Sneh Shah (sjs413) - Section 2

Rishi Shah (rrs141) - Section 5

Intro to AI Project 4

()

May 5, 2021

Certification of Academic Integrity

I certify that the work done on this project is my own and is not copied or taken from online or any other student's work. - Rishi Shah
I certify that the work done on this project is my own and is not copied or taken from online or any other student's work. -Sneh Shah

Breakdown of Work

Rishi coded the advanced agent functions. He worked on the advanced agent and advanced agent bonus sections of the write up.

Sneh coded the basic agent functions. She worked on the basic agent and basic agent bonus sections of the write up.

We worked together to come up with the pseudocode for all functions together and split up the coding as mentioned above.

The following image is the image used to train and test our basic agent and advanced agent.

1. Basic Agent

The final result (shown below) is somewhat visually satisfying in that the objects in the resulting picture are colored according to the colors that is represented in the original image. The trees are green, the water and sky are blue, and the sand is tan. There does seem to be some areas where the coloring is definitely wrong, like on the right side of the beach, where you see a darker color representing the shadows of the trees, that is colored blue. It seems as though in areas where there are small patches of different color, such as in the shadows area, those areas are more likely to be incorrect. This image is not numerically satisfying, as we are only restricting ourselves to 5 colors so the loss is going to be very high. This "loss" that we measured was simply the difference between the pixel value in the resulting image and the original.

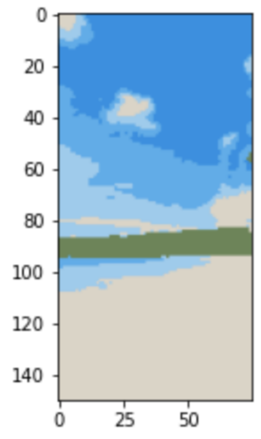


Figure 1: Training Data

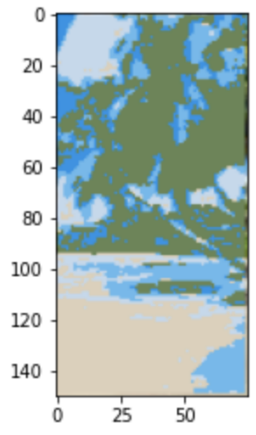
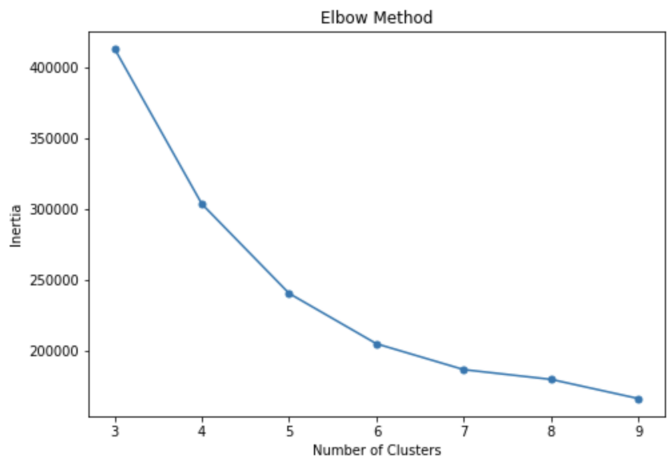


Figure 2: Testing Data

2. Basic Agent Bonus

We used the elbow method to determine the number of clusters that would be ideal when doing k-means. For the elbow method, we tried 3-9 clusters and we calculated the inertia for each number of clusters. We calculate the inertia by summing up the difference between the actual rgb value and the cluster center that the data point was assigned to. We then graphed these differences and found that having 5 clusters is ideal because the inertia starts plateauing after 5 clusters.



3. Advanced Agent

Our advanced agent picks up the shading of the original image, but does not output different colors. Both the testing and training data are colored with different shades of the same color. We believe this is happening because our weight vectors are trained over the entire image and therefore are trained to output an average of all the colors present. The result image is colored with a light blue color since most of the original image is blue and tan.

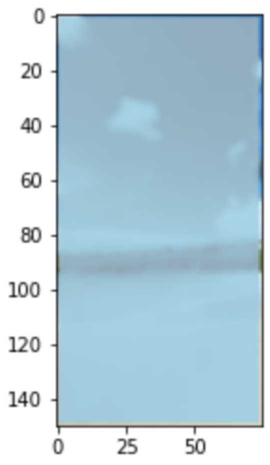


Figure 3: Training Data

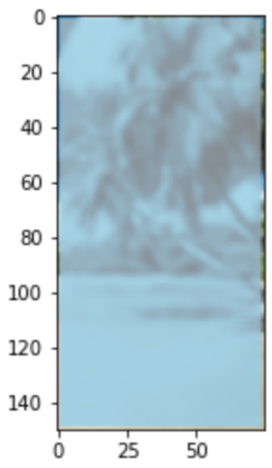


Figure 4: Testing Data

1. For our advanced agent, we used logistic regression with a sigmoid function to model red, green, and blue values. Our input is a vector of size 9 with grayscale values representing a 3x3 grayscale patch. Our output is one red value, one green value, and one blue value to represent a color. We calculate the loss by subtracting the predicted value from the sigmoid function from the actual value and squaring the difference. We then train our weights (one for red, one for green, one for blue) using stochastic gradient descent. This involves subtracting the learning rate multiplied by the derivative of the loss function from the current weight vector.

2. We choose our parameters mostly by guessing and checking. For the learning rate, we saw that when alpha was too large would all be one shade of a color. When alpha was too small, the output again would be one shade of a color. For the alphas we tested out, a large alpha made the output all white and a small alpha made the output all gray. The alpha that we ended with was picked because it gave us values that resulted in a somewhat clearer output picture.

   Regarding the weights, we noticed that when the weights were too large, the output pixel would be white. Because of that, we made the weights really small (values close to 0).

3. To preprocess our data, we divide the rgb values and the grayscale value by 255 before inputting these values into derivative of the loss. We do this to scale our numbers to be between 0 and 1 so that we can avoid really large outputs.

4. We train our model using stochastic gradient descent. This involves using alpha multiplied by the derivative of the loss function. This value is then subtracted from the weight vector. We prevent over fitting by not going through the training image multiple times. We only run through it once, creating a situation where we don't train our data too much.

5. We can compare the basic agent and advanced agent by checking what the loss is for each and then compare these losses. We can also look at the output pictures and notice the differences. Our basic agent does not do well with any shading that happens in the picture. it only captures the general colors of the objects. We can see that the advanced agent picks up shading and looks a lot more natural than the output of the basic agent.

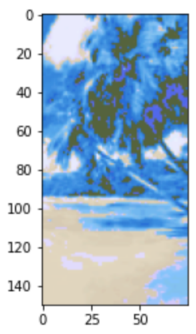
6. This model could be improved with sufficient time and resources by using patches that are bigger than 3x3. By looking at more gray scale pixels, you can get a better idea of what a color would look like and thus have a better chance of predicting it correctly. Another improvement that could be made is using some kind of object recognition to classify things in the image and use that to identify its color.

4. Advanced Agent Bonus

1. A linear model would be a bad idea because the same gray scale value can correspond to multiple colors.

2. We use scikit-learn's SVC or support vector classifier to improve upon the regular advanced agent. The kernel that is used here is an rbf (radial basis function) kernel which is non-linear and is thus appropriate for the situation at hand. Our results were fairly accurate, as the image does hold the same shading as the original image and has most of the same coloring as well. The only part of the image that is glaringly inaccurate is the trees, but that is likely due to the fact that the full image isn't perfectly symmetric.

   SVC with the rbf kernel works by using the kernel trick and soft margin classifier idea to use existing features and performing some kind of transformation to create new features. These new features are what helps the model find the proper nonlinear decision boundary.



5. Clever Acronym

basic agent: KOOP - KMeans on Output Picture

advanced agent: LOOMS - Logistic On Output Minus Sigmoid