Image Segmentation Using Markov Random Fields: An Implementation

Jake Callahan, James Griffin, McKay Harward, Taylor Paskett, Isaac Robbins, Mingyan Zhao

April 20, 2020

Abstract

The goal of this project was to evaluate current use of Markov random fields in image segmentation, then to implement one of the more advanced methods. In this paper we discuss how we implemented the equations from a paper based on an unsupervised method that used metropolis Hastings algorithm to find the best parameters for the Markov random field as well as the best number of classes. In short, we coded up a hands-off method of segmenting an input image.

1 Background and Motivation

This paper was motivated by the desire for a hands-off image segmentation algorithm. Primary motivations for wanting a hands-off image segmenter have to do with applications in neural photo editing using introspective adversarial networks as well as image recognition and a few other computer graphics applications. The idea for using Markov random fields for image segmentation has been around for a long time; in fact, the paper we decided to implement was written in 2000, but the unsupervised method doesn't seem to have any study since then. The background research we conducted mostly only returned methods for segmenting images based on pretraining a Markov random field on a class, and then estimating which pixels in a given image are most likely to be of that class. While this is useful, the more hands-off the approach the easier it is to apply to new scenarios. The drawback of our unsupervised method is that it trades an initial investment of creating pertained Markov random fields for a computationally expensive algorithm using metropolis Hastings to find those best parameters. So there are definite drawbacks to both methods, but with recent advances in computing power, we believe this method can provide results it previously couldn't.

2 Data

This method is not trained on a dataset but rather trains itself on the single image it plans to segment. We obtained a couple of images from the paper we read as well as a couple from James' phone.

3 Methods

Our method can be outlaid in 6 steps:

1. Random initialize each Markov random field for a randomly drawn number of classes
2. Segment the image based on these MRFs
3. Sample noise parameters μ and σ
4. Sample MRF parameters β_0 and β_1
5. Sample number of classes
6. Repeat steps (2) - (5) until convergence

Thought the process is simple, each of these steps required extensive coding and interpretation of equations. Step (5) was the hardest step by far, as the equations referenced various parts of the text with little explanation. In repeating steps (2)-(5), we're calculating an acceptance probability with each sampled parameter and using metropolis Hastings to decide whether or not we want to adopt the new parameters or keep our old parameters. These acceptance probabilities are defined as follows:

3.1 Noise

The noise parameters μ and σ are sampled using Metropolis-Hastings with the following likelihood:

$$\begin{aligned} p\left(\mu_c, \sigma_c | Y, X\right) &\propto \prod_{s: x_s = c} p\left(y_s | \mu_c, \sigma_c\right) p_r\left(\mu_c\right) p_r\left(\sigma_c\right) \\ &= \frac{1}{\sigma_c\left(2\pi\sigma_c^2 T_t\right)^{n_c}} \exp\left\{-\frac{1}{2T_t} \sum_{s: a_s = c} \left(\frac{y_s - \mu_c}{\sigma_c}\right)^2\right\} \end{aligned} \tag{1}$$

3.2 MRF

The noise parameters μ and σ are sampled using Metropolis-Hastings with the following probability:

$$\begin{aligned}
&p\left(\beta_c^{(0)}, c \in \Lambda | X\right) \\
&\quad = p\left(X | \beta_c^{(0)}, c \in \Lambda\right) p_r\left(\beta_c^{(0)}, c \in \Lambda\right) \\
&\quad = \textstyle\prod_{(c \in \Lambda, \psi_\eta)} \left(\frac{\exp\left(-\frac{1}{T_t}\left[\beta_c^{(0)} + \beta^{(1)} V(c,\eta)\right]\right)}{\sum_{i \in \Lambda} \exp\left(-\frac{1}{T_t}\left[\beta_i^{(0)} + \beta^{(1)} V(i,\eta)\right]\right)}\right)^{n_{(c,n)}} \\
&\quad = \textstyle\prod_{(c \in \Lambda)} \left(\frac{\exp\left(-\frac{1}{T_t}\left[\beta_c^{(0)}\right]\right)}{\sum_{i \in \Lambda} \exp\left(-\frac{1}{T_t}\left[\beta_i^{(0)}\right]\right)}\right)^{n_c} \\
&\quad \times \textstyle\prod_{(c \in \Lambda, \psi\eta)} \left(\frac{\exp\left(-\frac{1}{T_t}\left[\beta^{(1)} V(c,\eta)\right]\right) \sum_{i \in \Lambda} \exp\left(-\frac{1}{T_t}\left[\beta_i^{(0)}\right]\right)}{\sum_{i \in \Lambda} \exp\left(-\frac{1}{T_t}\left[\beta_i^{(0)} + \beta^{(1)} V(i,\eta)\right]\right.}\right)^{n_{(c,n)}}
\end{aligned} \tag{2}$$

3.3 Number of classes

The number of classes are sampled with likelihood

$$\begin{aligned}
&\frac{p\left(X = x^+, \psi^+, k^+ | Y = y\right)}{p(X = x, \psi, k | Y = y)} \frac{1}{p_\beta\left(u_1\right) p_\beta\left(u_2\right) p_\beta\left(u_3\right)} \\
&\quad \times \frac{\partial\left(\Psi_{c1}, \Psi_{c2}\right)}{p(\text{segmentation})} \left|\frac{1}{\partial\left(\Psi_c, u_1, u_2, u_3\right)}\right|
\end{aligned} \tag{3}$$

4 Results

5 Analysis

Our methods turned out to be fairly effective. They don't par with current state of the art technologies but they definitely do well for something done by a few undergraduate students in a single project, not research. The model we've created does a fairly decent job of segmenting images on par with what the original authors achieved, though sometimes in a few extra iterations. Looking at how it works, I do think it would work in a GAN for neural photo editing to maybe train the GAN to focus itself more closely on one area while keeping the rest of the image the same, but even with faster computers, it seems it still takes too long for any practical applications. Given enough computing power it can be used.

6 Conclusion

The initial intent of getting a completely unsupervised method of image segmentation using Markov Random Fields was successful and could probably be improved upon for efficiency. If given the opportunity to try image segmentation again, I would probably opt to use a neural network as they tend to require an initial investment of training time but can then be used to segment an image very quickly. Something we could do is have more informed priors for each parameters. We could also implement for color images (may be?)