice of programming language with which to implement our application. This choice will be made based on several factors such as performance, simplicity, portability, and selection of relevant packages. Choices for this topic have been limited to Python, Java, and R.

2.1 Python

Python is currently one of the most popular programming languages in use [4]. The main draw of this language is its simple syntax and abundance of packages. This makes the language very accessible, allowing those in fields other than computer science to easily write code to accomplish tasks suited to their needs. In many cases, the same application written in other traditional programming languages could be done so in Python using a fraction of the lines of code.

This simplification comes at a cost, however. Python is an interpreted language, as opposed to a compiled language. This means that each instruction is interpreted at run time, resulting in slower execution time [5]. Since our project will need to operate in real time, performance will be important.

2.2 Java

Even more popular than Python is the Java programming language [4]. As opposed to Python and R, Java is compiled with the use of a Just-In-Time (JIT) compiler. This means that Java code is first compiled into an intermediate representation at compile time [6]. At run time, this representation is compiled further into machine code and executed [6]. The result is faster execution time compared to interpreted languages.

The inclusion of Java as a choice was done in part to the fact that the Waikato Environment for Knowledge Analysis (Weka) is written in this language [7]. If our application were to be developed in Java, this would give us the benefit of leveraging Weka's machine learning libraries in our own code [7]. Our client has stated that part of this project will be to use Weka for data analysis and visualization. Since this requirement will ensure that the team gains some degree of familiarity with Weka regardless, it may be simpler to also include it in Java code as well, resulting in less time spent researching additional packages.

2.3 R

As far as the most popular machine learning programming languages go, R has consistently ranked first for some time now [8]. Among the reasons for this is that R offers many useful built-in methods [9]. These methods continue to grow as many researchers develop and maintain them [9]. In the performance category, it is an interpreted language like Python giving it comparable results.

The main drawback to using R for this project is that the team is not as experienced in it as the other two options. Apart from this, it shares similar pros and cons to choosing Python.

2.4 Comparison of Options

The first metric that should be discussed for these language choices is performance. As previously stated, the three choices can be grouped into two categories: interpreted or compiled. Python and R are both interpreted languages while Java is compiled, giving Java the benefit of faster run time over the others.

Next, the aspect of simplicity should be considered. Java is ranked last in this category due to the fact that static typing is required and the more complicated syntax results in less readability. Python and R are somewhat similar in syntax, but Python appears more intuitive making it the winner here. The reasoning behind this is that R requires more built-in calls to create or manipulate data while Python implements data as objects which each have their own simple methods for accomplishing similar goals.

The portability of each of the possible choices is another important aspect to consider. Portability here means how easy it is to run code on different machines. Java utilizes the Java Virtual Machine (JVM) to compile bytecode so that, in theory, applications can be successfully run on any operating system that has the JVM installed [10]. Further, the Python and R interpreters are both written in the C programming language. This means that the host machine must have a proper C compiler or have pre-built binaries specific to their operating system in order to run the interpreter [10].

All languages offer a good selection of machine learning packages. Among the most popular for each language include scikit-learn for Python, Weka for Java, and e1071 for R. Due to additional popular packages such as TensorFlow and

PyTorch for deep learning classifiers, Python is preferred for this category. However, it is important to note that only our neural network implementation could leverage any deep learning packages, so this distinction is not as important.

For this project, we were given relevant code from another application. Since this code was written in Python, our client suggested that we also implement our application in Python so we can use this existing code. Also, since this project is intended to be a proof-of-concept, issues such as portability and performance are less important. Instead, the team is mostly concerned with the clarity of the code. The reasoning behind this is that if the code is simple and readable, future teams will be able to understand the general concepts easily. Then, performance-sensitive implementations could be done using the team's application as a reference. It can be concluded from these reasons that Python should be the programming language of choice for this project.

REFERENCES

[1] J. Brownlee. (Apr. 10, 2016). Naive bayes for machine learning, Machine Learning Mastery, [Online]. Available: https://machinelearningmastery.com/naive-bayes-for-machine-learning/ (visited on 11/09/2019).

[2] 1.9. naive bayes — scikit-learn 0.21.3 documentation, [Online]. Available: https://scikit-learn.org/stable/modules/naive_bayes.html (visited on 11/09/2019).

[3] J. Rennie, L. Shih, J. Teevan, and D. Karger, "Tackling the poor assumptions of naive bayes text classifiers," *Proceedings of the Twentieth International Conference on Machine Learning*, vol. 41, 2003.

[4] M. Priyadarshini. (Aug. 23, 2019). 10 most popular programming languages in 2019: Learn to code, Fossbytes, [Online]. Available: https://fossbytes.com/most-popular-programming-languages/ (visited on 11/09/2019).

[5] (Jul. 15, 2018). Why is python so slow? [Online]. Available: https://hackernoon.com/why-is-python-so-slow-e5074b6fe55b (visited on 11/09/2019).

[6] (Jun. 27, 2019). JIT in java: Understanding just-in-time compiler, Edureka, [Online]. Available: https://www.edureka.co/blog/just-in-time-compiler/ (visited on 11/09/2019).

[7] Use weka in your java code - weka wiki, [Online]. Available: https://waikato.github.io/weka-wiki/use_weka_in_your_java_code/ (visited on 11/09/2019).

[8] J. Brownlee. (May 9, 2014). Best programming language for machine learning, Machine Learning Mastery, [Online]. Available: https://machinelearningmastery.com/best-programming-language-for-machine-learning/ (visited on 11/09/2019).

[9] ——, (Jan. 14, 2016). Use r for machine learning, Machine Learning Mastery, [Online]. Available: https://machinelearningmastery.com/use-r-for-machine-learning/ (visited on 11/09/2019).

[10] Which is more portable - java or python? - quora, [Online]. Available: https://www.quora.com/Which-is-more-portable-Java-or-Python (visited on 11/09/2019).

4.2 Tech Review - Sean Gillen

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.

4.3 Tech Review - Yihong Liu

CS CAPSTONE TECHNOLOGY REVIEW

NOVEMBER 3,2019

REDUCING PATIENT DOSE FROM DIAGNOSTIC IMAGING USING MACHINE LEARNING

PREPARED FOR

OREGON STATE UNIVERSITY

STEVE REESE

PREPARED BY

GROUP 16

YIHONG LIU

Abstract

This report introduces the possible technology to implement the Linear Regression algorithms for our project, with different models, and the detailed solution about how to implement the algorithms step by step, and the visualization of data. Then summarizing the best suitable possible technology that can be used in the project.

CONTENTS

1 Introduction

The technology review examines three components of the ”Reducing Patient Dose from Diagnostic Imaging Using Machine Learning” project. For each component, three alternatives are explored for possible use in the project and each alternative will be evaluated based on their specifications.

2 Problem Description

Diagnostic imagery is a useful tool for viewing internal bodily structures. Radiation is required in order to create an image with high enough saturation that any objects of interest are clearly visible by humans. If the dosage is not enough, these objects won't be visible due to low saturation. However, the dosage also needs to be limited in order to avoid harmful side effects in the patient. If a doctor orders a diagnostic scan, they believe that the potential findings outweigh the risk imposed by additional radiation exposure.

Current diagnostic imagery tools emit radiation until an exposure threshold has been reached as detected by a radiation detector. This leads to needless exposure because a point is reached where additional radiation will not yield satisfactorily better imagery. The high likelihood of generating an image with enough quality comes at the expense of excess irradiation of the patient. Current tools are not sophisticated enough to know the optimal dosage with enough accuracy. Also, since the process happens so quickly, it wouldn't be feasible for a person to monitor the image quality and stop the machine manually. Finding a way to stop the imagery process earlier will in theory yield the same results with less radiation exposed to the patient.

3 Piece 1: Linear Regression Machine Learning Model

3.1 Overview of Criteria

The Linear Regression is one of the most important machine-learning algorithms in our project, regression analysis is a set of statistical processes fro estimating the relationships among variable, the focus is on the relationship between a dependent variable and one or more independent variables, in simple words, we want to predict y(the dependent variable, or target) based on a set of x's(dependent variables, or features), where y is continuous. with this algorithms, the system can train the data set with features to, and correct the machine learning model as the data goes wrong or inaccurately, then after optimizing the model, the machine learning model can automatically detect the peak point (target variable) of the specific images of radiations, which the system can know the right point to stop the unnecessary does of radiation, to make the patient suffer less pain.

3.2 Simple Linear Regression

With simple linear regression when we have a single input, we can use statistics to estimate the coefficients.

This requires that you calculate statistical properties from the data such as means, standard deviations, correlations and co-variance. All of the data must be available to traverse and calculate statistics.

3.3 Ordinary Least Squares

When we have more than one input we can use Ordinary Least Squares to estimate the values of the coefficients.

The Ordinary Least Squares procedure seeks to minimize the sum of the squared residuals. This means that given a regression line through the data we calculate the distance from each data point to the regression line, square it, and sum all of the squared errors together. This is the quantity that ordinary least squares seeks to minimize.

This approach treats the data as a matrix and uses linear algebra operations to estimate the optimal values for the coefficients. It means that all of the data must be available and you must have enough memory to fit the data and perform matrix operations.

3.4 Gradient Descent

When there are one or more inputs you can use a process of optimizing the values of the coefficients by iteratively minimizing the error of the model on your training data.

This operation is called Gradient Descent and works by starting with random values for each coefficient. The sum of the squared errors are calculated for each pair of input and output values. A learning rate is used as a scale factor and the coefficients are updated in the direction towards minimizing the error. The process is repeated until a minimum sum squared error is achieved or no further improvement is possible.

When using this method, the system must select a learning rate (alpha) parameter that determines the size of the improvement step to take on each iteration of the procedure.

it is useful when there is a very large data set either in the number of rows or the number of columns that may not fit into memory. [1]

4 Piece 2: The implementation of Linger Regression machine learning model

4.1 Overview of Criteria

The implementation of Linger Regression machine learning model describes the detailed (but still in a high level) design about how to achieve our goal with the algorithms.

4.2 Reading the data set file

Since the whole project is based on Python programming language, the Pandas is the one of the most suitable and necessary tool to read the data file. Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. the process of reading data can be viewed as the way to collect data into our project, it can be achieved with something sudo code like this ” example = data.values[0:,1:]”.

4.3 Processing the data with python

Processing data is the way to correct and manipulate the data, one of the common utilization is feature extraction. For example, if clients only want to figure the correlation between size of house and the price of the house, then just extracting the ”size of house” only from bunch of variables, such as area, decoration, etc. except that, there are some method can be used to process data set, such as Normalization, one-hot encoding, regularization. The goal of normalization is to change the values of numeric columns in the dataset ti a common scale, without distorting differences in the ranges of values. The one-hot encoding representing each piece of data in a way that the computer can understand, which is a process by which categorical variables are converted into a from that could be accepted by computer.

Regularization is the process of adding information in order to solve an ill-popsed or to prevent over fitting. The major reason that Regularization is necessary is that the model will have a low accuracy it it is over fitting, because the model is trying too hard to capture the noised in the training dataset. This technique discourages learning a more complex or flexible model, so as to avoid the risk of over fitting. With the Scikit-learn, it provides the class linear Regression classifier, which can implement many processing function without writing any code extra. [2] [3]

4.4 evaluating the performance of the model

The Root mean squared error (RMSE) and coefficient of determination (R square score) will be the main method being used to determine the accuracy. RMSE is the square root of the average of the sum of the squares of residuals. R square score or the coefficient of determination explains how much the total variance of the dependent variable can be reduced by using the least square regression. Those value can tell whether the linger regression model achieves the accuracy requirements or not. [4]

5 Piece 3: visualization of data

5.1 WEKA

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 also contains tools for data pre-processing, classification, regression.

5.2 R

R is a programming language and software environment for statistical computing and graphic supported by R foundation for statistical computing, R is widely used among data miners and statisticians for developing statistical software and data analysis. The way R code implement in our project is for analyzing the data set, it has extensive visualization capabilities.

6 Recommendations

For the piece 1, those are some potential regression models, but previous two can not handle the dataset with too many dependent variables, so Gradient Descent will be the best choice for our project. For the piece 2, there is no recommendations, it's just all the detailed design about how to achieve implementing the linear regression model step by step. For piece 3, WEKA might be better than R in our project, although we might need to use both of R and WEKA, but WEKA can Handel most of algorithms that we need to implement, it has more selection, more packages, than R. [5]

References

[1] J. Brownlee, "Linear regression for machine learning." [Online]. Available: https://machinelearningmastery.com/linear-regression-for-machine-learning/

[2] P. Gupta, "Regularization in machine learning." [Online]. Available: https://towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a

[3] A. AI-Masri, "How does linear regression actually work?" [Online]. Available: https://towardsdatascience.com/how-does-linear-regression-actually-work-3297021970dd

[4] A. Agarwal, “Linear regression using python.” [Online]. Available: https://towardsdatascience.com/linear-regression-using-python-b136c91bf0a2

[5] J. Brownlee, “A gentle introduction to scikit-learn: A python machine learning library.” [Online]. Available: https://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/

5 Weekly Blog Posts

5.1 Ian Brown's Blog Posts

Ian Brown
CS 461
18 October 2019

Individual Progress, Problems, and Plans Blog Week 3

Progress: My team met with our client for the first time last week. We discussed broad, high-level details about the project. At the time, we had not signed the required Non-disclosure Agreement (NDA) so we were not able to go into much detail. This week, we all signed our NDAs and scheduled our next meeting for next week. We also met with our TA for the first time and went over his expectations for our future group work.

Problems: The biggest problem we have been facing recently is lack of information. We still know very little about our project which has made the Problem Statement and Requirements Document assignments somewhat difficult. We also still don't know the restrictions on our writing due to our NDA. Our group is the first group that our TA has worked with that was required to sign an NDA, so we weren't able to get much information related to this topic from our meeting with him.

Plans: We plan to meet with our client next week to go over more detailed requirements. We also hope to clarify questions about our NDA. We talked with our TA about potentially having him sign an NDA as well so we will be able to discuss our project in more detail. Following our next client meeting, we also plan to start work on the project.

Ian Brown
CS 461
25 October 2019

Individual Progress, Problems, and Plans Blog Week 4

Progress: My team met with our client for the second time and obtained more information about the requirements for the project. We also clarified the restrictions on the NDA. Since it only requires that information stay within OSU, we will be able to include details in our assignments related to the project.

Problems: Our current problems are currently lack of experience with Weka and other machine learning concepts. We also don't yet have any sample data to work with. This is something we will have to wait until next week for.

Plans: All of us plan to gain some experience with Weka. Also, we will each specialize in one of the machine learning approaches. We will research our specific approach and think about how we could create an implementation. My team will meet with our client next week to gather sample data.

Ian Brown
CS 461
1 November 2019

Individual Progress, Problems, and Plans Blog Week 5

Progress: Over the last week, I have mostly been working on learning Weka through online tutorials. My team met with our client again as well as two other people related to the project. We agreed that we would create a architecture diagram of our software outlining inputs and outputs. This document will be due next Wednesday.

Problems: We are still waiting for sample data from our client. Hopefully, this will be available on Monday, but may take longer. This will limit what we are able to work on in the meantime, but should not be too detrimental. We are still unsure about how our project will be implemented due to the constraint that our machine learning algorithm will need to operate in real time. This appears to be a less common case as opposed to creating classifiers beforehand. Hopefully, we will figure this out when drafting our diagram.

Plans: Right now, our priority is to work on our tech review document and architecture design diagram for our client. Once we have sample data, we will be able to do some testing using Weka and machine learning packages.

Ian Brown
CS 461
8 November 2019

Individual Progress, Problems, and Plans Blog Week 6

Progress: On Wednesday, my team met with our client to receive our first sample of data. This will allow us to get familiar with the data format. Knowing this format is necessary to begin any work on our serializer component.

Problems: The sample file contains counting information about a single element that is not easy to classify. The purpose of this was just to get familiar with the data format, without attempting to create a classifier. In order to progress any further, we will need more data on elements that can be classified. This will be the next step.

Plans: My team plans to look into the sample file and start working on some simple code that can parse the file into some sort of data structure. We will also look into our git flow for collaborating. Since the client expressed concerns about using cloud storage, we will clarify this with him and discuss further if necessary.

Ian Brown
CS 461
15 November 2019

Individual Progress, Problems, and Plans Blog Week 7

Progress: After receiving Python code and a sample of radiation counting data, our team created a simple program that plots the data after serializing it to objects. We also created a private GitHub repository for hosting our code.

Problems: As with the last blog, our next challenge will be waiting for more helpful sample data so that we can begin building and testing our classifiers.

Plans: Each of us will need to get more famililar with the existing code and data format of the counting files. We will also need to decide on a specific schema for the objects that will hold this data. This will ensure that our classifiers know what data to expect.

Ian Brown
CS 461
22 November 2019

Individual Progress, Problems, and Plans Blog Week 8

Progress: Due to other assignments this week, not as much work was done for the project. I created code in our GitHub repository that enforces code quality and formatting guidelines by using a linter. I also fixed remaining formatting issues in our code.

Problems: As with the last blog, our next challenge will be waiting for more helpful sample data so that we can begin building and testing our classifiers.

Plans: The team plans to continue working on understanding and improving the code. We will begin implementing classifiers once we have adequate sample data.

Ian Brown
CS 461
29 November 2019

Individual Progress, Problems, and Plans Blog Week 9

Progress: Due to the holiday break and the client's schedule, the group was not able to make much progress this week. We still require more data to begin creating classifiers. We won't have this until our next meeting.

Problems: The group is still waiting for more helpful sample data so that we can begin building and testing our classifiers.

Plans: Once the group has received better data, we can begin working on the classifiers.

Ian Brown
CS 462
9 January 2020

Week 1 Progress Report

Progress: Over the break and first week, the group has read over selected sections of a radiation detection textbook to become more familiar with gamma spectroscopy. This will help us when creating our machine learning implementation later.

Problems: Currently, the main problem that remains is receiving more data for use in our development. Also, the group still needs more experience with the machine learning packages that we will be using.

Plans: The group will meet on Friday to collect more data. This will be our first meeting with our client since the winter break. We hope to discuss our future plans for development with our client.

Ian Brown
CS 462
17 January 2020

Week 2 Progress Report

Progress: The group has performed energy calibrations and efficiency calibrations of the high-purity germanium detector using a few sample materials. We have also written code to parse the binary file emitted by the detector software. For visualizing the data and performing basic analyses, we have compiled and use OpenGammaX and made progress with compiling other software sent to us by the client.

Problems: The main problem that the group faces is getting the ListPro software to compile. After this, we will need to modify the software to allow shared read access when creating the output file rather than locking it. This will allow our application to eventually read in the file as it is being written to by the software.

Plans: The group plans to continue attempting to get the ListPro software working and eventually modified to suit our needs. This will then be tested in the laboratory with an actual detector. After this, we will begin building the machine learning models.

Ian Brown
CS 462
24 January 2020

Week 3 Progress Report

Progress: The previous energy calibration was slightly off due to a data entry error. The points were corrected and the regression equation was recalculated yielding much higher accuracy. The ListPRO software was successfully edited to allow read access sharing on the open file and compiled by Sean. However, when we attempted to run the executable on another computer, some errors related to missing libraries on the computer prevented the program from running. During this meeting with the client, we also gathered additional spectra files.

Problems: Figuring out the library-related issues will be an issue although it shouldn't be much of a blocker. From our application's perspective, a static file will be handled in much the same way as a file being written to in real time.

Plans: After meeting with our client, the group agreed to produce a short summary detailing our progress so far and our plans for implementing our machine learning model. This will be presented in the next meeting including another professor from Georgetown University who will be on a conference call. We will receive feedback on our plans which will be helpful when beginning the actual implementation.

Ian Brown
CS 462
7 February 2020

Week 5 Progress Report

Progress: The group has created a short summary detailing our current progress and plans for our machine learning implementation. We have also investigated multi-label classification. Fortunately, the package we are using for our implementation supports this natively.

Plans: We will need to manipulate our data into the right format before using the multi-label classifier. Then, we will want to train using the mix of samples that we have taken so far and evaluate the model's performance. If the accuracy isn't satisfactory, we will look more into adding some preprocessing to account for this.

Problems: If the accuracy of the multi-label classifier is not very good, adding preprocessing may be difficult. This is because we will need to account for the branching factors of the nuclides. This will involve converting channels to energy levels and applying some sort of weights to the features. Hopefully, this won't be necessary.

Ian Brown
CS 462
14 February 2020

Week 6 Progress Report

Progress: After some work, I was able to implement the multi-label classification. The accuracy was very poor when continuing to normalize the data but after removing this, the model is extremely accurate for our small dataset. I also refactored code to make it easier to add or remove test files to train the model.

Plans: Next, we will need to test our application in the lab on new data. We will also need to implement the real-time file parsing behavior. Our performance estimations predict that our application will perform at a significantly faster rate than the rate of data being written to the file. This may change slightly depending on if the machine is using a hard drive or SSD. In any case, we will need to handle the logic for waiting for new data from the radiation detector.

Problems: No problems at this time.

Ian Brown
CS 462
21 February 2020

Week 7 Progress Report

Progress: My team was mostly preoccupied with working on the slides for our design review which was presented on Tuesday. I also created some new issues in GitHub along with a new milestone for our beta release.

Plans: Our next steps will be to set up a meeting with our client. We plan on going over our progress and the contents of our design review with him. Hopefully, we will be able to meet with the Georgetown University professor to discuss our machine learning models. We will also need to work on improving the accuracy of the decision tree and neural network models.

Problems: Depending on the client's and other professor's schedules, it may take some time to get these meetings set up. Another issue will be tuning our neural network and decision tree models since we will need to experiment with data preprocessing.

Ian Brown
CS 462
28 February 2020

Week 8 Progress Report

Progress: This week, I have contacted the client and scheduled a meeting for next Friday. I have also began reviewing changes for adding the file watcher.

Plans: More review is needed for the changes. Then, I'll work on creating some persistence for the machine learning models. This will most likely be done by assigning names to the trained models and checking for existing models with the same name to use rather than training new ones.

Problems: The main issue now is improving the accuracy of the neural network and decision tree models.

Ian Brown
CS 462
6 March 2020

Week 9 Progress Report

Progress: I finished reviewing my teammates changes to add file-watcher functionality. We also met with our client and discussed our progress.

Plans: I plan to mostly just work on implementing model persistence. There don't appear to be many other issues we'll need to work on at the moment. Our client also requested us to create another short summary detailing our progress to share with the other professor and former student we have been working with.

Problems: The only real problems remain to be how much we can improve the accuracy in the decision tree and neural network models. We will get some input from the machine learning professor about this.

Ian Brown
CS 462
13 March 2020

Week 10 Progress Report

Progress: We met with our client again along with the professor from Georgetown university and the former PhD student. They were both satisfied with our progress and think that the project is mostly done. We discussed the model performance and our next steps.

Plans: Our plans are to meet after spring break and compare the model's predictions against the GammaVision software which uses peak-finding algorithms to indicate if isotopes are present. We will take several samples at different time lengths and find when the earliest time is that GammaVision can correctly identify isotope presence. If our model can do this faster, we will have satisfied the main criteria of the project.

Problems: The decision tree and neural network models continue to be slow. The client does not view this as much of an issue, as the machine learning professor confirmed that this performance is expected. We will use the Naive Bayes model primarily unless we can improve the speed of the other two models.

5.2 Sean Gillen's Blog Posts

Fall: Week 4	This week has had the most progress of any so far. Since we signed the NDA the Thursday before this one, we have had the chance to meet Dr. Reese and receive details about what we will be doing, and make plans. Our main NDA problem is out of the way. However, I appear to be the least versed in the machine learning techniques we'll be using out of my group members, so I will need to spend more time learning them so as to not hinder our progress. We plan to learn the WEKA machine learning software using online tutorials, and then start trying to apply it with sample data that we will receive from Dr. Reese at upcoming meetings.
Fall: Week 5	Progress Introduced to the two remote associates who will be advising project. Received dissertation and some code of remote associate whose work is similar to ours. Problems Clients want us to find an expert that we can consult (aside from them) on our system architecture. We're thinking of finding an OSU professor to ask, but we're not sure. Plans Submit a system architecture document to Dr. Reese and 2 remote associates by Wednesday. Use system architecture to come up with tools needed for Tech Review.
Fall: Week 6	Progress: Got sample spectroscopy data for depleted uranium, so that we can investigate file format. Problems: Need to figure out place to store code since it sounds like we're not supposed to use cloud storage. We think we can probably use OSU cloud resources, like the GSuite and maybe engineering servers. Plans: Create code to parse sample data and graph it.
Fall: Week 7	Progress: Adapted code from client's colleague to be able to plot sample data. Got approval to use GitHub, which will help a lot. Problems: Nothing major right now. Plans: Go through code/sample data more to understand output from analog to digital converter (ADC). Add to design document.

Fall: Week 8	Progress: We now understand the method of parsing data better and have worked out some more of the design for the overall system. Problems: Nothing major. Plans: Continue analysis of sample data file. Get design doc finished and send to client.
Fall: Week 9	Progress: Emailed Dr. Reese a progress report. Ian got the project more set up by adding a code linter and python pip file. Problems: Wasn't able to meet with Dr. Reese in last few weeks because he had to leave the state, but we sent him a progress report. Plans: Same as last week more or less, continue analysis of sample data file. Revise docs for Dr. Reese and hopefully go through parts with him in person next week or week after.

Winter: Week 1	Progress: During the break, we all read chapters 3, 4, and 13 of Glenn Knoll's Radiation Detection and Measurement, helping us understand the theory and operation of semiconductor detectors. On Friday, we met with Dr. Reese for the first time since last term. We collected one run of sample data with 3 samples present: Cesium-137, Cobalt-60 (?), and Barium-133. Steve also gave us a copy of the program used to generate/parse the data files, which is part of a software package sold by Ortec for about $2500, called the "CONNECTIONS Programmer's Toolkit." Problems: The program that Dr. Reese gave us has some difficulties running on my machine, and I am not sure if all the documentation is present. This could make it difficult to test with it outside the lab. Plans: Investigate programs received Friday, try to get running. - Do they actually output at 10 ms intervals (which is the supposed interval of data they get straight from DAC)? Or are they buffered? - Can we recompile them from the sources provided? Complete analysis of data files collected today by next Wednesday. - Generate histogram from collected counting data, categorize its main features - Calculate energy calibration and efficiency calibration - Background calculations: gamma yield of our samples, based on: 1. radioactive decay formula from element type and age of sample 2. Cs-137 gamma rays per decay event Work on program to parse data files, and play around with more machine learning models.

Winter: Week 2	Progress: ListPRO recompilation: Tried compiling sample code from Ortec that reads in data from the detector in real time, but couldn't because they left out 2 header files. I contacted their support, who immediately emailed a zip of the correct directory structure with the right files. ListPRO opening on our laptops: We have been struggling with the precompiled Ortec program, ListPRO.exe. It's used in the Radiation Center but was crashing whenever we opened it on our laptops, which was one of the reasons I was interested in recompiling it. It turned out that it was crashing without any error message because an assert() statement looked for a driver that's installed with other Ortec software. This means we just need to install that driver on our laptops to at least open up the program for testing with our system (Run ConnectionsSetup.exe, don't need to check any of the instrument families to install). HPGe detector calibration: Wrote code to generate histogram of detector radiation counts from list-mode file from ListPRO, allowing us to visually find peaks in order to perform calibration. Got most of the way to finding equations for energy and efficiency calibrations. We showed our work to Dr. Reese and he told us how to get the rest of the way. Problems: File lock: Our current design watches a file that is continually appended to by ListPRO. We do not have that part of our system operational yet, so when we were at the Radiation Center this week with Dr. Reese, we just tried opening the file while ListPRO was running to see if it was locked. Notepad++ gave an error, so it appears that a file lock is preventing it from being opened. Plans: File lock: Since we can now recompile ListPRO, and Ian found the part of the code that contains the Windows file open() call, we plan on recompiling it with the flags set to share the output file for reading. Hopefully, this means that we can then read the file in real time. If not, we can do other modifications of the ListPRO code to send the data directly to our application with some kind of Windows interprocess communication. Calibration: Email Dr. Reese before meeting 1/23 Thursday with fixed efficiency calculations. Background knowledge: In a decent spot now, but continue reading about detector physics.

Winter: Week 3	Progress Tested recompiled ListPRO on radiation center computer, but I was dumb and left it in debug mode so it failed looking for a VS2017 debug .dll. I recompiled it and tested it on another computer, and it worked. Detector calibration: Energy calibration problem turned out to be data entry problem. Energy calibration done. Efficiency calibration is better, but there's still a point or two causing issues with the equation, where its peak is closer to 200 KeV when it should be closer to 100 KeV. Problems A little annoying to test our stuff since it needs to run on a computer in the Radiation Center. Today I collected info on its software so we can mirror it easier. Plans Retest ListPRO.exe on radiation center computer with single change of FILE_SHARE_READ arg in CreateFile() call. Dr. Reese said I can use remote desktop if I get email approval first to see if they're not using it. This is potentially very helpful for other testing in the future, if we want to just run our program on that computer without needing to find a time to meet Dr. Reese. Meet Dr. Reese and call Ophir Frieder, other main client next week, possible Tue 8am. This is only 2nd time we've talked to him, and Dr. Reese wants us to provide him an overview of our current plans/system architecture that he can critique. We will need to spend time over weekend going over our documents and compiling a new summary. Work on code for file watching/data import modules. Might create a test that uses file modules to just show a plot of current data from detector, updated once a second?
Winter: Week 5	Progress Created progress/plans document with team for client Ophir. Problems I've had some difficulty with feature selection for the decision tree, but I think that's just my inexperience with the tool. Hopefully, just playing around with the parameters more will get me closer. Plans Get real time file watching working. Finish decision tree basic working prototype, with some degree of classification success.

Winter: Week 6	Progress Got file watcher prototype working. It logs when a file is changed and polls every 10 ms. A complementary file writer prototype was made that adds to a file every 10 ms, to simulate ListPRO. Dr. Reese said that I should be able to remotely access the radiation center computer, d102-03, to test our system over the weekend. Set up project in portable installation on usb drive, using WinPython, which was used to setup a portable pipenv. Hopefully, this will work without any missing .dlls on the d102-03 computer. Problems When making the decision tree prototype, I struggled somewhat with the fact that they generally do not come with probabilities and are just discrete classifiers. Our current design states that we will continually run the three classifiers until their confidence values are high enough. I think we can just do that for the other 2 classifiers, and then weight in whether or not the decision tree classifier agrees with them since it does not provide a confidence. Plans Test WinPython setup on d102-03 computer 2/15 saturday. If ADC is on and hooked up, get junk ListPRO data. Refine decision tree classifier parameters to improve accuracy, details of output from each classification.
Winter: Week 7	Progress Design review Tested on d102-03 with remote desktop: Portable installation File watching With real output from ListPRO (I guess they leave the instruments connected to the computer all the time, which is good for us) With ListPRO simulator script that just appends to a file like ListPRO does. Problems Tree accuracy Plans Submit PR for file watcher Submit PR for portable installation documentation

Winter: Week 8	Progress Submitted PR for: Manager prototype that watches file and has option of plotting real time cumulative spectrum ListPRO simulator that writes to a file incrementally like ListPRO would, with past collected data Problems Decision tree accuracy still not good. Plans Continue working on other decision trees to get better intuition for tweaking them. Submit PR for portable installation documentation. Submit PR for turning each classifier test into a Classifier object that can be used from the Manager component, and then add classification checking to Manager file watching process. This will get us mostly to beta functionality. Once above stuff done, work on system user interface.
Winter: Week 9	Progress Submitted PR for installation/running documentation. Merged existing PR for manager + listpro simulator. Met Dr. Reese Friday. Seems that we're doing pretty well at the moment. Problems Nothing major. Plans Decision tree improve. PR for manager with classifiers added, for basically full beta functionality. Status update for Ophir, Jessica by Monday. Meet Tue at 11 am for phone call. Revise docs for final report. Start system video.

Winter: Week 10	Progress Got pickle working for simple saving of prepared data set for easier playing with models. Started beta video, decided general layout and assigned sections. Started final progress report. Problems Partition data better for training (although not essential, the models all work ok). Plans Progress report. Beta video. Work on manager and the classifiers. Meet Dr. Reese next Wednesday at 2:30pm, test out GammaVision to see how long it takes to classify isotopes. Take another group picture.

Yihong Liu's Blog Posts

Fall: Week 4	progress met with client and we have talked about more details about the project, get the rough framework about the whole project. problems we need to practice to use WEKA, and be good at using it to manipulate our data. But we are not sure how to set up environments yet, need some time to self-learning that. Plans getting more practice on WEKA and trying to implement some simple machine learning models using WEAK , such as decision-tree model.
Fall: Week 5	progress got some practice with WEKA implementation and more detailed information about the project. Problems we don't have any python machine learning algorithm implementation experience before, but we need to design the algorithms for our project. Plans going to read the python algorithm for machine learning sample code, and trying to figure out the diagram design for the project first.

Fall: Week 6	Progress Met with client, and got some sample data files, we know the what the data format is for now. Problems Not really, maybe we need more background knowledge about radiation stuff when we try to design the diagram to meet client's requirement. Plans going to use weka, tensorflow, python or other framework to build some easy machine learning model to test with the sample data files.
Fall: Week 7	Progress Got the sample parsing code from client and figured out what the correct format of dataset should be used and how we should pre-process the data file. Problems I can not compile the python sample parsing code on my local machine due to some unknown reasons. Plans Debug the parsing code and make it compiled on my local machine.
Fall: Week 8	Progress Not much, just turn the progress port for last week to client (client's requirement), talked about what we learned in our tech reviews. Problems Not much, just trying to finish design document with the feedback. Plans meet with client, and get more data for classification.
Fall: Week 9	Progress Got some practice with tensorflow for neural network machine learning algorithm, and self-learning some neural network basic concepts and theory. Problems No much. Plans meeting with client next week to talk about progress and discuss the next step for our implementation plan.
Winter: week1	progress meeting with client and took some new data and got the listmode source code file, also introduced what we have learned for radiation knowledge during the winter break. Problems try to figure out the math equation that is used to calculate for frequency and energy , such as frequency calibration. Plans try to get the value correct before next week meeting and show the result to the client.

Winter: Week 2	Progress meet with client and talked about the efficiency and energy calibration and used open-gama software to pull out the plot graph. Problems the number we got is not consistent with graphs though. we might need to re-calculate to make sure the result is consistent. Plan Try to get the number correct.
Winter: Week 3	Progress Got the open gama vision software compiled,got the correct data file from the images. Plans Start to design core algorithm and have a presentation with clients. Problems Not much
Winter: Week 5	Progress create the neural network machine learning model to test the accuracy focus on single element data source file. Problems although the accuracy is good, but the time is much slower than excepted. Plans going to create the neural networking model can be used for detecting more than one element within a data source file.
Winter: Week 6	Progress finished the alpha functionality and the first draft poster, work is going well for now and we should meet our goals for Alpha if we can stick on our plan. Problems still testing the model see it works with multi label classifier. Plans figure it out and get it done before next meeting with client maybe.
Winter: Week 7	Progress Make the PowerPoint for presentation and did review feedback for other groups. Problems The accuracy of different machine learning models differ widely and we need to do some analysis and try to figure out the reason. Plans trying to apply the real-time function into our project and optimizing the accuracy of neural network.Also trying to figure out the reason why the accuracy of different machine learning models differ widely.

Winter: Week 8	Progress Improved accuracy for neural network and test our model on clients' computer. Problems not much. Plans continue to work on the improvement and meet with client next Friday and may look at the GUI stuff if needed.
Winter: Week 9	progress made some progress on improvement of neural network model's accuracy and time spent. Problems the running time is still much longer than other two models. Plans going to test it on client's computer and do the persistence for our models.
Winter: Week 10	Progress make the improvement and figured out a new way to do the optimization, which is called GindsearchCV from scikit learn learn package. Problems Not much, just making some organizations for the files we have. Plans implement the new package and see what's the output parameter result are.

6 Final Poster

RADIATION COUNTING USING MACHINE LEARNING

Applying machine learning to improve detection of radioactive materials

INTRODUCTION AND BACKGROUND

Gamma-ray spectrometers record the energy levels of incoming photons from a radioactive sample. Once a spectrum has been recorded by the detector, specialized software is able to analyze it and identify which materials are present by looking for the peaks where a lot of radiation came in at a certain energy level. The detector will gather data on a sample for a predetermined amount of time, commonly a few minutes.

Our project involved using machine learning to predict which materials are present in each sample faster than conventional algorithms. These predictions are made in real time, while the detector is still gathering data.

IMPORTANCE

Fast

- Machine learning models can identify materials much faster than humans as well as conventional isotope identification software

Adaptable

- Training data can be changed for the application, allowing it to fit a variety of conditions and detectors

Scalable

- The data set could be extended to include information from large databases
- Samples with many different materials would not add much additional time for calculation while this would be a problem for human analysis

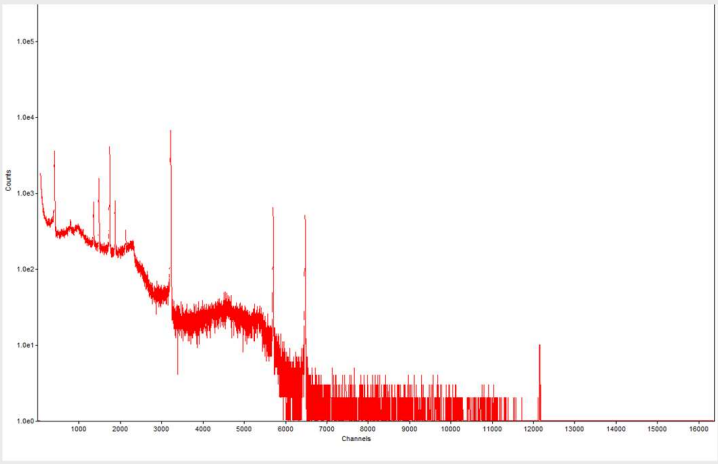


Figure 1: An example spectrum generated from a radiation detector. Each count represents one gamma-ray photon hitting the detector, and each channel is a different energy level of photon.

DESCRIPTION

Several combinations of materials were placed in a radiation detector. The detector then sampled each of these combinations and created files containing data about these samples. These sample files were used as the training data set for our machine learning models. Three different machine learning classifiers were used in our application.

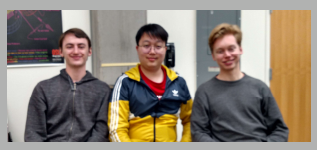
Our application uses our trained models to make predictions in real time. Once a detector starts writing data to a file, the models repeatedly make predictions until it is confident enough that it has identified the materials in the detector. If it does not become confident enough after a certain amount of time, it stops itself.

RESULTS/CONCLUSION

- Improved isotope identification times at detector in OSU Radiation Center
- Using real-time data from detector allows for continuous analysis
- Machine learning is a viable method of radiation spectrum analysis

ACKNOWLEDGEMENTS

- Dr. Steve Reese
  Director, OSU Radiation Center
  Capstone Project Client
- Ophir Frieder
  Professor at Georgetown University
- Jessica Curtis
  Recent OSU PhD



Project members listed from left to right:

Ian Brown; CS Senior
browni@oregonstate.edu

Yihong Liu; CS Senior
liuyih@oregonstate.edu

Sean Gillen; CS Senior
gillens@oregonstate.edu

Oregon State University

7 Project Documentation

7.1 Basic Usage

Requires Python version 3.8. If this is not available, it is suggested to install it using pyenv: https://github.com/pyenv/pyenv

1) Follow the instructions to install pipenv here: https://pipenv.pypa.io/en/latest/install/#installing-pipenv. Make sure that pipenv is added to your PATH if installing using pip by following the instructions in the pipenv documentation. If you don't want to add it to your path, replace pipenv with python -m pipenv in future instructions, making sure python corresponds with the version of pip used to install pipenv.
2) Install required dependencies using "pipenv install"
3) Run one of the classifier using "pipenv run python [script]" where script is either serializer_test_NB.py, serializer_test_tree.py, or serializer_test_NN.py

7.2 Testing

Check static typing with mypy and run unit tests with pytest: "make test"

7.3 Docker

This application can be deployed in a Docker container with: "./docker.sh [script]" where script is defined the same as in the Basic Usage section.

7.4 Run using WinPython

See the instructions in docs/WinPython.md

8 Resources

what I would recommend is that https://scikit-learn.org/stable/getting_started.html, the website contains all the introduction about the machine learning package it provides, with very detailed explanation with real examples. Also, getting familiar with python is very important for this project, since our environment is based on python 3.7, the following website introduces the basic use of python 3.7. https://docs.python.org/3.7/. The book we would recommend is called 'python machine learning'(3st edition, published September 23rd 2015,Publisher: Packt Publishing, Language: English, ISBN-10: 1783555130) , which is a famous and useful machine learning book based on python programming language, This book Leverage Python's most powerful open-source libraries for different machine learning models with deeper under insights. it's available free here: https://github.com/rasbt/python-machine-learning-book-3rd-edition, it not covers the basic machine design and some basic algorithms behind each model, but also introduces some examples of using scikit-learning libraries to make a machine learning model, which is the library we used for the project. There are some professors or TAs for machine learning class was helpful, always good to try to get contact with them if you are stuck.

9 Conclusion and Reflection

9.1 Ian Brown's Conclusion and Reflection

This project was an excellent opportunity to get some experience with machine learning and data science. I learned some of the basics of training ML models and data preprocessing. Also, I learned a good amount of the physics and chemistry behind radiation detectors. Our client required us to do some reading on these concepts to better understand the application.

This project also helped me deal with a situation where I need to deliver work while also learning about new material. The beginning was particularly stressful since machine learning and radiation detection were areas that I had no experience with. Despite this, I was able to contribute to the project by starting with the parts I did know and making adjustments as some of the more complicated topics started making more sense. If I could do the project over again, I would spend more time discussing what the client needed towards the beginning of the project. There were changes to some of the deliverables, so it could have been simpler to figure these out earlier if possible.

9.2 Sean Gillen's Conclusion and Reflection

This project has proved to be fun and challenging. I learned about machine learning, and how most of the work is usually in data preparation before the model is involved. Working with the Ortec spectrometer and software taught me about mixing new development with code that's 20 years old and not discussed on the internet. I especially enjoyed getting the file IO working; it was like a puzzle trying to decipher the format used by Ortec ListPRO until I had proper documentation for it. I've come to the same conclusion as a guy I once talked to at a career fair: you don't want to use a binary file format to send data back and forth unless you have a very special use case where you need high throughput or some requirement like that, which somehow you cannot reach with a somewhat human readable format. The debugging of this stage would have gone faster if they had just used ASCII or something a little more legible, at the cost of having a 5 MB file instead of a 2 MB file.

I also learned about the non-technical parts of projects. I learned how important it is for me personally to critically go through documents before meetings, so I can come with better questions and comments. This probably varies by the team, but make sure that anything that needs to be divided up is divided up ahead of time so that people aren't working on the same stuff or not starting because they want to avoid that (guilty here). If doing it over again, I'd probably focus more on the training data save feature, as that would have sped up development. I'd echo what Ian and Yihong said, that it's important to get the science background done soon as well as clear up requirements early in the project. I'm glad that I read the work of someone else who worked with an Ortec code sample, who mentioned that their support is very helpful if you are having difficulties with part of it. Their support team was instant in providing me with additional files that it turned out we were missing.

9.3 Yihong Liu's Conclusion and Reflection

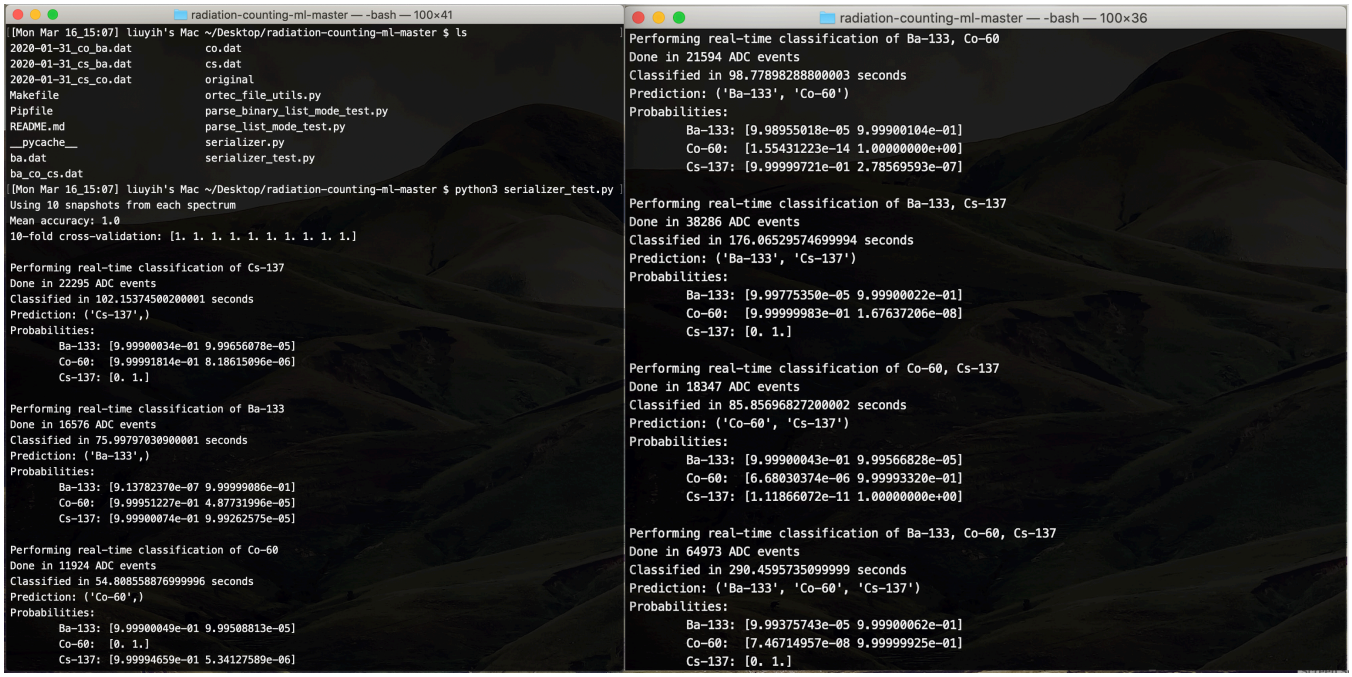
I have learned a lot from developing the machine learning based project throughout the year, what I feel most grateful part was that we are able to really implement the machine learning models into some real life application, to know more about what the machine learning model can help us to in the industry. I have also learned some radiation knowledge, such as what is energy, what does frequency represent. I have learned the work flow of detailed process about building

a brand new project or research, also the way to schedule the plan for each milestones. not only about the project, but the way we handle the different time point is very impressive for me, cut the big project into many small slices and make it organized and set time point, then package all the slices to form the whole project makes the management much easier than I used to do group project, I have also learned that working in teams help me to realize what I didn't see what I made mistakes, also sufficient contact is very important for keeping the whole project on the schedule. If I can do it all over, I would say not much I want to do differently, but what I would like to do differently is try to get enough background knowledge needed for radiation stuff as earlier as possible then we can work more on the machine learning model itself more.

10 APPENDIX

10.1 Project Screenshots of results

10.1.1 Neural Network Demo Result



10.1.2 Naive Bayes Demo Result

```
Using 10 snapshots from each spectrum
Mean accuracy: 1.0
10-fold cross-validation: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

Performing real-time classification of Cs-137
Done in 82 ADC events
Classified in 0.04520080000008875 seconds
Prediction: ('Cs-137',)
Probabilities:
        Ba-133: [9.99941732e-01 5.82675581e-05]
        Co-60:  [9.99976018e-01 2.39816773e-05]
        Cs-137: [2.34670124e-53 1.00000000e+00]

Performing real-time classification of Ba-133
Done in 48 ADC events
Classified in 0.02657909999993535 seconds
Prediction: ('Ba-133',)
Probabilities:
        Ba-133: [1.01889981e-25 1.00000000e+00]
        Co-60:  [9.99998544e-01 1.45587051e-06]
        Cs-137: [9.99946522e-01 5.34775823e-05]

Performing real-time classification of Co-60
Done in 24 ADC events
Classified in 0.01104859999998098 seconds
Prediction: ('Co-60',)
Probabilities:
        Ba-133: [9.99923297e-01 7.67031731e-05]
        Co-60:  [8.19995936e-29 1.00000000e+00]
        Cs-137: [9.99999718e-01 2.82290043e-07]

Performing real-time classification of Ba-133, Co-60
Done in 22 ADC events
```

```
Classified in 0.010074799999983952 seconds
Prediction: ('Ba-133', 'Co-60')
Probabilities:
        Ba-133: [2.19715566e-05 9.99978028e-01]
        Co-60:  [5.36234081e-14 1.00000000e+00]
        Cs-137: [9.99964748e-01 3.52517817e-05]

Performing real-time classification of Ba-133, Cs-137
Done in 51 ADC events
Classified in 0.02318209999998544 seconds
Prediction: ('Ba-133', 'Cs-137')
Probabilities:
        Ba-133: [1.14123704e-09 9.99999999e-01]
        Co-60:  [9.99916470e-01 8.35300382e-05]
        Cs-137: [7.00486797e-14 1.00000000e+00]

Performing real-time classification of Co-60, Cs-137
Done in 39 ADC events
Classified in 0.017938099999923907 seconds
Prediction: ('Co-60', 'Cs-137')
Probabilities:
        Ba-133: [9.99939592e-01 6.04076343e-05]
        Co-60:  [6.72227925e-09 9.99999993e-01]
        Cs-137: [7.88738448e-14 1.00000000e+00]

Performing real-time classification of Ba-133, Co-60, Cs-137
Done in 43 ADC events
Classified in 0.019558599999982107 seconds
Prediction: ('Ba-133', 'Co-60', 'Cs-137')
Probabilities:
        Ba-133: [4.74504136e-05 9.99952550e-01]
        Co-60:  [1.68289895e-07 9.99999832e-01]
        Cs-137: [3.28054729e-11 1.00000000e+00]
```

10.1.3 Decision Tree Demo Result

```
Using 10 snapshots from each spectrum
Loading training data...
Mean accuracy: 1.0
10-fold cross-validation: [0.71428571 1.         1.         0.85714286 0.71428571 1.
 1.         1.         0.85714286 1.        ]

Performing real-time classification of Cs-137
Done in 12985 ADC events
Classified in 8.518903300049715 seconds
Prediction: ('Ba-133', 'Cs-137')
Probabilities:
        Ba-133: [0. 1.]
        Co-60:  [1. 0.]
        Cs-137: [0. 1.]

Performing real-time classification of Ba-133
Done in 14696 ADC events
Classified in 8.969158099964261 seconds
Prediction: ('Ba-133',)
Probabilities:
        Ba-133: [0. 1.]
        Co-60:  [1. 0.]
        Cs-137: [1. 0.]

Performing real-time classification of Co-60
Done in 46779 ADC events
Classified in 21.327716100029647 seconds
Prediction: ('Ba-133', 'Co-60', 'Cs-137')
Probabilities:
        Ba-133: [0. 1.]
        Co-60:  [0. 1.]
        Cs-137: [0. 1.]

Performing real-time classification of Ba-133, Co-60
```

```
Done in 29892 ADC events
Classified in 13.651077499962412 seconds
Prediction: ('Ba-133', 'Co-60', 'Cs-137')
Probabilities:
        Ba-133: [0. 1.]
        Co-60:  [0. 1.]
        Cs-137: [0. 1.]

Performing real-time classification of Ba-133, Cs-137
Done in 24060 ADC events
Classified in 10.968293400015682 seconds
Prediction: ('Ba-133', 'Cs-137')
Probabilities:
        Ba-133: [0. 1.]
        Co-60:  [1. 0.]
        Cs-137: [0. 1.]

Performing real-time classification of Co-60, Cs-137
Done in 19083 ADC events
Classified in 9.343986100051552 seconds
Prediction: ('Ba-133', 'Co-60', 'Cs-137')
Probabilities:
        Ba-133: [0. 1.]
        Co-60:  [0. 1.]
        Cs-137: [0. 1.]

Performing real-time classification of Ba-133, Co-60, Cs-137
Done in 27739 ADC events
Classified in 14.699496100074612 seconds
Prediction: ('Ba-133', 'Cs-137')
Probabilities:
        Ba-133: [0. 1.]
        Co-60:  [1. 0.]
        Cs-137: [0. 1.]
```

10.2 Essential Code

10.3 Misc

10.4 Code Review

Category	Description	Reviewers Comment	Action taken by reviewed group
Build	Could you clone from Git and build using the README file?	Yes. Project ran correctly. Note that there is an oversight with regards to installation using pipenv, assuming that it will be set as an environmental variable. Make sure to account for the possibility that it needs to be set as such.	The README.md now specifies that pipenv should be present in the user's path, and it provides a link to the official documentation.
Legibility	Was the flow sane and were variable names and methods easy to follow? Does the code adhere to general guidelines and code style?	Code follows good practices and is clear and readable. There is a good sense of separation and abstraction via functions.	<no change requested> Additional directories created for clearer separation of different files.
Implementation	is it shorter/easier/faster/cleaner/safer to write functionally equivalent code? Do you see useful abstractions?	There do not appear to be any changes to code structure that would result in a significant change in quality. Much of the code appears to rely on exisiting frameworks and API's, which results in less room for mistakes or optimization.	<no change requested>
Maintainability	Are there unit tests? Should there be? Are the test covering interesting cases? Are they readable?	There do not appear to be unit tests. Unit tests would work to verify certain elements of the system, such as data and error handling, but much of the system relies on the ML models being correctly trained, for which unit testing would not suffice.	Unit tests added using python packages unittest and pytest. They live under tests/ and are run with "make test."
Requirements	Does the code fulfill the requirements?	I notice that check_classification is not defined yet. Does this still need to be defined? And if it has been made unnecessary, it and the call made by watch_file should be removed.	Removed call and definition for check_classification().
Other	Are there other things that stand out that can be improved?	No other significant changes are needed.	<no change requested>