Computer Vision CS543/ECE549
Homework 4

Due Date: April 3, 2012

1 Meanshift Segmentation (35 pts)

For this problem, you will implement the Meanshift algorithm and use it to segment images. Meanshift finds maxima for a non-parametric density function, by iteratively moving the mode estimation to the weighted mean of nearby points. To use it for image segmentation, first represent image pixels as data points with both spatial and color features. Then you will define a color and a spatial kernel of your choice and apply Meanshift algorithm to the datapoints. After getting the modes from Meanshift, you will merges those modes that are too near to each others and have the similar colors. Refer to Mean Shift: A Robust Approach Toward Feature Space Analysis for more details. Please refrain from using external source code.

1.1 Problem Description

You are given two images, included in the supplementary material, implement Meanshift algorithm and segment them. Bear in mind that being slow is kind of expected. My naive implementation takes 30 seconds to segment one image. Include in the writeup:

- The spatial and color kernel you used, as well as the Meanshift vector derived from your kernels. Write them in math expressions. (5pt)
- Briefly describe 1) Your stopping criteria for Meanshift. 2) Your strategy to merged the modes. (5pt)
- The "filtered" image after running Meanshift, before the mode merging step. (5pt)
- Your final segmentation. To display the segmented image, every pixel should have the average color of all pixels in the same segment. (15pt)
- Report run time and number of segments for each test image. (5pt)

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

Dealing with noisy annotations is a common problem in computer vision, especially when using crowdsourcing 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.

1. Report the indices corresponding to "bad" annotators (j for which m_j is most likely 0)
2. Report the estimated value of σ
3. 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-cuts (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 constrast-sensitive smoothing term. Your unary terms should be $-log[\frac{P(pixel|foreground)}{P(pixel|background)}]$. 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:

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