Course: MATH 456: Mathematical Modeling

Image Segmentation By Means of a Gaussian Mixture Model

Authors: Liam Donovan & Makar Pronin
Instructor: **Professor Markos Katsoulakis**

Date : May, 2024

Introduction

Gaussian Mixture Models (GMM) are the most widely used form of mixture models. Such models seek to represent a given, complex, distribution as a weighted sum of simpler distributions. In the case of GMMs, the simpler distributions are Gaussian; in practice it is often more applicable to use multivariate Gaussian distributions. That is, for a given distribution, for a given set of parameters, denoted θ one can write

$$p(\mathbf{y}|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{y}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{1}$$

, where π_k is the weight for the k^{th} normal distribution. We may also observe that K is the total of distributions. To ensure that p is a valid probability distribution, we must levy another condition on the model. (1) can be seen as the conditional probability of two random variables, where $\pi_k \sim Cat(\pi_1, \ldots, \pi_K)$, therefore,

$$\sum_{k=1}^{K} \pi_k = 1 \tag{2}$$

We seek to assign each point to one of the K normal distributions. This is called *clustering*. We then optimize the likelihood of the given distribution in order to fit the model. This turns out to be quite difficult, therefore an algorithm called *Expectation-Maximization (EM)* is often used.

Motivation

Image segmentation is a process in which items in an image are able to be selected individually. This is desirable in cases in which one may want to remove an item from an image, or make it stand out. In Photoshop$^{\mathrm{TM}}$, a user can manually separate out parts of an image, using the "lasso" tool. However, this is tedious and prone to error, therefore we would like a way for a model to "understand" different items in an image. To perform this, we group similar colors together, relative to their location. This is often referred to as a mask. There are several types, depending on the intentions of the model, one of which is the saliency map, or the heat map. This targets the area of most "interest" in an image. Then, a user can interact with each individual object, as they would with the use of the lasso tool. This also has uses in medical research, helping medical professionals locate tumors and different types of tissue. Another use lies in the most recent self-operating vehicles. The computer must be able to tell the difference between a pedestrian, vehicles, signs, and road markings. This requires image segmentation. GMM gives a method to segment an image, based on color and location. That is, each normal distribution of color (specifically color values) is assigned to each object.

Summary of Techniques Used

First, we must find a way to cluster the data points. In order to assign a data point to a given distribution, we introduce a latent variable, $z_n \in 1, \ldots, K$, which specifies the distribution for which a given point, n, belongs. To find the probability of a given point being in a given distribution, we use the posterior. The following follows from Bayes' Theorem.

$$p(z_n = k|y_n, \theta) = \frac{p(z_n = k|\theta)p(y_n|z_n = k,\, \theta)}{\sum_{k'=1}^{K} p(z_n = k'|\theta)p(y_n|z_n = k',\, \theta)}$$

In the context of clustering, this posterior is called the *responsibility*. It gives the probabilities of a data point being in the k$^{\text{th}}$ distribution. Issues of optimizing this can be seen, in general.

Optimizing this model requires the optimization of several, high dimensional objects; namely, the mean vectors, and the covariance matrices. This makes taking derivatives particularly difficult. In addition: the log likelihood function is given, writing the set of all the parameters using θ:

$$\begin{aligned} l(\theta) &= \sum_{n=1}^{N} \log(p(y_n|\theta)) \\ &= \sum_{n=1}^{N} \log\left(\sum_{z_n=1}^{K} p(y_n, z_n|\theta)\right) \end{aligned}$$

, which contains two sums. Therefore, taking derivatives of this is computationally expensive. The EM algorithm gives a more concise way to optimize this function.

One can compute the most probable cluster by taking the cluster with the greatest probability, but one could also assign a data point, fractionally, to different clusters, relative to their probability. The former is called *hard clustering* and the ladder is called *soft clustering*. Soft clustering is a bit more flexible, therefore we use it in this application.

The logic of the algorithm relies on Jensen's inequality. In particular, we consider the case in which $f(x) = \log(x)$. We then introduce an arbitrary distribution, q, with domain over then latent variable. We simply multiply p by q/q. This is because we require another factor, which sums to 1, over the domain of the sum. Since the logarithm is a concave function, we can write

$$\begin{aligned} l(\theta) &= \sum_{n=1}^{N} \log\left(\sum_{z_n=1}^{K} q_n(z_n)\frac{p(y_n, z_n|\theta)}{q_n(z_n)}\right) \\ &\geq \sum_{n=1}^{N}\sum_{z_n=1}^{K} q_n(z_n)\log\left(\frac{p(y_n, z_n|\theta)}{q_n(z_n)}\right). \end{aligned}$$

Note that rewriting the logarithm as a difference and factoring out gives:

$$= -D_{KL}(q_n \,||\, p(z_n|y_n, \theta)) + \sum_{n=1}^{N} \log(p(y_n|\theta))$$

, where $D_{KL}(p \,||\, q)$ refers to the Kullback-Leibler divergence of p to q. Since q and $\log(\cdot)$ are non-negative, $D_{KL}(q_n \,||\, p(z_n|y_n, \theta))$ is non-negative; moreover it is a statistical distance (but

not a metric). This forms a lower bound on the log likelihood and is called the *evidence lower bound* or *ELBO*. Therefore, to optimize the log likelihood, the Kullback-Leibler divergence must be equal to 0. This occurs if and only if we set $q_n^* = p(z_n|y_n, \theta)$, the responsibility. This is the expectation step in the EM algorithm. This will be done iterative, so we introduce some notation. Let $\theta^{(t)}$ denote optimized parameters at the t^{th} step and θ refer to the most recent proposed optimized parameters.

$$\theta^{t+1} := \arg\max_{\theta} Q(\theta, \theta^{(t)})$$

Q is called the *surrogate function*. It immediately follows:

$$l(\theta^{t+1}) \geq Q(\theta^{t+1}, \theta^{(t)})$$

since $Q(\theta^t, \theta^{(t)}) = l(\theta^{(t)})$

Next, we optimize the likelihood function, given the new $q = q^*$. We may rewrite the ELBO in a more convenient form. By factoring, we find and noting that $\sum_{z_n=1}^{K} q_n(z_n)p(y_n, z_n|\theta)$: is simply the expected value of $\log(p(y_n, z_n|\theta))$ over q_n.

$$l(\theta) \geq \sum_{n=1}^{N} \mathbb{E}_{q_n}(\log(p(y_n, z_n|\theta)) + \mathbb{H}(q_n)$$

where $\mathbb{H}$ refers to the information entropy. Since $\mathbb{H}$ does not depend on θ, when optimizing the ELBO, we need only consider the other term. That is, we find

$$\arg\max_{\theta} \sum_{n=1}^{N} \log(p(y_n|\theta))$$

This is the M step of the EM algorithm.

For the GMM, the E step is straightforward, since the responsibility the responsibility of the GMM is simply. We know $p(z_n = k|\theta) = \pi_k$, thus we may rewrite, for the t^{th} step

$$p^*(z_n = k|y_n, \theta^{(t)}) = \frac{\pi_k^{(t)} p(y_n|\theta_k^{(t)})}{\sum_{k'=1}^{K} \pi_{k'}^{(t)} p(y_n|z_n = k',\, \theta^{(t)})}$$

Of the resources used, one used frames from a video of a football game. The goal was for the model to detect the different players. This specific resource considered changing the pixels to greyscale before running the algorithm. Moreover, we see a very naive approach to computing K. The user simply guesses, by observing the histogram of color, where the x axis is the intensity of color, whereas the y axis is the frequency of the gien color.

For greyscale, we observe: We note that the color separation is more effective than the gray. Below is the color separated image and below that, the gray separated image.

We note that the model works fairly well, however, we note that rerunning the model can cause the coloring to flip, due to the labelling of each Gaussian. The source was unable to find a way to correct this issue, trying to find the average frame of the video, then use

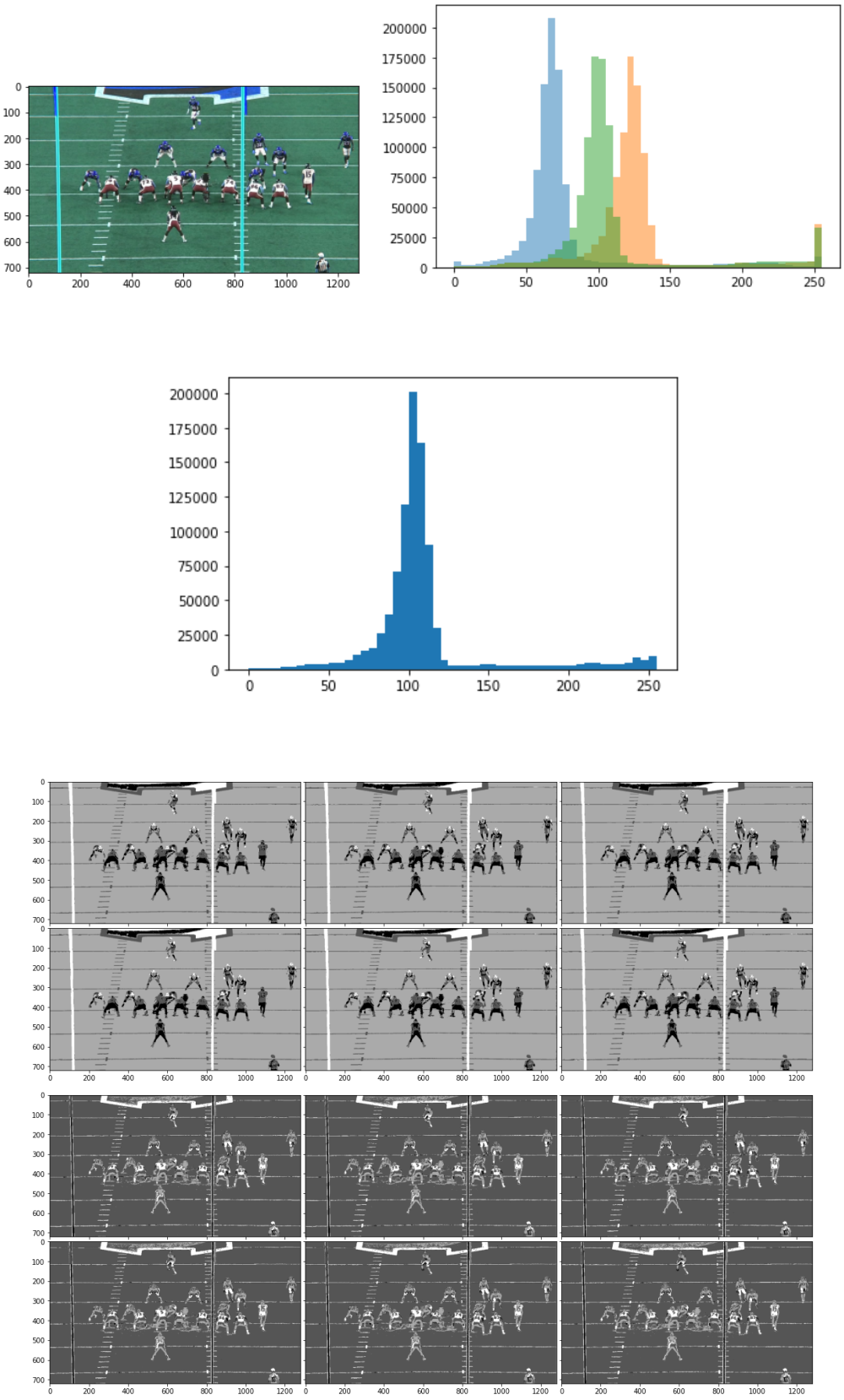

Figure 1: We note a more distinct separation in the color photo. This makes sense, since the color plot generally is more descriptive, but more computationally expensive.

this to average the training image was hypothesised to cause more standardization, but did not work well.

The second model was originally used on an image of a moose and a fox. We can see that output here.

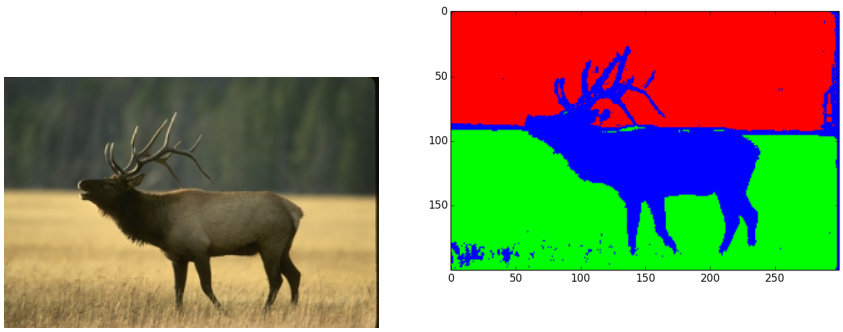

Figure 2: We note that the model accurately finds the difference betweent the moose, grass, and background.

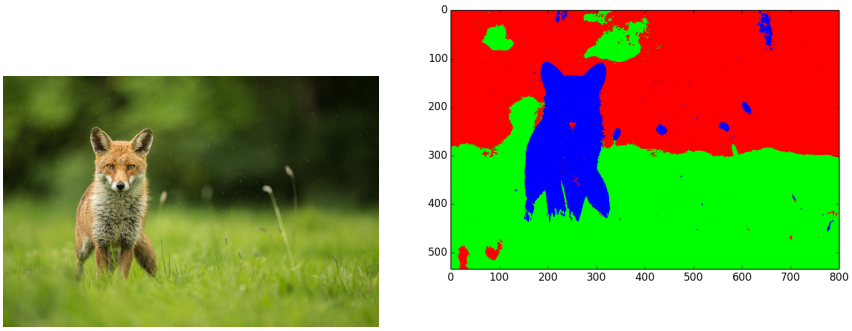

Figure 3: Similarly, this mdoel does quite well.

We note that the images, this time, are a bit more simple than those used for the football example. Moreover, this algorithm initializes the parameters using K-Means, from the sklearn library. This is a hard clustering method, which simply uses the Euclidean distance to compute the most probably distribution. This results in a faster convergence, since the initial parameters are closer to the real distribution.

We also note a few problem spots, such as the fox's nose being too dark, so it is deemed a part of the background. The model also has issues with the darker parts of the grass.

Discussion

All of the methods discussed did converge, however there was an issue of the process not being a involution in the first model. We may have been able to treat this case by conditioning the initial parameters using the k-means approach used in the second model. The second model was significantly more successful, although I am curious to see how it deals with more complex images. A good approach may be to also consider how certain pixels are related to

the pixels which surround it. This was an issue that I noticed in the fox picture, where the nose was treated as part of of the dark background.

I am also interesting in analyzing different types of mixture models. Is there an applicable situation where normal distributions are not the optimal choice for the mixture. If this is the case, what is the application?

Also, as mentioned in the introduction, such algorithms are used for tumor spotting. I am curious how accurate such a model is, since the consequences for failure in the model are significantly more severe than Photoshop applications. How can they make the loss function punish more heavily, while still having meaningful results.

I have also notices that the technology used in self-driving cars create, what appears to be, a 3d map. I am curious how this works. For example, how does the car collect information from around it well enough for the compute to make decisions quickly, or how does the computer quickly compute values for this purpose.

I would also like to research more about the ADAM optimization algorithm in this specific use. This is commonly used, in practice, in many deep learning algorithms. I am curious how this could improve the image segmentation algorithm.

References