Instructions

You can use your own laptop if you want but you will need the **statistical toolbox** to be installed. Otherwise, please use one of the computer in the room. You have 1h 45 minutes. You have to submit your files by 10.01AM. Any late submission will result in a penalty of -5 points (total is 100 points) for every minute.

Notify one of the assistants before leaving the room.

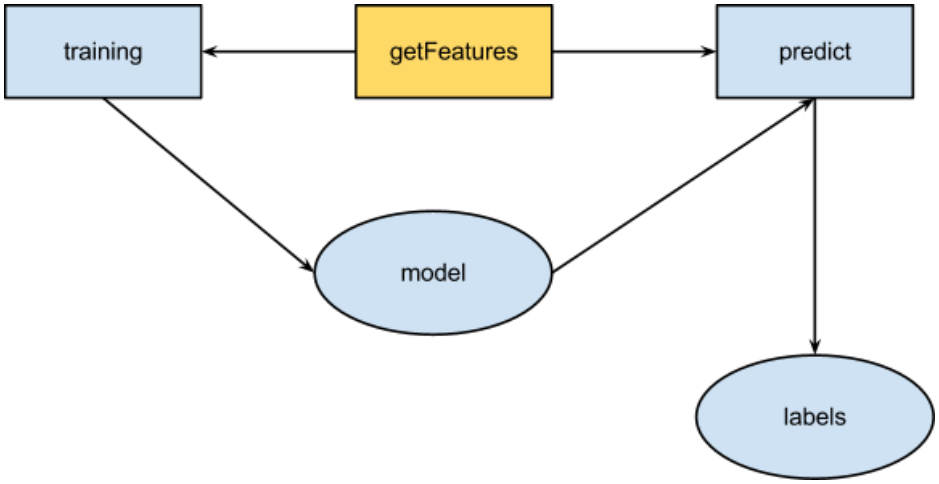
Once you are done, make sure that your files are successfully uploaded to the system by checking with one of the assistants.

Access to Internet resources is forbidden.

Introduction

The goal of this exercise session is to segment interesting objects in biomedical images.

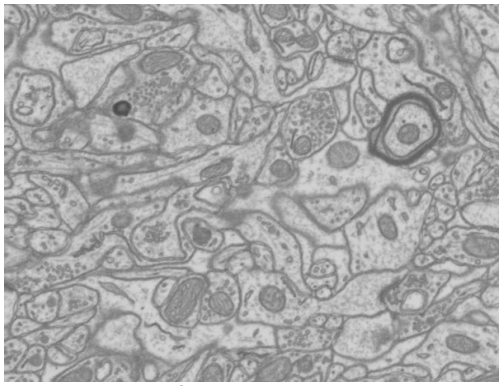
We use a binary classifier called a decision tree to classify patches extracted from each image. The patches are square regions of size YxY.



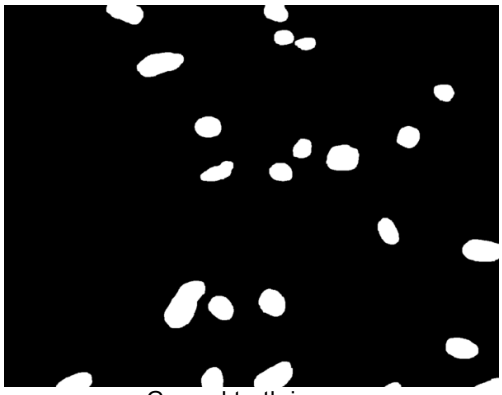
As illustrated in the above diagram, the decision tree is first trained to recognize the target objects indicated by white pixels in the annotation images (see the folder train/annotations_x). After training, we obtain a model that can then be used to predict a label for a given image patch. We use the decision tree function from matlab (classregtree) so you are not required to know what the model is. You are asked to construct the feature vectors that are passed to the classifier. You should write your code only in the getFeatures.m file for the first Exercise and in the computeHessianEig.m file for the second one.

Exercise 1

This first exercise will focus on segmenting mitochondria in biomedical images. A sample image is given below with the ground truth annotation in which white pixels indicate mitochondria while the black pixels indicate the background class.

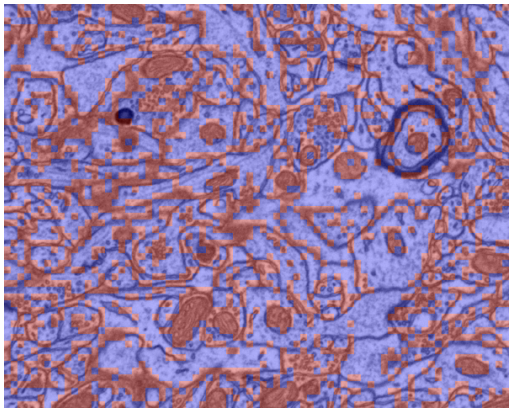


Original image



Ground truth image

First, download and unzip the file named ex4.zip. Start matlab and run the function called “main_exercise.m” that will train the classifier and then test it on a single image. The default example uses the intensity of the pixel at the center of the patch as the only feature. The result is shown on the image below and should also appear in the directory named “predictions” after you run the matlab script.

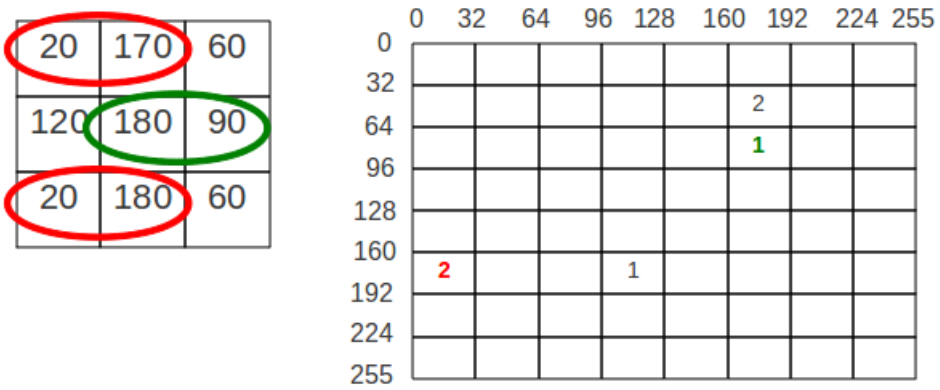


Classifier prediction using the default feature. The red overlay indicates what the decision tree classifies as mitochondria.

We will now construct more discriminative features that will lead to a better segmentation by following the steps below:

1. Write a new feature that will extract the mean and variance intensity in a patch. You have to implement your code in the getFeatures function (the location is indicated in the code by the following comment : “Question 1.1”).

2. You are now asked to write a co-occurrence feature that will count the number of times pixel intensities co-occur next to each other in the horizontal direction.

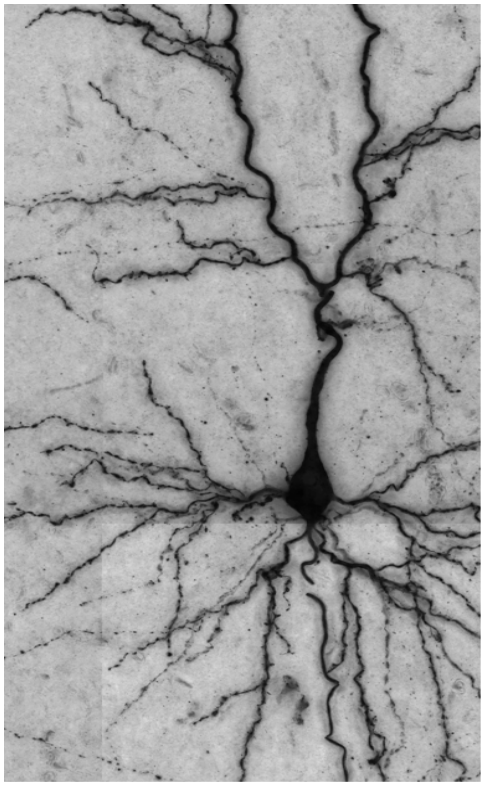


An illustration is given above. The matrix on the left side represents the pixel intensities of a 3 by 3 patch extracted from the image while the matrix on the right is the co-occurrence feature we wish to construct. Note that we discretize the pixels intensities into 8 intervals, each interval covering 256/8=32 pixel intensities. For example, the first interval covers all the pixel intensities between 0 and 31 while the last interval covers values from 223 to 255. The feature vectors returned by your function will be the serialized co-occurence matrix.

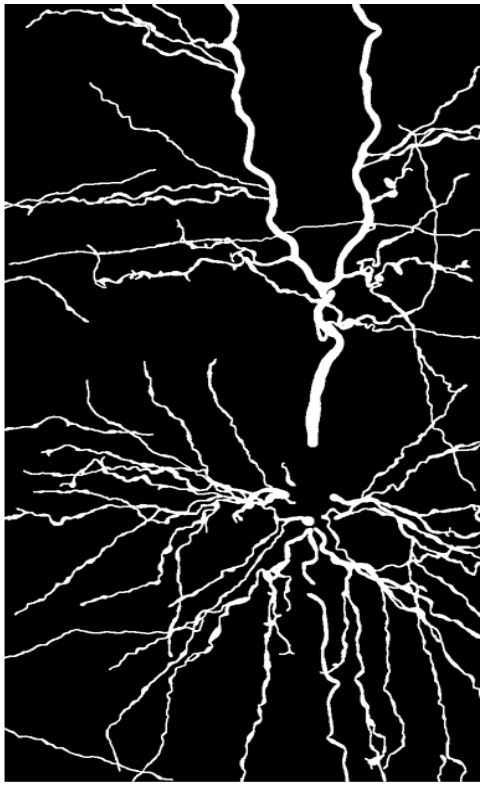
3. We will now make two changes to the previous feature of item 2. First, include also the vertical co-occurrences in the same co-occurence matrix. Second, the minimum and maximum pixel intensities used to construct the intervals in the previous question are not adapted to the actual range of pixel intensities of the training image. You have to compute these values from the training image itself and fill them in the matlab file.

Exercise 2

This exercise focuses on segmenting elongated structures in biomedical images. Examples of such structures are blood vessels in retinal/CT scans and neurites in light microscopy images. In this exercise, we will detect dendritic branches in 2D light microscopy images, such as the ones shown below:



Original Image



Ground Truth Segmentation

We will use the same classification system as the previous exercise. You are asked to implement a new feature to measure the *ridgeness* of an image pixel and its neighborhood. The computation of this feature will be based on the eigenvalues of the Hessian matrix:

H = [Ixx Ixy; Ixy Iyy];

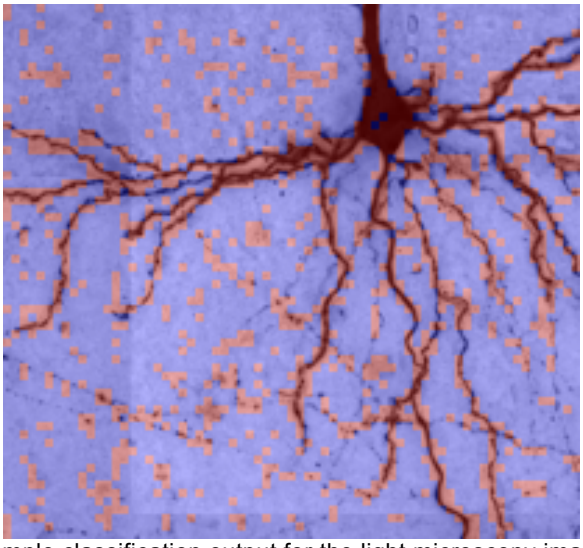
where Ixx and Iyy are the second order image derivatives along the x and the y axis, respectively. For dark structures on a bright background, a frequently used ridgeness measure is defined as the absolute value of the maximum eigenvalue. In this exercise, rather than using such a hand-designed metric, we will train a decision tree classifier to learn the ridge-like appearance of the elongated structures in the image based on both eigenvalues defined as:

lamda_1 = 0.5 * (Ixx + Iyy - sqrt((Ixx - Iyy).^2 + 4*Ixy.^2));
lamda_2 = 0.5 * (Ixx + Iyy + sqrt((Ixx - Iyy).^2 + 4*Ixy.^2));

Prior to computing the image derivatives, we will smooth the image with a Gaussian filter of certain standard deviation in order to get rid of the background noise and obtain a smooth response on the elongated structures. Since the width of dendrites vary considerably, we will adapt a multi-scale approach to detect them. More specifically, we will compute multiple eigenvalue pairs for a given set of predefined standard deviations *S*, where each value in the set corresponds to a possible dendrite radius. Therefore, for a given number |*S*| of standard deviations, we will compute *2*|S|* features (i.e., eigenvalues) for each pixel. In this exercise, the standard deviation set *S* is equal to *{1.0, 2.0, 3.0}*.

Although in the previous exercise, we have computed the feature vectors for each image patch separately, we will now speed-up this procedure by pre-computing them densely for the whole image. The feature vector for a patch will then be taken as the concatenation of the feature vectors of all the pixels within this patch.

Your task is to implement the dense feature extraction step encapsulated in the *computeHessianEig* function. Use the Sobel operator for the derivative computations.



A sample classification output for the light microscopy images.