HIGH SCHOOL EDITION
Journal of Student Research
Volume 10 Issue 4 (2021)

Applications of Feature Engineering and Logistic Regression to Analyze Microchip Performance

Agranya Ketha[1] and Guillermo Goldsztein[#]

[1]Morris County School of Technology, Denville, NJ, USA
[#]Advisor

ABSTRACT

Machine learning is an adaptive concept that has applications in several other industries. Businesses can use it to create real-time and predictive analyses of corporate data to identify consumer trends, calculate product success rates, visualizing short- and long-term business models, and more. This paper takes machine learning algorithms to analyze microchip performance. Beginning with an introduction of fundamental concepts of machine learning — including feature, labels, and classification vs regression models — the paper's example is introduced with a scatterplot visualization, a definition of the loss function binary cross entropy, and the parameters of the dataset. The third and fourth sections explain the critical function of feature engineering and its utility in refining the decision boundary to best separate the data classes with minimal error. The final section reiterates the main idea that machine learning is ubiquitously applicable and, like the example in this paper, has many cross-industrial functions.

Introduction

Machine learning is a technological concept that encompasses an expansive majority of industries across the world. Its main purpose is to facilitate the process of identifying a pattern in existing data to create a future projection, notably beyond what humans can feasibly achieve. Sorting this discipline's subsets comes in various forms, from an algorithm's predictive capacity (supervised, unsupervised, or reinforced learning) to the type of data it is given (classification or regression modeling).

As automated analytics fundamentally changes the ways consumers and businesses alike perceive data analysis, communication, and security, it is important the capabilities and limits of the tools that one can use to analyze information and accordingly posit a conclusion for a real-life phenomenon. This research will contribute to this continual discovery by taking a dataset of microchip performance metrics and creating, testing, and discussing a few algorithms that analyze this data.

From a technical standpoint, the goal of this research is to generate the most adequate decision boundary against this dataset. These are the primary research questions that are discussed in conjunction with the algorithms:

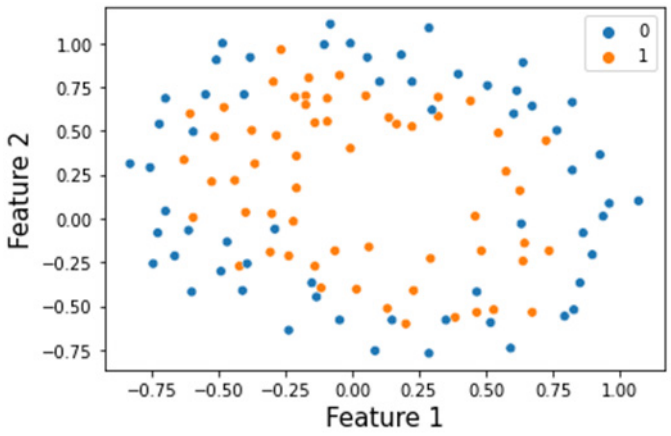
1. What is the best method of determining the most accurate and precise decision boundary?
2. How do different data classifications alter the algorithm development process?

Materials and Methods

The dataset consists of two numerical 'features' (which fall between -0.8 and 1.2) that are used to calculate the efficacy of a microchip's performance. The performance (denoted by 'pass') is either a 'yes' or a 'no', implying its status as a binary classification. In order for a program to be able to computationally differentiate any given microchip's status, each 'yes' and 'no' had to be replaced by '1' and '0', respectively; this process is known as one-hot encoding.

	feature 1	feature 2	*pass*	pass (OHE)
0	0.051267	0.699560	*yes*	1
1	-0.092742	0.684940	*yes*	1
2	-0.213710	0.692250	*yes*	1
...	...	...	...	...
115	-0.484450	0.999270	*no*	0
116	-0.006336	0.999270	*no*	0
117	0.632650	-0.030612	*no*	0

The first and last five rows of the dataset before and after one-hot encoding (OHE).



The data visualized as a scatter plot.

The scatter plot visualization was rendered using Google Colaboratory (or Colab for short), which is the software used for analyzing and visualizing all of the data & creating all of the algorithms used in this paper. Colab is a web- and cloud-based Python editing tool and notebook; it is based on Jupyter Notebook's editing environment but is largely used for its portable ease of use and collaborative access.

The technical aspects of this project were created using Python, a high-level easily comprehensible programming language. The following data libraries were accessed for this project:

- Matplotlib, '*matplotlib*' → used for graphing and visualizing data points and decision boundaries
- NumPy, '*numpy*' → used for computationally operating on arrays
- pandas, '*pandas*' → used for transporting *.csv* files to manipulable code
- Seaborn, '*seaborn*' → used for initializing scatterplots for data visualization

- scikit-learn, '*sklearn*' → used for scaling arrays, splitting datasets into train-test sections, and generating classification reports
- Keras & TensorFlow, '*tensorflow.keras*' → used for analyzing loss and activation functions

Mathematically, there are a few core functions used to create a decision boundary. The first one is known as the sigmoid function.

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

This equation is used as an activation function to execute the binary cross entropy loss function and determine the best fitting decision boundary. The reason the sigmoid function is used is because it restricts the graph between 0 and 1 (the lower and upper limits of probability); also, while x is increasing, σ(x) is also increasing.
The second is a series of expressions that represents the rudiments of feature engineering. The label prediction function for a linear decision boundary is as follows:

$$\hat{y} = \sigma(w_1x_1 + w_2x_2 + b)$$

ŷ represents the label prediction, and σ represents the aforementioned sigmoid function. The expression within the parentheses represent the parameters of the function. The most comparable form of this expression is the classic slope-intercept form of a linear equation: y = wx + b; because of this, the decision boundary generated by the above equation will be linear.

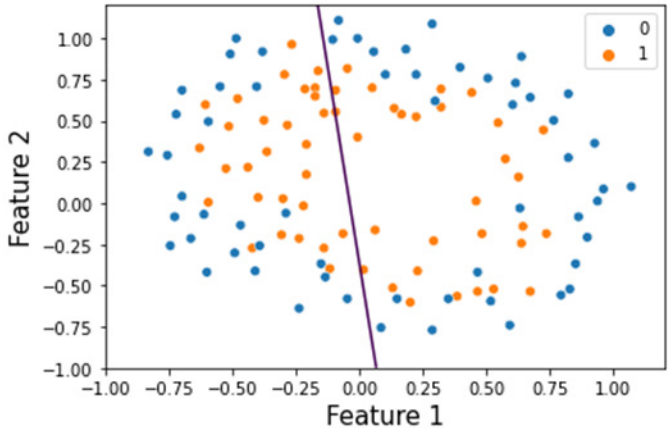
It can be easily noted, however, that a linear decision boundary is not visually adequate for the scatter plot data above. Instead, a few better alternatives include a polynomial expression or, as with the following expression used for the adjusted algorithm, an expression for an ellipse.

$$\hat{y} = \sigma(w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_1x_2 + w_5x_2^2 + b)$$

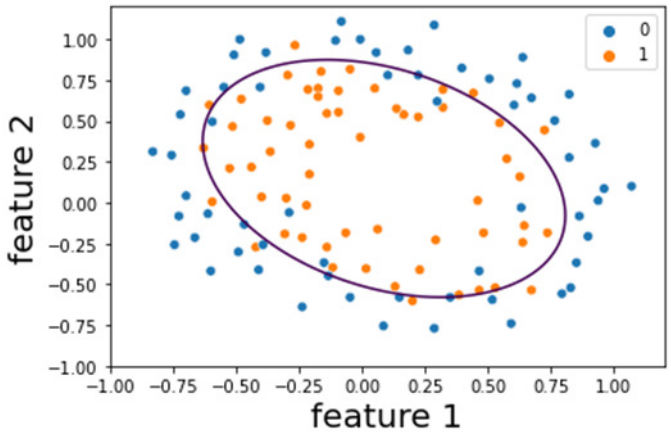
ŷ and σ maintain their same function, but the expression embedded within the parentheses now contains enough parameters to ensure that the resulting decision boundary will be elliptical instead of linear. The first equation has a degree of 1 (meaning that the power of the hyperparameter is 1), and the second equation has a degree of 2 (the hyperparameter power is 2).

Results

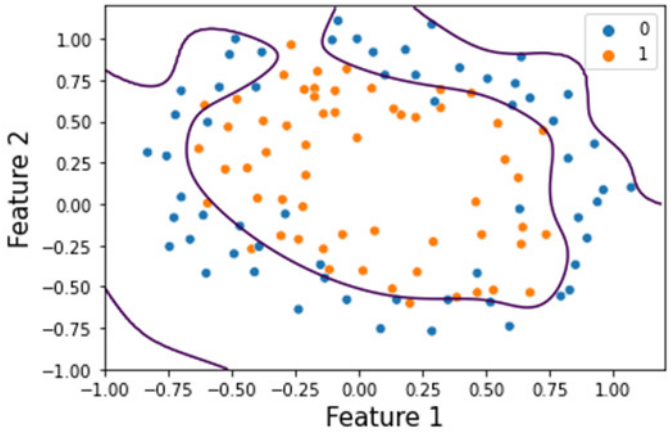
With the given dataset, there were three different variations of a decision boundary generated.



The data scatterplot with the best-fitting *linear* decision boundary.



The data scatterplot with the best-fitting *elliptical* decision boundary.



The data scatterplot with the best-fitting *20th-degree nonlinear* decision boundary.

The first figure shows a first-degree (linear) decision boundary for the scatter-plotted dataset; the second figure shows a second-degree (elliptical) decision boundary; and the third figure shows a twentieth-degree decision boundary. All

three of these were algorithmically designed to be the best fit for the corresponding data; however, there are clear visual discrepancies between the three examples.

Discussion

By design, a classification decision boundary is created to be the most accurate separation between two categories of plotted data. While the decision boundary in the first scatter plot algorithmically demonstrates this idea, it is not a good visual separation. This linear boundary is said to underfit the current data because it cannot properly capture the separation between the two classifications; it also would not accommodate any new data that follows the current data's trends well.

The decision boundary in the second scatter plot, which is a second-degree ellipse, is a much better separation (visually and algorithmically). It fits the current data and would accommodate new data well. The decision boundary in the third scatter plot, which is a 20^{th} degree boundary, is an interesting case that is unlike either of the former two. While it fits the current data well, it would not serve as an efficacious boundary for any new data that even roughly follows the trends of this current data. This idea is known as overfitting, and it is a common characteristic of decision boundaries of extremely high degrees.

There are two other methods of verifying the efficacy of decision boundaries, both of which do not rely on a visual analysis of the boundary itself against a scatter plot. The first method is by using a classification report, which displays the precision, recall, and accuracy metrics of a decision boundary on a given dataset. *scikit-learn*'s classification report functions based on a confusion matrix (a 2x2 matrix separating results into true positive, true negative, false positive, and false negative), and the three aforementioned ratios are defined as follows:

- Precision: the number of true positives over the total number of positive predictions (true positives and false positives)
- Recall: the number of true positives over the total number of positives (true positives and false negatives)
- Accuracy: the number of correct predictions (true positives and true negatives) over the total number of predictions

A value closer to 1 for any of these values indicates a more effective decision boundary. These are the results for the first two boundaries above:

	First Power (Linear)			Second Power (Nonlinear)		
	0	1	W.A.	0	1	W.A.
Precision	0.55	0.54	0.54	0.86	0.85	0.86
Recall	0.57	0.52	0.54	0.85	0.86	0.86
Accuracy	0.54			0.86		

Note: W.A. is weighted average of features 0 and 1

The classification report for the first two decision boundaries.

The weighted averages of the precision, recall, and accuracy for the elliptical decision boundary are all closer to 1 than those of the linear decision boundary. This reiterates the idea that the elliptical dataset is the better decision boundary for the current data than the linear boundary.

However, the classification does not take into account a decision boundary's separating capacity for newly introduced data. Because it overfits the current data, the 20th-degree boundary (if analyzed using a classification report) would produce precision, recall, and accuracy values closer to 1 than any of the ratios for the elliptical boundary. To analyze predictive efficacy, the dataset can be split into a training set (to train the algorithm for creating the best-fitting decision boundary) and a validation set (to test the generated decision boundary). The algorithm used in the paper splits the dataset into 75% training and 25% validation. In order to quantifiably determine the best decision boundary, the binary cross entropy values for each degree decision boundary can be calculated and stored in two separate lists; the first, *J_list*, stores the BCE values of each model against the training sets, and the second, *J_deg*, stores the BCE values of each model against the validation set. The first seven degrees and their corresponding *J_train* and *J_deg* values are shown in the table below:

Degree	1	2	3	4	5	6	7
J_train	0.690773	0.353295	0.348901	0.307736	0.262820	0.251863	0.233709
J_deg	0.692732	0.341296	0.342659	0.431231	0.591593	0.592112	0.741034

The first seven values of *J_train* and *J_deg*.

For context, a larger BCE value means that there is a larger discrepancy between the decision boundary and its corresponding data point; in other words, the larger the BCE value, the more average error the degree decision boundary has. As the degree of the decision boundary increases, there are two different trends almost immediately established. For *J_train*, the BCE values continually decrease because boundaries of higher degree overfit the data they are trained against. For *J_deg*, the BCE values initially decrease then continue to increase. The initial decrease marks the shift from a linear decision boundary (which has been proven to be ineffective) and an elliptical decision boundary. However, the increase from the third degree onwards shows that there is significant predictive error for higher-degree decision boundaries when introduced to new data.

These trends once again prove that a 20th-degree decision boundary is an extremely overfitted and predictively inaccurate decision boundary for this dataset. In order to determine the best decision boundary based on these values (one that fits the current data and can accommodate for newly introduced data well), the degree corresponding to the lowest error value of *J_deg* should be chosen; in this case, that is the second-degree (elliptical) decision boundary.

Conclusion

Through a detailed example using a microchip dataset, the topics were practically analyzed using decision boundaries, feature engineering, classification analysis, and more. Altogether, the research collected and tested can stand as a basis for the effective simplicity of machine learning and its cross-industrial applications.

Acknowledgments

Professor Guillermo Goldsztein (Associate Academic Chair of the Department of Mathematics, Georgia Institute of Technology). We discussed and collaboratively researched statistical models, computer science concepts, and the

functions of Google Colaboratory, all of which I utilized for this project. He also guided me through the research process itself, helping me evolve my academic research interests and overall experiential outlook.

References

Burkov, Andriy. *The Hundred-Page Machine Learning Book*. Andriy Burkov, 2019.

Nargesian, Fatemeh, et al. "Learning Feature Engineering for Classification." *Ijcai*. 2017.

Trawiński, Bogdan, et al. "Nonparametric statistical analysis for multiple comparison of machine learning regression algorithms." *International Journal of Applied Mathematics and Computer Science* 22 (2012): 867-881.