Analyzing Decision Trees to Understand MNIST Misclassification

Alexis Comeau and Christopher McDonald
Department of Math and Computer Science
McDaniel College
Westminster, Maryland 21157, USA
{amc025, ccm009}@mcdaniel.edu

2019 IEEE MIT Undergraduate Research Technology Conference (URTC) | 978-1-7281-5818-1/19/$31.00 ©2019 IEEE | DOI: 10.1109/URTC49097.2019.9660504

***Abstract* – The MNIST dataset of handwritten digits is a classic tool for the study of machine learning. In this paper we use MNIST and a decision tree classifier to make a case study of misclassification for the digits three and five, analyzing the nodes of the decision tree plot to understand the process of classification. The paper concludes with a discussion of possibilities for our research and a look at some related, higher-level research that has been done.**

Index Terms* – *Machine Learning; Classification; MNIST; Decision Trees

I. Introduction

Machine learning algorithms and approaches to classification problems are becoming increasingly sophisticated, but tasks that seem simple can produce errors that are bizarre from a human perspective, such as mistaking two clearly distinct digits. Human brains have been taught to recognize the symbols of human communication, and our learning tools have been passed along, refined, and adapted through generations. Our group of student researchers began to study machine learning; we ascertained that the machine does not see the data the way we see it. We need to do more to understand the complexity of the process. The more we understand, the better we can refine our use of machine learning and preemptively address the consequences of any errors in their varied real-world applications, from spam filters and targeted marketing to digital check deposits and medical diagnosis.

The MNIST dataset is a widely-used and well-supported educational tool for the study of machine learning at various levels, and it served as the foundation for our research. It consists of 70,000 instances of scanned and preprocessed handwritten digits provided by high school students and employees of the US Census Bureau. They are labeled from 0 to 9, with 784 pixels (with values for pixel density ranging from 0 to 255) serving as the features [1].

We wanted to see and understand how a classifier trained and tested on the MNIST dataset makes its decisions, using tools like confusion matrices, feature importance graphs, and decision tree classifier plots to do so. This paper describes the information obtained during the last few weeks of the research period, where we used these tools to try and answer questions that we developed from our observations of the MNIST data.

We will discuss initial observations and our research questions, provide a description of the training and testing sets, and our reasoning for choosing the decision tree classifier as our predictive model. We then look at the feature importance graph and analyze the diagram of the decision tree for our training set. Finally, we conclude the paper with thoughts about the directions this research could take in the future and a brief overview of related works.

II. Misclassification Research Questions

We split the MNIST data into two sets: a training set consisting of the first 60,000 digits, and a testing set consisting of the last 10,000 digits. While testing various classifiers we noticed that certain digits were persistently mistaken for one another. Fig. 1 shows the confusion matrix for the MNIST dataset using a decision tree classifier.

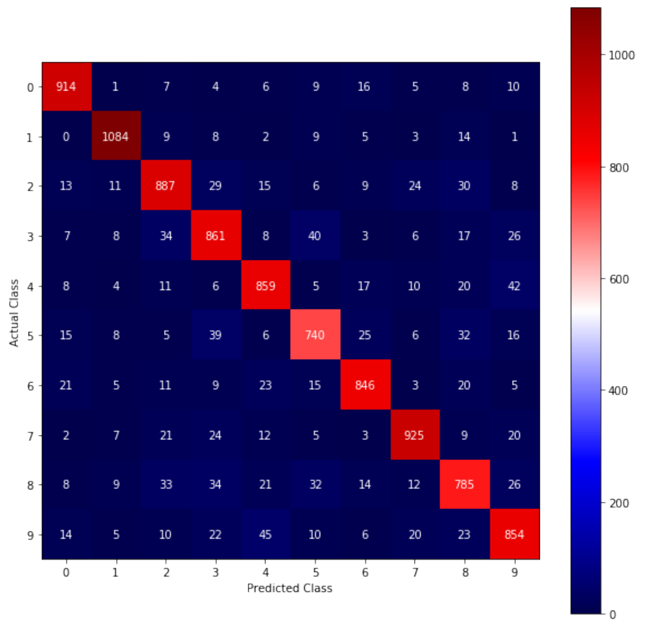

Fig. 1 The confusion matrix for the MNIST dataset trained and tested using a decision tree classifier.

From the matrix we see that there are 40 threes predicted as fives and that there are 39 fives predicted as threes. This shows that threes and fives are commonly misclassified with each other. These misclassifications also occur with other pairs of digits such as fours and nines. We decided to focus on threes and fives and set out to answer the following questions:

- Why does the classifier mistake threes for fives and fives for threes?
- Can we pinpoint the source or sources of these mistakes in the classification process?
- What changes can be made such that a misclassified digit gets classified correctly?

We wanted to crack open the black box to answer these questions.

III. Methods

As previously stated, we chose to train and test the classifiers on the digits three and five. Ideally there would be an even number of threes and fives in the MNIST dataset; however, this is not the case. The ratio of fives to threes is 6,313 to 7,141, about 0.884. We wanted to keep the ratio of fives to threes approximately the same throughout the training and testing sets to minimize sampling error. The threes and fives in the first 60,000 digits formed the training set, and those in the last 10,000 digits became the testing set. This worked out nicely: the ratio of fives to threes in the training set was 5,421 to 6,131, approximately 0.884 and the ratio of fives to threes in the testing set was 892 to 1,010, about 0.883.

We tested several classifiers on our dataset and found that the accuracy score for the random forest classifier was the highest (Table 1). Random forests are an example of an ensemble learning method, so-named because they take a group (or *ensemble*) of predictors and combine the results. This classifier is composed of a number of decision tree classifiers that are trained from random subsets of the training data, and the final output of the forest is the class most often predicted by the trees [2]. A decision tree is formed as the classifier splits its training data into smaller sets, and the decision points where these splits are made become the *nodes* of the decision tree that test instances pass through during classification. The structure of an individual decision tree, and how the nodes are defined, will be shown in more detail when we discuss our analysis of one in section IV.

Since its accuracy was the highest, we tried to use the random forest classifier and take a closer look at its process. We found that we could isolate the individual decision trees that composed the forest. With the utility to plot each of these decision trees, we hoped to see where classifications went wrong and how we might be able to correct them. However, this process would be tedious to do so with an entire forest, and not practical to look at them all together. Therefore, we used the decision tree classifier to get a picture of the decision-making process, and what we lost in the accuracy of the classifier we make up for with a greater level of detail.

Table 1
Comparison of different classifiers' accuracy scores

Predictive Models	**Accuracy**
Decision Tree Classifier	95.95%
Linear Support Vector Classifier	95.48%
Logistic Regression	95.95%
One Vs One Classifier	96.06%
Random Forest Classifier	98.37%

IV. Observations and Results

In order to understand how threes and fives overlapped from the perspective of the algorithm, we looked at the important features for the classifier trained on those digits (Fig. 2) and the decision tree (Fig. 3) that was built from the entire training set and visualized using Graphviz. Each node of the decision tree corresponds to a feature (pixel) of the feature importance graph. The more important pixels are the ones considered earliest in the process, in the highest nodes of the tree.

The node at the top of Fig. 3 is one of two that immediately follows from the root node (pixel325, which is the white pixel in Fig. 2). Each node asks whether the value of each *pixel**n*** is less than or equal to a threshold number, proceeding deeper down the path to the left if true and to the right if false. We can see that the number of *samples* for which the condition "pixel319 <= 70.5" is true is 5143. This is out of the 7304 samples that were false for the condition in the root node, which in turn had a number of samples equal to the number of threes and fives in the training set (11,552 instances).

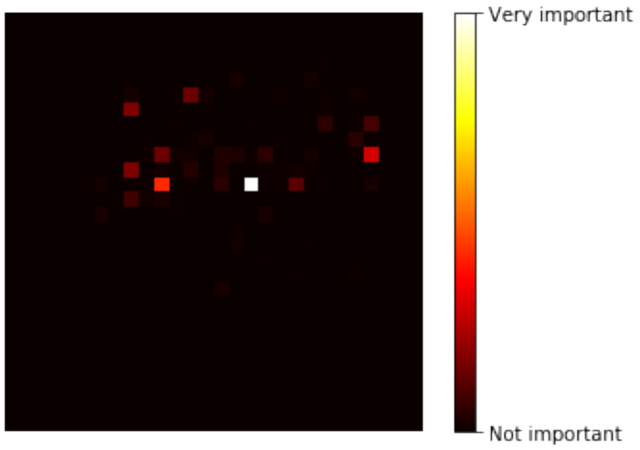

Fig. 2 Feature importance graph of threes and fives trained using a decision tree classifier.

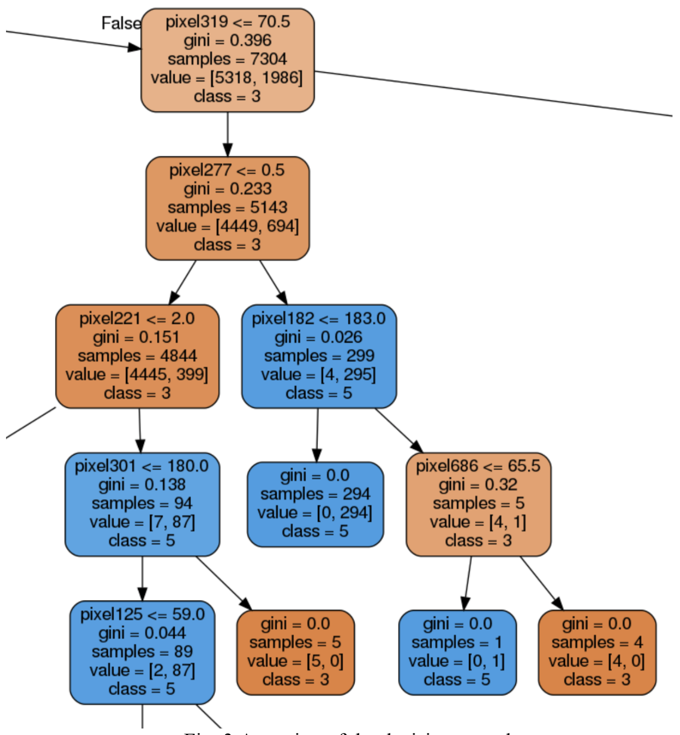

Fig. 3 A portion of the decision tree plot.

The *gini* score is a measure of class purity in the nodes [3], and in more detail *values* tells us the distribution of the samples between each class. For two classes, the gini score will vary between 0 (all of the samples belong to one class) and .5 (there are an even number of each class in the samples).

The *class* of each node shows the predicted class at that interval, determined by which class has plurality in the samples (or in this case, just a majority). There is not an actual prediction until one of the smaller leaf nodes is reached and a digit's classification is finalized. The classification of a given digit is the result, then, of a sequence of decisions; the most important feature is simply the place that classification starts and the biggest split can be made. It is the only pixel that is guaranteed to be considered.

Changing a pixel or a few pixels of a misclassified digit can change the classification of that digit. A misclassified digit's path can be changed anywhere along the tree and you can get the digit to be classified correctly.

In Fig. 4, we show a five from the testing set that was misclassified as a three (figure on the left). We also show the five where pixel325 was changed (figure on the right). By changing this one pixel, the digit becomes correctly classified as a five.

Similarly, in Fig. 5, we show a three from the testing set that was originally misclassified as a five (figure on the left) and the three where we changed pixel95 (figure on the right). This resulted in the three being correctly classified.

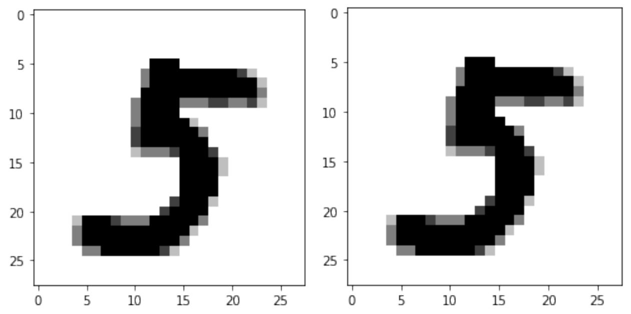

Fig. 4 On the left is a five from the testing set that was wrongly classified and on the right is the modified five that was correctly classified.

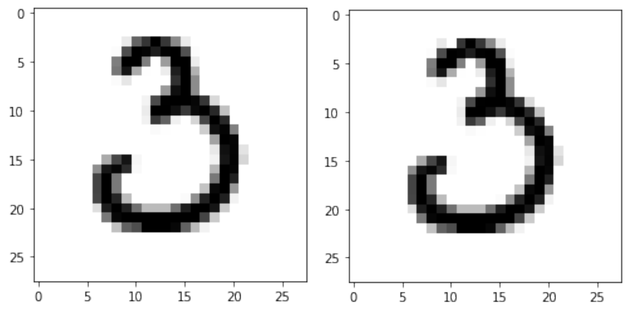

Fig. 5 On the left is a three from the testing set that was wrongly classified and on the right is the modified three that was correctly classified.

As you can see the changes that we made to the images were small and are barely noticeable, however, the results after altering the digits were significant.

V. CONCLUSION

Earlier in our research, when we began to look more specifically at MNIST, our group tried to improve the performance of our classifiers in broad strokes. We created altered or expanded versions of MNIST with all the digits shifted, darkened, lightened, with binary pixel values, etc. The results varied, and did not provide insight as to how the classifiers made decisions.

Narrowing our focus to one algorithm's classification between two particular digit classes helped us to understand the mechanics of classification a little more concretely; we could see how specific changes could produce different results, and how the machine sees the instances.

The following are some possible directions we could take in the future:

We would have liked to find a way to parse the dot file generated for a decision tree in order to automate the identification and manipulation of misclassified digits, rather than manually identifying the sequence of nodes and changing the pixel values accordingly. We would also like to learn more about the training process for the decision tree classifier and how it determines the particular threshold for each node.

VI. RELATED WORKS

Here we briefly discuss a small number of more advanced works that caught our attention as we finalized our work. We could not have used them as the foundation for our basic research or meaningfully expanded on them, but these works provide context on some of the real-world applications that we refer to in the introduction and present intriguing perspectives and solutions concerning misclassification that could guide our future learning.

For the problem of interpretation and correction that we tried to address by visualizing decision trees, the authors of Reference [4] define a *correctable learning* approach to classification measuring the *correctability* of an algorithm in response to feedback. They found that they were able to improve results as well as the ease of interpreting results which are achieved more quickly through independently trained local models that restrict the impact of mistakes and allow greater parallelization [4].

With a similar focus on efficient correction, Reference [5] discusses the use of *peer networks* which, rather than independently producing results from which the mode is taken (like the random forest classifier), correct and retrain each other. The authors found that they could efficiently locate mistakes, glean the faults in the decision-making process, and generate new instances that would be misclassified [5].

In the context of broader research on defending system security and high-stakes decision-making from adversarial attacks that seek to manipulate classifiers, Reference [6] distinguishes between "taking secure decisions as opposed to making a classification systems robust" with the concept of two-layered *bounded misclassification* which insulated decision-making from classification [6]. The authors found that more work was needed, especially "to make sure that bounded misclassification is achieved on top of and not in place of accuracy" when dealing with random noise [6]. Nonetheless, the *weighted-loss function* neural network based on their concept showed significant improved (if still far from optimal) results over a typical Deep Neural Network when both are faced with adversarial input.

Directly or indirectly, most of the papers we looked at sought to address security problems created by intentional, *adversarial* efforts to deceive or manipulate classifiers, rather than the results of routine misclassification. For example, one of the papers mentions the classification of "digits on a bank cheque" and the "recognition of objects on the road" by a self-driving car; the consequences can range from huge losses from fraud to the loss of life [6].

Although the papers are at a higher level of learning and methodological complexity, the problems they address can be related to our basic questions about digit misclassification and improve our understanding of the real-world dilemmas related to misclassification and the means by which they are addressed.

ACKNOWLEDGMENT

The authors would like to thank Dr. Chia-En Lin and our fellow researchers, SoYoung Jeon and Mason White, for the opportunity to research machine learning with them.

REFERENCES

[1] A. Géron, *Hands-On Machine Learning with Scikit-learn & Tensorflow:Concepts, Tools, and Techniques*, 1st ed., Sebastopol: O'Reilly Media, Inc., 2017, p. 81.

[2] A. Géron, *Hands-On Machine Learning with Scikit-learn & Tensorflow:Concepts, Tools, and Techniques*, 1st ed., Sebastopol: O'Reilly Media, Inc., 2017, p. 183.

[3] F. Ceballos, "Scikit-Learn Decision Trees Explained: Training, Visualizing, and Making Predictions with Decision Trees," February 22, 2019, https://towardsdatascience.com/scikit-learn-decision-trees-explained-803f3812290d.

[4] K. Raman, K. M. Svore, R. Gilad-Bachrach, and C. J. C. Burges, "Learning from Mistakes: Towards a Correctable Learning Algorithm," CIKM '12: Proceedings of the 21st ACM international conference on Information and knowledge management, pp. 1930-1934, October 2012. doi: 10.1145/2396761.2398546

[5] S. Baluja, M. Covell, and R. Sukthanar, "The Virtues of Peer Pressure: A Simple Method for Discovering High-Value Mistakes," in Computer Analysis of Images and Patterns: 16th International Conference, CAIP 2015, Valletta, Malta, September 2-4, 2015 Proceedings, Part I, pp. 96-108, G. Azzopardi and N. Petkov, Eds., Springer, 2015. doi: 10.1007/978-3-319-23117-4_9

[6] S. Sengupta, A. Dudley, T. Chakraborti, and Subbarao Kambhampati, "An Investigation of Bounded Misclassification for Operational Security of Deep Neural Networks," presented at AAAI-18 Workshop on Engineering Secure and Dependable Machine Learning Systems, 2018. Retrieved from https://sailik1991.github.io/files/aaai_wlf_bounded_misclassification.pdf on August 2, 2019.