

The deadline will be enforced strictly, please talk to me now if you will have work commitments.

Introduction

Each of the sections below constitute one problem. If the problem asks for images, you should turn in a print-out with the requested images. Ideally, your assignment should be composed in a word-processor, such as L^AT_EX or Microsoft Word. You are welcome to write out derivations by hand.

In your writeup, describe the steps you completed for each problem and show the results. Readability will be part of your grade.

Because this is a midterm, you are not permitted to collaborate with anyone else on completing these problems

1 Learning to Detect Edges

In this problem, you will train an “optimal” edge-detector. From the Berkeley Segmentation Database, we have extracted a set of 13×13 image patches, along with a label describing whether an edge is present at the center of the image or not.

Note that in this problem you will be training a logistic regression model rather than a boosting-type model. There are no weak learners in this model.

1. Write a function that takes an image patch and generates a feature vector. Remember, in the logistic regression model, the probability of an item taking the label +1 is estimated by

$$p(l_i = +1 | \mathbf{x}_i) = \frac{1}{(1 + \exp(-(\mathbf{x}_i \cdot \mathbf{w})))}$$

where \mathbf{w} is a vector of weights that you will learn using the training code from Problem Set 2 (This was denoted using θ in PS2, here we use θ as an angle for the filters). In this first problem, you are constructing the feature vector \mathbf{x}_i for each image patch.

For features, you should use the absolute value of the response of the image patch to a set of oriented derivative of Gaussian features. A first derivative of Gaussian filter, with orientation θ can be computed as:

$$f(x, y) = \left[\cos(\theta) 2xe^{-\frac{(x^2+y^2)}{\sigma^2}} + \sin(\theta) 2ye^{-\frac{(x^2+y^2)}{\sigma^2}} \right] \quad (1)$$

A second derivative of Gaussian filter, with orientation θ can be calculated as

$$f(x, y) = \left[\cos^2(\theta) \left(2\frac{x^2}{\sigma^2} - 1 \right) e^{-\frac{(x^2+y^2)}{\sigma^2}} - 2 \cos(\theta) \sin(\theta) \frac{xy}{\sigma^2} e^{-\frac{(x^2+y^2)}{\sigma^2}} + \sin^2(\theta) \left(2\frac{y^2}{\sigma^2} - 1 \right) e^{-\frac{(x^2+y^2)}{\sigma^2}} \right] \quad (2)$$

You may choose the orientations to use, but there should be at least 3 orientations in the set and at least two scales (which correspond to values of σ). Solving problem 3 will be much easier if you use “valid” convolution to generate each feature.

Next, write a wrapper function that builds a feature *matrix* out of all of the training examples. To add a bias term, you should ensure that your matrix of features has a column where every value is equal to one.

2. Modify your code from Problem Set 2 to train a classifier using this data. Now that you have more than two dimensions, you will need to disable the visualization code.

Use the test set of patches provided to plot an ROC curve by changing the varying the threshold for classifying a patch as containing an edge or not. Your logistic regression model will return the probability of each patch containing an edge pixel. You will then compare this probability to a threshold to decide if that patch contains an edge.

To produce each point on the ROC curve, you will evaluate $P[D]$ and $P[FA]$ for a certain threshold. The curve is produced by plotting the points generated for different thresholds. $P[D]$, or probability of detection, is the percentage of examples labeled +1 that your classifier labels +1. $P[FA]$, or probability of false alarm, is the percentage of examples labeled -1 that your classifier labels +1.

3. Adapt your code to use entire images. In other words, build an edge-detector that takes a whole image as an input and uses the weights that were learned to predict if an edge is at every pixel or not. Run on some of the images from the Berkeley Segmentation Database, which is linked from the course homepage. How does your edge detector compare with the results from humans?
4. Create a new type of feature, e.g. the local max of a filter response. Retrain the classifier and compare to the old classifier using either ROC curves or classification performance.