Intro to AI Project 4 Write Up

May 5, 2021

On my honor, I have neither received nor given any unauthorized assistance on this assignment. All of our work and ideas are our own. We did not use any other external sources.
- Michael Zhang, mbz27, Section 1

On my honor, I have neither received nor given any unauthorized assistance on this assignment. All of our work and ideas are our own. We did not use any other external sources.
- Prasanth Balaji, pmb162, Section 3

Group members: Michael Zhang, Prasanth Balaji

(a) Original Image (b) Grayscaled Image

(a) Sampled Multiple Similar Patches: Euclidean Tested area error - 136737.5029637309

(b) Sampled Most Similar Patches: Euclidean Tested area error - 132803.04478065128

Figure 1: Observation - It seems that the algorithm that sampled from multiple similar patches did marginally better than the one that only chose the most similar patch. Which intuitively makes sense since one algorithm had more patches to choose from.

Basic Coloring Agent

How good is the final result?

Both versions of the basic agent produced an almost identical image and it is quite similar to the original.

How could you measure the quality of the final result?

We used the euclidean distance formula to measure the "distance" to compare two different RGB values. To measure the quality of an image we simply add all of these "distances" together to get an overall quality value.

Improved Coloring Agent: S.C.O.L.M - Soft classification of lega man

Why is a linear model a bad idea?

A linear model would produce values that are not RGB values and the linear model will definitely not "fit" with the given data.

A specification of your solution, including describing your input space, output space, model space, error / loss function, and learning algorithm.

For our solution we chose to use a soft classification algorithm where the algorithm will choose from the 5 colors from the clustered image.

Input Space: Xi = [1, G1, G2, ..., G9]
This represents the 3 by 3 greyscale patch.

Output Space: Yi = [C1, C2, ..., C5]
This represents the 5 colors from the clustered image.

Model Space: F(X)=(f0(x), ... ,f4(x))
These 5 functions represent the 5 colors from the clustered image where:

$$f_j(x) = \frac{e^{w_j \cdot x}}{\sum_{k=0}^{9} e^{w_j \cdot x}}$$

Loss Function:

$$L[i] = \sum_{i=0}^{N} \sum_{c=0}^{4} -y_c * ln(\frac{e^{wc * x[i]}}{\sum_{k=0}^{4} e^{w_k x[i]}})$$

Learning Algorithm:

$$\textit{General Formula}: \; w_{t+1} = w_t - \alpha \nabla_w L[i]$$

Solving for $\nabla_w L[i]$:

$$L[i] = \sum_{c=0}^{4} - y_c * ln(\frac{e^{w_c \cdot x[i]}}{\sum_{k=0}^{4} e^{w_k \cdot x[i]}})$$

Expand out :

$$L[i] = -y_0[i] * ln(\frac{e^{w_0 \cdot x[i]}}{e^{w_0 \cdot x[i]} + e^{w_1 \cdot x[i]} + \ldots + e^{w_4 \cdot x[i]}}) + \ldots + (-y_4[i] * ln(\frac{e^{w_4 \cdot x[i]}}{e^{w_0 \cdot x[i]} + e^{w_1 \cdot x[i]} + \ldots + e^{w_4 \cdot x[i]}}))$$

Use Logarithmic Rule :

$$L[i] = -y_0[i] * (ln(e^{w_0 \cdot x[i]}) - ln(e^{w_0 \cdot x[i]} + e^{w_1 \cdot x[i]} + \ldots + e^{w_4 \cdot x[i]})) + \ldots$$
$$(-y_4[i] * (ln(e^{w_4 \cdot x[i]}) - ln(e^{w_0 \cdot x[i]} + e^{w_1 \cdot x[i]} + \ldots + e^{w_4 \cdot x[i]})))$$

Distribute the y's and group together variables :

$$L[i] = -y_0[i] * ln(e^{w_0 \cdot x[i]}) + \ldots + (-y_4[i] * ln(e^{w_0 \cdot x[i]})) +$$
$$(y_0[i] + \ldots + y_4[i]) * (ln(e^{w_0 \cdot x[i]} + e^{w_1 \cdot x[i]} + \ldots + e^{w_4 \cdot x[i]}))$$

$(y_0[i] + \ldots + y_4[i])$ *equals* 1 *since the total of all probabilities must sum up to* 1 :

$$L[i] = -y_0[i] * ln(e^{w_0 \cdot x[i]}) + \ldots + (-y_4[i] * ln(e^{w_4 \cdot x[i]})) + (ln(e^{w_0 \cdot x[i]} + e^{w_1 \cdot x[i]} + \ldots + e^{w_4 \cdot x[i]}))$$

Simplify logs :

$$L[i] = -y_0[i] * (w_0 \cdot x[i]) + \ldots + (-y_4[i] * (w_4 \cdot x[i])) + (ln(e^{w_0 \cdot x[i]} + e^{w_1 \cdot x[i]} + \ldots + e^{w_4 \cdot x[i]}))$$

We will be doing the derivative with respect to w_0 *and generalize later* :

$$\frac{\partial}{\partial w_0}[-y_0[i] * (w_0 \cdot x[i]) + \ldots + (-y_4[i] * (w_4 \cdot x[i])) + (ln(e^{w_0 \cdot x[i]} + e^{w_1 \cdot x[i]} + \ldots + e^{w_4 \cdot x[i]}))]$$

Distribute the partial derivatives and simplify :

$$\frac{\partial}{\partial w_0}[-y_0[i] * (w_0 \cdot x[i])] + \ldots + \frac{\partial}{\partial w_0}[(-y_4[i] * (w_4 \cdot x[i]))] + \frac{\partial}{\partial w_0}[(ln(e^{w_0 \cdot x[i]} + e^{w_1 \cdot x[i]} + \ldots + e^{w_4 \cdot x[i]}))]$$

$(-y_1[i] * (w_1 \cdot x[i])) \ldots (-y_4[i] * (w_4 \cdot x[i]))$ *derive down to* 0 *since it's respect to* w_0

$$-y_0[i] * x[i] + \frac{1}{e^{w_0 \cdot x[i]} + e^{w_1 \cdot x[i]} + \ldots + e^{w_4 \cdot x[i]}} * \frac{\partial}{\partial w_0}[e^{w_0 \cdot x[i]} + e^{w_1 \cdot x[i]} + \ldots + e^{w_4 \cdot x[i]}]$$

$$-y_0[i] * x[i] + \frac{1}{e^{w_0 \cdot x[i]} + e^{w_1 \cdot x[i]} + \ldots + e^{w_4 \cdot x[i]}} * \frac{\partial}{\partial w_0}[e^{w_0 \cdot x[i]}]$$

$$-y_0[i] * x[i] + \frac{e^{w_0}}{e^{w_0 \cdot x[i]} + e^{w_1 \cdot x[i]} + \ldots + e^{w_4 \cdot x[i]}} * x[i]$$

$$\nabla_{w_0} L[i] = (\frac{e^{w_0}}{e^{w_0 \cdot x[i]} + e^{w_1 \cdot x[i]} + \ldots + e^{w_4 \cdot x[i]}} - y_0) * x[i]$$

$$\nabla_{w_0} L[i] = (f_0(x[i]) - y_0) * x[i]$$

Generalize :

$$w_{t+1} = w_t - \alpha \nabla_w L[i]$$

Final Training Algo. Equation :

$$w_{t+1} = w_t - \alpha((f_c(x[i]) - y_c) * x[i])$$

How did you choose the parameters (structure, weights, any decisions that needed to be made) for your model?

Input Space:

We decided to copy the concept from the basic agent where it samples from a 3 by 3 patch of grayscale values to generate a RGB value. So in our input space we would have the 3 by 3 grayscale patch represented by a 10 dimensional vector.

Output Space:

Since soft classifying all 256 RGB values and 3 different combinations that they can be arranged in would be a bad idea, we decided to soft classify the 5 colors that were picked by the clustering algorithm. Therefore, in our output space we chose to soft classify the 5 colors that were picked by the clustering algorithm. As a result, in our output space we have the 5 colors represented by a one hot encoded vector.

Model Space:

Obviously for a soft classification algorithm that only identifies 5 colors we will need 5 equations to represent the probability of being one of the 5 colors.

Any pre-processing of the input data or output data that you used?

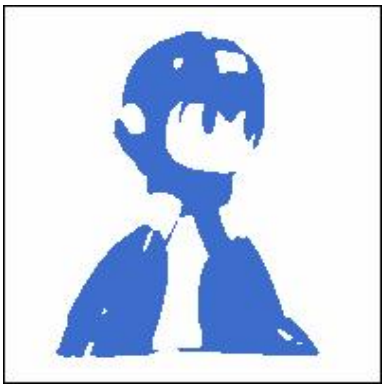
When implementing the learning algorithm there are many scenarios where we needed to take the exponential value of the dot product of the weight vector which resulted in an extremely large number. In fact this number was so large that the double data type could not store this large number and it was simply round up to infinity, which was a very poor representation of the data. So we came up with a solution to scale down the grayscale values by simply dividing them by 255 which resulted in better values that represent the data.

How did you avoid overfitting?

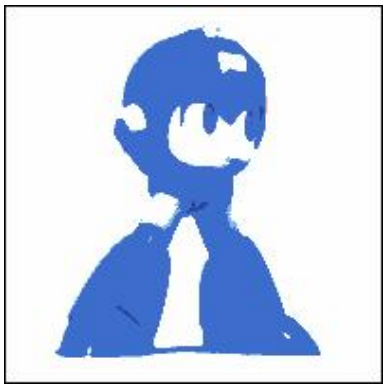
There wasn't a whole lot of concern regarding over fitting, as the training data was so small. In addition to this, we weren't dealing with many features that were added to the dataset.

An evaluation of the quality of your model compared to the basic agent. How can you quantify and qualify the differences between their performance? How can you make sure that the comparison is 'fair'?

Since both the basic agent and the improved agent uses the clustered image as their training data and that both can only produce the 5 colors from the clustered image to make a fair comparison, we will consider the clustered image as the "original image". To compare the difference in quality between one RGB value from the original image and the RGB value produced by the algorithm, we will take the euclidean distance of the two RGB values to quantify the quality of the chosen pixel. To compare the original image with the image produced by the algorithm we will simply sum up all the euclidean values of all the RGB values.



(a) Improved Agent without any features: Euclidean Tested area error - 383555.31481374253



(b) Improved Agent with quadratic feature: Euclidean Tested area error - 365190.7329143515

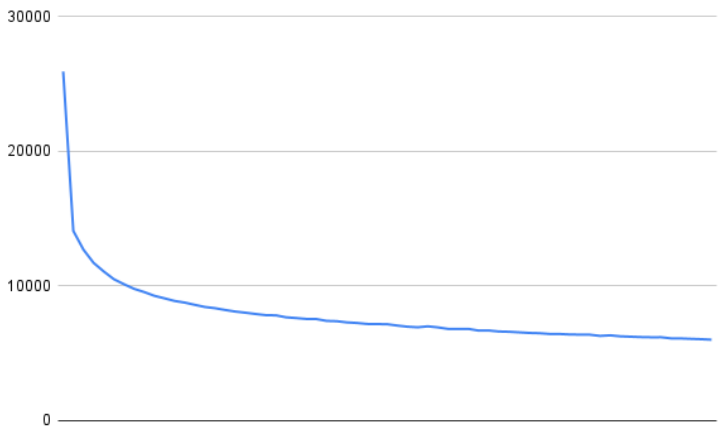


Figure 2: Loss Value Graph over time for Improved Agent (Learning Rate 0.01)

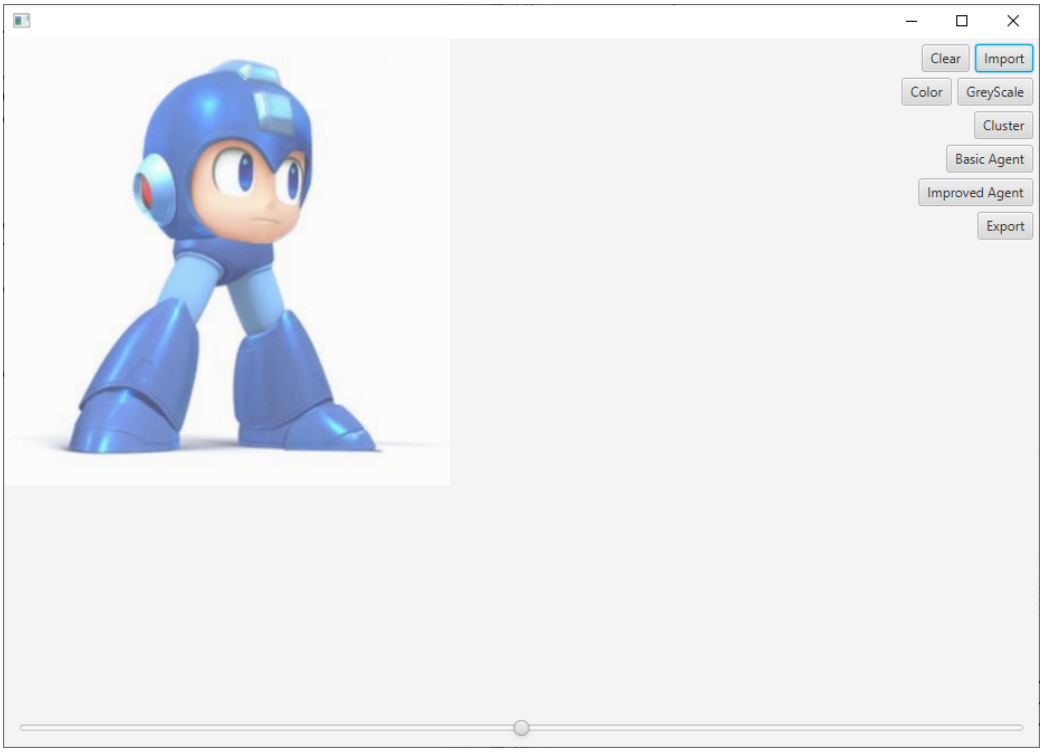
The reason why the results of the improved agent were not as good as the basic agent is most likely due to the fact that our model does not have many features which made the model not fit the data as well. In addition, since we had limited time to run the training algorithm it might have not produced the best weights for our model.

How might you improve your model with sufficient time, energy, and resources?

To improve our model we should add more features to the data set and also it would be a worthwhile try to optimize the implementation of the training algorithm just by using more efficient data structures. Also given more time we can run the training algorithm for a longer duration to get better results and augment our training algorithm so it can consider more than a 3 by 3 patch of an image.

Documentation

We used javafx to get a GUI that displays the image to the screen. So in order to run the program, Javafx 15 needs to be installed in Eclipse and –module-path "[Path to javafx sdk lib folder]" –add-modules javafx.controls,javafx.fxml needs to be added in the VM arguments.



Notice: Only square images work and please keep in mind that the higher the resolution of the image the longer it takes to run the different algorithms.

To import an image click on the import button and select a square image. This will automatically convert the image to a grayscale image and run the Cluster algorithm. You will be able to see the grayscale image by clicking on the grayscale button and you will be able to see the clustered image by clicking on the clustered button. The slider on the bottom of the GUI will zoom in and out the image (though it isn't scaling the image it just spaces the pixels out or squishes the pixels together). To run the basic agent click on the "Basic Agent" button. To run the improved agent click on the "Improved Agent" button. (Notice: Both the Basic Agent and especially the Improved Agent will take quite a while to run depending on the image's resolution and size). You can export the images by clicking on the "Export" and selecting a folder to save the images to and giving the images a name (This will export the grayscale image, clustered image, basic agent's image (if the algorithm ran), and the improved agent's image (if the algorithm ran)).

Implementation

We have an object that stores the RGB values and location of a pixel. This will be our abstraction of a pixel in the images that we are generating. This object will be in a 2D array that will represent a whole image. As for tracking the weights of our improved agent we simply have a 2-dimensional array which represents the individual equations that give us the probability of the corresponding color and the columns are the weights values.