IMLD: A Python-Based Interactive Machine Learning Demonstration

T. Cap[1], A. Kreitzer[2], M. Miranda[2], D. Vadimsky[2] and J. Picone[1]

1. Neural Engineering Data Consortium, Temple University, Philadelphia, Pennsylvania, USA
2. College of Engineering, Temple University, Philadelphia, Pennsylvania, USA
{thao.cap, aaron.kreitzer, matthew.miranda, dakota.vadimsky, picone}@temple.edu

The related fields of machine learning and pattern recognition have enjoyed significant success in recent years due to the impact of deep learning algorithms [1]. Pattern recognition is the automatic recognition of regularities or trends in data. Machine learning, a closely related field that has evolved considerably in the past two decades, refers to the ability of a machine to learn and adapt to data, improving a system's ability to detect patterns and perform inference. These methods are ubiquitous in engineering today impacting diverse fields including signal and image processing, human language technology, bioinformatics, and finance. The ISIP Machine Learning Demo (IMLD) is a tool used to introduce the basics of machine learning using a highly interactive environment in which users can easily visualize the performance of an algorithm. IMLD was first developed as a Java applet in the late-1990's when Java applets were envisioned as the future of interactive computing [2] and there was an emphasis on web-based educational tools [3]. The Institute for Signal and Information Processing, then located at Mississippi State University, developed a suite of interactive demos to teach important concepts in signal processing [4].

However, today, due primarily to security issues, Java applets have fallen in disfavor and are no longer being supported. Instead, it makes more sense to deliver such applications in Python, where the bulk of machine learning research is being done. This allows the application to integrate a wider range of algorithms. Hence, a major focus of this work, being conducted at the Neural Engineering Data Consortium at Temple University, is the conversion of this application from Java to Python. However, as we will discuss in this work, delivering a complex interactive application in Python is not as easy as one might think. Major concerns include the stability of the graphical programming languages available and web accessibility.

A typical screenshot of the user interface is shown in Figure 1. IMLD allows users to create unique two-dimensional data sets that can be easily visualized. Users can choose from a wide selection of predefined data sets such as multivariate Gaussian data, draw custom data sets, or create a combination of the two. Data can be saved to a file or uploaded, allowing users to experiment with their own data sets and use IMLD as a reference implementation. A set of standard algorithms are available including fully supervised approaches such as Principal Components Analysis [5], unsupervised approaches such as K-MEANS clustering [6], and popular neural network algorithms such as multilayer perceptrons [7]. The number of classes, classification modules, and other key parameters of the data generation are user defined. A dialog box is included that displays step-by-step computations for the

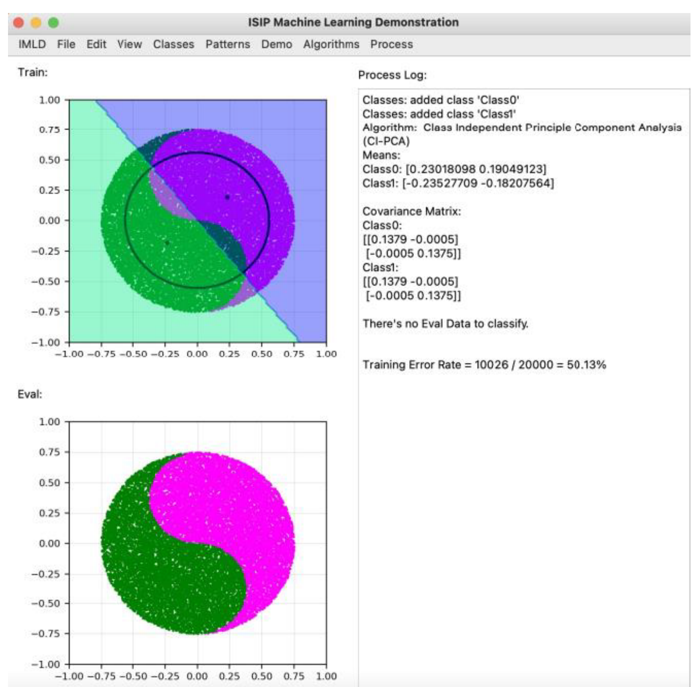

Figure 1. The IMLD user interface

algorithm. Users can step through the algorithms or run them in their entirety. Decision surfaces are rendered, and error rates are computed on the training and evaluation sets.

An overview of IMLD's software architecture is shown in Figure 2. IMLD is dependent on a variety of third-party libraries. Many third-party libraries were considered based on the wide variety of components required to generate the applet. The following libraries were decided based on the specific needs of IMLD. PyQT5 is the framework used to create the front-end design of the application, chosen for its ability to handle large quantities of data. The backend uses NumPy for calculations, Sklearn/Scipy for implementing algorithms, and Matplotlib for interactive graphs. The software is organized into three major components: the graphical user interface (GUI), data handler, and machine learning algorithms.

The first part of the architecture is the GUI, where three key modules handle the window design, events, and parameters. The window module manages all the front-end design for the application. The module includes the menu bar, input/output graphical displays, and output log. The menu bar holds all the functionality for adding/deleting classes, choosing an algorithm, and giving users the option to either import, export, or create test data. The two graphical displays allow users to click and drag their mouse to create either point-like data or Gaussian-like data. The last section for this module is the output log which records all interactions between users and the application, i.e., adding a class or choosing an algorithm. The event module handles execution for the buttons displayed on the GUI. Lastly, the parameter module handles all secondary user information and allows users to configure the data generators.

Another part of the application is the data handler. One of the features that the Python version of IMLD enhances from the Pattern Recognition Applet written in Java is the software that allows users to generate data. The tool provides two underlying mechanisms of the generation of the data: generation by drawing points or generation using a functional form such as a Gaussian distribution. The training and evaluation sets are generated independently in separate windows so that a wide range of machine learning scenarios

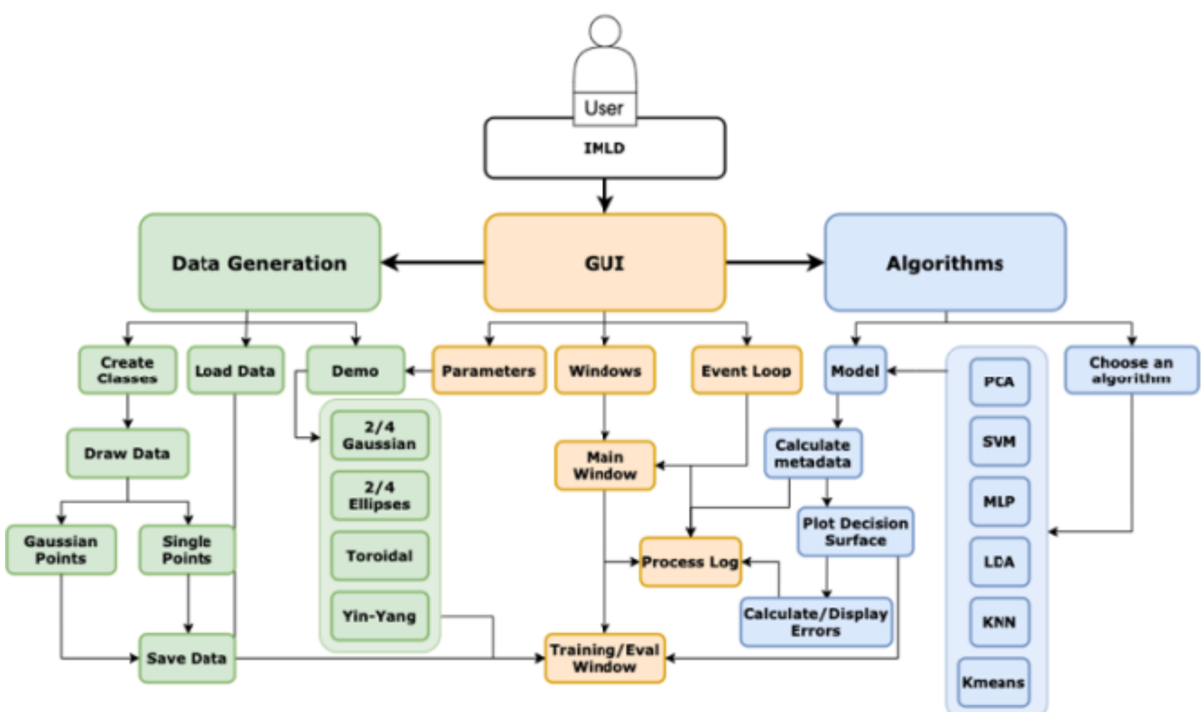

Figure 2. The IMLD software architecture

(e.g., generalization) can be evaluated. A dictionary stores the data where the key is the name of the class added, and the value is an array that holds the class color and an array of x and y coordinates.

Newly created data can be exported into a CSV file where comment fields are used to hold class names and class color, while each line holds the x and y coordinates. This is a simple format we have used for our machine learning class that makes it easy for novice programmers to interface to the data. It was also a preferred format based on a survey we conducted with the user community. The application also allows CSV files to be imported for further study and modification.

IMLD supports the configuration of data sets and analysis parameters through drop-down menus. Class, Color, and Scale are all tools that allow users to uniquely configure their data. While adding a new class, users are prompted to add a name, and then choose from over the 150 colors options. Once set, users can now draw data points. Users can add a new class at any time by navigating to the Classes menu and adding another class. The Scale of the data can also be managed through the Classes tool while in the Classes menu. Users can at any time choose which of the added classes they would like to delete.

Users can also choose from a selection of prestored, or canned, demos that include classic machine learning datasets such as overlapping Gaussian distributions. toroidally shaped distributions that cannot be classified with a linear classifier, and a yin-yang distribution that requires a nonlinear decision surface. Finally, after creating the data, users can choose the algorithm under the algorithm section and select the 'run' or 'run by step' options to start classification.

Currently, IMLD does not allow users to insert their own algorithms. In the future, we plan to implement a method by which users can do this. This feature would significantly enhance the educational capacity of IMLD, as users will not be restricted to the algorithms offered only by the tool.

IMLD is an educational tool with the ability to walk users through a step-by-step process to visualize various machine learning algorithms. It has been used by a machine learning class we have been teaching since the late 1990's (*https://www.isip.piconepress.com/courses/temple/ece_8527/*). It is easily installed on a platform that includes Anaconda v3 and Python v3.7 or later. The source code is available from the course web site at: *https://www.isip.piconepress.com/courses/temple/ece_8527/resources/imld/*. A detailed user manual demonstrating use of the tool and instructional videos are also available. A demonstration will be provided at the symposium.

ACKNOWLEDGMENTS

Research reported in this publication was most recently supported by the National Science Foundation's Industrial Innovation and Partnerships (IIP) Research Experience for Undergraduates award number 1827565. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the official views of any of these organizations. Open source libraries that were used to develop the IMLD are: NumPy v1.15.4, PyQT5 v5.13.1, SkLearn v0.20.0, Scipy v1.1.0, and Matplotlib v3.0.2.

REFERENCES

[1] W. Samek, G. Montavon, S. Lapuschkin, C. J. Anders, and K.-R. Müller, "Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications," *Proceedings of the IEEE*, vol. 109, no. 3, pp. 247–278, 2021. *https://ieeexplore.ieee.org/document/9369420*.

[2] D. May and J. Picone, "The ISIP Pattern Recognition Applet," Institute for Signal and Information Processing, College of Engineering, Mississippi State University, 2002. [Online]. Available: *https://www.isip.piconepress.com/projects/speech/software/demonstrations/applets/util/pattern_recognition/current/*. [Accessed: 05-Jul-2021].

[3] J. Picone, R. Duncan, and J. Hamaker, "Internet-Accessible Speech Recognition Technology," in O'Reilly Open Source Convention, 2001. *http://www.isip.piconepress.com/publications/conference_presentations/2001/oscon/software/*.

[4] J. Picone, "Internet-Accessible Technology Demonstrations," *Institute for Signal and Information Processing, College of Engineering, Mississippi State University*, 2000. [Online]. Available: *https://www.isip.piconepress.com/projects/speech/software/demonstrations/*. [Accessed: 05-Jul-2021].

[5] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. New York City, New York, USA: John Wiley & Sons, Inc, 2001. *https://www.wiley.com/en-us/Pattern+Classification%2C+2nd+Edition-p-9780471056690*.

[6] K. Mallapragada, R. Jin, and A. Jain, "Non-parametric Mixture Models for Clustering BT - Structural, Syntactic, and Statistical Pattern Recognition," E. R. Hancock, R. C. Wilson, T. Windeatt, I. Ulusoy, and F. Escolano, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 334–343. *https://link.springer.com/chapter/10.1007/978-3-642-14980-1_32*.

[7] C. Bishop, *Pattern Recognition and Machine Learning*, 2nd ed. New York, New York, USA: Springer, 2011. *https://www.springer.com/us/book/9780387310732*.