COMS4036: Project 2 - Detecting Coins (GMM)

Liam Pulles (855442)

October 18, 2017

1 Introduction

"Life does not ask what we want. It presents us with options"
- Thomas Sowell

The purpose of this assignment was to create a Gaussian Mixture Model (from here on described as GMM) which could assign a probability and then mask to a specified set of images containing coins on a desk. To create the GMM, we were expected to use Expectation Maximization in combination with a provided set of training masks.

Not everything went as expected in this project, and I shall attempt to detail these issues with my solutions.

2 General training process overview

A high level description of the training algorithm is provided as follows:

1. Load all the training images with their masks, and transform them into just two lists of HSV vectors. One list contains "coin" data, and the other "background" data (from here on described without quotes).
2. Keep only a random sampling of 10% of the training data, to keep memory and processing costs down.
3. Initialize two separate GMMs, one with the coin data and the other with the background data. We use KMeans for the initialization, using as many clusters as the specified number of multivariate normal distributions (hereon referred to as MNDs). The trained cluster centers will give us out initial means. We use diagonal covariance matrices for our MNDs, and initially we set all the elements on the diagonal to 100. The weights are all set equally and made to add up to 1 for each GMM.
4. Begin the EM cycle. Keep iterating until: the sum total in the covariance differences for all MNDs in each GMM is less than 10, until a singular matrix is detected, or an iteration limit is reached (the default is 100).
   (a) Perform the E step.
   (b) Perform the M step.
   (c) Calculate the sum changes in various relevant parameters and variables (including the total covariance change) and print the information to the screen. Sending a SIGINT is conveniently caught in the singular matrix detection phase and allows us to kill the cycle (but not the whole program) should we feel that the loop is not making any useful progress.
5. Determine a weighting to balance the two trained GMMs. This was the main source of contention in the project for this author.
6. Save the trained classifier.

3 General mask image creation process

A high level description of the classification algorithm is provided as follows:

1. Load the image and transform it into a single list of HSV vectors.
2. Run the list through each GMM separately to get two lists of probabilities.
3. Combine the two lists with the weighting determined previously to arrive at a single list of probabilities, describing the probability that the given pixel is a coin.
4. Scale the list of probabilities by 255 - putting them in greyscale pixel range - then reshape it back to the original image dimensions and output the now completed mask.

4 Individual Algorithm descriptions

For the sake of brevity I shall describe those algorithms which I think are most relevant to the given course and which are the least immediately intuitive.

As such I shall take it on faith that the reader understands what KMeans is, and that they trust that I understand this and other trivial concepts.

4.1 The HSV colorspace

The HSV colorspace may be considered as a polar coordinate space resulting from a linear transformation on the standard RGB colorspace. In this coordinate sphere the north and south poles represent intensity extremes, and the equator represents the most saturated possible colors. Hue describes the chrominance of the pixel, while saturation describes its vibrancy, and intensity the brightness. Note that peak saturation can only occur at mid-level brightness, along the equator.

The advantage of this colorspace over RGB is that the channels are more naturally independent - that is we can imagine possibly describing the difference in coins and desk using only the value channel, or segmenting based only on the hue channel. This is vital if we are going to make a diagonal covariance assumption (which we will).

4.2 The E Step

The E step is tasked with finding a responsibility matrix for all the given normal distributions and a selection of training data. For a given row (fixing on a pixel in the data) each element in the row describes the proportion that the corresponding column (normal distribution) is associated with the given pixel relative to the other columns. It is calculated in correspondence with formula (7.17) on pg. 110 of the prescribed textbook.

4.3 The M Step

The M step is responsible for taking the responsibility matrix, and using it to weight data points in the parameter calculations depending on the "strength" of the data point in relation to the given distribution. So for example, instead of a straight average to determine the mean - we do a weighted average using responsibility values. We do similar for the covariance matrix.

The weights for the distributions come from the relative row sum responsibility (or the total responsibility for a given distribution across all provided data points relative to the totals for all the distributions).

Again, the calculations are provided in formula (7.19) on pg. 113 of the prescribed textbook.

4.4 Aside: The point of Iteration

The reason we have to iterate the EM algorithm instead of just doing both steps once is that both are approximations that build on the other. As we accumulate iterations, the error in the approximation begins to disappear as we aggregate successive iterations onto previous ones.

We are guaranteed to reach a local minimum, and not necessarily a global minimum. As was discovered, the algorithm is very susceptible to getting caught in the local minimum nearest to the starting position and it is advisable to repeat the EM process a few times with different cluster configurations to try and get the best possible local minimum.

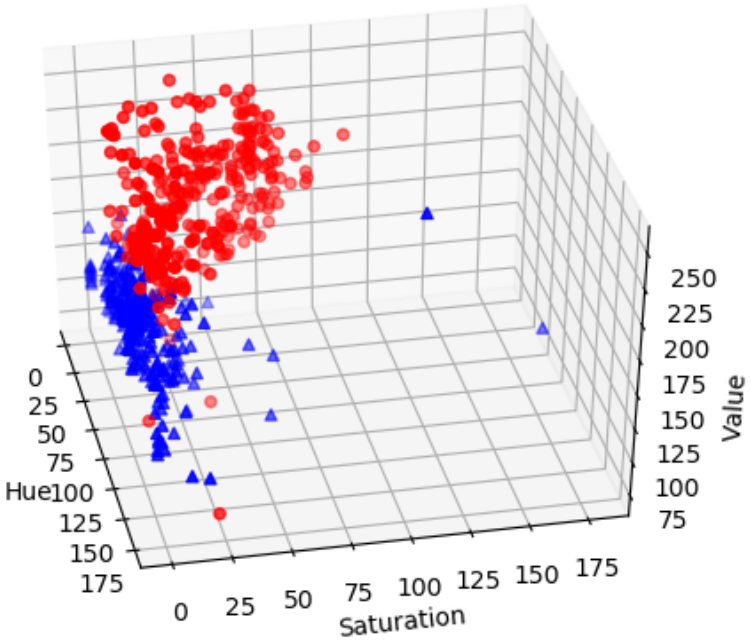

Figure 1: A graph showing a random selection of coin pixels(red) and background pixels (blue). It is clear that the HSV colorspace does a good job of separating most of them, though there is some tight overlap.

4.5 Determining the weighting

In order to calculate the probability that a given pixel is a coin, we need to use Bayes rule. Our desired probability is the posterior, the two trained GMMs provide likelihoods taken separately and part of the evidence taken together, but we need priors to weight them.

Since we are modeling a world of binary states, our prior is a simple Bernoulli distribution. The maximum likelihood solution for the necessary parameter lambda is given simply by the proportion of coin pixels relative to all pixels. This was attempted.

However, it did not produce the desired result. Infact, through trial and error, it was found that the correct parameter was several orders of magnitude away from the parameter suggested by the maximum likelihood solution.

To alleviate this issue, I decided to derive an approximation for the parameter by reversing the Bayes rule equation to put it in focus of the prior, and then get an average of these priors from the available training data. The algorithm can be found in the train() function in the LiamEM class.

The approximated value appears to remain consistent among repeat trainings with fixed cluster amounts, and the variance of the priors derived from the training data is six orders of magnitude less than the average, giving high confidence in the parameter's correctness.

Yet there are some strange effects. which sometimes (and not always) occur. The first main effect is that the mask is sometimes inverted. The other main effect is that the trained parameter is sometimes also inverted. It's also worth noting that although (as it seems to the author) solving one of these issues directly solves the other, this is not always the case.

The author is unsure as to why this is the case, but has provided a "hack" solution by allowing command line parameters to invert the parameters for the mask and/or Bernoulli parameter.

5 Performance

5.1 Results

The average Mean Squared Error between the provided masks and the guessed masks is about 16, indicated an average difference in intensity of 4 units across all pixels. This is "okay".

5.2 Interpretation

The system seems far more adept at picking up silver coins than bronze coins. This would seem to indicate that the value dimension in the HSV vector is seen by the algorithm as being an important separator. We must remember that we do not have actual covariances within our limited covariance matrices. This means that complex groupings of the data are not possible, and there is a likely overlap of desk colour and bronze pixel colour which is difficult for the system to resolve with diagonal covariance matrices.

The system also picks up "shiny desk" as "silver coin", which is likely due to the same issues that "bronze coin" vs "brown desk" produced as described in the previous paragraph.

"Rare" objects such as the ruler and blue pen are picked up quite confidently as foreground. It is unclear from the spec whether these items can be considered as foreground or background, however the provided labeling data would indicate that they should be considered background. Their brightness would indicate that they are picked up more strongly by the foreground GMM, which would seem to indicate that they've been unduly penalized from the background GMM (since these object have no presence in the foreground training data). The reasons for this are not clear, but it's worth noting that the amount of training data for these items is very limited, and them being seen as outliers by the Background GMM is a rational result of a generalizing algorithm.

5.3 Example images

6 Conclusion

Although unconventional and incomplete means were needed to get a balancing term between the two GMMs, the result is probably quite close to the optimal possible implementation in terms of accuracy.

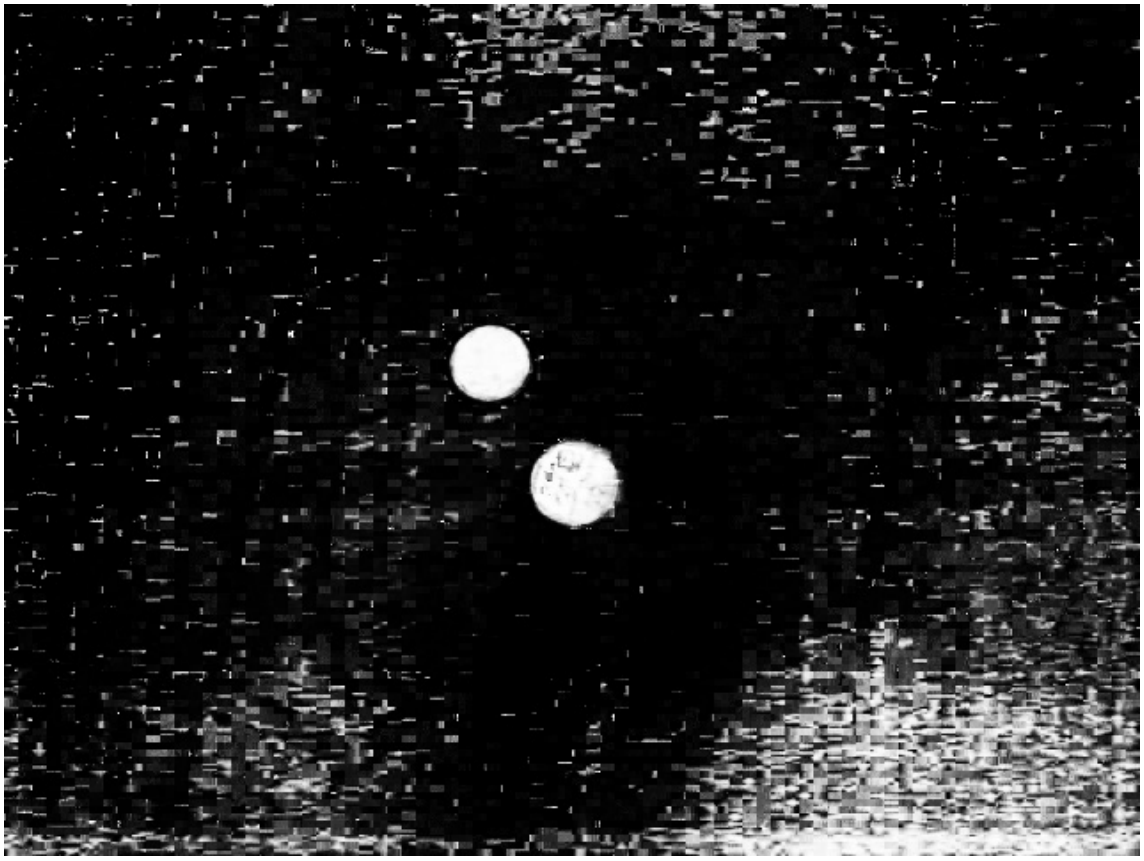

Figure 2: A relatively "good" image. Note that the system still picked up the shiny desk as a coin.

That leads the author to conclude that if the results seen are the best that a mixture of gausssians model can provide in this context, then it alone will not be sufficient to identify coins in the image. Further success may be achieved by moving to full diagonal covariance matrices, but this has a steep associated increased processing cost.

This author is satisfied that their own non-mixed normal distributions classifier is supreme in this case, as it was arrived at directly using maximum likelihood, and uses additional world states and intelligence (multiple categories of material, detected circles, random forests) to identify coins. Though this has a steeper cost than the given EM model in terms of processing time, it is likely less than the cost of using a full diagonal covariance matrix here when the result is not even guaranteed.

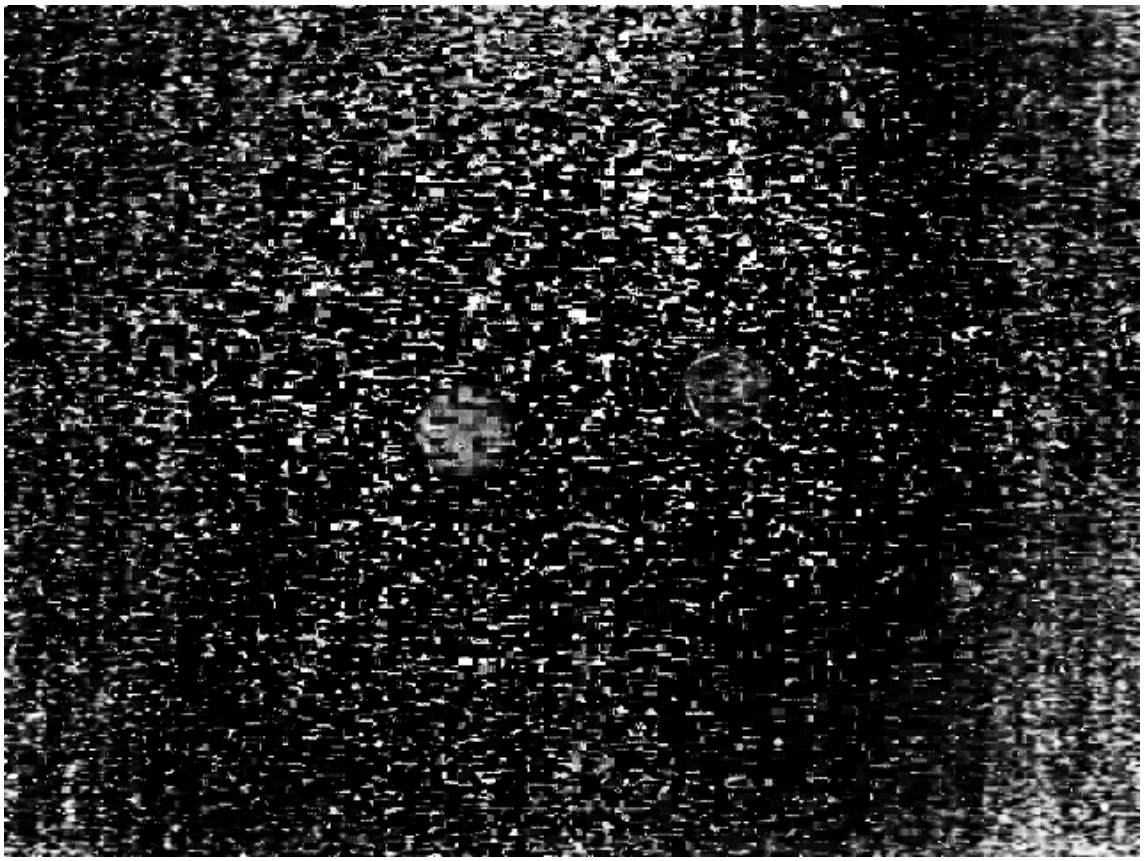

Figure 3: A relatively "bad" image. The wrongly detected JPEG noise obscures the two coins in the center.