Supervised Machine Learning for Color Classification

Benjamin Hall

***Abstract*—Color classification is typically achieved using hand-tuned algorithms. However, the algorithms generally need to be retuned when the environment changes, and the simple boundaries produced may not be sufficiently accurate. This study analyzes the effectiveness of using supervised machine learning to replace these traditional algorithms. The k-nearest neighbor, Bayesian linear classifier, and multiclass single-layer perceptron (SLP) algorithms were tested. Additionally, the colors were tested in both the RGB format and the complex HSV format, where numbers on the complex unit circle are used to represent hue. Results show that a multiclass SLP, adjusted to support complex numbers, can most accurately and efficiently classify colors, and the complex HSV format outperforms RGB with the data tested.**

Further research is needed to prevent the SLP algorithm from entering a local minimum, as well as to compare its accuracy on the RGB and complex HSV formats with more advanced color groups, such as classifying into various categories of blue.

I. Introduction

COLOR classification is used extensively in robotics and image processing. Companies use it to automate the detection of certain product defects. Autonomous vehicles use color classification to help detect road signs, determine the color of a traffic light, and so on. FIRST robotics teams use color detection for vision targeting and autonomous robot operations.

Color classification can be done with fairly reasonable accuracy using image processing and fairly simple algorithms, such as checking if the HSV values of a color are within a certain range. However, the image processing can be an expensive operation, and the simple algorithms often need to be retuned if the environment changes, such as with new lighting. Additionally, the algorithms typically only target one or two colors, as hand-tuning the algorithm becomes more time-consuming and prone to error as the list of colors grows. Furthermore, the algorithms often draw simple, rectangular boundaries between color classifications.

AI provides an alternative method of color classification. Rather than relying on hand-tuning of algorithms whenever the environment changes and dealing with human error, an AI based on supervised machine learning can efficiently tune an algorithm to identify colors given enough training data. This can result in massive time savings, as humans now only need to collect data samples to feed the AI, and it has the potential to increase the accuracy of the algorithm, as it can create advanced, mathematically optimized classification boundaries.

II. Technical Discussion

When it comes to color classification, there are several supervised machine learning algorithms available, and there are multiple methods to digitally represent colors, many of which involve nonlinear transformations.

A. Supervised Machine Learning

There are three fairly simple supervised machine learning algorithms that stand out: k-nearest neighbors, the Bayesian linear classifier, and the single-layer perceptron algorithm.

k-nearest neighbors is a very simple algorithm to implement and can handle nonlinear classification boundaries. The algorithm calculates the distance between the test data point and each training data point, and then it partial sorts those distances until it has the k nearest neighbors. The data gets whichever classification is most prominent among those neighbors. However, the algorithm slows down as the training data set grows, as both calculating the distances and performing the partial sort take linear time relative to the size of the training data. The algorithm also has a high memory cost, as it needs to keep the training data in memory during the entire program runtime to perform the distance calculations and partial sort.

The Bayesian linear classifier fixes both the time and memory costs by calculating ahead of time the mean vectors of the classifications. The algorithm only needs to keep a vector of classification means during program runtime, and classifying a test data point is simple: for each classification, take the dot product of the data point with the classification's mean vector and add the result of the classification's mean vector dotted with itself. The data point then gets the classification with the largest output. However, the algorithm assumes a linear boundary between classifications, and it assumes that the data's class conditional density functions follow a Gaussian distribution [1], which is often not the case.

The single-layer perceptron algorithm relaxes the assumption of a Gaussian distribution to provide a more general linear classifier. Instead of using a vector of means, the algorithm trains a vector of weights by testing the training points against the weights, calculating the error, and adjusting the weights accordingly. In its simplest form, the SLP only splits data into two classifications using a linear boundary and a single vector, assigning positive outputs to one class and negative outputs to the other. However, the algorithm can also be generalized into a multiclass algorithm by storing weight vectors for each classification, as is the case with the Bayesian linear classifier.

B. Color Representation

There are many ways to digitally represent colors, but the two most interesting from a classification perspective are RGB and HSV. The data read from a color sensor or a camera pixel

is represented in RGB, as most cameras and color sensors use red, green, and blue sensors to convert an image to a digital format. However, HSV more accurately represents how we humans perceive colors: a hue from red to violet, a saturation from gray to colorful, and a value from dark to light. The transformation from RGB to HSV is also nonlinear, so it is possible that one of the two results in linear boundaries between color classifications, while the other requires more complicated boundaries.

A color can be converted from RGB to HSV by firstly calculating the chroma of the RGB value:

$$M = \max(R, G, B)$$
$$m = \min(R, G, B)$$
$$C = M - m$$

From there, the HSV value can be calculated [2]:

$$V = M$$
$$H = 60^\circ \times \begin{cases} 0, & \text{if } C = 0 \\ \dfrac{G-B}{C} \mod 6, & \text{if } V = R \\ \dfrac{B-R}{C} + 2, & \text{if } V = G \\ \dfrac{R-G}{C} + 4, & \text{if } V = B \end{cases}$$
$$S = \begin{cases} 0, & \text{if } V = 0 \\ \dfrac{C}{V}, & \text{otherwise} \end{cases}$$

Since hue is an angle, it can also be converted from a non-continuous range to the (continuous) unit circle by mapping it to the complex plane using Euler's formula:

$$e^{i\theta} = \cos(\theta) + i\sin(\theta)$$

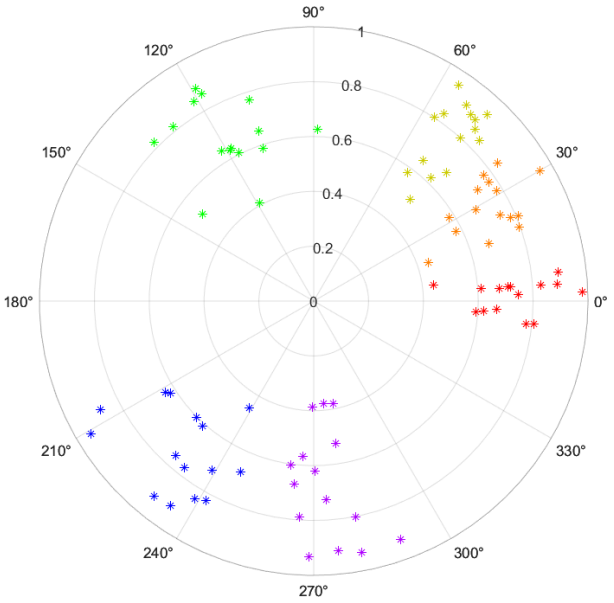

Fig. 1. MATLAB [3] plot of training data HSV hues (θ) and saturations (r) in polar form, colored by classification.

III. Results

A training dataset of 90 RGB colors was collected and classified into six groups: red, orange, yellow, green, blue, and purple. Plotting the data in MATLAB (see Appendix B), there are some noticeable clusters of colors, but the groups are not cleanly separated by linear boundaries.

The data was then converted into the HSV format and plotted in MATLAB (see Appendix B). Unlike the RGB colors, the HSV colors are clearly divided by hue with the classifications used. As a result, we can hypothesize that the linear classifiers will perform better with the HSV colors than the RGB colors with the given classifications. It is important

TABLE I
RGB K-NEAREST NEIGHBOR CLASSIFICATION RESULTS AT K=3.

Color	Classified	Misclassified	Success Rate
Red	3	0	1.00
Orange	3	0	1.00
Yellow	3	0	1.00
Green	3	0	1.00
Blue	3	0	1.00
Purple	3	0	1.00
Total	**18**	**0**	**1.00**

TABLE II
RGB K-NEAREST NEIGHBOR CLASSIFICATION RESULTS AT K=5.

Color	Classified	Misclassified	Success Rate
Red	3	0	1.00
Orange	2	1 (1 yellow)	0.67
Yellow	3	0	1.00
Green	3	0	1.00
Blue	3	0	1.00
Purple	3	0	1.00
Total	**17**	**1**	**0.94**

TABLE III
HSV K-NEAREST NEIGHBOR CLASSIFICATION RESULTS AT K=3.

Color	Classified	Misclassified	Success Rate
Red	3	0	1.00
Orange	3	0	1.00
Yellow	3	0	1.00
Green	3	0	1.00
Blue	3	0	1.00
Purple	3	0	1.00
Total	**18**	**0**	**1.00**

TABLE IV
HSV K-NEAREST NEIGHBOR CLASSIFICATION RESULTS AT K=5.

Color	Classified	Misclassified	Success Rate
Red	3	0	1.00
Orange	3	0	1.00
Yellow	3	0	1.00
Green	3	0	1.00
Blue	3	0	1.00
Purple	3	0	1.00
Total	**18**	**0**	**1.00**

TABLE V
RGB BAYESIAN CLASSIFICATION RESULTS.

Color	Classified	Misclassified	Success Rate
Red	3	0	1.00
Orange	2	1 (1 yellow)	0.67
Yellow	2	1 (1 orange)	0.67
Green	3	0	1.00
Blue	3	0	1.00
Purple	3	0	1.00
Total	**16**	**2**	**0.89**

TABLE VI
HSV BAYESIAN CLASSIFICATION RESULTS.

Color	Classified	Misclassified	Success Rate
Red	0	3 (2 orange, 1 purple)	0.00
Orange	2	1 (1 yellow)	0.67
Yellow	3	0	1.00
Green	2	1 (1 red)	0.67
Blue	3	0	1.00
Purple	3	0	1.00
Total	**13**	**5**	**0.72**

TABLE VII
COMPLEX HSV BAYESIAN CLASSIFICATION RESULTS.

Color	Classified	Misclassified	Success Rate
Red	3	0	1.00
Orange	2	1 (1 yellow)	0.67
Yellow	3	0	1.00
Green	3	0	1.00
Blue	3	0	1.00
Purple	3	0	1.00
Total	**17**	**1**	**0.94**

to note that some of the red colors sit close to 0 degrees on the hue axis, while others sit close to 360 degrees. In reality, the reds are very close in hue, as shown in Fig. 1. As a result, any algorithm that does not handle wrapping at 360 degrees may misclassify red, as the mean of the data points is roughly that of the green colors.

A small test set of 18 colors (three per classification) was collected and saved in both RGB and HSV formats. The training and test data for both RGB and HSV were then fed into the k-nearest neighbor algorithm (see Appendix A) at k=3 and k=5, with the results shown in Table I to IV. An example program output is shown in Appendix C. With the given test data, the HSV data was classified with a perfect success rate, as was the RGB data at k=3. The RGB data misclassified one orange color as yellow at k=5, but the success rate was still high at 94%. Given that both the RGB and HSV data could be accurately classified by the k-nearest neighbor algorithm, both data sets were tested in the other algorithms.

From there, both data sets were tested in the Bayesian linear classifier, the results of which are shown in Table V and VI.

Since the Bayesian algorithm relies on means, the HSV data set performs poorly, as the red's hue wrapping at 360 is not handled correctly. This can be resolved by mapping the hues to the complex plane. Doing so requires adjustments to the Bayesian algorithm (see Appendix A):

- The means used by the algorithm need to be the conjugate of the training data means. When a test data point closely matches the classification, $e^{i\theta} * e^{-i\theta} = 1$, so the data is rotated onto the positive real plane. When a test data point does not closely match the classification, the data is rotated away from the positive real plane. As a result, the real component of the dot product can be used to determine classification.
- The mean offset μ_0 needs to be calculated by taking the dot product of the vector of means with the vector of mean conjugates, resulting in a real-valued offset.

The data was then converted to complex HSV values and run through the Bayesian classifier, the results of which are shown in Table VII. This results in more reasonable classifications, so the complex HSV values were used instead of the regular HSV values in the remaining algorithms.

For both the RGB data in Table V and the complex HSV data in Table VII, the Bayesian classifier achieved a high success rate (89% and 94%, respectively), but there is still room for improvement given the success rate of the k-nearest neighbor algorithm. As a result, the single-layer perceptron algorithm was tested on both data sets.

Before running a multiclass SLP algorithm, however, a subset of the data was run through a standard, binary class SLP algorithm to test the behavior of the system.

Just like with the Bayesian algorithm, the SLP algorithm requires adjustments to support complex numbers:

- When training the weights, the conjugates of the training data point components are used to adjust the weight. Once again, this makes it so data points that match the classification are rotated towards the positive real axis, strengthening their classification values.
- When running the classifier on test data, the real component of the calculated classification value is used to determine classification.

The red and green classifications were chosen for testing. While testing the data, it was observed that the weights would occasionally fail to converge or oscillate around convergence.

TABLE VIII
RGB BINARY SLP CLASSIFICATION RESULTS, $\eta = 1$.

Color	Classified	Misclassified	Success Rate
Red	3	0	1.00
Green	3	0	1.00
Total	**6**	**0**	**1.00**

TABLE IX
COMPLEX HSV BINARY SLP CLASSIFICATION RESULTS, $\eta = 1$.

Color	Classified	Misclassified	Success Rate
Red	3	0	1.00
Green	3	0	1.00
Total	**6**	**0**	**1.00**

TABLE X
RGB MULTICLASS SLP CLASSIFICATION RESULTS, $\eta = 1$.

Color	Classified	Misclassified	Success Rate
Red	9	0	1.00
Orange	8	1 (1 yellow)	0.89
Yellow	9	0	1.00
Green	8	1 (1 yellow)	0.89
Blue	9	0	1.00
Purple	9	0	1.00
Total	**52**	**2**	**0.96**

TABLE XI
COMPLEX HSV MULTICLASS SLP CLASSIFICATION RESULTS, $\eta = 1$.

Color	Classified	Misclassified	Success Rate
Red	9	0	1.00
Orange	9	0	1.00
Yellow	9	0	1.00
Green	9	0	1.00
Blue	9	0	1.00
Purple	9	0	1.00
Total	**54**	**0**	**1.00**

After adjusting the algorithm to subtract the mean of the entire testing data set from all data points, the behavior improved significantly. The results of the algorithm are shown in Table VIII and IX.

As can be seen with the complex HSV data, the adjustments to account for complex numbers were successful. Additionally, all data was successfully classified, so it was worth proceeding with the multiclass SLP algorithm.

Compared to the binary SLP algorithm, the multiclass SLP algorithm stores multiple weight vectors—one for each classification—and utilizes the "one-vs.-rest" [4] method of training the weights. While training, the data point is tested against all the weight vectors, and the classification with the largest calculated value is assigned to the data point. If the classification is correct, then the weights are not adjusted. Otherwise, the weights of the correct class are increased by adding $\eta(k)\,\mathbf{y}_k$, where $\eta(k)$ is the learning rate and $\mathbf{y}_k$ is the training data point, and the weights of the other classes are decreased by subtracting that same value. When testing data against the weights, whichever classification has the largest calculated value is assigned to the data point.

Once again, the algorithm requires adjustments to support complex numbers (see Appendix A):

- When training the weights, the conjugates of the data point components are used to adjust the weights.
- The real component of the calculated classification value is used to determine the classification of data points.

The results of running the multiclass SLP algorithm are shown in Table X and XI. Each test was run three times to account for the initial random state of the weights.

For both the RGB and the complex HSV data sets, the multiclass SLP classifier shows a significantly better classification rate for the given training and test data than the Bayesian classifier. Additionally, the complex HSV data set has a perfect success rate with the given classifications.

IV. CONCLUSION

Looking at the results in Table X and XI, the multiclass SLP algorithm is an accurate and efficient method of classifying colors. Comparing the multiclass SLP results to those of the Bayesian algorithm in Table V and VII, it is safe to assume that in the general use case, color data's class conditional density functions do not perfectly follow a Gaussian distribution [1]. Additionally, comparing the results of Table X and XI show that mapping RGB colors to complex HSV colors can result in better linear classification.

One issue with the implementation of the SLP algorithms in this paper is that the algorithm often falls into a local minimum. Because of the random starting weights, the algorithm is not guaranteed to find the absolute minimum, resulting in the algorithm varying in accuracy each time it is retrained. This was observed with the RGB multiclass results in Table X; one run classified everything perfectly, while the other two each misclassified one different data point. This can be improved by adding momentum to the weight adjustments during training.

There is also room for further research on the accuracy of classification using RGB colors vs. complex HSV colors. The classifications used in this paper strongly lined up with hue, but it is possible that a different classification system, such as differentiating between several types of blue, will perform better using RGB than with HSV.

APPENDIX A
SUPERVISED MACHINE LEARNING ALGORITHMS

The k-nearest neighbor algorithm is shown in Figure 2. The Bayesian linear classifier algorithm is shown in Figure 3. Lastly, the multiclass single-layer perceptron algorithm is shown in Figure 4.

The Bayesian linear classifier and the single-layer perceptron algorithms have been adjusted to support complex data.

APPENDIX B
PLOTS OF TRAINING DATA

Fig. 5 shows plots of the RGB training data, and Fig. 6 shows plots of the HSV training data.

Algorithm k-nearest neighbors

Input : T, a set of training data points;
$\mathbf{d}$, the test data point;
k, number of nearest neighbors;
Output: The classification guess of d;

compute distances between $\mathbf{d}$ and all $\mathbf{t} \in T$;
partial sort k smallest computed distances;
select k-nearest training points corresponding to k-smallest distances;
count number of occurrences of each classification among k-nearest neighbors;
return the most frequent classification;

Fig. 2. k-nearest neighbors algorithm.

Algorithm Bayesian linear classifier

Input : T, a set of training data points;
$\mathbf{d}$, the test data point;
Output: The classification guess of $\mathbf{d}$;

▷ *Training*

for each classification c_i of $\mathbf{t} \in T$ **do**
  $\mathbf{u}_i \leftarrow \text{mean}\left(\{\mathbf{t} \in T : c_\mathbf{t} = c_i\}\right)$;
  $\mathbf{u}_i^* \leftarrow \forall u_j \in \mathbf{u}_i, u_j \mapsto u_j^*$;
  $(\mathbf{w}_i, w_{i0}) \leftarrow \left(2\mathbf{u}_i^*, \mathbf{u}_i^T \mathbf{u}_i^*\right)$;

▷ *Testing*

for all $(\mathbf{w}_i, w_{i0})$ **do**
  $g_i \leftarrow \mathbf{w}_i^T \mathbf{d} - w_{i0}$;
return classification c_i of the max computed $\text{Re}\left(g_i\right)$;

Fig. 3. Bayesian linear classifier algorithm.

APPENDIX C
PROGRAM DEMONSTRATION

Fig. 7 shows an example of the output from running the multiclass SLP program.

APPENDIX D
PROGRAM SOURCE CODE AND DATA FILES

The data in this study is plotted in MATLAB [3], and the algorithms are written in Rust. The source code and data files are available in benjamin_hall_color_classification.zip.

The data files used in the algorithms are as follows:

- k-nearest neighbor
  - RGB
    - color_training_full.dat
    - color_test_full.dat
  - HSV
    - color_training_full_hsv.dat
    - color_test_full_hsv.dat
- Bayesian
  - RGB
    - color_training_full.dat
    - color_test_full.dat
  - HSV
    - color_training_full_hsv.dat
    - color_test_full_hsv.dat
  - Complex HSV
    - color_training_full_hsv_complex.dat
    - color_test_full_hsv_complex.dat
- Binary SLP
  - RGB
    - color_training_binary.dat
    - color_test_binary.dat

Algorithm Multiclass single-layer perceptron

Input : T, a set of training data points;
$\mathbf{d}$, the test data point;
η, the learning rate;
ϑ, the threshold to stop training;
Output: The classification guess of $\mathbf{d}$;

Require: $\forall \mathbf{t} \in T, \mathbf{t} = \langle t_1, t_2, \ldots, t_n \rangle$;
$\forall \mathbf{t} \in T, c_\mathbf{t} =$ classification of $\mathbf{t}$;
$\mathbf{d} = \langle d_1, d_2, \ldots, d_n \rangle$;
$0 < \eta \leq 1$;
$\vartheta \geq 0$

for each classification c_i of $\mathbf{t} \in T$ **do**
  $\mathbf{w}_i \leftarrow (n+1)$-dimensional vector of random weights;

▷ *Training*

for $iter$ in $1 \ldots MaxIter$ **do**
  $cnt_{bad} \leftarrow 0$;
  for $\mathbf{t} \in T$ **do**
    $\mathbf{y} \leftarrow \langle 1, t_1, t_2, \ldots, t_n \rangle$;
    for $\mathbf{w}_i$ **do**
      $g_i \leftarrow \mathbf{w}_i^T \mathbf{y}$;
    $c_{guess} \leftarrow c_i$ of the max computed $\text{Re}\left(g_i\right)$;
    if $c_{guess} \neq c_\mathbf{t}$ **then**
      $cnt_{bad} \leftarrow cnt_{bad} + 1$;
      $\mathbf{y}^* \leftarrow \forall y_j \in \mathbf{y}, y_j \mapsto y_j^*$;
      $\mathbf{w}_{adj} \leftarrow \eta \mathbf{y}^*$;
      for each classification c_i of $\mathbf{t} \in T$ **do**
        if $c_i = c_\mathbf{t}$ **then**
          $\mathbf{w}_i \leftarrow \mathbf{w}_i + \mathbf{w}_{adj}$;
        else
          $\mathbf{w}_i \leftarrow \mathbf{w}_i - \mathbf{w}_{adj}$;
  if $cnt_{bad} \leq \vartheta$ **then**
    break;

▷ *Testing*

for all $\mathbf{w}_i$ **do**
  $\mathbf{y} \leftarrow \langle 1, d_1, d_2, \ldots, d_n \rangle$;
  $g_i \leftarrow \mathbf{w}_i^T \mathbf{y}$;
return classification c_i of the max computed $\text{Re}\left(g_i\right)$;

Fig. 4. Multiclass single-layer perceptron algorithm.

  - Complex HSV
    - color_training_binary_hsv_complex.dat
    - color_test_binary_hsv_complex.dat
- Multiclass SLP
  - RGB
    - color_training_full.dat
    - color_test_full.dat
  - Complex HSV
    - color_training_full_hsv_complex.dat
    - color_test_full_hsv_complex.dat

REFERENCES

[1] M. Farmer, "CS-481 Lecture 15: Learning and Connectionist Learning," [PDF], 2023, available: https://kettering.blackboard.com/bbcswebdav/pid-861079-dt-content-rid-6079321_1/xid-6079321_1, accessed Mar. 20, 2023.

[2] Wikipedia. (2023) HSL and HSV. Accessed Mar. 20, 2023. [Online]. Available: https://en.wikipedia.org/wiki/HSL_and_HSV

[3] The MathWorks, Inc. (2022) MATLAB version: 9.13.0 (R2022b). Accessed Mar. 20, 2023. [Online]. Available: https://www.mathworks.com

[4] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006, ch. 3, pp. 182–184.

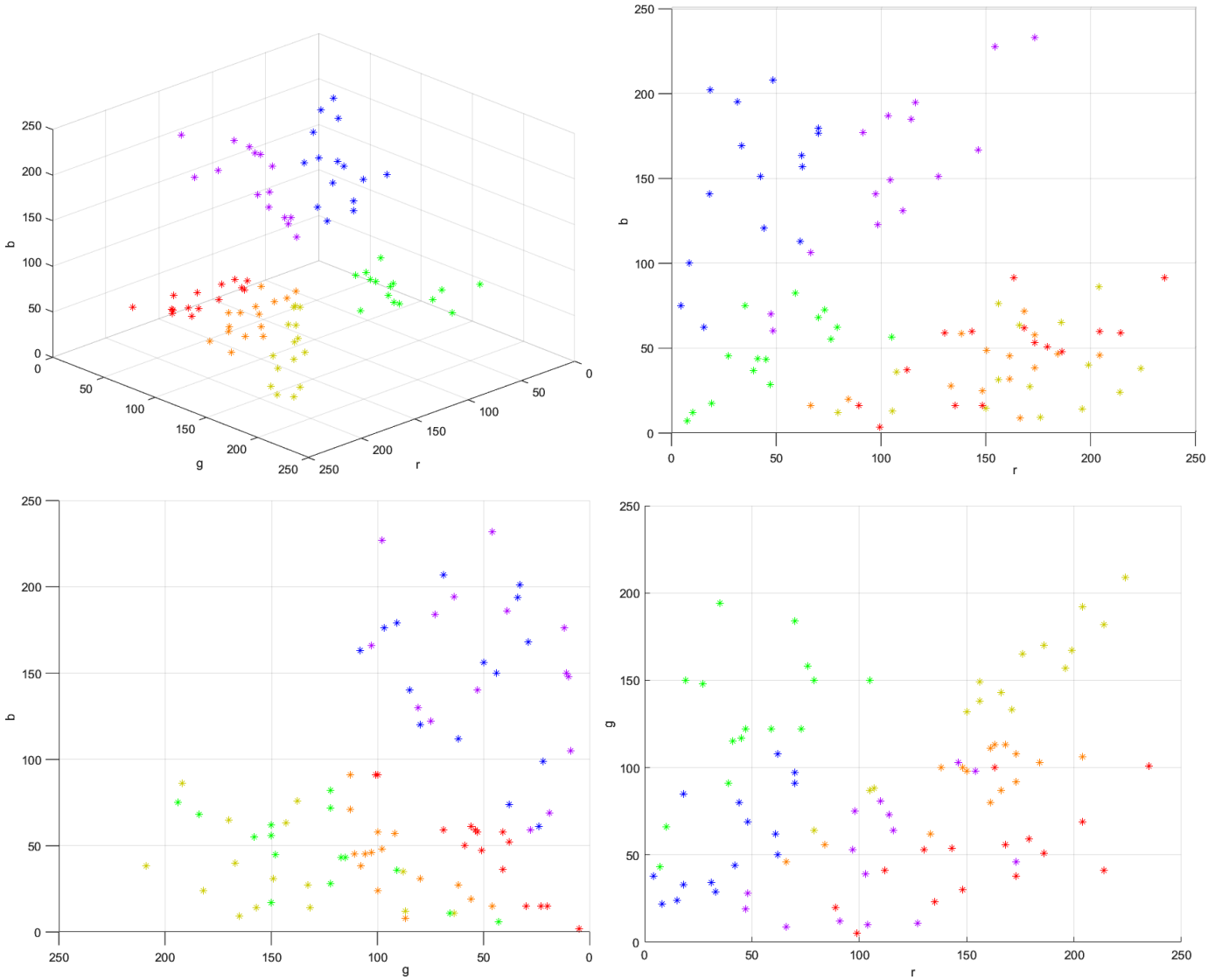

Fig. 5. MATLAB [3] plot of training data RGB values, colored by classification.

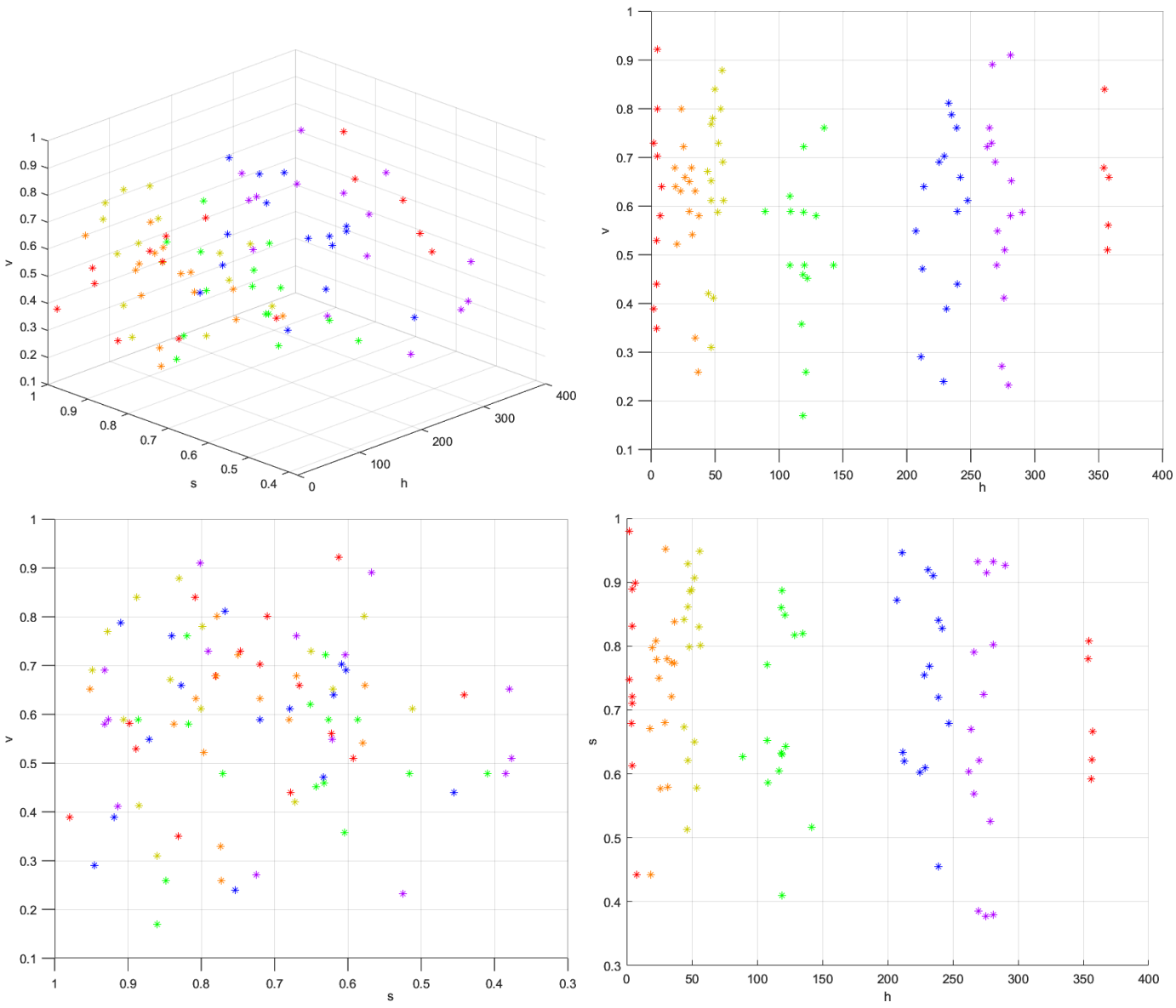

Fig. 6. MATLAB [3] plot of training data HSV values, colored by classification.

```
PS color_classification> ./target/release/color_class_slp color_training_full_hsv_complex.dat color_test_full_hsv_complex.dat
[
    Classification {
        data: DataPoint {
            point: [0.9951572723309918 + 0.09829548985960711i, 0.8181818181818182, 0.5607843137254902],
            class: "red",
        },
        class_guess: "red",
    },
    Classification {
        data: DataPoint {
            point: [0.9999204087556838 + 0.012616503234503532i, 0.6014492753623188, 0.5411764705882353],
            class: "red",
        },
        class_guess: "red",
    },
    Classification {
        data: DataPoint {
            point: [0.9948217482960331 - 0.10163507818280141i, 0.7422680412371134, 0.3803921568627451],
            class: "red",
        },
        class_guess: "red",
    },
    Classification {
        data: DataPoint {
            point: [-0.3133439679848821 + 0.94963969890031931, 0.5287356321839081, 0.3411764705882353],
            class: "green",
        },
        class_guess: "green",
    },
    Classification {
        data: DataPoint {
            point: [-0.76604444311897761 + 0.64278760968653981, 0.588785046728972, 0.4196078431372549],
            class: "green",
        },
        class_guess: "green",
    },
    Classification {
        data: DataPoint {
            point: [-0.039400459213775531 + 0.99922350043108151, 0.8857142857142857, 0.4117647058823529],
            class: "green",
        },
        class_guess: "green",
    },
    Classification {
        data: DataPoint {
            point: [-0.74314482547739421 - 0.66913060635885821, 0.6862745098039216, 0.4],
            class: "blue",
        },
        class_guess: "blue",
    },
    Classification {
        data: DataPoint {
            point: [-0.55864676580365261 - 0.82940568545020181, 0.6708860759493671, 0.6196078431372549],
            class: "blue",
        },
        class_guess: "blue",
    },
    Classification {
        data: DataPoint {
            point: [-0.96109563291487451 - 0.27621582936529311, 0.7204968944099378, 0.6313725490196078],
            class: "blue",
```

Fig. 7. Example program output of the multiclass SLP using complex HSV colors.