Computer Vision CS543/ECE549
Homework 4

Due Date: April 10, 2017

1 SLIC Superpixels (35 pts)

Superpixel algorithms group pixels into perceptually meaningful regions while respecting potential object contours, and thereby can replace the rigid pixel grid structure. Due to the reduced complexity, superpixels are becoming popular for various computer vision applications, e.g., multiclass object segmentation, depth estimation, human pose estimation, and object localization. In this problem, you will implement a simple superpixel algorithm called Simple Linear Iterative Clustering (SLIC) that clusters pixels in the five-dimensional color and pixel coordinate space (e.g., r, g, b, x, y). The algorithm starts with a collection of K cluster centers initialized at an equally sampled regular grid on the image of N pixels. For each cluster, you define for a localized window 2S x 2S centered at the cluster center, where S = sqrt(N/K) is the roughly the space between the seed cluster centers. Then, you check whether the pixel within the 2S x 2S local window should be assigned to the cluster center or not (by checking the distance in 5D space to the cluster center). Once you loop through all the clusters, you can update the cluster center by averaging over the cluster members. Then, iterate the previous pixel to cluster assignment process till convergence or maximum iterations reached. For more details, please see SLIC Superpixels Compared to State-of-the-art Superpixel Methods, PAMI 2011.

In your write up:

(a) Explain your distance function for measuring the similarity between a pixel and cluster in the 5D space. (5 pts)

(b) Choose one image, try three different weights on the color and spatial feature and show the three segmentation results. Describe what you observe. (5 pts)

(c) Choose one image, show the error map (1) at the initialization and (2) at convergence. Note: error: distance to cluster center in the 5D space. (5 pts)

(d) Choose one image, show three superpixel results with different number of K, e.g., 64, 256, 1024 and run time for each K (use tic, toc) (5 pts)

(e) Use evalSLIC.m to evaluate your algorithms on the subset of Berkeley Segmentation Dataset (BSD) and show the three figures of performance using different number of segment K: (1) boundary recall and (2) under-segmentation error (3) averaged run time per image for the BSD (15 pts)

(f) **[Extra credit up to 10 pts]** Try to improve your result on (e). You may try different color space (e.g., CIELab, HSV), more complex distance measure (Sec 4.5 in the paper), richer image features (e.g., gradients), or even other algorithms. Report the accuracy on boundary recall and under-segmentation error with K = 256. Compare the results with (e) and explain why you get better results. Just changing the color space would be worth up to 5 pts.

2 EM Algorithm: Dealing with Bad Annotations (35 pts)

Dealing with noisy annotations is a common problem in computer vision, especially when using crowd-sourcing tools, like Amazon's Mechanical Turk. For this problem, you've collected photo aesthetic ratings for 150 images. Each image is labeled 5 times by a total of 25 annotators (each annotator provided 30 labels). Each label consists of a continuous score from 0 (unattractive) to 10 (attractive). The problem is that some users do not understand instructions or are trying to get paid without attending to the image. These "bad" annotators assign a label uniformly at random from 0 to 10. Other "good" annotators assign a label to the i^{th} image with mean μ_i and standard deviation σ (σ is the same for all images). Your goal is to solve for the most likely image scores and to figure out which annotators are trying to cheat you.

In your write-up, use the following notation:

- $x_{ij} \in [0, 10]$: the score for i^{th} image from the j^{th} annotator
- $m_j \in \{0, 1\}$: whether each j^{th} annotator is "good" ($m_j = 1$) or "bad" ($m_j = 0$)
- $P(x_{ij}|m_j = 0) = \frac{1}{10}$: uniform distribution for bad annotators
- $P(x_{ij}|m_j = 1; \mu_i, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{1}{2}\frac{(x_{ij}-\mu_i)^2}{\sigma^2})$: normal distribution for good annotators
- $P(m_j = 1; \beta) = \beta$: prior probability for being a good annotator

2.1 Derivation of EM Algorithm (20 pts)

Derive the EM algorithm to solve for each μ_i, each m_j, σ, and β. Show the major steps of the derivation and make it clear how to compute each variable in the update step.

2.2 Application to Data (15 pts)

I've included a "annotation_data.mat" file containing

- annotator_ids(i) provides the ID of the annotator for the i^{th} annotation entry
- image_ids(i) provides the ID of the image for the i^{th} annotation entry
- annotation_scores(i) provides the i^{th} annotated score

Implement the EM algorithm to solve for all variables. You will need to initialize the probability that each annotator is good – set the initial value to 0.5 for each annotator.

(a) Report the indices corresponding to "bad" annotators (j for which m_j is most likely 0)
(b) Report the estimated value of σ
(c) Plot the estimated means μ_i of the image scores (e.g., plot(1:150, mean_scores(1:150)))

Tips

- For numerical stability, you should use logsumexp.m (http://www.cs.toronto.edu/pub/psala/Project/KPMtools/logsumexp.m) when computing the likelihood that an annotator is "good", given the current parameters. The joint probability of each annotators scores and "goodness" will be very small (close to zero by numerical precision). Thus, you should compute the log probabilities and use logsumexp to compute the denominator.
- Another useful functions is normpdf
- In each iteration, I like to show the bar plot of probability of "good" for each annotator and to show the plot of the mean score for each image. At the end, you should be very confident which annotators are "good".

3 Graph-cut Segmentation (30 pts)

Let us apply Graph-cuts for foreground/background segmentation. In the "cat" image, you are given a rough polygon of a foreground cat. Apply Graph-cuts to get a better segmentation.

First, you need an energy function. Your energy function should include a unary term, a data-independent smoothing term, and a contrast-sensitive smoothing term. Your unary terms should be $\log[P\frac{(\text{pixel}|\text{foreground})}{P(\text{pixel}|\text{background})}]$. Your pairwise term should include uniform smoothing and the contrast-sensitive term. To construct the unary term, use the provided polygon to obtain an estimate of foreground and background color likelihood. You may choose the likelihood distribution (e.g., color histograms or color mixture of Gaussians. Yes, you can use MATLAB GMM functions this time). Apply graph cut code for segmentation. You must define the graph structure and unary and pairwise terms and use the provided graph cut code or another package of your choice. Include in your writeup:

(a) Explain your foreground and background likelihood function. (5 pts)
(b) Write unary and pairwise term as well as the whole energy function, in math expressions. (5 pts)
(c) Your foreground and background likelihood map. Display $P(foreground|pixel)$ as an intensity map (bright = confident foreground). (10 pts)
(d) Final segmentation. Create an image for which the background pixels are blue, and the foreground pixels have the color of the input image. (10 pts)