Using Logistic Regression Method for Color Segmentation:

Yushen Bai
University of California, San Diego
yub025@eng.ucsd.edu

Abstract: This report presents an approach for color segmentation by logistic regression. In this work, we first obtained a training dataset. Then we trained the logistic regression model to label the region blue or not blue. Finally, we drew a bounding box of blue barrel. The results showed that it was a nice try.

Index terms: color segmentation, logistic regression

I. Introduction

Recently, implement of computer vision is more and more popular. A fundamental capability of computer vision system forming interpretations in terms of components of image is to segment in to several disjoint regions processing uniform characteristics. Color segmentation is the process of partitioning a digital image into multiple segments by different colors. It simplifies the vision problem by assuming that objects are colored distinctively, and that only gross color differences matter. When applying this approach, it will result in the simplified image which can be processed very rapidly. Color segmentation is very important in a lot of fields, such as robot motion.

In this report, logistic regression model was used for the training dataset. Two labels were set for the model (blue, not blue). The goal was to get the bounding box of the blue barrel in the test image. The mathematical method was introduced in Section II. The approach to color segmentation and barrel detection was described in Section III. Section IV shows the results and discussion.

II. Problem Formulations

The object we are going to find is the blue barrel in the images. We defined 2 classes of color: blue and not blue. For each example images, we obtain a training dataset D={x_i, y_i}, where x is a vector of one pixel's RGB data, y is the coordinate label. In the labels, y=1 is blue and y=-1 is not blue.

We used a discriminative model p(y | X, w) for the discrete labels y. That is a product of sigmoid functions:

$$\mathrm{p(y|X,w)} = \prod_{i=1}^{n} \sigma\left(y_i x_i^T w\right) = \prod_{i=1}^{n} \frac{1}{1+\exp(-y_i x_i^T w)}$$

It leads to maximum likelihood estimation for w:

$$w_{MLE} = argmax_w \ \log\left(\mathrm{p(y|X,w)}\right) = argmin_w \sum_{i=1}^{n} \log(1+\exp(-y_i x_i^T w))$$

It is clear that $\nabla_w(-\log(\mathrm{p(y|X,w)}))$=0 does not have a closed-form solution. Consider the composition of an affine function $f_i(w) = -y_i x_i^T w$ and a convex function: $g(z) = \log(1 + e^z)$, and it is convex. Besides, the sum of convex functions $\sum_{i=1}^{n} g(f_i(w))$ is convex. Thus, the negative log-likelihood can be minimized iteratively to obtain a global minimum:

$$w_{MLE}^{t+1} = w_{MLE}^{t} - \alpha \nabla_w \left(-\log\left(p(y|X, w)\right)\right) \mid_{w=w_{MLE}^{t}}$$

$$= w_{MLE}^{t} - \alpha \sum_{i=1}^{n} \frac{1}{1 + \exp(-y_i x_i^T w)} \exp\left(-y_i x_i^T w\right)(-y_i x)$$

$$= w_{MLE}^{t} + \alpha \sum_{i=1}^{n} \left(1 - \sigma\left(y_i x_i^T w\right)\right)$$

When we get the optimized parameters, we can predict the labels in a test image:

$$y = \begin{cases} 1 & x^T * w \geq 0 \\ -1 & x^T * w < 0 \end{cases}$$

III. Technical Approaches

In this section we discuss about algorithms used in this project.

A. Optimize parameters

Read the example images in python, and then used the function "Roipoly" to draw the frame of the blue barrel and labeled them for each color classes, as shown in Figure 1. After applying this process on enough training images, we can get a training dataset, D={x_i, y_i}. Each number in the matrix X is from 0 to 255, and y is 1 or -1. I set the initial w_i is zero. According to the equations about getting the global minimum in Section II, after some iterations, we could get the optimized parameters w. I have tried two different datasets. To simplify the calculation, I only analyzed 1000 blue pixels and 5000 not blue pixels. In the first set, I used all the images I could access, and labeled the blue barrels in each picture. For the other set, I only used the images whose blue barrels are obvious. That is there are nothing in front of the blue barrel, such as branch or shelf. For the first set, I got w=[2.61261, -2.33581, -64.6212]. For the other one, w=[-139.344, -78.2001, 115.254].

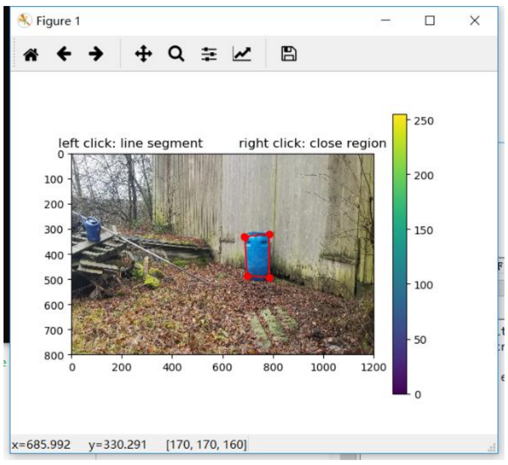
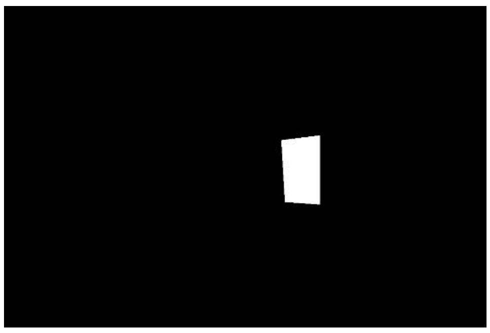


Figure 1
left click: line segment
right click: close region
x=685.992 y=330.291 [170, 170, 160]



(a) (b)

Figure 1 (a)getting the frame of blue barrel. (b) labeled color classes

B. Image Segment

Read the image we wanted to test in python. Get all the RGB data of the image, and put them into a matrix X ∈ R^{n*3}. Each column in X include all the data of R, G, B respectively. Each row of X represents a pixel of the image. For every row of X, multiple by w and calculated the result. For example, Xi is the ith row of X, calculated Xi*w. If the result is larger than zero, the pixel can be regarded as blue and labeled by 1. Otherwise, it was not blue and labeled by -1. For all the pixels labeled by 1, I changed the values of them to [255, 255, 255]. While I set the other pixels to [0, 0, 0]. Then reshaped the matrix into RGB and merged it, the segment image can be shown.

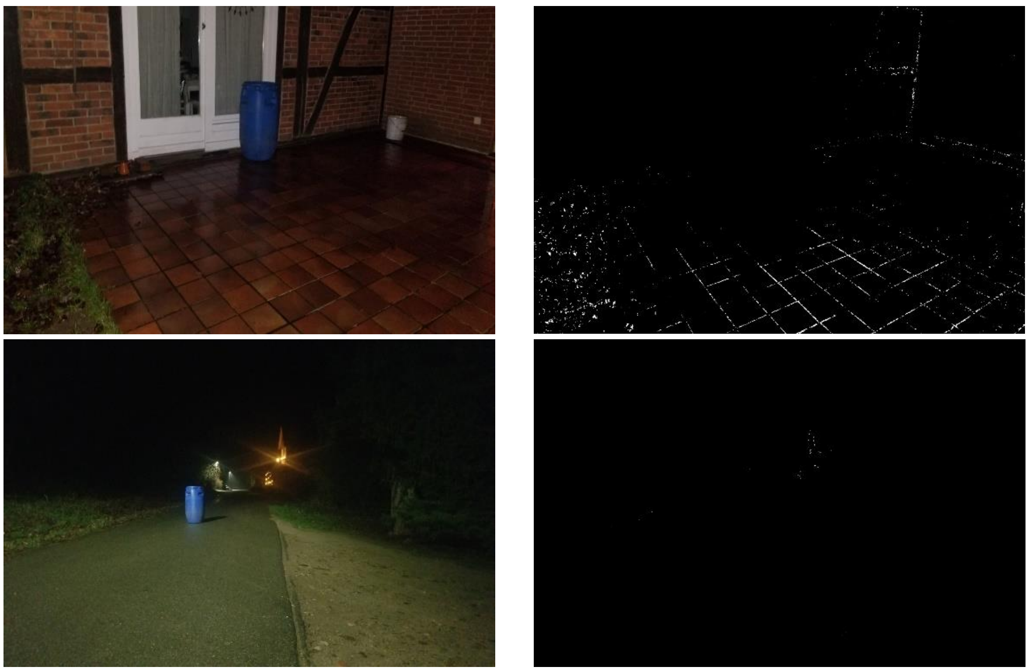
C. Get Bounding Box

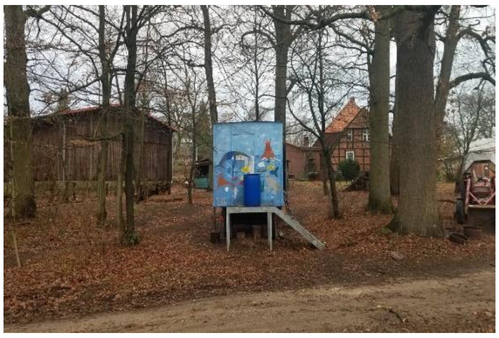
To enumerate blue region combinations and score them based on "barrelness", I used the function "label"and "regionprops" in "skimage". In order to make the result more accuracy, I only consider the regions whose area is larger than 1000. I used "props.bbox" to get the coordinates of the rectangular cover the region. The top left and bottom right coordinates were output.

IV. Results and Discussion

A. Dataset include all images

I got the optimized parameters w=[2.61261, -2.33581, -64.6212]. The results are shown in Figure 2.




(a)

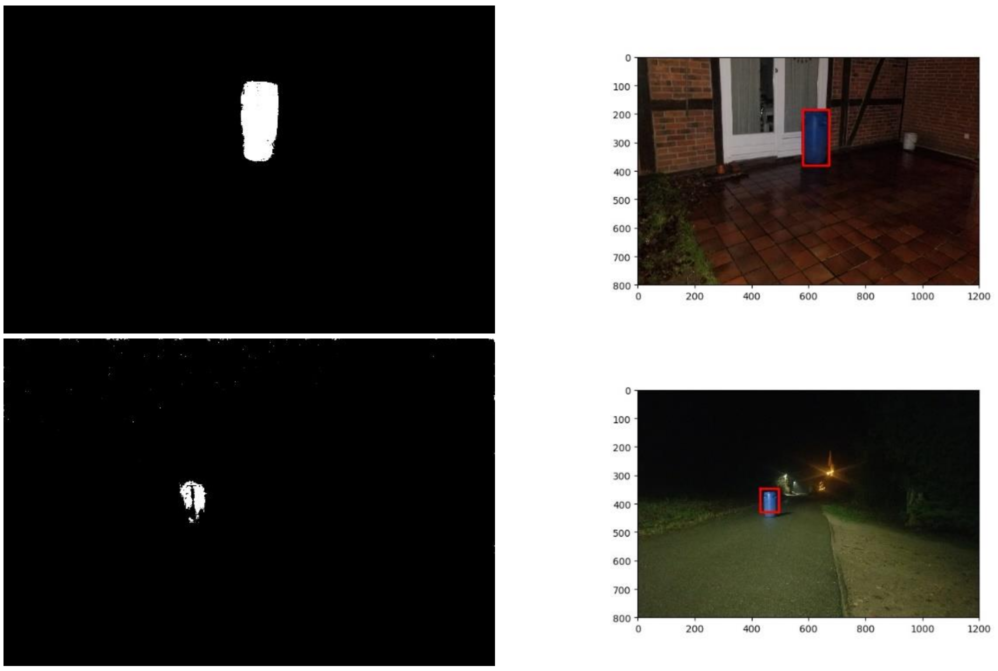
(b)

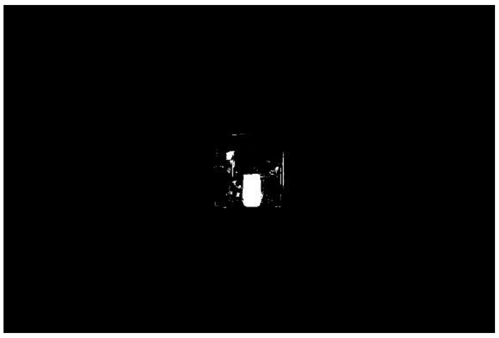
Figure 2 (a) the original images, (b)the segment images

As shown in the Figure, it cannot find the blue barrels. So, they cannot draw the bounding boxes. It might because I drew the frame of blue barrels in every image, however in some images, there is not only the blue barrel, but also something else. In some images, there is something in front of the blue barrel whose color is not blue. When I drew edges of blue barrels, these things were included. Thus, there were some color not blue but labeled with 1(blue). They had a great impact on the later calculation, and then leaded to the fact that we cannot use this “optimized parameters” to find blue barrel.

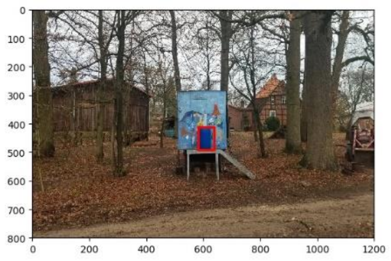
B. Dataset only include images with obvious blue barrel

This time, I only considered the images with obvious blue barrels as training images, and I got the optimized parameters w=[-139.344, -78.2001, 115.254]. The results of the same images in A are shown in Figure 3.





(a)

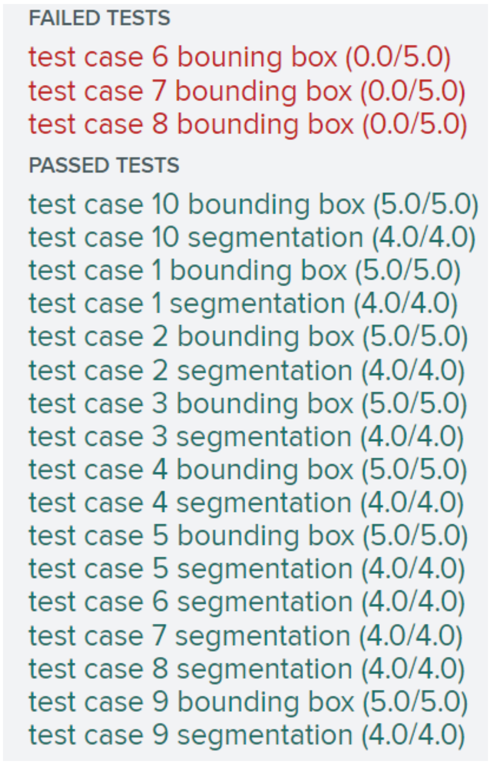


(b)

Figure 3. (a) the segment image, (b) bounding box

From the results above, it’s clear that we can find blue barrels this time, even in the dark situation or its background is nearly the same color. Thus, we can learn that drawing a frame include only blue barrel is very important.

C. Test Result in Gradescope



FAILED TESTS

test case 6 bouning box (0.0/5.0)
test case 7 bounding box (0.0/5.0)
test case 8 bounding box (0.0/5.0)

PASSED TESTS

test case 10 bounding box (5.0/5.0)
test case 10 segmentation (4.0/4.0)
test case 1 bounding box (5.0/5.0)
test case 1 segmentation (4.0/4.0)
test case 2 bounding box (5.0/5.0)
test case 2 segmentation (4.0/4.0)
test case 3 bounding box (5.0/5.0)
test case 3 segmentation (4.0/4.0)
test case 4 bounding box (5.0/5.0)
test case 4 segmentation (4.0/4.0)
test case 5 bounding box (5.0/5.0)
test case 5 segmentation (4.0/4.0)
test case 6 segmentation (4.0/4.0)
test case 7 segmentation (4.0/4.0)
test case 8 segmentation (4.0/4.0)
test case 9 bounding box (5.0/5.0)
test case 9 segmentation (4.0/4.0)

My segmentations are all right, but failed in some bounding boxes. It might because that I did not consider the shape of the segmentations, and got something other than barrel in the box.